Available online at www.sciencedirect.com
ScienceDirect
Neurocomputing 71 (2008) 2330–2343
NEUROCOMPUTING
www.elsevier.com/locate/neucom

Estimating the mixing matrix in Sparse Component Analysis (SCA) based on partial k-dimensional subspace clustering

Farid Movahedi Naini[a], G. Hosein Mohimani[a], Massoud Babaie-Zadeh[a,*,1], Christian Jutten[b,1]

[a]*Electrical Engineering Department, Sharif University of Technology, Tehran, Iran*
[b]*Laboratory of Images and Signals (CNRS UMR 5083, INPG, UJF), Grenoble, France*

Available online 15 February 2008

Abstract

One of the major problems in underdetermined Sparse Component Analysis (SCA) in the field of (semi) Blind Source Separation (BSS) is the appropriate estimation of the mixing matrix, $\mathbf{A}$, in the linear model $\mathbf{X} = \mathbf{AS}$, especially where more than one source is active at each instant of time. Most existing algorithms require the restriction that at each instant (i.e. in each column of the source matrix $\mathbf{S}$), there is at most one single dominant component. Moreover, these algorithms require that the number of sources must be determined in advance. In this paper, we proposed a new algorithm for estimating the matrix $\mathbf{A}$, which does not require the restriction of single dominant source at each instant. Moreover, it is not necessary that the exact number of sources be known a priori.

Keywords: Underdetermined Blind Source Separation (BSS); Sparse Component Analysis (SCA); Subspace fitting

1. Introduction

Because of its many applications, the problem of Blind Source Separation (BSS) has been extensively studied in the last 20 years (refer for example to the books [12,6,24]). This problem consists of separating a set of mixed signals from their mixtures, where there are prior information neither about the mixing system nor about the source signals (except their statistical independence).

The quality of source separation approaches may be significantly improved, if we pass from the totally 'blind' case to the semi-blind case, that is, if we take the advantage of a very weak *a priori* information available about the source signals [3]. For example, the priors 'non-stationarity' of the source signals [16] and their 'temporal correlation' [20,4,17] have been already used in the early literature of source separation.

Another prior information, which is relatively newer in source separation, is 'sparsity' of the source signals [11,23,5,9]. A sparse signal is a signal whose most samples are nearly zero, and just a few percent take significant values. Consequently, at each instant ('time' slot), only a few number of sources have significant values (say they are 'active'), and most of them are almost zero (say they are 'inactive'). This prior information is important because of two reasons. Firstly, in contrast to traditional source separation methods, it permits source separation for the case in which the number of sources exceeds the number of sensors [5,7–9,13–15,23]. Secondly, it is a very practical assumption for many sources: even if the sources are non-sparse in time domain, they may be sparse in another (linear) transformed domain. Since the mixing system is identical in time and transformed domains (because of linearity of the transform), the sources may be separated in the transformed domain. For example, the speech sources may not be sparse enough in time, but they are sparse in time-frequency (using Short-Time Fourier Transform = STFT) or time-scale (using wavelet packet) domains [11].

*Corresponding author.
E-mail addresses: mbzadeh@yahoo.com (M. Babaie-Zadeh), Christian.Jutten@inpg.fr (C. Jutten).
[1]This work has been partially funded by French Embassy in Tehran, by Center for International Research and Collaboration (ISMO) and by Iran National Science Foundation (INSF).

0925-2312/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2007.07.035

Source separation, based on the sparsity prior, mainly relies on the sparsity of sources, and not on their independence. Consequently, these methods (for extracting sparse components of the mixtures) are usually called Sparse Component Analysis (SCA) [9].

The problem of SCA can be stated as follows. Consider the linear model:

$$\mathbf{X} = \mathbf{AS}, \tag{1}$$

where $\mathbf{A} = [\mathbf{a}_1 \dots \mathbf{a}_n] \in \mathbb{R}^{m \times n}$ is the mixing matrix, $\mathbf{S} = [\mathbf{s}_1 \dots \mathbf{s}_T] \in \mathbb{R}^{n \times T}$ and $\mathbf{X} = [\mathbf{x}_1 \dots \mathbf{x}_T] \in \mathbb{R}^{m \times T}$ are the matrices of n sources and m observed signals. Each column of $\mathbf{S}$ and $\mathbf{X}$ corresponds to an instant of 'time' and T is the number of 'time' samples. Sparsity of source signals implies that in each column of $\mathbf{S}$, there are just a few significant values (active sources) and most of the elements are almost zero (inactive sources). The goal of SCA is then to estimate $\mathbf{A}$ and $\mathbf{S}$, only from $\mathbf{X}$ and the sparsity assumption. In this paper, each column of the mixing matrix, i.e. each $\mathbf{a}_i$, $1 \leqslant i \leqslant n$, is called a *mixing vector*.

Although the word 'time' is used in the above paragraphs ('time' samples, instant of 'time' and 'time' slot), and will be used in the continuation of this paper, the above model may be in another domain, in which the sparsity assumption holds. To see this, let $\mathscr{T}$ be a linear 'sparsifying' transform (like STFT or wavelet packet transforms for speech signals), and the mixing system is stated as $\mathbf{X} = \mathbf{AS}$ in the time domain. Then, we have $\mathscr{T}\{\mathbf{X}\} = \mathbf{A}\mathscr{T}\{\mathbf{S}\}$ in the transformed domain, and because of the sparsity of $\mathscr{T}\{\mathbf{S}\}$, it is in the form of (1).

Generally, more than one source may be active at each instant of time. The number of active sources at each instant is a random variable and its average[2] is denoted by k.

The SCA problem is usually solved in two steps. The first step is the estimation of the mixing matrix ($\mathbf{A}$), and the second step is the recovery of the source signals ($\mathbf{S}$) by knowing the mixing matrix. Note that in the underdetermined case, in which the number of sources exceeds the number of sensors, these two problems are not identical. In other words, knowing the mixing matrix does not directly result in the recovery of the sources [2]. In this paper, we address only the problem of the estimation of the mixing matrix.

In the field of SCA, two different cases should be distinguished for estimating the mixing matrix; single dominant component and multiple dominant components. In the former, the average number of active sources is less than or approximately equal to one, and the scatter plot of $\mathbf{X}$ geometrically shows the data concentration directions (see Fig. 1a). To see this, note that at each instant we have $\mathbf{x}(t) = \mathbf{As}(t) = s_1(t)\mathbf{a}_1 + \cdots + s_n(t)\mathbf{a}_n$, $1 \leqslant t \leqslant T$. For most instants, only one of values of s_i, $1 \leqslant i \leqslant n$ is dominant and the others are almost zero.

[2]In fact, this random variable may have non-integer average. In this case, k is the closest integer to this average.

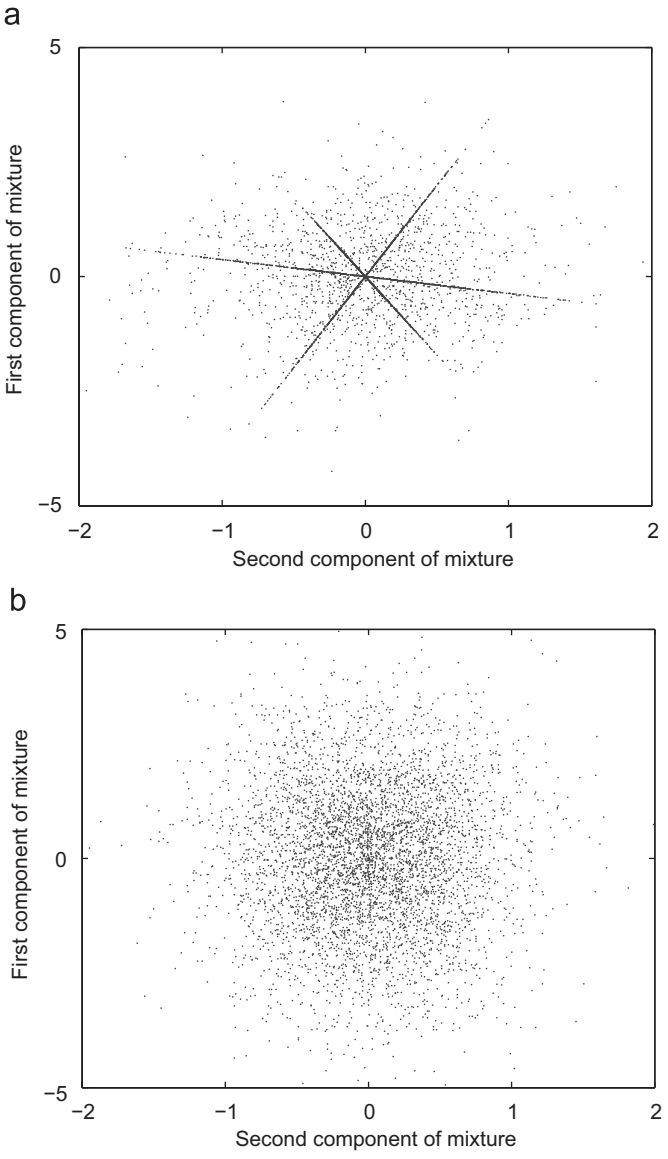

Fig. 1. Scatter plot of first mixture versus second mixture in the case $n = 3$ and $m = 2$. In (a) $k = 1$ and three data concentration directions are easily distinguished which is equal to the number of sources. In (b) $k > 1$, identifying the concentration directions is not easy.

Consequently, in most samples, $\mathbf{x}(t)$ is in the direction of one of the mixing vectors. Conversely, in the multiple dominant case, the average number of active sources is greater than one and the scatter plot of $\mathbf{X}$ is not sufficient for estimating $\mathbf{A}$ (see Fig. 1b). Up to now, many papers have addressed the single dominant case [21,11,5,9], while only a few researchers have considered the multiple dominant case [1,10,22]. In this paper, we focus on the latter case.

In the single dominant component SCA, the observed data in the m-dimensional scatter plot of mixtures concentrate along the directions of n mixing vectors. Similarly, in the multiple dominant components SCA, the observed data concentrate around k-dimensional subspaces

which are spanned by a set of k mixing vectors. In fact, if we assume that in a time sample, only k sources, $s_{i_1}(t)\dots s_{i_k}(t)$, are active and other sources are nearly zero, then the mixture vector $\mathbf{x}(t)$ concentrates around the k-dimensional subspace spanned by $\mathbf{a}_{i_1}\dots\mathbf{a}_{i_k}$.

$$\mathbf{x}(t)=\mathbf{A}\mathbf{s}(t)\simeq s_{i_1}(t)\mathbf{a}_{i_1}+\cdots+s_{i_k}(t)\mathbf{a}_{i_k}. \tag{2}$$

The total number of these subspaces is equal to $N_p=\binom{n}{k}$. We call these subspaces *concentration subspaces* throughout this paper. This concept is also used in [22].

The basic idea of the paper is to find these k-dimensional concentration subspaces, and then to estimate the mixing vectors using them. This general idea has also been used by Washizawa et al. in [22], but with our method this goal is achieved by another technique which has lower computational cost and lets us to solve the medium scale problems. In fact, in our method, it is not necessary to find all $N_p=\binom{n}{k}$ concentration subspaces. If some of these subspaces are found mistakenly, the estimating part of the mixing matrix is in a way robust to these errors. Moreover, contrary to the previous approaches [1,10,22], the number of sources, n, need not to be known in advance.

It should be emphasized that in this paper, k, the average number of active sources, is assumed to be determined a priori. Estimating k from the data is a subject for further investigation, which is currently being studied in our group and a method has been proposed in [18].

The paper is organized as follows. In the following section, we will explain the procedure of estimating the concentration subspaces. In Section 3, a method for estimating the mixing vectors from the estimated concentration subspaces is developed. In Section 4, the algorithm for estimating the matrix $\mathbf{A}$ is finalized, while Section 5 presents various computer simulations to justify the algorithm. After a short comment on choosing the parameters of the algorithm in Section 6, the paper will be concluded in Section 7.

2. Estimating concentration subspaces

In this section, we try to estimate k-dimensional concentration subspaces. Each k-dimensional subspace can be represented by an m by k matrix, whose columns form an orthonormal basis for the subspace.[3] In this paper, we do not distinguish between a subspace and its matrix representation.

Let $\mathbf{B}\in\mathbb{R}^{m\times k}$ be the matrix representation of an arbitrary k-dimensional subspace. We define the following function to detect whether $\mathbf{B}$ is a concentration subspace or not:

$$f_\sigma(\mathbf{B})=\sum_{i=1}^{\mathrm{T}}\exp\left(\frac{-d^2(\mathbf{x}_i,\mathbf{B})}{2\sigma^2}\right), \tag{3}$$

where $d(\mathbf{x}_i,\mathbf{B})$ is the distance of $\mathbf{x}_i$ from the subspace represented by $\mathbf{B}$ (the definition of this distance is presented in Appendix A).

[3]Note that this representation is not unique.

For small values of $d(\mathbf{x}_i,\mathbf{B})$ compared to σ, $\exp(-d^2(\mathbf{x}_i,\mathbf{B})/2\sigma^2)$ is about 1 and for large values of $d(\mathbf{x}_i,\mathbf{B})$, it is nearly zero. Therefore, for sufficiently small values of σ, the above function *is approximately equal to the number of data points close to* $\mathbf{B}$. Moreover, if the set of points are concentrated around several different k-dimensional concentration subspaces, f has a local maximum where $\mathbf{B}$ is close to the basis of each of them. These local maxima are very strong if σ is small enough. In fact, their values are approximately equal to the number of data samples which lie in that subspace. Therefore, by maximizing the function f, we actually maximize the number of data points close to $\mathbf{B}$. Now two examples are presented for demonstrating this function.

Example 1. Consider the problem for the case $n=3$, $m=2$ and $k=1$ (for the plots of this example, as well as Examples 2, 3 and 4, we have used $T=2000$ data samples). In this case, there are three sources, two mixtures and only one source is active in most instants of time. Therefore, the observed data in the scatter plot of the first mixture versus the second one concentrate along the directions of three mixing vectors (see Fig. 1a).

To have a demonstration of the function f, defined in (3), the data points near origin are first omitted (they lie in all concentration subspaces) and the remaining points are projected onto the surface of a unit semi-sphere (by normalizing the data in every sample and forcing sign of the first component to be positive. This operation does not change the subspace of a data sample). Then, $\mathbf{B}$ is set equal to $[\cos(\varphi),\sin(\varphi)]^{\mathrm{T}}$ and f is plotted versus φ for $0\leqslant\varphi\leqslant\pi$. The results are shown in Figs. 2a–c for different values of σ. The histogram of data concentration versus φ is also plotted in Fig. 3.

As it is seen in the figures, for the smallest σ, the function f is similar to the histogram and not smooth; the peaks represent the mixing vectors and can be distinguished in the figure.

However, for the smallest σ, because of the existence of many local maxima, maximization of f is not easy. For the medium σ, f is smooth and maximization is easier. For the largest one, the peaks are mixed and cannot be distinguished.

Example 2. In this example, another demonstration of f, defined in (3), for the case $n=5$, $m=3$ and $k=2$ is presented. In this case, there are five mixing vectors in the three-dimensional space. The data concentrate around $\binom{5}{2}=10$ two-dimensional concentration subspaces.

Given an arbitrary two-dimensional subspace, let $\mathbf{B}$ be its matrix representation and (x,y,z) be its normal vector[4] representation in Cartesian coordinate. This vector can be

[4]In this case, $\mathbf{B}$ is a hyperplane in the three-dimensional space. Its normal vector, by definition, is the vector on the unit sphere which is perpendicular to the hyperplane.

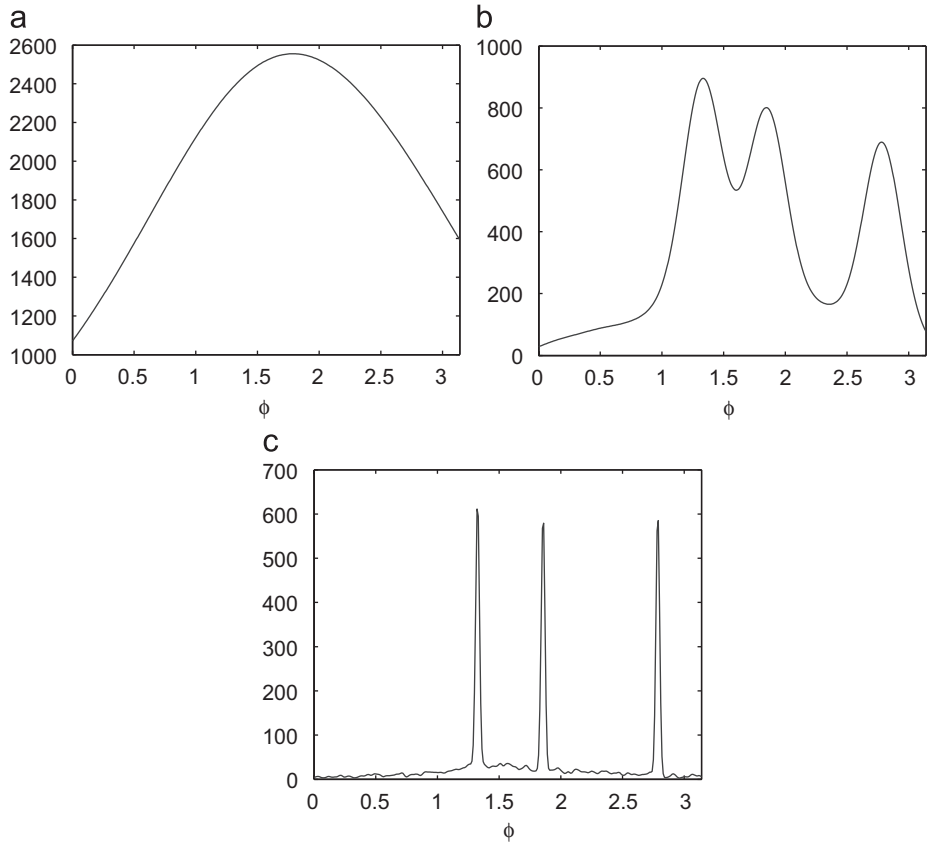

Fig. 2. Graph of the function $f([\cos(\varphi), \sin(\varphi)]^{\mathrm{T}})$ versus φ in the case $n = 3$, $m = 2$ and $k = 1$ for three different values of σ. In (a) $\sigma = 1$, the function is smooth and peaks are mixed. In (b) $\sigma = 0.15$. In (c) $\sigma = 0.015$, the function lacks smoothness while peaks are completely discriminated.

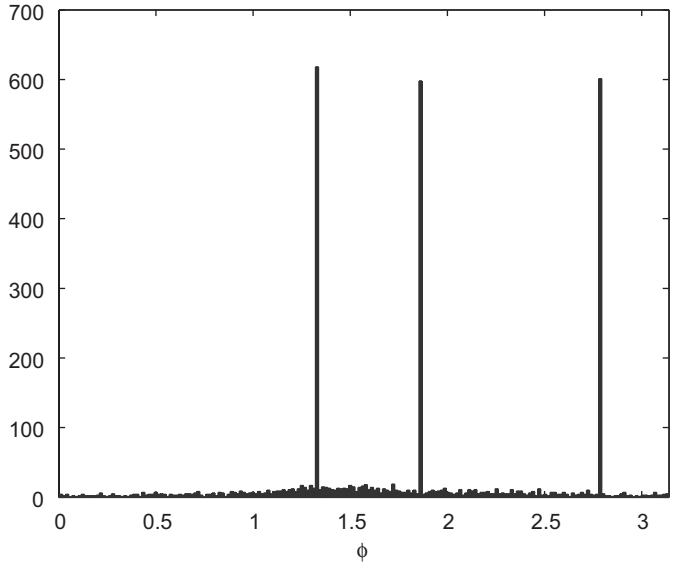

Fig. 3. Angular histogram of the data concentration in the case $n = 3$, $m = 2$ and $k = 1$. Note the similarities between this histogram and graph of the function f in the case $\sigma = 0.015$ in Fig. 2c.

represented by (φ, θ) satisfying:

$$\begin{cases} x = \sin(\varphi)\sin(\theta), \\ y = \cos(\varphi)\sin(\theta), \\ z = \cos(\theta) \end{cases} \tag{4}$$

The function f is plotted versus φ, θ for $0 \leqslant \varphi, \theta \leqslant \pi$ and different values of σ. The results are shown in Figs. 4a and b.

Note that for larger σ, f is smoother, but for smaller one, the peak locations are better distinguishable (all 10 concentration subspaces are separated). Therefore, they form a better representation for the actual concentration subspaces.

The idea is to maximize the function *f for a small value of* σ, using a maximization method. However, for small σ's, many local maxima exist which make this maximization difficult. But even in this case, if we have a good initial guess about the location of the maximum, then by starting from this initial point, the maximization algorithm may easily find the actual maximum. Our idea is then to use the maximum obtained from the maximization of f for a larger σ, as the initial guess for the location of the maximum for smaller σ. This suggests to use a decreasing sequence of σ in order to obtain an accurate estimation.

Up to now, estimating the concentration subspaces is discussed. The presentation of the final algorithm is delayed to Section 4, after introducing the method of estimating the mixing vectors from concentration subspaces in the next section.

3. Estimating mixing vectors

Now suppose all of the k-dimensional concentration subspaces are estimated and their representation matrices

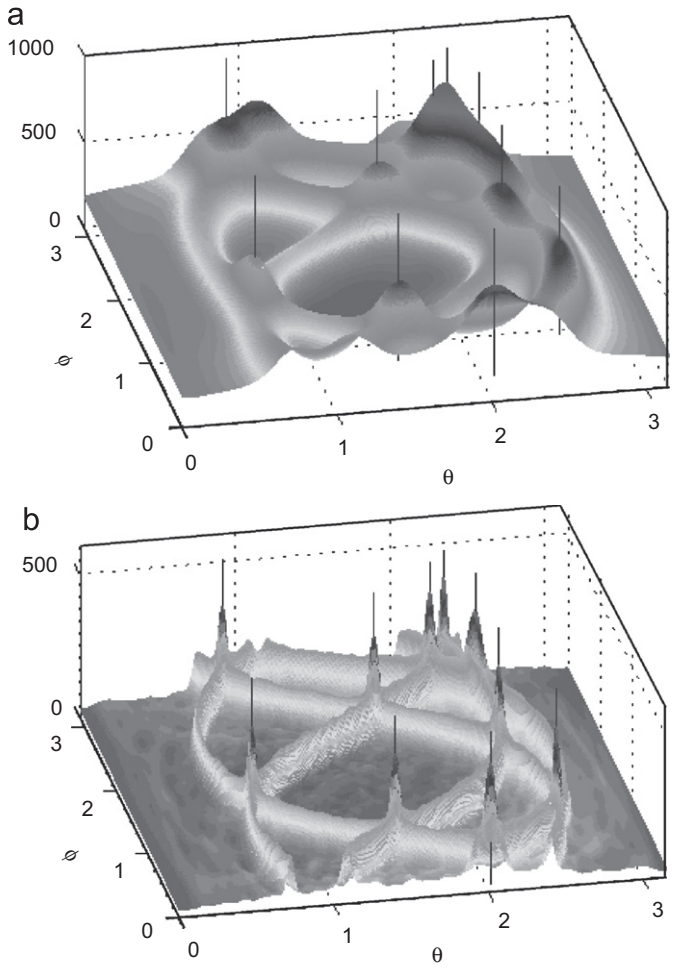

Fig. 4. Graph of the function f in the case $n = 5$, $m = 3$ and $k = 2$ for two different values of σ. In (a) $\sigma = 0.1$ and in (b) $\sigma = 0.02$. The exact location of the actual concentration subspaces are indicated by vertical lines. Similar to Fig. 2, by decreasing σ, discrimination increases while a decrease in smoothness is observed.

are $\mathbf{B}_i, i = 1 \ldots N_p$. This section is dedicated to estimating the mixing vectors using these subspaces. To do this, we use an idea similar to the idea we used in the previous section to find the concentration subspaces.

As mentioned before, every concentration subspace is spanned by a set of k mixing vectors. The number of concentration subspaces which include a certain mixing vector, equals to the number of choices of other $k-1$ mixing vectors from the total $n-1$ ones. Therefore, every mixing vector lies in $\binom{n-1}{k-1}$ of the subspaces. Given an arbitrary vector $\mathbf{v}$ in the m-dimensional space, we define the following function:

$$g_\sigma(\mathbf{v}) = \sum_{i=1}^{N_p} \exp\left(\frac{-d^2(\mathbf{v}, \mathbf{B}_i)}{2\sigma^2}\right), \tag{5}$$

where $d(\mathbf{v}, \mathbf{B}_i)$ is the distance of the vector $\mathbf{v}$ from the ith estimated concentration subspace (refer to Appendix A for the definition of this distance).

For small values of $d(\mathbf{v}, \mathbf{B}_i)$ compared to σ, $\exp\left(-d^2(\mathbf{v}, \mathbf{B}_i)/2\sigma^2\right)$ is about 1 and for large values of $d(\mathbf{v}, \mathbf{B}_i)$, it is almost zero. Thus, for sufficiently small values of σ, the function g *is approximately equal to the number of subspaces close to* $\mathbf{v}$. Note that the σ used in this formula is different from the previous one. In order to distinguish the two, the one related to finding subspaces is denoted by $\sigma_{\mathbf{B}}$ and the one related to finding mixing vectors is denoted by $\sigma_{\mathbf{A}}$. The next example explains the behavior of this function.

Example 3. Here, we demonstrate the function g for the case of example 2, in which $n = 5$, $m = 3$ and $k = 2$. There are $\binom{5}{2} = 10$ concentration subspaces, all of them are presumed to be already estimated. Each mixing vector is close to $\binom{5-1}{2-1} = 4$ of these estimated subspaces.

All of the surface points of the unit semi-sphere are spanned as follows. Each vector with Cartesian coordinate (x, y, z) is transformed to (φ, θ) satisfying (4). The function g is shown versus φ, θ for $0 \leqslant \varphi, \theta \leqslant \pi$ in Fig. 5. According to (4), a two-dimensional subspace defined by equation $z = ax + by$ can be represented in this system of coordinates by equation:

$$\cos(\theta) = a \sin(\varphi) \sin(\theta) + b \cos(\varphi) \sin(\theta).$$

Therefore, any arbitrary two-dimensional subspace is transformed to a curve in this figure. In fact, each distinguished curve in the figure represents one of the 10 concentration subspaces.

In this case every mixing vector lies in four subspaces. Therefore, the absolute maxima that are located in the intersection of four curves represent the mixing vectors and are easily distinguishable in the figure. Note that the values of these peaks are nearly 4, which is equal to the number of concentration subspaces in which they lie.

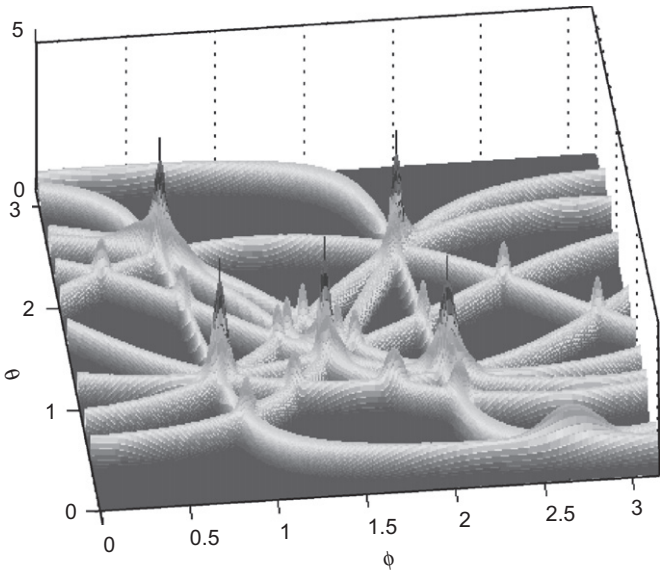

Fig. 5. Graph of the function g in the case $n = 5$, $m = 3$ and $k = 2$ for $\sigma = 0.02$. Distinguished curves represent 10 two-dimensional concentration subspaces and discriminated peaks represent five mixing vectors. The exact location of the actual mixing vectors are indicated by added vertical lines. Note that each mixing vector is the intersection of four concentration subspaces.

As observed in the figure, there exists local maxima which make it difficult to design a maximization algorithm for estimating the mixing vectors. In general, each mixing vector lies in $\binom{n-1}{k-1}$ concentration subspaces, but other vectors are close to relatively smaller number of concentration subspaces, say less than q concentration subspaces. Consequently, in order to identify the mixing vectors correctly, we detect the vectors which lie in at least q concentration subspaces.

Note that if q is set to $\binom{n-1}{k-1}$ then all of the concentration subspaces must be estimated accurately. However, the value of q can be set much less in some experiences, and the method accomplishes correctly without requiring all the concentration subspaces to be estimated.

Moreover, in determining the mixing vectors, instead of working with the function g (as defined in (5)), we define the following function to better discriminate between the picks corresponding to mixing vectors, and to force each detected mixing vector to lie in at least q concentration subspaces:

$$h_\sigma(\mathbf{v}) = \sum_{1 \leqslant i_1 < \cdots < i_q \leqslant N_p} u_\sigma(\mathbf{v}, \mathbf{B}_{i_1}) \cdots u_\sigma(\mathbf{v}, \mathbf{B}_{i_q}), \tag{6}$$

where

$$u_\sigma(\mathbf{v}, \mathbf{B}) = \exp(-d^2(\mathbf{v}, \mathbf{B})/2\sigma^2)$$

and $d(\mathbf{v}, \mathbf{B})$ is the distance of vector $\mathbf{v}$ from subspace $\mathbf{B}$. For sufficiently small values of σ, if $\mathbf{v}$ is close to $\mathbf{B}$ then $u_\sigma(\mathbf{v}, \mathbf{B})$ is about 1 and otherwise it is almost zero. Thus,

$$u_\sigma(\mathbf{v}, \mathbf{B}_{i_1}) \cdots u_\sigma(\mathbf{v}, \mathbf{B}_{i_q}) \approx \begin{cases} 1 & \text{if } \mathbf{v} \text{ is close to all } \mathbf{B}_{i_1} \cdots \mathbf{B}_{i_q}, \\ 0 & \text{otherwise.} \end{cases}$$

This means that $h_\sigma(\mathbf{v})$ is significant if at least one of the summands is significant, i.e. if $\mathbf{v}$ is near to the corresponding subset of q subspaces. Therefore, this function can be utilized for finding mixing vectors. Note that direct computation of (6) is too time-consuming, and should be avoided. Instead, a fast algorithm for computing (6) is presented in Appendix B. The function h is shown for the same case of example 2, with $q = 4$ and different values of σ in Fig. 6.

As it can be observed in the figures, the mixing vectors are more distinguishable and the maximization process is simpler for detecting these maxima.

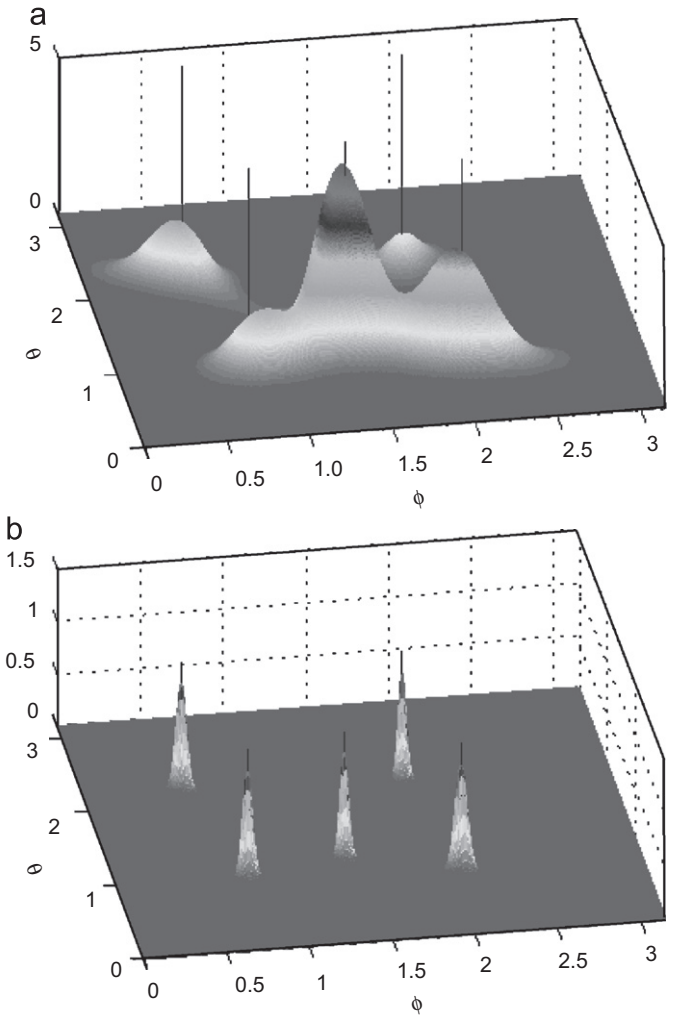

Fig. 6. Graph of the function h in the case $n = 5$, $m = 3$, $k = 2$ and $q = 4$ for two different values of σ. In (a) $\sigma = 0.15$ and in (b) $\sigma = 0.03$. The exact location of the actual mixing vectors are indicated by added vertical lines. As usual, value of σ experiences a trade-off between smoothness and discrimination.

4. Final algorithm of identifying the mixing matrix

As mentioned in Sections 2 and 3, two decreasing sequences of σ are used in this algorithm. We denote them by $[\sigma_1 \ldots \sigma_R]$. However, their lengths and their values can be different.

Usually, the estimation of all the $N_p = \binom{n}{k}$ concentration subspaces is not necessary. It is sufficient to estimate as many concentration subspaces to guarantee the existence of any mixing vectors in at least q of the estimated concentration subspaces.

Now presume that $N_{\mathbf{B}}$ of the k-dimensional concentration subspaces are chosen randomly from the set of all N_p concentration subspaces. Because each concentration subspace contains exactly k of the mixing vectors, the probability that a certain mixing vector is included in a certain subspace is k/n. Therefore, the probability that it is included in less than q of the $N_{\mathbf{B}}$ subspaces is

$$\sum_{r=0}^{q-1} \binom{N_{\mathbf{B}}}{r} \left(\frac{k}{n}\right)^r \left(1 - \frac{k}{n}\right)^{(q-1-r)} \approx G\left(\frac{q - 1 - N_{\mathbf{B}}(k/n)}{\sqrt{N_{\mathbf{B}}(k/n)(1 - k/n)}}\right), \tag{7}$$

where G is the Gaussian Cumulative Distribution Function.[5]

$N_{\mathbf{B}}$ should be chosen so large to make the above probability less than a certain amount α. This results in a

[5]Conditions required for the approximation is discussed in [19, pp. 70–73].

rough estimation of $N_{\mathbf{B}}$ which can be used in the experimental results for both cases of determined and undetermined number of sources.

Now we present the algorithm for estimating the mixing matrix. This algorithm is composed of two parts: identifying the concentration subspaces (Section 4.1) and then identifying the mixing vectors (Section 4.2). Suitable choice of the parameters and their effect on the performance are discussed in Section 6.

4.1. Algorithm for identifying the concentration subspaces

The idea of the following algorithm is to start from randomly different starting points, with the hope of finding different maxima. To achieve this, instead of trying $N_{\mathbf{B}}$ starting points, we use $L_{\mathbf{B}}$ starting points (where $L_{\mathbf{B}}$ is several times greater than $N_{\mathbf{B}}$), and then we take $N_{\mathbf{B}}$ of them which have greater f_{σ_R} (note that we are taking advantage that the actual number of concentration subspaces $N_p = \binom{n}{k}$ is large). Using this method, if some of the detected subspaces are false (because of getting trapped in local maxima), they will be ignored, too.

(1) Remove samples of the mixture matrix $\mathbf{X}$ which are near origin. In these samples, all of the sources are probably inactive. Then normalize every column of $\mathbf{X}$ (normalization simplifies the distance measurement).
(2) Assume an appropriate value of α and estimate $N_{\mathbf{B}}$ in (7). Choose a suitable decreasing sequence of $[\sigma_1 \dots \sigma_R]$.
(3) For $j = 1 \dots L_{\mathbf{B}}$
   (a) Choose a random starting subspace (an orthonormal m by k matrix $\mathbf{B}_j$).
   (b) Set $i = 1$.
   (c) Start with $\mathbf{B}_j$ and maximize the function f_{σ_i} using the steepest ascent method with gradients presented in Appendix C.[6] Orthonormalize $\mathbf{B}_j$ after each iteration. Update $\mathbf{B}_j$ to the argument that maximizes this function.
   (d) If $i < R$ (where R is the number of elements of the sequence$[\sigma_1 \dots \sigma_R]$), increment i and go back to (c).
(4) Omit the repeated subspaces.[7]
(5) Choose $N_{\mathbf{B}}$ of the obtained subspaces that have the largest value of the function f_{σ_R}.

At the end of algorithm, it would be still possible that some of the detected subspaces are incorrect, and the next part of the algorithm (estimating $\mathbf{A}$ from these $\mathbf{B}$'s) should be insensitive to these errors. However, it should be noted that even these incorrectly detected subspaces still contain some information about the mixing vectors, because they are usually close to at least some of the mixing vectors. This fact, in addition to the fact that accurate estimation of all the subspaces is not a necessity for the second part of the algorithm, gives us a great degree of freedom in the first part. This means that the error in the first part of the algorithm may be compensated in the second part to some extent. This is one of the most essential advantages of the proposed algorithm.

4.2. Algorithm for identifying mixing vectors

Similar to the previous subsection, in order to improve the performance of the algorithm, instead of estimating n mixing vectors, $L_{\mathbf{A}} \gg n$ vectors are estimated, after which the repeated vectors are omitted and actual mixing vectors are extracted via error detection process. This approach is especially advantageous for the cases where the exact number of sources is not known in advance.

(1) Choose a suitable decreasing sequence of $[\sigma_1 \dots \sigma_R]$.
(2) For $j = 1 \dots L_{\mathbf{A}}$
   (a) Choose a random normalized m by 1 vector $\mathbf{v}_j$.
   (b) Set $i = 1$.
   (c) Start with $\mathbf{v}_j$ and maximize the function h_{σ_i} using the steepest ascent method with gradients presented in Appendix C.[8] Normalize $\mathbf{v}_j$ after each iteration. Update $\mathbf{v}_j$ to the argument that maximizes this function.
   (d) If $i < R$, increment i and go back to (c).
(3) Error detection process: Omit the vectors that are near to less than q of the estimated subspaces.[9]
(4) Omit the repeated vectors.[10]

Note that the parameter n (number of sources) is not directly used in the above algorithm (the only usage of n is in the estimation of $N_{\mathbf{B}}$, that is, the number of concentration subspaces we detect in the first part). If n is not known, the above algorithm can be equally applied. In this case, $N_{\mathbf{B}}$ can be selected, for example, based on a priori known upper bound for n. As will be seen in the experimental results, obtaining more than n (actual number of) mixing vectors is rare, but obtaining less than n vectors happens more frequently, specially where the actual mixing vectors are close enough.

5. Experimental results

In this section, five simulations are presented to justify the algorithm.[11] In all of these simulations, sparse sources are generated independently and identically distributed

[6]Note that the maximization method is not an essential part of the algorithm. For example, we have successfully used the Nelder–Mead simplex method (the *fminsearch* function of MATLAB).

[7]A repetition is detected if the distance of the currently obtained subspace from one of the previously obtained subspaces is less than a certain amount. This distance is explained in Appendix A.

[8]Similar to the first part of the algorithm, maximization method is not an essential part.

[9]A vector is considered near to a subspace if their distance (presented in Appendix A) is less than a certain amount.

[10]A repetition is detected if the angle between the vector and another one is less than a certain amount.

[11]The MATLAB codes of the proposed method is available via email request to authors.

(i.i.d) by the sum of Gaussians model [2]:

$$s_i \sim p\mathcal{N}(0,\sigma_{\text{on}}) + (1-p)\mathcal{N}(0,\sigma_{\text{off}}), \tag{8}$$

where p is the probability of activity of the sources (and hence $k \approx np$). σ_{on} and σ_{off} are the standard deviations of the sources in active and inactive modes, respectively. In order to have sparse sources, the conditions $\sigma_{\text{on}} \gg \sigma_{\text{off}}$ and $p \ll 1$ should be applied. σ_{off} is to model a white noise. In all simulations, the values $\sigma_{\text{on}} = 1$ and $\sigma_{\text{off}} = 0.01$ have been used.

Generally, two types of error may occur in this method. We say *type one* error has occurred if one of the mixing vectors is not obtained by the method. This error occurs if that mixing vector is not close to at least q of the estimated subspaces. Type one error may be generated because of two reasons. First reason is that α (the acceptable probability of not having a mixing vector in the detected subspaces) is usually chosen greater than zero. Therefore, there is always a chance that a mixing vector is not close to q of the estimated subspaces. The second reason is error in subspace estimation process.

We say an error of *type two* has occurred if one of the obtained vectors is inaccurate, i.e. not close to any of actual mixing vectors. As will be seen, type two error has been rarely observed in our simulations.

In all of the simulations mixing matrices are generated randomly and each column of them is normalized. In most simulations, $L_{\mathbf{A}}$ has been set equal to $20n$ (unless specified otherwise).

In the use of the algorithm of Section 4.1, two subspaces are detected identical if their distance (presented in Appendix A) is less than 0.1, and in the use of the algorithm of Section 4.2, a vector is considered to lie in a subspace if its distance (presented in Appendix A) from that subspace is less than 0.03. In this section two vectors are called identical if their angle is less than 6°. This criterion forces any two mixing vectors to have a minimum angle of 6°. In all simulations, the mixing matrices which did not obey this criteria were omitted.

All simulations were performed in MATLAB 7 environment using an Intel Pentium 4, 2.4 GHz processor with 1 GB RAM under Microsoft Windows XP operating system.

Experiment 1: performance

In this experiment, the efficiency of our algorithm is demonstrated. One hundred simulations are performed for the case $n = 12$, $m = 6$, $k = 2$ ($p = 0.167$) and $T = 3900$. Parameters were chosen as follows: $q = 4$, $\alpha = 0.01$ (resulting in $N_{\mathbf{B}} = 58$), $L_{\mathbf{B}} = 10N_{\mathbf{B}} = 580$, $\sigma_{\mathbf{B}} = [0.15, 0.075, 0.037, 0.018]$ and $\sigma_{\mathbf{A}} = [0.1, 0.05, 0.025, 0.0125]$.

Fig. 7a shows the number of vectors obtained by the algorithm in all simulations. Note that in none of these simulations more than 12 vectors are obtained. However, in 11 of them less than 12 vectors are estimated which indicate the occurrence of type one error.

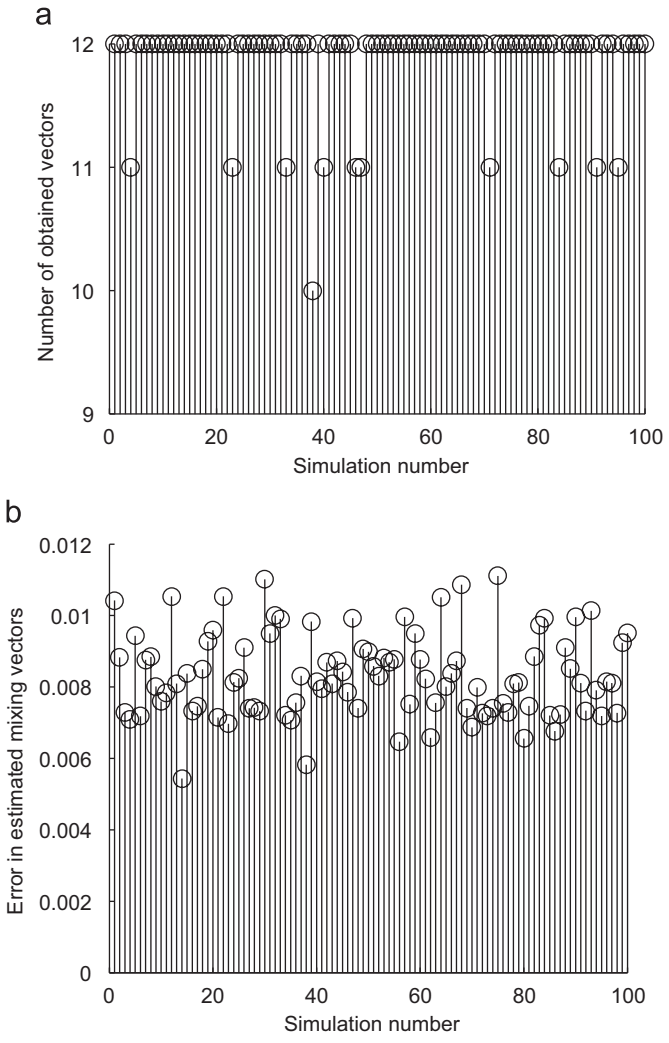

Fig. 7. Efficiency of the overall algorithm for all simulations in the case $n = 12$, $m = 6$, $k = 2$ and $T = 3900$ for 100 different simulations. In (a) number of obtained vectors in each simulation is shown. In some cases fewer than 12 vectors are obtained and type one error has occurred. In (b) error in obtaining the mixing vectors is measured using (9). Negligible errors show type two error has never occurred.

In all cases, the obtained vectors are compared with the mixing vectors. For the cases in which 12 vectors were obtained, the criterion

$$\mathscr{E} = \min_{\mathbf{P}\in\mathscr{P}} \|\mathbf{A} - \hat{\mathbf{A}}\mathbf{P}\|_2 \tag{9}$$

is used, where $\mathscr{P}$ is the set of all permutation matrices (this is the same criterion used in [22]). This estimation error is shown in Fig. 7b for all simulations.[12] The process took less than 3 minutes in average for estimating each mixing matrix. The fact that these errors are negligible implies that type two error has not occurred. The strict error detection

[12] In the cases where $n_0 < n$ vectors are obtained, the formula

$$\mathscr{E} = \min_{\mathbf{P}\in\mathscr{X}} \|\mathbf{A}\mathbf{P} - \hat{\mathbf{A}}\|_2 \tag{10}$$

is used where $\mathscr{X}$ is the set of all n_0 by n full-rank matrices in which all elements are zero except an element equal to 1 in each row.

process in Section 4.2, usually prevents occurrence of any type two error.

In the cases where less than 12 vectors were obtained, type one error has occurred. These cases can also be avoided by increasing $L_{\mathbf{B}}$, or the number of samples (if possible) or changing the sequence of $\sigma_{\mathbf{B}}$ or $\sigma_{\mathbf{A}}$.

Experiment 2: middle scale problem

To show that the method is capable of solving medium scale problems, two simulations are performed. In the first simulation, the parameters are set to $n = 15$, $m = 7$, $k = 3$, $T = 9000$, $q = 15$, $N_{\mathbf{B}} = 180$, $L_{\mathbf{B}} = 3N_{\mathbf{B}} = 540$, $\sigma_{\mathbf{B}} = [0.15, 0.075, 0.037, 0.018]$ and $\sigma_{\mathbf{A}} = [0.1, 0.05, 0.025, 0.012]$, whereas in the second experiment, they are set to $n = 30$, $m = 15$, $k = 2$, $T = 8500$, $q = 4$, $N_{\mathbf{B}} = 250$, $L_{\mathbf{B}} = 3N_{\mathbf{B}} = 750$, $\sigma_{\mathbf{B}} = [0.3, 0.15, 0.075, 0.037, 0.018]$ and $\sigma_{\mathbf{A}} = [0.1, 0.05, 0.025, 0.0125]$. The process took less than 40 minutes for the first case and less than two hours for the second case.

To measure the accuracy of the estimation, the angle between each estimated vector and its corresponding actual mixing vector (i.e. inverse cosine of their dot product) is shown in Figs. 8a and b. As it is seen in these figures, in both cases the algorithm has successfully detected all the mixing vectors. For example, in the second simulation, elements of the most erroneous estimated mixing vector and their actual values are shown in Fig. 9.

As it is seen in this experiment, neither type one nor type two error has occurred. All obtained vectors were close to one of the mixing vectors with high accuracy. Even larger scale problems can be solved by providing the algorithm with more time.

Experiment 3: unknown number of sources

As mentioned previously, the proposed method does not directly depend on the exact number of sources. Therefore, even in cases where n is not known, estimating $\mathbf{A}$ (and hence n) is still possible.

In these cases, occurrence of type one error is not detectable. Therefore, to choose a suitable $N_{\mathbf{B}}$ in this case, an upper bound for n is necessary. Using this upper bound and (7), a suitable $N_{\mathbf{B}}$ can be chosen. In order to prevent occurrence of type one error, $L_{\mathbf{B}}$ should be chosen several times larger than $N_{\mathbf{B}}$.

To demonstrate the algorithm in the case in which n is unknown, the parameters are set as follows. n is presumed less than 12 (upper bound for n is equal to 12), $\alpha = 0.01$ resulting in $N_{\mathbf{B}} = 58$, $q = 4$, $\sigma_{\mathbf{A}} = [0.1, 0.05, 0.025, 0.0125]$, $\sigma_{\mathbf{B}} = [0.2, 0.1, 0.05, 0.025, 0.0125]$ and $L_{\mathbf{B}} = 10N_{\mathbf{B}} = 580$. Seven different mixtures were given to the algorithm as input, all having $m = 6$, $k = 2$ and $T = 1300$. The mixtures were produced by mixing 6–12 number of sources, respectively.

The algorithm was able to correctly identify the number of sources in all of the cases. The mixing matrix was also

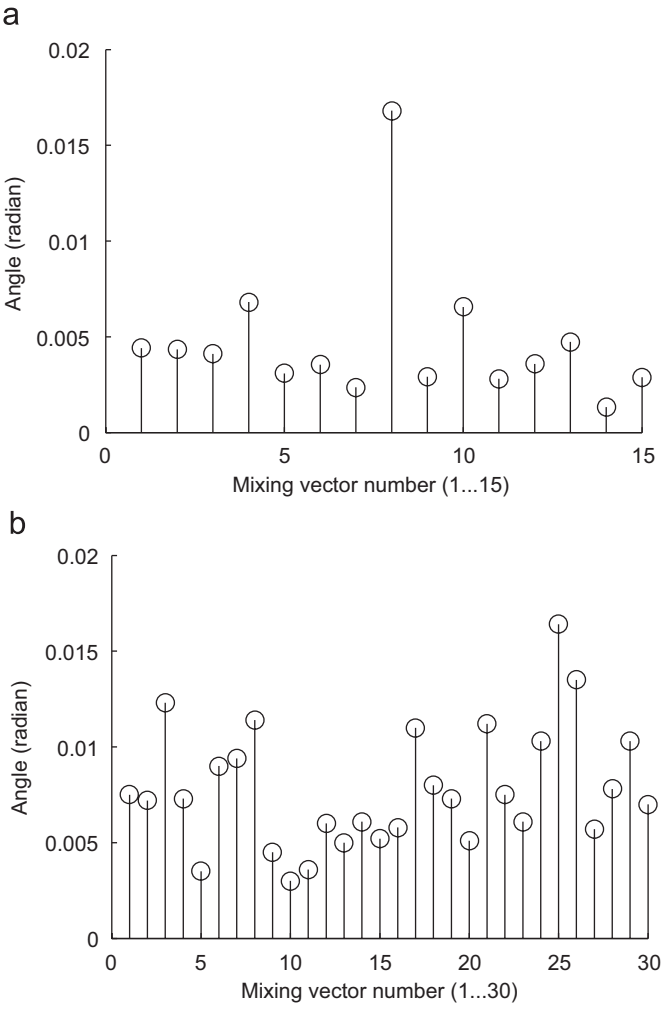

Fig. 8. The angle (in radian) between the mixing vectors and their corresponding estimation in middle scale problems. In (a) $n = 15$, $m = 7$ and $k = 3$. In (b) $n = 30$, $m = 15$ and $k = 2$.

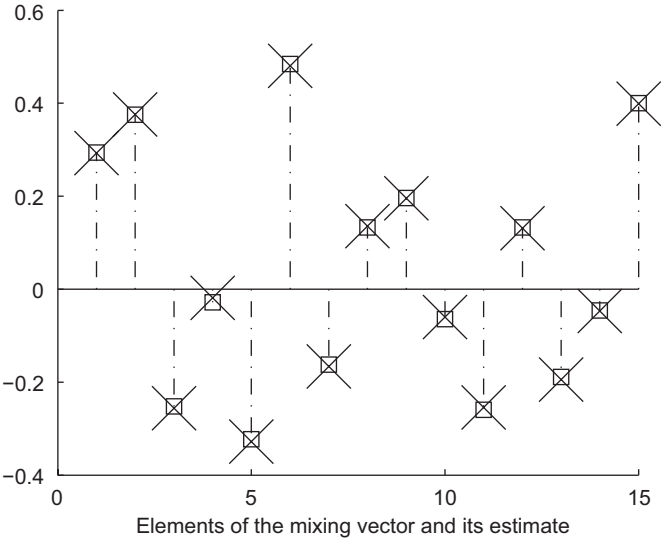

Fig. 9. Comparison between the most erroneous obtained vector and its corresponding mixing vector, for the second case ($n = 30$, $m = 15$ and $k = 2$) in Experiment 2. The crosses show the values of elements of the obtained vector and the squares show their actual values.

estimated accurately. The estimation errors, as defined in (9), are shown in the following table:

n	Estimation error
6	0.0180
7	0.0142
8	0.0184
9	0.0107
10	0.0101
11	0.0086
12	0.005

These small errors imply the correct identification of the mixing matrices. This experiment shows that the method is able to estimate the mixing matrix without knowing the exact number of sources.

Experiment 4: effect of the number of data samples on the performance

In this experiment the effect of the number of data samples, T, on the performance of the algorithms is analyzed. Performance can be measured using two criteria, the number of simulations in which the mixing matrix is successfully estimated and the error in these estimations.

In order to estimate the concentration subspaces, number of data samples should be proportional to N_p.[13] In this experiment, 15 different values of T between $4N_p$ to $60N_p$ were examined.

For each value of T, 100 simulations were performed in the case $n=12$, $m=6$ and $k=2$. Parameters were chosen as follows: $q=4$, $N_{\mathbf{B}}=58$, $L_{\mathbf{B}}=10N_{\mathbf{B}}=580$, $\sigma_{\mathbf{A}}=[0.1, 0.05, 0.025, 0.0125]$, $\sigma_{\mathbf{B}}=[0.15, 0.075, 0.037, 0.018]$. The number of successfully estimated mixing matrices for each value of T is shown in Fig. 10a. Average error of the successfully estimated mixing matrices is also shown in Fig. 10b.

For small values of T, type two error occurred. However, the percentage of type two error occurrence was less than 5%. For large values of T, type two error never occurred.

Experiment 5: comparison with existing methods

In this experiment the algorithm is applied to the same case presented in [22], as one of the most recent algorithms, where $n=5$, $m=3$ and $k=2$. Two kinds of sources were employed in this experiment. The first kind was the same as sources used in [22], where at each instant, exactly two sources were active and the inactive sources were forced to be zero. However, phase shifts were added to the sources to make them uncorrelated. The second kind of sources were

[13]The reason is that, the samples are distributed among all the concentration subspaces randomly. Considering the fact that strong peaks of the function f require the dense concentration of data samples around them, T should be at least several times greater than N_p.

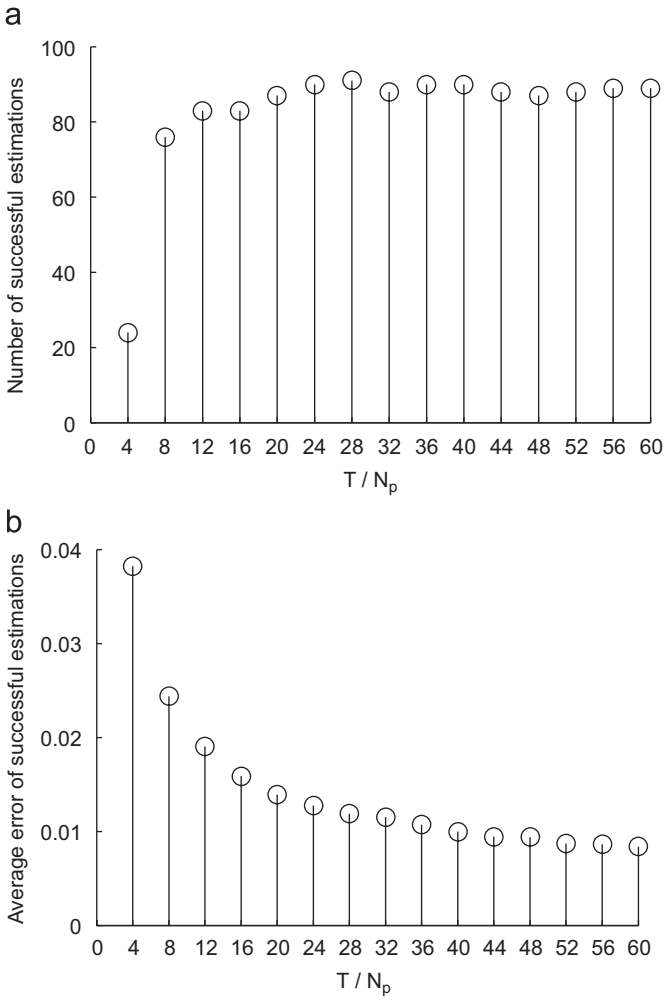

Fig. 10. The effect of number of data samples on the performance. In (a), the number of successfully estimated mixing metrics in 100 simulation versus T is shown. In (b), average error using (9) is shown for the successful cases. Note that more than 80% of the simulations were successfully performed for the case $T/N_p \geqslant 12$.

generated according to the general model of sum of Gaussians (8). The parameters were chosen as follows: $q=4$, $L_{\mathbf{B}}=400$, $\sigma_{\mathbf{A}}=[0.1, 0.05, 0.025, 0.0125]$, $\sigma_{\mathbf{B}}=[0.05, 0.025, 0.0125]$ and $N_{\mathbf{B}}=N_p=10$. The number of data samples for the first kind and the second kind of sources were chosen equal to 1000 and 400, respectively (as opposed to 2000 data samples used in [22]). The algorithm was capable to find all of the mixing vectors and the mixing matrix estimation error, as defined in (9), was 0.0056 and 0.0058, for the first kind and the second kind of the sources, respectively; as opposed to 0.2018 in [22]. The process took about 160 s for the first kind of the sources and 25 s for the second kind. The mixing matrix and its estimation using the second kind of sources are shown as follows:

$$\mathbf{A}=\begin{bmatrix} 0.5525 & 0.3919 & 0.5707 & 0.3934 & 0.6904 \\ 0.6863 & -0.6066 & 0.5166 & 0.8634 & -0.6007 \\ 0.4730 & 0.6917 & -0.6383 & -0.3158 & 0.4032 \end{bmatrix},$$

$$\hat{\mathbf{A}} = \begin{bmatrix} 0.5536 & 0.6930 & -0.3908 & 0.3892 & -0.5712 \\ 0.6829 & -0.5980 & -0.8644 & -0.6083 & -0.5154 \\ 0.4765 & 0.4027 & 0.3165 & 0.6917 & 0.6388 \end{bmatrix}.$$

As observed from this experiment, the proposed method is capable of achieving a better estimation of the mixing matrix using fewer number of data samples.

6. Comments on choosing the parameters

As observed in Section 4, the proposed algorithm possesses parameters such as decreasing sequences of $\sigma_{\mathbf{B}}$, $\sigma_{\mathbf{A}}$, q, $L_{\mathbf{B}}$, $L_{\mathbf{A}}$ and α which should be suitably chosen by the user. The choice of suitable values of the parameters and their effects are discussed in the following.

Suitable choice of starting elements of $\sigma_{\mathbf{B}}$ and $\sigma_{\mathbf{A}}$ are essential factors in the performance. If the starting element of $\sigma_{\mathbf{B}}$ is chosen too large, then peaks of the function f get mixed. This can be detected by the user through the observation of a large number of repeated subspaces in the output of the first part of the algorithm (in these cases, the number of different estimated subspaces is much less than $N_{\mathbf{B}}$). On the other hand, if the starting element is chosen too small, it makes the function f having a lot of local maxima. This results in a lot of incorrect estimated subspaces (in these cases, the number of repeated subspaces is very small).

If the starting element of $\sigma_{\mathbf{A}}$ is chosen too large, then peaks of the function h get mixed. In this case the number of repeated vectors in the output of the second part is very large. On the other hand, if the starting element is chosen too small, it makes the function h having a lot of local maxima. This results in a lot of incorrect estimated vectors (in these cases the number of repeated vectors is very small).

As observed in the experimental results, the algorithm usually accomplishes properly with a starting σ between 0.05 and 0.5 (in the cases where columns of the matrix $\mathbf{X}$ are normalized in advance). Other elements of the sequence of $\sigma_{\mathbf{B}}$ and $\sigma_{\mathbf{A}}$ are not that much essential in the performance. As observed in the experimental results, the decreasing sequences of $\sigma_{\mathbf{B}}$ and $\sigma_{\mathbf{A}}$ can usually be chosen a geometrical sequences with a scale factor of 0.5. The suitable stopping element of sequences is related to the power of the additive noise. However, a sufficiently small value of stopping σ always works. In the experimental results, (having $\sigma_{\text{off}} = 0.01$), a value of stopping σ close to 0.01 is always adequate.

As mentioned previously, $L_{\mathbf{B}}$ and $L_{\mathbf{A}}$ should be chosen several times larger than $N_{\mathbf{B}}$ and n (or its upper bound), respectively. In all experimental results, $L_{\mathbf{B}} = 10N_{\mathbf{B}}$ and $L_{\mathbf{A}} = 20n$ usually works. Choosing much larger values of $L_{\mathbf{B}}$ and $L_{\mathbf{A}}$, increases computation cost without improving the performance. However, sufficiently large values of $L_{\mathbf{B}}$ and $L_{\mathbf{A}}$, decreases the sensitivity of the algorithm to some parameters such as $\sigma_{\mathbf{B}}$, $\sigma_{\mathbf{A}}$ and T.

Another parameter in the algorithm of Section 4.2 is q. The optimum theoretical value of q has not been obtained. If q is chosen too small, many incorrect vectors can pass the error detection process in the second part which results in type two error. On the other hand, if q is chosen too large, many concentration subspaces must be estimated (to prevent type one error) which makes the algorithm more time-consuming. In our experimental results, we observed that $q = 4$ and $q = n$ are adequate for the cases $k = 2$ and $k = 3$, respectively.

7. Discussion and conclusion

Most existing algorithms assume single dominant case at each instant for estimating the mixing matrix in SCA. Moreover, the number of sources is assumed to be known in most of them. In this paper, we presented a method which omits these two restrictions. On the contrary, in our simulations we assumed that the averaged number of active sources, k, is known in advance.[14]

In Section 2, we observed the fact that graph of the function f is similar to the histogram of data concentration when σ is sufficiently small. In fact, as stated in Section 2, for sufficiently small σ's, the value of f in (3) is equal to the number of data points in the subspace specified by $\mathbf{B}$. Consequently, in Fig. 2, f is the same as the histogram (note the similarity of Figs. 2c and 3). Therefore, (3) can be seen as a generalization of histogram: firstly, it is smoother (with the degree of smoothness under our control. To find its maximum for small σ's, we decrease the smoothness gradually to escape from local maxima) and secondly, it is applicable to higher dimensions.

Some other related works in solving the SCA problem are [1,22,21]. In [1], K-SVD is introduced which is essentially a generalization of the K-means that fulfils the requirements of solving the multiple dominant case. This method works for the middle scale problem, is fast, achieves acceptable performance, requires relatively few number of data samples and depends on few parameters. However, the number of sources, n, is required to be known a prior. Moreover, in many cases, some of the mixing vectors may be detected incorrectly (both type one and type two error). The method presented in [22] is based on estimation of the concentration subspaces. It solves the multiple dominant case. However, it cannot work in medium scale. In [21], a generalization of the histogram method is introduced that takes advantage of information obtained from more than two mixtures. However, single dominance is still an essential assumption. Our method is also a generalization of the histogram method. However, instead of linear concentration assumption like in [21], we have planar concentration assumption like in [22]. Another difference is in the way of escaping from getting trapped in local maxima which is a suffering in all histogram-like methods.

[14] A method for estimating k has been proposed in [18].

Unfortunately, like many SCA methods, our method suffers from exponential growth in computation cost. The reason is that in order to estimate the concentration subspaces, the number of data samples should be proportional to N_p. Therefore, it is burdensome to solve the problem in the large scales. The large scale case still remains an open problem.

As observed in the experimental results, correct choice of the parameters prevents any occurrence of type two error. This means that any vector estimated by the algorithm would be one of the mixing vectors with a high probability: the algorithm rarely gives an incorrect vector and is trustable. However, some mixing vectors might not be found in the first try, either because of the badly chosen parameters, or because some of the actual mixing vectors are close to each other, or because of lack of sufficient data samples.

As mentioned before, the presented method uses a decreasing sequence of σ for estimating the concentration subspaces. The first and largest σ in the sequence is an essential factor in the quality. A suitable starting σ depends on many factors such as n, m and k. If chosen too large, it may cause mixed peaks and if too small many local maxima exist. The decreasing sequence of σ's are also important.

The performance of our method depends on several factors, such as the condition number of mixing matrix, the number of samples, the number of observations, the average number of active sources, the sequences of $\sigma_{\mathbf{A}}$ and $\sigma_{\mathbf{B}}$. A proper estimation of these sequences is an essential factor in the performance. An optimum choice of the parameters is still an open problem.

Appendix A

In the algorithm it is required to calculate the distance between a subspace and a vector or two subspaces. Let $\mathbf{B}=[\mathbf{b}_1 \dots \mathbf{b}_k]$ be the matrix representation of a k-dimensional subspace of the m-dimensional space, that is, $\{\mathbf{b}_1,\dots,\mathbf{b}_k\}$ is an orthonormal basis for this subspace. Let $\mathbf{v}$ be a unit-norm m-dimensional vector. Then, as a measure of the distance between subspace $\mathbf{B}$ and vector $\mathbf{v}$, we have used:

$$d(\mathbf{v},\mathbf{B})=\sqrt{1-[(\mathbf{v}\cdot\mathbf{b}_1)^2+\cdots+(\mathbf{v}\cdot\mathbf{b}_k)^2]}, \tag{11}$$

where $\mathbf{v}\cdot\mathbf{b}_i$ represents the dot product of $\mathbf{b}_i$ and $\mathbf{v}$.

Now let $\mathbf{B}=[\mathbf{b}_1 \dots \mathbf{b}_k]$ and $\hat{\mathbf{B}}=[\hat{\mathbf{b}}_1 \dots \hat{\mathbf{b}}_k]$ be two k-dimensional subspaces of m-dimensional space represented in orthonormal form. Then, as a measure of the distance between these two subspaces we have used:

$$d(\mathbf{B},\hat{\mathbf{B}})=\sqrt{d^2(\mathbf{b}_1,\hat{\mathbf{B}})+\cdots+d^2(\mathbf{b}_k,\hat{\mathbf{B}})},$$

where $d(\mathbf{b}_i,\hat{\mathbf{B}})$ is the distance between vector and subspace, stated above.

Appendix B

In (6) it is necessary to calculate an expression of the form

$$\sum_{1\leqslant i_1<\cdots<i_q\leqslant N} a_{i_1}\dots a_{i_q},$$

where $N=N_p$ and $a_j=u_\sigma(\mathbf{v},\mathbf{B}_j), 1\leqslant j\leqslant N_p$. If directly computed, it requires a cost computation of order $\binom{N}{q}$. However, by implementing a recursive algorithm, cost computation can be decreased to the order of Nq.

By defining:

$$sum_q(a_1\dots a_N)=\sum_{1\leqslant i_1<\cdots<i_q\leqslant N} a_{i_1}\dots a_{i_q},$$

we have

$$sum_q(a_1\dots a_N)=sum_q(a_1\dots a_{N-1})+a_N\cdot sum_{q-1}(a_1\dots a_{N-1}).$$

Using this formula the recursive algorithm can be designed.

Appendix C

As mentioned in Section 4, this method is based on the maximization of two functions, f_σ and h_σ. This appendix is dedicated to the development of the steepest ascent methods for their maximization.

To maximize the function $f_\sigma(\mathbf{B})$ using the steepest ascent method, its matrix gradient with respect to $\mathbf{B}=[\mathbf{b}_1\dots\mathbf{b}_k]$ can be presented by

$$\frac{\partial f_\sigma}{\partial \mathbf{B}}=\left(\frac{\partial f_\sigma}{\partial \mathbf{b}_1}\cdots\frac{\partial f_\sigma}{\partial \mathbf{b}_k}\right),$$

where

$$\frac{\partial f_\sigma}{\partial \mathbf{b}_j}=\sum_{t=1}^{\mathrm{T}}\frac{\partial}{\partial \mathbf{b}_j}\{\exp(-d^2(\mathbf{x}_t,\mathbf{B})/2\sigma^2)\}$$
$$=-\frac{1}{2\sigma^2}\sum_{t=1}^{\mathrm{T}}\frac{\partial d^2(\mathbf{x}_t,\mathbf{B})}{\partial \mathbf{b}_j}\exp(-d^2(\mathbf{x}_t,\mathbf{B})/2\sigma^2) \tag{12}$$

for $1\leqslant j\leqslant k$.

The vector gradient of $d^2(\mathbf{x}_t,\mathbf{B})$ versus $\mathbf{b}_j$ can be calculated using (11):

$$\frac{\partial d^2(\mathbf{x}_t,\mathbf{B})}{\partial \mathbf{b}_j}=-2\mathbf{x}_t(\mathbf{x}_t\cdot\mathbf{b}_j)$$

and by utilizing this equation in (12) the final formula can be achieved

$$\frac{\partial f_\sigma}{\partial \mathbf{b}_j}=\frac{1}{\sigma^2}\sum_{t=1}^{\mathrm{T}}\mathbf{x}_t(\mathbf{x}_t\cdot\mathbf{b}_j)\exp(-d^2(\mathbf{x}_t,\mathbf{B})/2\sigma^2),\quad 1\leqslant j\leqslant k. \tag{13}$$

This equation can be utilized to compute gradient in each iteration of the steepest ascent maximization. Each iteration of this algorithm is composed of:

- Set $\mathbf{b}_j\leftarrow\mathbf{b}_j+\mu(\partial f_\sigma/\partial \mathbf{b}_j)/T$ for $1\leqslant j\leqslant k$ using (13).

- Orthonormalize $\mathbf{B}$; Set $\mathbf{B} \leftarrow \mathbf{B}(\mathbf{B}^{\mathrm{T}}\mathbf{B})^{-1/2}$ (refer to [12, Section 6.5]).

In our simulation, we have chosen the step size of the algorithm (μ) proportional to σ^2, to have smaller step sizes for more complicated functions (which is the case for smaller σ's).

Note that orthonormality of the matrix representation is assumed in (11) and it is necessary to orthonormalize $\mathbf{B}$ in each iteration.

We have also used the steepest ascent method also for maximizing the function $h_\sigma(\mathbf{v})$ versus $\mathbf{v}$. The details are as follows.

Using (6), the function h can be represented by

$$h_\sigma(\mathbf{v}) = \sum_{1 \leqslant i_1 < \cdots < i_q \leqslant N_p} \left(\prod_{j=1}^{q} \exp(-d^2(\mathbf{v}, \mathbf{B}_{i_j})/2\sigma^2) \right)$$

so:

$$\frac{\partial h_\sigma}{\partial \mathbf{v}} = \sum_{1 \leqslant i_1 < \cdots < i_q \leqslant N_p} \left(\sum_{l=1}^{q} \left(\frac{\partial}{\partial \mathbf{v}} \{\exp(-d^2(\mathbf{v}, \mathbf{B}_{i_l})/2\sigma^2)\} \times \prod_{\substack{j=1 \\ j \neq 1}}^{q} \exp(-d^2(\mathbf{v}, \mathbf{B}_{i_j})/2\sigma^2) \right) \right).$$

By rearranging the sums:

$$\frac{\partial h_\sigma}{\partial \mathbf{v}} = \sum_{l=1}^{N} \frac{\partial}{\partial \mathbf{v}} \{\exp(-d^2(\mathbf{v}, \mathbf{B}_{i_l})/2\sigma^2)\} \times \left(\sum_{\substack{1 \leqslant i_1 < \cdots < i_{q-1} \leqslant N_p \\ l \notin \{i_1 \ldots i_{q-1}\}}} \prod_{j=1}^{q-1} \exp(-d^2(\mathbf{v}, \mathbf{B}_{i_j})/2\sigma^2) \right) \quad (14)$$

and using (11)

$$\frac{\partial}{\partial \mathbf{v}} \{\exp(-d^2(\mathbf{v}, \mathbf{B})/2\sigma^2)\} = \frac{1}{\sigma^2} \left(\sum_{i=1}^{k} \mathbf{b}_i (\mathbf{b}_i \cdot \mathbf{v}) \right) \exp(-d^2(\mathbf{v}, \mathbf{B})/2\sigma^2). \quad (15)$$

To calculate (14), the method introduced in Appendix B can be applied. In fact:

$$\sum_{\substack{1 \leqslant i_1 < \cdots < i_{q-1} \leqslant N_p \\ l \notin \{i_1 \ldots i_{q-1}\}}} \prod_{j=1}^{q-1} \exp(-d^2(\mathbf{v}, \mathbf{B}_{i_j})/2\sigma^2) = sum_{q-1}\{\exp(-d^2(\mathbf{v}, \mathbf{B}_i)/2\sigma^2) | 1 \leqslant i \leqslant N, i \neq l\}.$$

This simplification makes the steepest ascent method computationally possible for the function h.

Similar to the previous maximization, a suitable step μ proportional to σ^2 has been chosen and the following steps should be performed in each iteration.

- Set $\mathbf{v} \leftarrow \mathbf{v} + \mu(\partial h_\sigma/\partial \mathbf{v})/\|\partial h_\sigma/\partial \mathbf{v}\|_2$ using (14) and (15).
- Normalize $\mathbf{v}$ by setting $\mathbf{v} \leftarrow \mathbf{v}/\|\mathbf{v}\|_2$.

In the experimental results, in the first maximization μ is set to $10^4\sigma^2$ and in the second, μ is set to $100\sigma^2$. Another reason for choosing μ proportional to σ^2 is that, in (13) and (15) there is a factor of $1/\sigma^2$ in calculation of the gradient and by choosing μ proportional to σ^2 it would be compensated.

References

[1] M. Aharon, M. Elad, A. Bruckstein, The K-SVD: an algorithm for designing of overcomplete dictionaries for sparse representation, IEEE Trans. Signal Process. 54 (11) (2006) 4311–4322.

[2] A.A. Amini, M. Babaie-Zadeh, C. Jutten, A fast method for sparse component analysis based on iterative detection-projection, in: Proceedings of 26th International Workshop on Bayesian Inference and Maximum Entropy Methods in Science and Engineering (MaxEnt), 2006.

[3] M. Babaie-Zadeh, C. Jutten, Semi-blind approaches for source separation and independent component analysis, in: Proceedings of ESANN'06, 2006, pp. 301–312.

[4] A. Belouchrani, J.-F. Cardoso, Maximum likelihood source separation by the expectation-maximization technique, in: NOLTA 95, Las Vegas, USA, 1995, pp. 49–53.

[5] P. Bofill, M. Zibulevsky, Underdetermined blind source separation using sparse representations, Signal Processing 81 (2001) 2353–2362 ⟨citeseer.ist.psu.edu/bofill01underdetermined.html⟩.

[6] A. Cichocki, S.-i. Amari, Adaptive Blind Signal and Image Processing: Learning Algorithms and Applications, Wiley, New York, 2002.

[7] D.L. Donoho, For most large underdetermined systems of linear equations the minimal l^1-norm solution is also the sparsest solution, Technical Report ⟨http://www-stat.stanford.edu/~donoho/Reports/2004/⟩, 2004.

[8] D.L. Donoho, M. Elad, Optimally sparse representation from overcomplete dictionaries via l^1 norm minimization, Proc. Natl. Acad. Sci. 100 (5) (2003) 2197–2202 ⟨citeseer.ist.psu.edu/pati93orthogonal.html⟩.

[9] P.G. Georgiev, F. J. Theis, A. Cichocki, Blind source separation and sparse component analysis for over-complete mixtures, in: Proceedings of ICASSP'04, Montreal, Canada, 2004, pp. 493–496.

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

[11] R. Gribonval, S. Lesage, A survey of sparse component analysis for blind source separation: principles, perspectives, and new challenges, in: Proceedings of ESANN'06, 2006, pp. 323–330.

[12] A. Hyvarinen, J. Karhunen, E. Oja, Independent Component Analysis, Wiley, New York, 2001.

[13] T. Lee, M. Lewicki, M. Girolami, T. Sejnowski, Blind source separation of more sources than mixtures using overcomplete representations, IEEE Signal Process. Lett. 4 (4) (1999) 87–90.

[14] Y. Li, A. Cichocki, S. Amari, Sparse component analysis for blind source separation with less sensors than sources, in: ICA2003, 2003, pp. 89–94.

[15] Y. Li, A. Cichocki, S. Amari, Analysis of sparse representation and blind source separation, Neural Computation 16 (6) (2004) 1193–1234.

[16] K. Matsuoka, M. Ohya, M. Kawamoto, A neural net for blind separation of nonstationary signals, Neural Networks 8 (3) (1995) 411–419.
[17] L. Molgedey, H. Schuster, Separation of a mixture of independent signals using time delayed correlations, Phys. Rev. Lett. 72 (23) (1994) 3634–3637.
[18] N. Noorshams, M. Babaie-Zadeh, C. Jutten, Estimating the mixing matrix in Sparse Component Analysis (SCA) based on converting a multiple dominant to a single dominant problem, in: Proceedings of the Seventh International Conference on Independent Component Analysis and Signal Separation (ICA2007), Lecture Notes in Computer Science, vol. 4666, September 2007, London, pp. 397–405.
[19] A. Papoulis, Probability and Statistics, Prentice-Hall, Englewood Cliffs, NJ, 1990.
[20] L. Tong, V. Soon, R. Liu, Y. Huang, AMUSE: a new blind identification algorithm, in: Proceedings of ISCAS, New Orleans, USA, 1990, pp. 1784–1786.
[21] L. Vielva, Y. Pereiro, D. Erdogmus, J. Principe, Inversion techniques for underdetermined BSS in an arbitrary number of dimensions, in: Proceedings of the 6th International Conference on Independent Component Analysis and Signal Separation (ICA'03), Nara, Japan, 2003, pp. 131–136.
[22] Y. Washizawa, A. Cichocki, On-Line k-plane clustering learning algorithm for sparse component analysis, in: Proceedings of ICASSP'06, Toulouse, France, 2006, pp. 681–684.
[23] M. Zibulevsky, B.A. Pearlmutter, Blind source separation by sparse decomposition in a signal dictionary, Neural Computation 13 (4) (2001) 863–882 ⟨citeseer.ist.psu.edu/article/zibulevsky01blind.html⟩.
[24] M. Zibulevsky, B. Pearlmutter, P. Bofill, P. Kisilev, Blind source separation by sparse decomposition, in: Independent Component Analysis: Principles and Practice, Cambridge, 2001.

Farid Movahedi Naini was born in Mashad, Iran, in 1985. He received his B.Sc. degree in communication systems from Sharif University of Technology, Tehran, Iran, in 2007. He is currently a Master student in the Department of Computer and Communication Sciences at the Ecole Polytechnique Fédérale de Lausanne (EPFL). His research interests include communication systems and signal processing.

G. Hosein Mohimani was born in Bushehr, Iran, in 1985. He is currently a B.Sc. double major student of Electrical Engineering and Mathematics, at Sharif University of Technology. His research interests include signal processing, communication systems, information theory and coding.

Massoud Babaie-Zadeh received the B.S. degree in electrical engineering from Isfahan University of Technology, Isfahan, Iran, in 1994, and the M.S. degree in electrical engineering from Sharif University of Technology, Tehran, Iran, in 1996, and the Ph.D. degree in signal processing from Institute National Polytechnique of Grenoble (INPG), Grenoble, France, in 2002 (for which, he received the best Ph.D. thesis award of INPG).

Since 2003, he has been an assistant professor of the Department of Electrical Engineering at Sharif University of Technology, Tehran, Iran. His main research areas are statistical signal processing, blind source separation (BSS) and independent component analysis (ICA).

Christian Jutten received the Ph.D. degree in 1981 and the Docteur ès Sciences degree in 1987 from the Institut National Polytechnique of Grenoble (France). He taught as associate professor in Ecole Nationale Supérieure d'Electronique et de Radioélectricité of Grenoble from 1982 to 1989. He was visiting professor in Swiss Federal Polytechnic Institute in Lausanne in 1989, before to become full professor in Université Joseph Fourier of Grenoble, more precisely in Polytech'-Grenoble Institute. He is currently associate director of the images and signals laboratory (100 people). For 25 years, his research interests are blind source separation, independent component analysis and learning in neural networks, including theoretical aspects (separability, source separation in nonlinear mixtures), applications in signal processing (biomedical, seismic, speech) and data analysis. He is author or co-author of more than 40 papers in international journals, 16 invited papers and 100 communications in international conferences. He has been associate editor of IEEE Transactions on Circuits and Systems (1994–95), and co-organizer with Dr. J.-F. Cardoso and Prof. Ph. Loubaton of the 1st International Conference on Blind Signal Separation and Independent Component Analysis (Aussois, France, January 1999). He is currently member of a technical committee of IEEE Circuits and Systems Society on blind signal processing. He is a reviewer of main international journals (IEEE Transactions on Signal Processing, IEEE Signal Processing Letters, IEEE Transactions on Neural Networks, Signal Processing, Neural Computation, Neurocomputing, etc.) and conferences in signal processing and neural networks (ICASSP, ISCASS, EUSIPCO, IJCNN, ICA, ESANN, IWANN, etc.).