Signal Processing 89 (2009) 1011–1022

Contents lists available at ScienceDirect

Signal Processing

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

K-hyperline clustering learning for sparse component analysis

Zhaoshui He [a,b,*], Andrzej Cichocki [a,d,e], Yuanqing Li [c], Shengli Xie [b], Saeid Sanei [f]

[a] Laboratory for Advanced Brain Signal Processing, RIKEN Brain Science Institute, Saitama 3510198, Japan
[b] School of Electronics and Information Engineering, South China University of Technology, Guangzhou 510641, China
[c] Automation Science and Engineering Institute, South China University of Technology, Guangzhou 510641, China
[d] System Research Institute, Polish Academy of Sciences (PAN), Warsaw 00-901, Poland
[e] Warsaw University of Technology, Warsaw 00-661, Poland
[f] Centre of Digital Signal Processing, Cardiff University, Cardiff CF24 3AA, Wales, UK

ARTICLE INFO

Article history:
Received 7 September 2007
Received in revised form
1 December 2008
Accepted 2 December 2008
Available online 24 December 2008

Keywords:
Sparse component analysis (SCA)
Disjoint orthogonality condition
K-means clustering
Blind source separation
Sparse representation
K-SVD

ABSTRACT

A two-stage clustering-then-ℓ_1-optimization approach has been often used for sparse component analysis (SCA). The first challenging task of this approach is to estimate the basis matrix by cluster analysis. In this paper, a robust K-hyperline clustering (K-HLC) algorithm is developed for this task. The novelty of our method is that it is not only able to implement hyperline clustering, but also is capable of detecting the number of hidden hyperlines (or sparse components). K-HLC seamlessly integrates "the hyperline clustering" and "hyperline number detection" in the same algorithm. In addition, three strategies are proposed to tackle this problem: (1) reject the outliers by overestimating the number of hyperlines; (2) escape from local minima by using a multilayer initialization and (3) suppress the noise by a multilayer K-HLC. By taking these strategies into account, the robust K-HLC procedure can be briefly described as follows: first, we overestimate the number of hyperlines; then, a confidence index is given to evaluate the significance of each hyperline. Subsequently, we determine the number of hyperlines by checking the gap in the sorted confidence indices. Moreover, we select those hyperlines corresponding to large confidence indices with high rank priority and remove spurious ones with small confidence indices. The high performance of our clustering scheme is illustrated by extensive numerical experiments including some challenging benchmarks, e.g., very ill-conditioned basis matrix (Hilbert matrix), or the observations with strong outliers.

1. Introduction

Sparse component analysis (SCA), also known as sparse coding or sparse representations (SR), can be modeled as

$$\boldsymbol{x}(t)=\boldsymbol{A}\boldsymbol{s}(t)+\boldsymbol{e}(t) \quad \text{or} \quad \boldsymbol{X}=\boldsymbol{A}\boldsymbol{S}+\boldsymbol{E}, \tag{1}$$

where $\boldsymbol{X}=[\boldsymbol{x}(1),\ldots,\boldsymbol{x}(T)]\in\mathbb{R}^{m\times T}(T\gg m)$ is an observable data matrix, $\boldsymbol{A}=[\boldsymbol{a}_1,\ldots,\boldsymbol{a}_n]\in\mathbb{R}^{m\times n}$ is an unknown full row rank basis matrix, which contains n basis vectors $\boldsymbol{a}_1,\ldots,\boldsymbol{a}_n$, and $\boldsymbol{S}\in\mathbb{R}^{n\times T}$ is an unknown matrix which represents n sparse sources or hidden sparse components. $\boldsymbol{E}\in\mathbb{R}^{m\times T}$ is the noise matrix, where $\boldsymbol{e}(t)=(e_1(t),\ldots,e_m(t))^{\mathrm{T}}$, $t=1,\ldots,T$, T is the number of samples, m is the number of observations and n is the number of sources. Note that n is generally unknown. When $m<n$, the columns of $\boldsymbol{A}$ form a set of overcomplete bases in $\mathbb{R}^m$ so that the linear model (1) is underdetermined. The main objective of SCA is to estimate the basis matrix $\boldsymbol{A}$ and the sources $\boldsymbol{S}$ such that $\boldsymbol{S}$ is as sparse as possible or has specified sparsity profile. Without loss of generality, assume that the columns of $\boldsymbol{A}$ are normalized [1–3], i.e., $\|\boldsymbol{a}_i\|_2=1,\ i=1,\ldots,n$.

* Corresponding author at: Laboratory for Advanced Brain Signal Processing, RIKEN Brain Science Institute, Saitama 3510198, Japan. Tel.: +81 48 467 9665; fax: +81 48 467 9694.
E-mail address: he_shui@tom.com (Z. He).

0165-1684/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.sigpro.2008.12.005

SCA has already found many applications such as electromagnetic and biomagnetic imaging, feature extraction, filtering, wavelet denoising, time–frequency representation, image processing, neural and speech coding, spectral estimation, estimation of direction of arrival (DOA), vector quantization, and fault diagnosis [4–9]. Especially, SCA can be applied in the underdetermined blind source separation (BSS), where the sparsity of sources is used as the additional information to achieve BSS in the case that the number of observations is less than the number of sources. A typical example was given by Bofill and Zibulevsky [10]: six flute signals were separated from only two observations. Regarding the applications of SCA in BSS, more results were reported in [1,5,10–16].

Notice that when the sources $\boldsymbol{s}(t)$ are sparse enough to approximately satisfy *disjoint orthogonality condition* [15,17], i.e., $s_i(t)\cdot s_j(t)=0$ (or $s_i(t)\cdot s_j(t)\approx 0$ in noisy case), then there will be at most only one significantly nonzero source in $s_1(t),\ldots,s_n(t)$. Without loss of generality, suppose that only $s_i(t)$ is significant at time instant t, i.e., $s_i(t)\neq 0$ and $s_j(t)=0$, $j\neq i$, $j=1,\ldots,n$. In the noise-free case, model (1) can be simplified as follows:

$$\boldsymbol{x}(t)=\boldsymbol{A}\boldsymbol{s}(t)=\boldsymbol{a}_1\cdot s_1(t)+\cdots+\boldsymbol{a}_n\cdot s_n(t)=\boldsymbol{a}_i\cdot s_i(t). \tag{2}$$

Eq. (2) can be re-written as

$$\frac{x_1(t)}{a_{i1}}=\frac{x_2(t)}{a_{i2}}=\cdots=\frac{x_m(t)}{a_{im}}=s_i(t), \tag{3}$$

where $\boldsymbol{a}_i=[a_{i1},\ldots,a_{im}]^{\mathrm{T}}$. Obviously, formula (3) is a linear equation, which means that all columns $\boldsymbol{a}_1,\ldots,\boldsymbol{a}_m$ of $\boldsymbol{A}$ are the hyperline directions in the scatter plot of the observed data $\boldsymbol{x}(t)$.

In this case, finding the basis matrix $\boldsymbol{A}$ can be cast into a hyperline clustering problem [5,10]. Hence a two-stage framework "cluster-then-ℓ_1-optimization" is developed [1,5,10,14]: in the first stage, the basis matrix $\boldsymbol{A}$ is first identified by hyperline clustering; in the second stage, the sparse components are estimated using other methods such as linear programming (LP) [1,5,18–20], FOCUSS algorithms [6,8,21,22] or shortest path decomposition [10,20]. So the hyperline clustering plays an important role in SCA.

There have been several clustering methods to find $\boldsymbol{A}$ in the two-stage SCA, e.g., K-means, fuzzy-C clustering [5,23], median-based clustering [24], linear geometric ICA-based method [25]. However, as mentioned in [14], these methods require the sources to be very sparse, so that they well satisfy the disjoint orthogonality condition in the analyzed transformed domain. By extending the "degenerate unmixing estimation technique" (DUET) and the method called "time–frequency ratio of mixtures (TIFROM)", Li et al. [14] proposed a new algorithm in which the precondition of sparseness can be considerably relaxed. The disadvantage of this method is that it has five free unknown parameters or thresholds which need to be determined in advance or found empirically. Generally, it is not easy to set these parameters. Recently, principle component analysis (PCA) was employed to identify the basis matrix in SCA [26–30]: first, the samples of the observed data are partitioned into several clusters; then for each cluster, the PCA is applied and the eigenvector corresponding to the largest eigenvalue is chosen as the estimate of a basis vector. When the samples of observed data are correctly partitioned, PCA can find the right estimation of the basis matrix $\boldsymbol{A}$. Furthermore, several confidence indices based on eigenanalysis were constructed to evaluate the validity of the clusters [27–29].

Although the methods mentioned above are promising, their performance may not be perfect due to the following factors in practice: noise, outliers, insufficient sparseness of the sources (e.g., only a small fraction of the samples well satisfy the disjoint orthogonality condition), ill-conditioned basis matrix $\boldsymbol{A}$, high dimensional basis matrix $\boldsymbol{A}$, etc. In addition, the performance of these methods is often limited because of the local minima. Moreover, another problem is that the dimension of basis matrix $\boldsymbol{A}$ (or the number of sources) is probably unknown. If the number of sources is underestimated, it will be impossible to obtain satisfactory results.

In order to solve such problems, the hyperline clustering is studied in this paper. A robust K-hyperline clustering (K-HLC) method is implemented by improving the K-SVD method for hyperline clustering [9]. It is easy to implement. The remainder of this paper is organized as follows: First, we discuss the basic K-HLC in Section 2. The robust K-HLC and its extensions are given in Section 3. Simulation examples are provided in Section 4. The conclusions and discussion are presented in Section 5.

2. K-hyperline clustering

Consider the K-HLC problem as: given a set of m-dimensional points $\{\boldsymbol{x}_t\}_{t=1}^{T}$ drawn from some unknown hyperlines, how to find these hyperlines and group these points into them? This problem can be divided into three sub-problems:

(1) Detect the number K of the hyperlines;
(2) Determine the K hyperlines $L(\boldsymbol{l}_k)$, $k=1,\ldots,K$, where $L(\boldsymbol{l}_k):=\{\boldsymbol{x}\,|\,\boldsymbol{x}\in\mathbb{R}^m, x_1/l_{k1}=x_2/l_{k2}=\cdots=x_m/l_{km}=c_k, c_k\in\mathbb{R}\}$, $\boldsymbol{l}_k=[l_{k1},\ldots,l_{km}]\in\mathbb{R}^m$ is the directional vector of hyperline $L(\boldsymbol{l}_k)$;
(3) Partition $\{\boldsymbol{x}_t\}_{t=1}^{T}$ into K different hyperlines $L(\boldsymbol{l}_k)$, $k=1,\ldots,K$.

2.1. Distance formula from a point to a hyperline

Given a point $P(p_1,\ldots,p_m)$ and a hyperline $L(\boldsymbol{l})$ in the m-dimensional space, of which the direction vector is $\boldsymbol{l}=(l_1,\ldots,l_m)^{\mathrm{T}}\in\mathbb{R}^m$, the distance $d(\boldsymbol{p},\boldsymbol{l})$ from P to $L(\boldsymbol{l})$ is

$$d^2(\boldsymbol{p},\boldsymbol{l})=\langle\boldsymbol{p},\boldsymbol{p}\rangle-\frac{\langle\boldsymbol{l},\boldsymbol{p}\rangle^2}{\langle\boldsymbol{l},\boldsymbol{l}\rangle}, \tag{4}$$

where $\langle\cdot,\cdot\rangle$ denotes inner product. Especially, $d^2(\boldsymbol{p},\boldsymbol{l})=\langle\boldsymbol{p},\boldsymbol{p}\rangle-\langle\boldsymbol{l},\boldsymbol{p}\rangle^2$ when $\langle\boldsymbol{l},\boldsymbol{l}\rangle=1$ and $d^2(\boldsymbol{p},\boldsymbol{l})=1-\langle\boldsymbol{l},\boldsymbol{p}\rangle^2$ when $\langle\boldsymbol{l},\boldsymbol{l}\rangle=1$ and $\langle\boldsymbol{p},\boldsymbol{p}\rangle=1$.

2.2. The basic K-HLC algorithm

In this subsection, the "basic K-HLC algorithm" is presented. The complete K-HLC algorithm will be given later in Section 3. The basic K-HLC algorithm is as follows:

The Basic K-HLC Algorithm.

$[\{\boldsymbol{l}_k\}_{k=1}^K, \{f_k\}_{k=1}^K] = \text{Basic- KHLC}(\boldsymbol{X}, K, \boldsymbol{L}^{(0)})$

Step 1: *Initialization.* Initialize the direction matrix as $\boldsymbol{L}^{(0)} = [\boldsymbol{l}_1^{(0)}, \ldots, \boldsymbol{l}_K^{(0)}]$. The initialization is flexible. We will discuss it in more details later.

Step 2: Extract a submatrix $\tilde{\boldsymbol{X}}$ from $\boldsymbol{X} = [\boldsymbol{x}(1), \ldots, \boldsymbol{x}(T)] \in \mathbb{R}^{m\times T}$ such that the norm of its each column is greater than a specific threshold ξ, where ξ is a positive constant chosen in advance. Suppose that $\tilde{T}$ columns of $\boldsymbol{X}$ are extracted.

As mentioned in [14], *the purpose of this step is twofold*: *first, it can reduce the computational burden of the estimation process, and second, it removes those columns that are disproportionately influenced by noise or outliers.*

This step can be omitted if the data are not very noisy.

Step 3: To suppress the outliers, normalize $\tilde{\boldsymbol{X}}$ such that each column $\|\tilde{\boldsymbol{x}}(t)\|_2 = 1,\ t = 1, \ldots, \tilde{T}$ if $\tilde{\boldsymbol{x}}(t) \neq \boldsymbol{0}$.

Step 3 *is also optional and we can directly go to step* 4. *However, this step is essential if the sources are not very sparse or they are corrupted by outliers* (*see Examples* 2 *and* 3).

Step 4: *Cluster assignment.* Assign the sample points $\{\boldsymbol{x}_t\}_{t=1}^T$ into K different clustering sets $\Omega_k,\ k = 1, \ldots, K$, where Ω_k is a vector set. The estimation of line direction of set Ω_k is $\boldsymbol{l}_k, k = 1, \ldots, K$. Based on the distance formula (4), compute each distance $d(\tilde{\boldsymbol{x}}(t), \boldsymbol{l}_k)$ from $\tilde{\boldsymbol{x}}(t),\ t = 1, \ldots, \tilde{T}$ to $\boldsymbol{l}_k,\ k = 1, \ldots, K$. The sample time index $t \in \Omega_k$ if and only if it satisfies $k = \arg\min_{i=1,\ldots,K}\{d^2(\tilde{\boldsymbol{x}}(t), \boldsymbol{l}_i)\}$. Suppose that $\tilde{T}_k$ points $\tilde{\boldsymbol{x}}(t_1^k), \ldots, \tilde{\boldsymbol{x}}(t_{\tilde{T}_k}^k)$ are assigned to set Ω_k, respectively. So we obtain a set of submatrices $\tilde{\boldsymbol{X}}_k = [\tilde{\boldsymbol{x}}(t_1^k), \ldots, \tilde{\boldsymbol{x}}(t_{\tilde{T}_k}^k)],\ k = 1, \ldots, K$.

In step 4, *the "hard assignment" strategy is used, where the winner hyperline takes all.*

Step 5: *Direction vector update.* Update $\boldsymbol{l}_k,\ k = 1, \ldots, K$ and their corresponding confidence indices f_k as follows:

For $k = 1, \ldots, K$

Compute the first eigenvector and its corresponding largest eigenvalue of the matrix $\tilde{\boldsymbol{X}}_k \cdot (\tilde{\boldsymbol{X}}_k)^{\mathrm{T}}/T$. Applying eigenvalue decomposition (EVD) or singular value decomposition (SVD), we have $\tilde{\boldsymbol{X}}_k \cdot (\tilde{\boldsymbol{X}}_k)^{\mathrm{T}}/T = \boldsymbol{U}_k\boldsymbol{\Lambda}_k(\boldsymbol{U}_k)^{\mathrm{T}}$, where $\boldsymbol{U}_k = [\boldsymbol{u}_{1k}, \ldots, \boldsymbol{u}_{mk}]$ are the set of eigenvectors.

End

Denote $\boldsymbol{\Lambda}_k = \mathrm{diag}(\lambda_{1k}, \ldots, \lambda_{mk})$ and suppose $\lambda_{1k} \geqslant \cdots \geqslant \lambda_{mk}$, where eigenvalues $\lambda_{1k}, \ldots, \lambda_{mk}$ correspond to $\mathbf{u}_{1k}, \ldots, \mathbf{u}_{mk}$, respectively. It should be noted that for each cluster, the eigenvectors $\mathbf{u}_{1k}$ corresponding to the largest eigenvalue λ_{1k}, are only used to update the line directions $\boldsymbol{l}_k$ of cluster Ω_k, i.e., $\boldsymbol{l}_k = \boldsymbol{u}_{1k}$. So we have

$$\boldsymbol{l}_k \leftarrow \boldsymbol{u}_{1k}, \quad f_k \leftarrow \lambda_{1k},\ k = 1, \ldots, K. \tag{5}$$

Step 6: Return to step 4, and repeat step 4, 5 until convergence.

Step 7: Output the estimated hyperlines $\boldsymbol{l}_k,\ k = 1, \ldots, K$ and their corresponding confidence indices $f_k,\ k = 1, \ldots, K$. So we have $\boldsymbol{L} = [\boldsymbol{l}_1, \ldots, \boldsymbol{l}_K]$ and $\boldsymbol{f} = [f_1, \ldots, f_K]^{\mathrm{T}}$.

The basic K-HLC is equivalent to solve the following optimization problem:

$$\min_{\boldsymbol{l}_k, \Omega_k,\ k=1,\ldots,K} J(\boldsymbol{l}_k, \Omega_k,\ k = 1, \ldots, K)$$
$$= \min_{\boldsymbol{l}_k, \Omega_k,\ k=1,\ldots,K} \sum_{t=1}^{T}\sum_{k=1}^{K} d^2(\tilde{\boldsymbol{x}}(t), \boldsymbol{l}_k) \cdot I_{t\in\Omega_k}, \tag{6}$$

where the indicator function $I_{t\in\Omega_k}$ is given as

$$I_{t\in\Omega_k} = \begin{cases} 1, & t \in \Omega_k, \\ 0, & t \notin \Omega_k,\ k = 1, \ldots, K. \end{cases} \tag{7}$$

It works in a similar way as *expectation maximization* (EM) algorithm [31,32]. The cost function $J(\cdot)$ in (6) decreases monotonically with the number of iterations until convergence is achieved. We stop the iterative procedure when $\|\boldsymbol{L}^{(\mathrm{iter})} - \boldsymbol{L}^{(\mathrm{iter}-1)}\| < \varepsilon$.

Remark. It is worth mentioning that the K-SVD also can be adopted for hyperline clustering by setting $T_0 = 1$ [9]. Furthermore, similar to the basic K-HLC, the largest singular values of SVD can be analogously used as the confidence indices to evaluate the significance of each hyperline although this point is not discussed in [9]. From this point of view, the basic K-HLC algorithm can be seen as an improved version of K-SVD for hyperline clustering. However, K-SVD contains two alternating steps in each iteration: basis vector update and coefficient update (sparse coding). The sparse coding stage of K-SVD corresponds to "clustering assignment step" in the basic K-HLC. The first difference between K-SVD and K-HLC is that K-HLC and K-SVD perform EVD or SVD on different matrices, respectively. In more details, for a given $m \times T$ matrix $\boldsymbol{X}$, the K-SVD directly performs SVD on $\boldsymbol{X}$ as $SVD(\boldsymbol{X}) = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}}$, where $\boldsymbol{U}$ is $m \times m$, $\boldsymbol{\Sigma}$ is $m \times T$ and $\boldsymbol{V}$ is $T \times T$. Then matrix $\boldsymbol{U}$ is used to update the dictionary and matrix $\boldsymbol{V}$ is used to update the sources. However, for K-HLC, we do not calculate $\boldsymbol{V}$. The K-HLC just performs EVD or SVD on a relatively small-dimensional symmetrical matrix $\boldsymbol{X}\cdot\boldsymbol{X}^{\mathrm{T}}$ to derive $EVD(\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}) = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{\mathrm{T}}$ or $SVD(\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}) = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{\mathrm{T}}$. For K-HLC, there is no need to compute the high dimensional matrix $\boldsymbol{V}$ in each iteration, which causes K-HLC to be computationally more efficient and saves storage space especially when $T \gg m$, which happens often for hyperline clustering. As an example, it cost more than 56.41 seconds in Example 1 of this work using K-SVD, whereas it took only about 26.875 seconds by K-HLC. In addition, since only the eigenvector corresponding to the largest eigenvalue is used in the K-HLC, we can compute it without performing full EVD/SVD decomposition.

The basic K-HLC algorithm is available under the condition that the number of hyperlines is exactly known (or given). Next, we would like to discuss the robust hyperline clustering and how to detect the number of hyperlines.

3. Robust hyperline clustering

In this section, we consider the practical implementation of the robust hyperline clustering, which is concerned with the following issues:

(i) Detection of the number of the hyperlines;
(ii) Development of efficient method to escape from the local minima;
(iii) Initialization of the K-HLC.
(iv) Suppression of the noise.

3.1. Robust K-HLC method

The largest eigenvalues $\lambda_{1k},\ k=1,\ldots,K$ in (5) can be chosen as the confidence index to evaluate how significant the corresponding identified hyperlines $L(\boldsymbol{l}_k),\ k=1,\ldots,n$ are. Moreover, we have the following theorem:

Theorem 1. *For SCA problem* (1), *suppose the following conditions*:

(i) *the sparse components* $s_1(t),\ldots,s_n(t)$ *satisfy disjoint orthogonality condition and have the same second-order moment* σ_s^2;
(ii) *the noises* $e_1(t),\ldots,e_m(t)$ *are zero-mean and statistically independent with the same second-order moment* σ_e^2 $(\sigma_e^2<\sigma_s^2)$;
(iii) *for arbitrary* i $(i=1,\ldots,n)$ *and* j $(j=1,\ldots,m)$, $s_i(t)$ *and* $e_j(t)$ *are mutually independent*;
(iv) $\boldsymbol{a}_1,\ldots,\boldsymbol{a}_n$ *are accurately identified by the basic K-HLC algorithm, i.e.,* $\{\boldsymbol{a}_1,\ldots,\boldsymbol{a}_n\}\subseteq\{\boldsymbol{l}_1,\ldots,\boldsymbol{l}_K\}$ $(K\geqslant n)$. *In other words, n of K hyperlines* $\boldsymbol{l}_1,\ldots,\boldsymbol{l}_K$ *correspond to true ones and they are* $\boldsymbol{a}_1,\ldots,\boldsymbol{a}_n$, *respectively*; *the remaining* $K-n$ *hyperlines of* $\boldsymbol{l}_1,\ldots,\boldsymbol{l}_K$ *are spurious.*

Then we have the following conclusions: (i) *if* $L(\boldsymbol{l}_i)$ *corresponds to a true hyperline, i.e.,* $\boldsymbol{l}_i\in\{\boldsymbol{a}_1,\ldots,\boldsymbol{a}_n\}$, *then* $\sigma_s^2\leqslant f_i\leqslant\sigma_s^2+\sigma_e^2$; (ii) *if* $L(\boldsymbol{l}_j)$ *is a spurious estimation, i.e.,* $\boldsymbol{l}_j\notin\{\boldsymbol{a}_1,\ldots,\boldsymbol{a}_n\}$, *then* $f_j\leqslant\sigma_e^2$.

Proof. see Appendix A.

Remark. Suppose that $L(\boldsymbol{l}_i)$ is a true hyperline and $L(\boldsymbol{l}_j)$ corresponds to a spurious one. Since $\sigma_s^2>\sigma_e^2$, we have $f_i\geqslant\sigma_s^2>\sigma_e^2\geqslant f_j$. For simplicity, suppose $L(\boldsymbol{l}_1),\ldots,L(\boldsymbol{l}_n)$ correspond to n true hyperlines and the remaining $L(\boldsymbol{l}_{n+1}),\ldots,L(\boldsymbol{l}_K)$ are spurious. Then we have $f_1=\cdots=f_n\geqslant\sigma_s^2>\sigma_e^2\geqslant f_{n+1}=\cdots=f_K$, which means that there will be a gap in the sorted confidence indices between f_n and f_{n+1} (see Figs. 2, 5 and 6). In fact, the gap is still noticeable even if the sparse sources do not perfectly satisfy disjoint orthogonality condition or data are corrupted by noise and outliers (see Example 2). If the gap is not very significant, we suggest to determine the number of hyperlines more precisely by means of computing the difference $\mathrm{gap}(k)=f_k-f_{k+1},\ k=1,\ldots,K_{max}-1$, of the confidence indices.

As mentioned in Section 1, usually we have no information about the number of the hidden hyperlines. For this reason, we overestimate the number of hyperlines first. From Theorem 1, we select only those hyperlines with larger confidence indices. Furthermore, we derive the "complete K-HLC algorithm" or "robust K-HLC algorithm":

The Complete K-HLC Algorithm.
$[\{\boldsymbol{l}_k\}_{k=1}^K,\{f_k\}_{k=1}^K,n]=\text{K-HLC}(\boldsymbol{X},K,\boldsymbol{L}^{(0)})$

Step 1: Overestimate the number of hyperlines by setting a large K (typically, $K>10n$, where n is the rough estimation of the number of the hyperlines).

Step 2: Apply "the basic K-HLC algorithm" on the observed data set $\boldsymbol{X}$ and obtain

$$[\{\boldsymbol{l}_k\}_{k=1}^K,\{f_k\}_{k=1}^K]=\text{Basic-KHLC}(\boldsymbol{X},K,\boldsymbol{L}^{(0)}).$$

Step 3: Determine the number n of hyperlines by searching the gap in the plot of the sorted confidence indices $\{f_k\}_{k=1}^K$.

Step 4: Extract n significant hyperlines from $\{\boldsymbol{l}_k\}_{k=1}^K$ according to their confidence indices $\{f_k\}_{k=1}^K$ and remove the spurious ones.

Since the cost function $J(\boldsymbol{l}_k,\Omega_k,\ k=1,\ldots,K)$ in (6) is not globally convex, the local minima occur very often. For the hyperline clustering, in addition to help detection of the number of hyperlines, overestimating the number of hyperlines is also helpful to escape from local minima actually, which is due to the fact that generally the larger the pre-specified number K is, the larger the possibility, that all true hyperlines are involved in the $m\times K$ initialization matrix $\boldsymbol{L}^{(0)}$, is.

3.2. Fast multilayer initialization (FMI)

Generally speaking, the hyperline $L(\boldsymbol{l}_k)$ can be easily identified by the basic K-HLC algorithm if $\boldsymbol{l}_k$ falls into the ε-neighborhood of a certain initial point $L(\boldsymbol{l}_i^{(0)}),\ i=1,\ldots,K$, i.e., $\boldsymbol{l}_k\in O(\boldsymbol{l}_i^{(0)},\varepsilon)$. From (6), we know that, although the basic K-HLC algorithm cannot guarantee the global minima, it can decrease the value of cost function in (6) monotonically. So the basic K-HLC algorithm can find the local minima after sufficient number of iterations and keep them during the iterative procedure. Since the true hyperline $L(\boldsymbol{l}_k)$ is optimal, it also corresponds to a certain local minimum. The basic K-HLC algorithm will keep $L(\boldsymbol{l}_k)$, once the hyperline $\boldsymbol{l}_k$ falls into the ε-neighborhood of a certain initial point $L(\boldsymbol{l}_i^{(0)})$.

The number of true hyperlines $L(\boldsymbol{l}_k),\ k=1,\ldots,n$ is finite and fixed. Theoretically, we are always able to cover all $\boldsymbol{l}_k,\ k=1,\ldots,n$ by ε-net $\bigcup_{k=1}^K O(\boldsymbol{l}_k^{(0)},\varepsilon)$ by setting a very large K (typically, $K>T/2$). Without loss of generality, we assume that the entries $l_{1k},\ldots,l_{mk}$ of direction vector $\boldsymbol{l}_k$ to be in the interval $[-1,1]$, i.e., $\boldsymbol{l}_k\in[-1,1]^m$, because we can always do this by normalization. So we can sufficiently segment the m-dimensional geometrical body $[-1,1]^m$ to obtain a very dense ε-grid, in which each node corresponds to an initial vector $\boldsymbol{l}_k^{(0)}$. In this way, n true direction vectors $\boldsymbol{l}_k,\ k=1,\ldots,n$ theoretically will be covered by ε-net $\bigcup_{k=1}^K O(\boldsymbol{l}_k^{(0)},\varepsilon)$ if K is sufficiently large so that n

hyperlines $L(\boldsymbol{l}_k)$, $k=1,\ldots,n$ could be identified by the K-HLC.

In practice, we have several ways to produce such an ε-net. First, we consider the following well-known simple initialization schemes:

Scheme 1: Extract a submatrix $\boldsymbol{L}^{(0)}=[\boldsymbol{x}(t_1),\ldots,\boldsymbol{x}(t_K)]_{m\times K}$ from $\boldsymbol{X}$ as an initial value of $\boldsymbol{L}$, i.e., select K column vectors from $\boldsymbol{X}$.

Scheme 2: Uniformly and randomly generate an m by K matrix as the initial value $\boldsymbol{L}^{(0)}$, whose entries are uniformly valued in the interval $[-1, 1]$. This scheme is simple, but works well.

Scheme 3: The K-HLC can work with other existing clustering algorithms. For example, we can use other clustering algorithms to roughly estimate the direction matrix $\boldsymbol{L}^{(0)}$ (with sufficiently high dimension) of the hyperlines as the initial value. Then we further precisely refine the hyperlines and determine the dimension of the direction matrix by the K-HLC.

Scheme 4: In order to considerably increase the possibility of covering all true line directions $\boldsymbol{l}_k$, $k=1,\ldots,n$ by the ε-net $\bigcup_{k=1}^{K} O(\boldsymbol{l}_k^{(0)},\varepsilon)$, it is better to set a very large K, for example, $K=T/2$. However, for large T (e.g., $K>2000$), the computational cost would be high and also it would requires large storage space. To overcome this problem, we employ the idea of multilayer or hierarchical system with some rough analogy to [33,34] and implement this idea in our "*fast multilayer initialization* (*FMI*)" method to produce a more efficient initialization $\boldsymbol{L}^{(0)}$ for the K-HLC.

Suppose that we have a high dimensional pre-initial direction matrix $\bar{\boldsymbol{L}}^{(0)}=[\bar{\boldsymbol{l}}_1^{(0)},\ldots,\bar{\boldsymbol{l}}_N^{(0)}]$, whose size is $m\times N$ (usually $N\gg n$ and $N\gg K$, typically, $N=100\,000$, $K=400$ and $n=6$). In many situations, we can take $\bar{\boldsymbol{L}}^{(0)}=\boldsymbol{X}$. In this case, $N=T$. We also can uniformly randomly generate $\bar{\boldsymbol{L}}^{(0)}$ in the interval $[-1, 1]$. For simplicity, we normalize $\boldsymbol{l}_k^{(0)}$, $k=1,\ldots,K$ such that $\|\bar{\boldsymbol{l}}_k^{(0)}\|_2=1$, $k=1,\ldots,N$. The FMI algorithm is as follows:

The FMI algorithm. *Step* 1: *Segmentation*. The original $m\times N$ direction matrix $\bar{\boldsymbol{L}}^{(0)}$ is segmented into M lower dimensional matrices $\bar{\boldsymbol{L}}_1^{(0)},\ldots,\bar{\boldsymbol{L}}_M^{(0)}$, whose sizes are $m\times N_1,\ldots,m\times N_M$, respectively. Obviously, $\bar{\boldsymbol{L}}^{(0)}=[\bar{\boldsymbol{L}}_1^{(0)},\ldots,\bar{\boldsymbol{L}}_M^{(0)}]$. Here, $N_1,\ldots,N_M$ are much smaller than N. For simplicity, we usually can take $N_1=\cdots=N_M$ (e.g., $N_1=\cdots=N_M=400$).

Step 2: *Normalization*. Normalize the observed matrix $\boldsymbol{X}$ to $\tilde{\boldsymbol{X}}$ such that $\|\tilde{\boldsymbol{x}}(t)\|_2=1$, $t=1,\ldots,\tilde{T}$ if $\tilde{\boldsymbol{x}}(t)\neq\boldsymbol{0}$.

Step 3: *Searching direction matrix* $\boldsymbol{L}^{(0)}$. In this step, we attempt to select K possibly optimal columns from $\bar{\boldsymbol{L}}^{(0)}$ to construct the direction matrix $\boldsymbol{L}^{(0)}$.

Set $\boldsymbol{L}^{(0)}=$ null;
For $i=1,\ldots,M$
  Construct $\boldsymbol{L}_i^{(0)}=[\boldsymbol{L}^{(0)},\bar{\boldsymbol{L}}_i^{(0)}]$, where $\boldsymbol{L}_i^{(0)}$ is the $m\times(K+N_i)$ size;
  Extract K possible columns (which are the candidates of the optimal columns) from $\boldsymbol{L}_i^{(0)}$ to construct $\boldsymbol{L}^{(0)}$ (the detailed operations are as follows);
End

To select K columns from $\boldsymbol{L}_i^{(0)}=[\boldsymbol{l}_i^{(0)}(1),\ldots,\boldsymbol{l}_i^{(0)}(K+N_i)]$, *we perform the following operations*: *compute all correlation coefficients* $c(\boldsymbol{l}_i^{(0)}(k),\tilde{\boldsymbol{x}}(t))=[\boldsymbol{l}_i^{(0)}(k)]^{\mathrm{T}}[\tilde{\boldsymbol{x}}(t)]$, $t=1,\ldots,T$, $k=1,\ldots,K+N_i$; *assign all samples* $\tilde{\boldsymbol{x}}(t)$, $t=1,\ldots,T$ *to the* $K+N_i$ *different sets* $\Omega(\boldsymbol{l}_i^{(0)}(k))$, $k=1,\ldots,K+N_i$; $\tilde{\boldsymbol{x}}(t)\in\Omega(\boldsymbol{l}_i^{(0)}(k))$ *if and only if* $\tilde{\boldsymbol{x}}(t)$ *satisfies* $k=\arg\max_{j=1,\ldots,K+N_i}\{|c(\tilde{\boldsymbol{x}}(t),\boldsymbol{l}_i^{(0)}(j))|\}$, *and compute the number of entries in each set* $\Omega(\boldsymbol{l}_i^{(0)}(k))$, $k=1,\ldots,K+N_i$. *Extract K columns from* $\boldsymbol{L}_i^{(0)}$ *to construct* $m\times K$ *matrix* $\boldsymbol{L}^{(0)}$, *where these K columns correspond to the K sets that have most number of entries*.

Step 4: *Output*. Output the $m\times K$ initial direction matrix $\boldsymbol{L}^{(0)}$ to the K-HLC.

$$\mathrm{SIR}(\boldsymbol{A},\widehat{\boldsymbol{A}})=-20\log_{10}\left(\frac{1}{n}\sum_{i=1}^{n}\times\min_{j=1,\ldots,n}\frac{\min\{\|\boldsymbol{a}_i-\widehat{\boldsymbol{a}}_j\|_2,\|\boldsymbol{a}_i+\widehat{\boldsymbol{a}}_j\|_2\}}{\|\boldsymbol{a}_i\|_2}\right)\ (\mathrm{dB}). \quad (8)$$

The FMI method can rapidly scan the high dimensional matrix $\boldsymbol{L}^{(0)}$. It can be used to produce efficient initializations $\boldsymbol{L}^{(0)}$ for the relatively large scale K-HLC.

3.3. Multilayer K-HLC

In the noise-free case, the K-HLC works well. In order to improve its robustness to noise, we consider a multilayer K-HLC scheme: given an initialization $\boldsymbol{L}^{(0)}$, the K-HLC obtains a set of hyperlines $\boldsymbol{L}^{(1)}=[\boldsymbol{l}_1^{(1)},\ldots,\boldsymbol{l}_K^{(1)}]$ and their corresponding confidence indices $\boldsymbol{f}^{(1)}$; then it removes the spurious estimations according to their confidence indices, and input the remaining hyperlines $\tilde{\boldsymbol{L}}^{(1)}=[\tilde{\boldsymbol{L}}_1^{(1)},\ldots,\tilde{\boldsymbol{L}}_{K1}^{(1)}]$ ($K1<K$) as the new initial direction matrix, the K-HLC will obtain another set of estimations and their corresponding confidence indices. In this way, we can refine the clustering results repeatedly until we can see a relatively clear gap in the confidence indices. The multilayer K-HLC is intuitive and heuristic. It improves the performance in the hyperline clustering by gradually decreasing the dimension of the problem (see Example 3).

4. Numerical experiments and analysis of results

In this section, we demonstrate that the K-HLC algorithm is used to identify the basis matrix $\boldsymbol{A}$ for SCA. We compare it with some existing methods in Example 1. Next, from Examples 2–5 we test the K-HLC only for some much more challenging benchmarks that are not achievable or difficult to achieve for the conventional methods. These examples are challenging due to the following reasons:

(i) Only few samples satisfy *disjoint orthogonality* condition (typically, see Example 2);
(ii) The observations are contaminated by noises (e.g., Example 3);
(iii) The basis matrix $\boldsymbol{A}$ is very ill-conditioned (see Example 4);
(iv) The problem is relatively large scale (see Example 5).

In all examples, the FMI is used to give the initializations, where the pre-initialized matrix $\bar{\boldsymbol{L}}^{(0)}$ is set as $\bar{\boldsymbol{L}}^{(0)}=\boldsymbol{X}$

unless otherwise mentioned. One-layer K-HLC is used if no specific information explanation is given. All examples are performed on a PC with Intel Pentium 4 CPU 2.20 GHz.

To check how well the basis matrix is estimated, we compute the signal to interference ratio (SIR) between the true $\boldsymbol{A}$ and its estimation $\widehat{\boldsymbol{A}}$, which is defined in (8) at the bottom of this page. The SIR between source s and its estimate $\widehat{s}$ is calculated to measure the accuracy of the estimations of the sparse components:

$$\mathrm{SIR}(s,\widehat{s}) = 10\log_{10}\frac{\sum_{t=1}^{T} s^2(t)}{\sum_{t=1}^{T}(s(t)-\widehat{s}(t))^2}\ (\mathrm{dB}). \tag{9}$$

Since the estimated sparse source $\widehat{s}$ may have arbitrary scale, we rescale $\widehat{s}$ to have the same energy level as s before computing their SIR.

Example 1. We begin with an easy example of blind speech separation in time–frequency domain. The considered four sources (65 536 samples in the time domain) are from the experiment "FourVoices" in [10]. The mixing matrix was randomly generated and followed by normalization:

$$\boldsymbol{A} = \begin{bmatrix} -0.7798 & -0.3703 & 0.1650 & 0.5585 \\ -0.0753 & 0.8316 & 0.6263 & 0.3753 \\ -0.6215 & -0.4139 & 0.7619 & -0.7398 \end{bmatrix}.$$

Three mixtures were obtained by $\boldsymbol{X} = \boldsymbol{AS}$. To satisfy the sparseness assumption, we performed SCA in the time–frequency domain. A short time Fourier transform (STFT) with Hanning window was used. The STFT parameters were set to be the same as those in the experiment "FourVoices" in [10]: the window length of STFT was 2048, and the hop distance was $d = 614$. In more details, a $3 \times T$ observed matrix $\boldsymbol{X}$ of the mixtures are first segmented into a series of 3×2048 frames $\boldsymbol{X}^i$, $i = 1, 2, \ldots$; then we perform STFT on each frame as $STFT(\boldsymbol{X}^i) = \boldsymbol{A} \cdot STFT(\boldsymbol{S}^i)$ and denote it as $\boldsymbol{X}_f^i = \boldsymbol{A} \cdot \boldsymbol{S}_f^i$. In this way, $\boldsymbol{X} = \boldsymbol{AS}$ was transformed to the time–frequency domain as $\boldsymbol{X}_f = [\boldsymbol{X}_f^1, \boldsymbol{X}_f^2, \ldots] = \boldsymbol{A} \cdot [\boldsymbol{S}_f^1, \boldsymbol{S}_f^2, \ldots] = \boldsymbol{A} \cdot \boldsymbol{S}_f$. By taking the real part and imaginary part, we have $\mathrm{Real}(\boldsymbol{X}_f) = \boldsymbol{A} \cdot \mathrm{Real}(\boldsymbol{S}_f)$ and $\mathrm{Imag}(\boldsymbol{X}_f) = \boldsymbol{A} \cdot \mathrm{Imag}(\boldsymbol{S}_f)$. They can be arranged in the format as the model (1):

$$\begin{aligned} \underline{\boldsymbol{X}}_f &= [\mathrm{Real}(\boldsymbol{X}_f)\ \ \mathrm{Imag}(\boldsymbol{X}_f)] \\ &= \boldsymbol{A} \cdot [\mathrm{Real}(\boldsymbol{S}_f)\ \ \mathrm{Imag}(\boldsymbol{S}_f)] = \boldsymbol{A} \cdot \underline{\boldsymbol{S}}_f. \end{aligned} \tag{10}$$

The scatter plot of $\underline{\boldsymbol{X}}_f$ is shown in Fig. 1(a). We solved this blind separation problem in the same way as in [10] except that instead of the potential function-based method, here the K-HLC is employed to estimate the mixing matrix $\boldsymbol{A}$ from $\underline{\boldsymbol{X}}_f$ in (10). K was set to $K = 30$. So, $\widehat{\boldsymbol{A}}$ is 3×30 (i.e., $\widehat{\boldsymbol{A}}_{3\times 30}$) 50 Monte Carlo runs were conducted using the K-HLC with random initializations. Thirty corresponding sorted confidence indices of one of Monte Carlo tests are shown in Fig. 1(c), where we can see that the first 4 are significantly greater than the others. So the first four columns $\widehat{\boldsymbol{A}}_{3\times 4}$ of $\widehat{\boldsymbol{A}}_{3\times 30}$ are chosen as the estimation of mixing matrix $\boldsymbol{A}$. The average SIR of 50 Monte Carlo tests is 38.03 dB.

In addition, we compare the K-HLC with the TIFROM and the K-means algorithm in this example. TIFROM is an algorithm to identify the mixing matrix in the time–frequency domain for blind sparse source separation [16]. When the size of STFT is 2048 for this example, the TIFROM algorithm achieved the best estimation with SIR = 30.59 dB. For K-means algorithm, 50 Monte Carlo tests were also performed, where the SIRs of 12 Monte Carlo runs were below 13 dB. More details are shown in Table 1, where we can see that the estimation obtained by the K-HLC is more reliable than those of TIFROM and K-means algorithm.

Example 2. This example was originally given by Li et al. [14] to test the robustness of SCA algorithms in insufficient sparsity situation, where only 10% of samples

Table 1
Comparison results of 50 Monte Carlo runs of the K-HLC and the K-means (in dB).

Methods	SIR_{min}	SIR_{max}	Average value of 50 SIRs
K-means algorithm	11.96	33.85	20.64
K-HLC algorithm	33.77	43.64	38.03

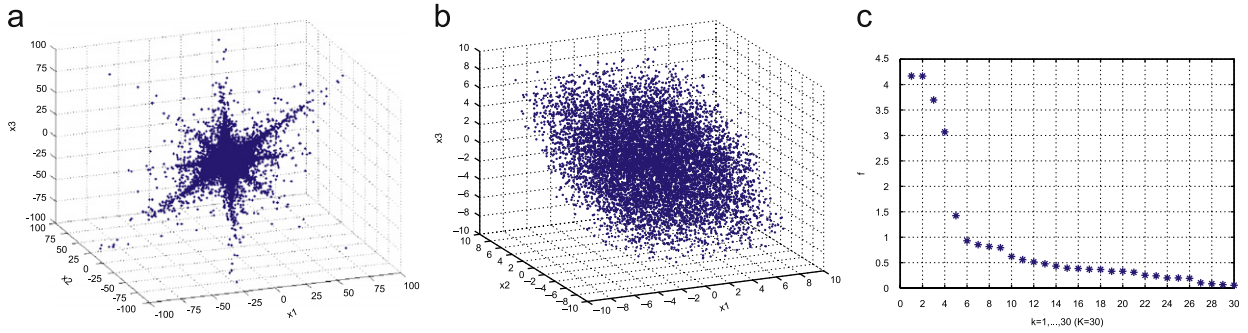

Fig. 1. (a) The scatter plot of $\underline{\boldsymbol{X}}_f$ in Example 1. (b) The scatter plot of $\boldsymbol{X}$ in Example 2. (c) Twenty sorted confidence indices. There is a noticeable gap between the first four and the others.

satisfy *disjoint orthogonality* condition [15,17]. The source matrix $\boldsymbol{S} \in \mathbb{R}^{5\times 10\,000}$ has two types of columns. That is, 9000 columns of $\boldsymbol{S}$ have their entries drawn from a uniform distribution valued in $[-5, 5]$; the other 1000 columns have only one nonzero entry and four zero entries. Furthermore, the 1000 columns are divided into five sets, the ith set has 200 columns with their ith entries being nonzeros (also drawn from the uniform distribution) $(i = 1,\ldots,5)$. The column indices of the 1000 columns in the matrix are disjoint. More precisely, the column indices of the 200 columns of the ith set are as follows: $(i-1)\times 10+1, (i-1)\times 10+51,\ldots,(i-1)\times 10+9951$; $i = 1,\ldots,5$. The mixtures $\boldsymbol{X} = \boldsymbol{AS}$, where the randomly generated mixing matrix $\boldsymbol{A} \in \mathbb{R}^{3\times 5}$ was as follows:

$$\boldsymbol{A} = \begin{bmatrix} 0.8412 & -0.0298 & -0.0750 & -0.1735 & 0.7240 \\ -0.5025 & 0.8305 & -0.8294 & 0.3621 & 0.4088 \\ 0.1997 & 0.5563 & 0.5536 & 0.9158 & -0.5556 \end{bmatrix}.$$

Set $K = 200$. We applied simply one-layer K-HLC because of no noise. After 43 iterations, we obtained $\widehat{\boldsymbol{A}}_{3\times 200}$ and its corresponding 200 confidence indices. The gap curve in Fig. 2 shows that the number of hyperlines should be $K = 5$. Hence we obtained the estimated $\widehat{\boldsymbol{A}}_{3\times 5}$ as follows:

$$\widehat{\boldsymbol{A}} = \begin{bmatrix} 0.8424 & -0.0281 & -0.0749 & -0.1777 & -0.7241 \\ -0.5002 & 0.8289 & -0.8300 & 0.3598 & -0.4084 \\ 0.2004 & 0.5586 & 0.5528 & 0.9159 & 0.5557 \end{bmatrix},$$

with $\mathrm{SIR}(\boldsymbol{A}, \widehat{\boldsymbol{A}}_{3\times 5}) = 52.0$ dB.

In comparison to Example 1, this example is much more challenging. There are only 10% of samples satisfying disjoint orthogonality condition [15,17] and the other 90% of samples are outliers. We cannot see any noticeable line directions in the scatter plot in Fig. 1(b), whereas four lines are clear in the scatter plot of Example 1 (see Fig. 1(a)). Additionally, the sources are not sparse in the time–frequency domain in this example. The TIFROM algorithm cannot be used [14,16]. K-means clustering algorithm also failed. We tried 50 Monte Carlo tests to estimate $\boldsymbol{A}$ by the K-means clustering algorithm, where the average SIR is only 7.00 dB. The algorithm proposed in [14] is available for this example. Comparing with it, K-HLC has fewer parameters (thresholds) to set. Relatively, Li et al.'s algorithm [14] is advantageous when there are only few samples in the observed data.

In order to test the robustness of our approach, we further conducted the Monte Carlo tests. For each K, 50 Monte Carlo runs were conducted. In each Monte Carlo run, the initial direction matrices were generated by the FMI, where the elements of $\bar{\boldsymbol{L}}^{(0)}$ were uniformly randomly drawn from $[-0.5, 0.5]$. The results are shown in Table 2, where we can see that the K-HLC consistently succeeded in all Monte Carlo trials when $K \geqslant 50$.

The overestimation of the number of hyperlines is essential. If the pre-set K is too small, there may be no clear gap (see Fig. 3) so that K-HLC may even fail. For instance, we took $K = 20$ and applied K-HLC. In this case, $\mathrm{SIR}(\boldsymbol{A}, \widehat{\boldsymbol{A}}_{3\times 5}) = 12.5$ dB. Also, Table 2 showed that the performance of K-HLC is not good when $K \leqslant 20$, while it always succeeded when $K \geqslant 50$. Generally speaking, the larger the pre-set K is, the more accurate the estimation of basis matrix is in the noise free case. For example, set $K = 500$, and we ran our algorithm again. A 3×500 matrix $\widehat{\boldsymbol{A}}_{3\times 500}$ was obtained. We chose again the first five columns of $\widehat{\boldsymbol{A}}_{3\times 500}$ as the estimation of $\boldsymbol{A}$. In this case $\mathrm{SIR}(\boldsymbol{A}, \widehat{\boldsymbol{A}}_{3\times 5}) = 60.9$ dB. However, as mentioned previously, if K is large, the computational cost is relatively high.

Example 3. In this example, we perform multilayer K-HLC for the noisy data. Here both the basis matrix $\boldsymbol{A}$ and the sparse sources $\boldsymbol{S}$ are the same as Example 2, but the different level (SNR) white Gaussian noises were added to the observations $\boldsymbol{X}$, respectively. A four-layer K-HLC was

Table 2
Results of 50 Monte Carlo runs for different K in the noise free case.

K	5	6	8	10	20	50	100	200	300	500
SIR (dB)	11.0	12.8	12.3	11.7	13.8	35.0	42.7	49.8	54.4	60.3
Runtime (s)	2.4	2.5	4.0	4.4	9.8	17.3	36.8	53.7	84.0	122.8

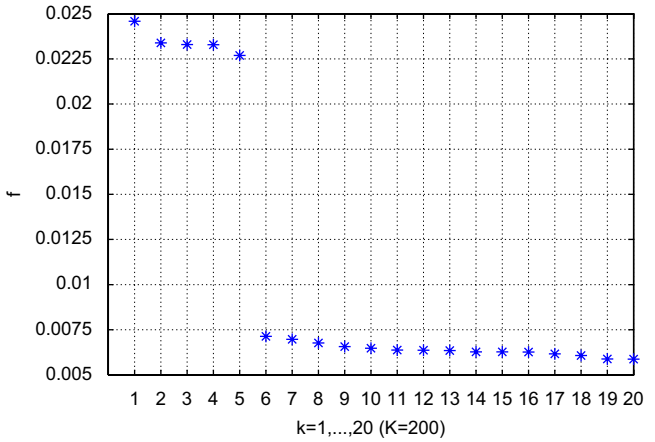

Fig. 2. Distribution of the first 20 confidence indices. There are totally 200 confidence indices associated with the columns of $\widehat{\boldsymbol{A}}_{3\times 200}$, respectively. We can see a clear gap which shows that the first five columns $\widehat{\boldsymbol{a}}_1,\ldots,\widehat{\boldsymbol{a}}_5$ of $\widehat{\boldsymbol{A}}_{3\times 200}$ correspond to the mixing matrix.

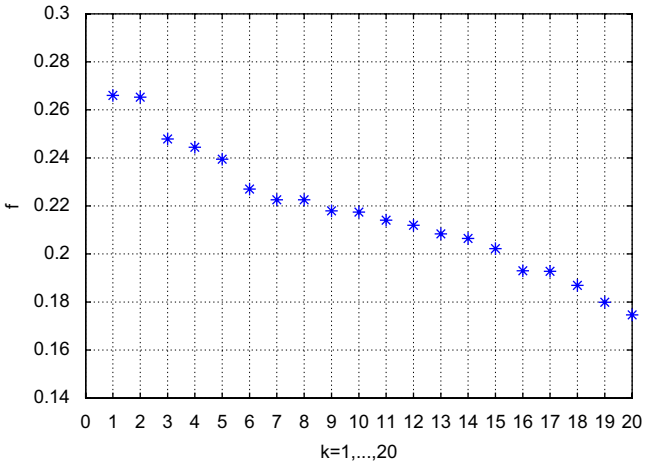

Fig. 3. The sorted confidence indices for $K = 20$. In this case, we do not observe any significant gap.

Table 3
The average SIR$(\boldsymbol{A}, \widehat{\boldsymbol{A}}_{3\times 5})$ of 50 Monte Carlo runs in different noise level (in dB).

Noise level (dB)	20	25	30	35	40	Noiseless
The 1st layer ($K = 500$)	5.8	9.4	18.7	36.4	48.1	58.7
The 2nd layer ($K = 200$)	4.8	12.6	33.6	43.0	46.4	49.8
The 3rd layer ($K = 100$)	6.9	13.9	36.5	39.0	40.1	41.5
The 4th layer ($K = 50$)	11.2	29.2	32.9	34.0	35.0	35.9

performed, where $K = 500$, 200, 100 and 50, respectively. For different noise level, 50 Monte Carlo runs were conducted. In each Monte Carlo run, the initializations were also generated by the proposed FMI, where the pre-initial matrix $\tilde{\boldsymbol{L}}^{(0)}$ was also drawn from a uniform distribution. The results are shown in Table 3, in which the four-layer K-HLC worked well when $SNR > 20$ dB. It is shown that the multilayer K-HLC can gradually improve the performance in the higher noise level cases. It is observed that for the noisy data, the gap of confidence indices is less noticeable, but it is still possible to identify them (see Fig. 4).

Note that the multilayer approach did not improve the performance, and even degraded the performance when the noise level is 30 dB or higher. However, the SIR is always higher than 32 dB in this case. In other words, although the multilayer approach degrades the performance to some extend, the performance is sufficiently good in this case. The reason for this problem is that this benchmark is very challenging because 90% of samples are outliers (only 10% of samples are desirable). It is worth noting that the number K of clusters was reduced from 500 to 50 layer by layer in the multilayer architecture. At the same time, 90% outliers must be always assigned to K clusters at each layer. If K is not large enough, it will be difficult to suppress the outliers.

Example 4. A normalized 10×10 Hilbert matrix with a coherence parameter 0.99986 (the condition number 8.4797e + 012) was chosen as the basis matrix generated by MATLAB function 'hilb' [35]. Since the *coherence parameter* of $\boldsymbol{A}$ is very close to "1", the angles between certain columns $\boldsymbol{a}_i$ of $\boldsymbol{A}$ are quite small [36]. The sources $\boldsymbol{S} \in R^{10\times 10\,000}$ are generated in the same way as in Example 2. Similar to previous examples, only 10% of samples of the sources satisfy the *disjoint orthogonality condition* and the rest are corrupted by random outliers or large noise. This example is much more challenging than Example 2 because Hilbert matrix is extremely ill-conditioned.

We set $K = 500$, and performed K-HLC. The confidence indices of the first 10 columns of $\widehat{\boldsymbol{A}}_{10\times 500}$ are significantly larger than the others. So, the first 10 columns of $\widehat{\boldsymbol{A}}_{10\times 500}$ is the estimation of $\boldsymbol{A}$ (see Fig. 5). SIR$(\boldsymbol{A}, \widehat{\boldsymbol{A}}_{10\times 10}) = 67.5$ dB. The SIRs between the sources and their estimations were 29.3, 25.8, 26.1, 23.2, 31.0, 27.4, 32.9, 26.7, 27.7, 27.3 dB, respectively. Both the basis matrix and the sparse components were well estimated.

Example 5. In this example, we consider a medium scale SCA problem: the basis matrix $\boldsymbol{A}$ is 30×50 and the sparse source matrix $\boldsymbol{S}$ is $50 \times 10\,000$. Fifty Monte Carlo runs were conducted for this example. In each Monte Carlo run, $\boldsymbol{A}$ was randomly generated with different seed by MATLAB commands: "$A = rand(30, 50) - 0.5$" and followed by normalization. $\boldsymbol{S} \in \mathbb{R}^{50\times 10\,000}$ was generated by the following MATLAB command [35]:

$$\boldsymbol{S} = -\log(rand(50, 10\,000)) \times \max(0, \mathrm{sign}(rand(50, 10\,000) - p)). \quad (11)$$

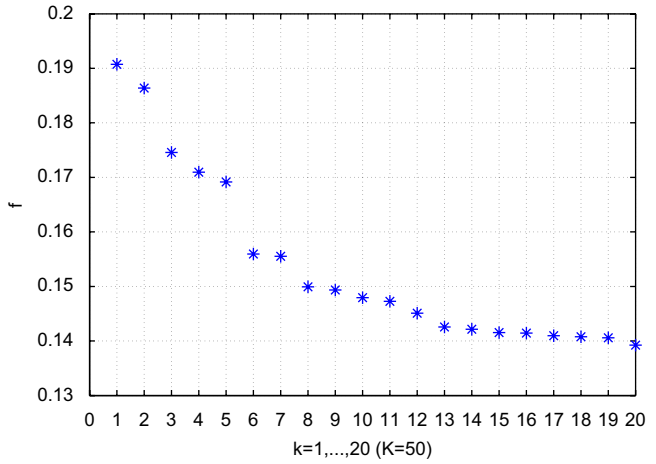

Fig. 4. The first 20 confidence indices of the four-layer K-HLC for the noisy data with $SNR = 25$ dB. We still can extract five true hidden hyperlines although there is no as clear gap as in noiseless case.

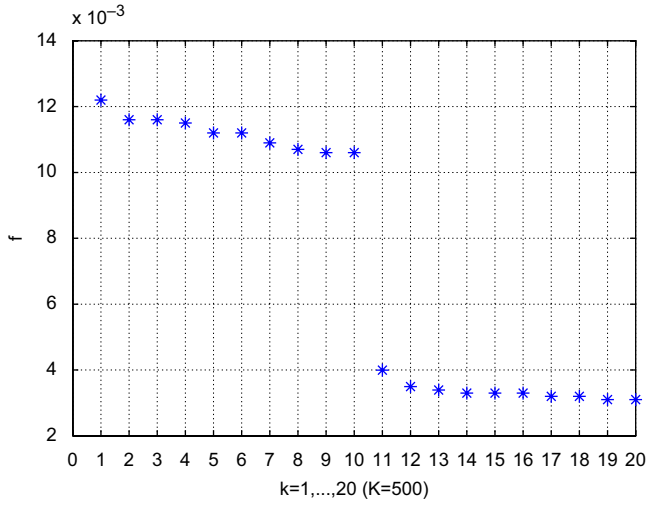

Fig. 5. The first 20 largest confidence indices selected from the 500 confidence indices associated with the column vectors of $\widehat{\boldsymbol{A}}_{10\times 500}$, where the first 10 confidence indices corresponding to $\widehat{\boldsymbol{a}}_1,\ldots,\widehat{\boldsymbol{a}}_{10}$ are very significant.

Table 4
The average SIR of 50 Monte Carlo runs of the two-layer K-HLC in noisy SCA (dB).

Noise level (dB)	15	20	25	30	40	50	Noiseless
The 1st layer ($K = 1000$)	18.5	24.7	29.0	31.0	33.2	34.0	34.2
The 2nd layer ($K = 500$)	21.4	25.4	26.9	27.8	28.6	28.9	28.9

By choosing different parameter p ($0 \leqslant p \leqslant 1$) in (11), we can obtain the sources with different sparseness degree. The probability that a source entry is zero is p. In this example, we set $p = 0.95$. We generated the observations as $\boldsymbol{X} = \boldsymbol{A} \cdot \boldsymbol{S} + \boldsymbol{E}$, where the levels of white Gaussian noise $\boldsymbol{E}$ are shown in Table 4.

In all Monte Carlo runs, the two-layer K-HLC was performed with $K = 1000$ and 500, respectively. The gap curve of one of the Monte Carlo runs is plotted in Fig. 6. The results are shown in Table 4. Even with relatively large level SNR = 15 dB, we still obtained satisfactory solutions. The more detailed Monte Carlo results regarding SIR and runtime are shown in Fig. 7. It is seen that comparing with the single layer K-HLC, the two layer K-HLC gradually improves the SIR as SNR of the mixtures decreases.

5. Conclusions

Hyperline clustering was discussed in this work. An efficient hyperline clustering method "K-HLC" was proposed. K-HLC is robust to the outliers. Interestingly, it also can simultaneously detect the number of hyperlines during searching the hyperlines. Based on K-HLC, a multilayer K-HLC framework was further developed, which can improve the performance in the noisy case. In

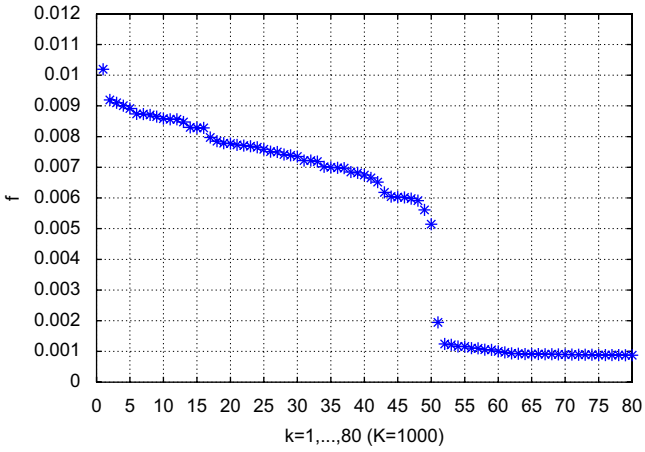

Fig. 6. The first 80 largest confidence indices. There are totally 1000 confidence indices associated with the column vectors of $\hat{\boldsymbol{A}}_{30\times1000}$. The first 50 confidence indices corresponding to $\hat{\boldsymbol{a}}_1,\ldots,\hat{\boldsymbol{a}}_{50}$ are greater than the others.

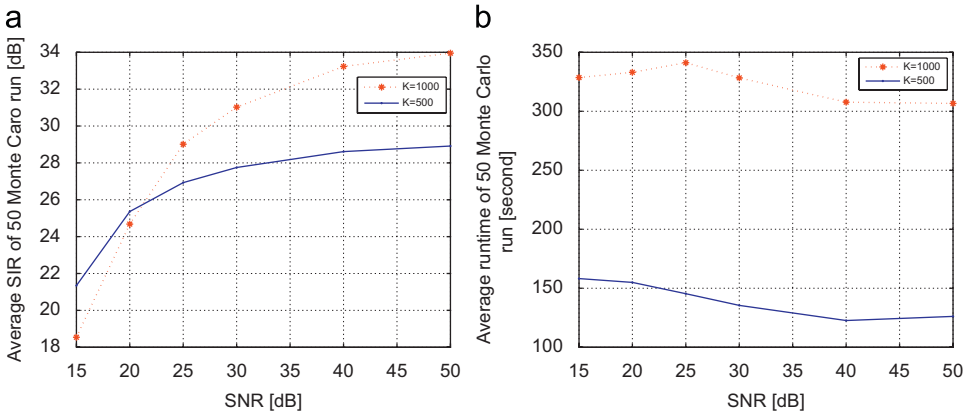

Fig. 7. The results of 50 Monte Carlo runs. (a) The average SIR versus SNR of the mixtures. (b) The average runtime versus SNR.

addition, the proposed FMI can seamlessly work with the K-HLC and produces efficient initializations for the K-HLC.

The K-HLC-based SCA is applicable to those applications such as underdetermined BSS, in which the number of the sources is unknown. Also it is suitable for those cases where not all the hyperlines are equally important and we need to extract only significant ones best represented by a given data.

Acknowledgements

The work is supported in part by National Natural Science Foundation of China (Grants 60505005, 60874061, 60825306), the Natural Science Fund of Guangdong Province, China (Grant 05103553) and CPSF (Grants 20070410237, 200801248).

Appendix A. Proof of Theorem 1

(1) We first prove the first part: if $L(\boldsymbol{l}_i)$ is a true hyperline, i.e., $\boldsymbol{l}_i\in\{\boldsymbol{a}_1,\ldots,\boldsymbol{a}_n\}$, then $\sigma_s^2\leqslant f_i\leqslant\sigma_s^2+\sigma_e^2$.

In the K-HLC algorithm presented in Section 2.2, note that f_i is the largest eigenvalue of $\lim_{T\to+\infty}\tilde{\boldsymbol{X}}_i\cdot(\tilde{\boldsymbol{X}}_i)^{\mathrm{T}}/T$. Therefore, f_i is the solution of the following optimization problem (see [37, Chapter 4]):

$$f_i=\lambda_{1i}=\max_{\boldsymbol{u}\neq0}\frac{\boldsymbol{u}^{\mathrm{T}}\cdot\lim_{T\to+\infty}[\tilde{\boldsymbol{X}}_i\cdot(\tilde{\boldsymbol{X}}_i)^{\mathrm{T}}/T]\cdot\boldsymbol{u}}{\boldsymbol{u}^{\mathrm{T}}\boldsymbol{u}}$$
$$=\max_{\boldsymbol{u}\neq0}\frac{\boldsymbol{u}^{\mathrm{T}}\cdot\lim_{T\to+\infty}[\sum_{t\in\Omega_i}\tilde{\boldsymbol{x}}(t)\cdot\tilde{\boldsymbol{x}}^{\mathrm{T}}(t)/T]\cdot\boldsymbol{u}}{\boldsymbol{u}^{\mathrm{T}}\boldsymbol{u}}. \tag{12}$$

Since $L(\boldsymbol{l}_i)$ is a true hyperline (i.e., $\boldsymbol{l}_i\in\{\boldsymbol{a}_1,\ldots,\boldsymbol{a}_n\}$), without loss of generality, suppose $\boldsymbol{l}_i=\boldsymbol{a}_r$. Then we have $s_\tau(t)=0$ if $\tau\neq r$, $t\in\Omega_i$ for all $\tau=1,\ldots,n$ and $t=1,\ldots,T$ under the assumption that the sparse components $s_1(t),\ldots,s_n(t)$, $t=1,\ldots,T$ satisfy *disjoint orthogonality* condition and $\boldsymbol{a}_1,\ldots,\boldsymbol{a}_n$ are accurately identified by the K-HLC. So we can derive

$$\tilde{\boldsymbol{x}}(t)=\sum_{\tau=1}^{n}\boldsymbol{a}_\tau\cdot s_\tau(t)+\boldsymbol{e}(t)=\boldsymbol{a}_r\cdot s_r(t)+\boldsymbol{e}(t),$$
$$t\in\Omega_i\text{ and }\boldsymbol{l}_i=\boldsymbol{a}_r. \tag{13}$$

From (13), we have

$$\tilde{\boldsymbol{x}}(t)\cdot\tilde{\boldsymbol{x}}^{\mathrm{T}}(t)=\boldsymbol{a}_r\boldsymbol{a}_r^{\mathrm{T}}\cdot s_r^2(t)+2\boldsymbol{a}_r\cdot\boldsymbol{e}^{\mathrm{T}}(t)\cdot s_r(t)+\boldsymbol{e}(t)\cdot\boldsymbol{e}^{\mathrm{T}}(t),\quad t\in\Omega_i \text{ and } \boldsymbol{l}_i=\boldsymbol{a}_r. \tag{14}$$

Then

$$\frac{1}{T}\sum_{t\in\Omega_i}\tilde{\boldsymbol{x}}(t)\cdot\tilde{\boldsymbol{x}}^{\mathrm{T}}(t)=\frac{1}{T}\sum_{t\in\Omega_i}[\boldsymbol{a}_r\boldsymbol{a}_r^{\mathrm{T}}\cdot s_r^2(t)+2\boldsymbol{a}_r\cdot\boldsymbol{e}^{\mathrm{T}}(t)\cdot s_r(t)+\boldsymbol{e}(t)\cdot\boldsymbol{e}^{\mathrm{T}}(t)]. \tag{15}$$

Since $s_r(t)=0$ if $t\notin\Omega_i$, we have $\sum_{\substack{t\notin\Omega_i\\ t=1,\ldots,T}}\boldsymbol{a}_r\boldsymbol{a}_r^{\mathrm{T}}\cdot s_r^2(t)=0$. Thus, we can get

$$\frac{1}{T}\sum_{t\in\Omega_i}\boldsymbol{a}_r\boldsymbol{a}_r^{\mathrm{T}}\cdot s_r^2(t)=\frac{1}{T}\sum_{\substack{t\in\Omega_i\\ t=1,\ldots,T}}\boldsymbol{a}_r\boldsymbol{a}_r^{\mathrm{T}}\cdot s_r^2(t)+\frac{1}{T}\sum_{\substack{t\notin\Omega_i\\ t=1,\ldots,T}}\boldsymbol{a}_r\boldsymbol{a}_r^{\mathrm{T}}\cdot s_r^2(t)=\frac{1}{T}\sum_{t=1}^{T}\boldsymbol{a}_r\boldsymbol{a}_r^{\mathrm{T}}\cdot s_r^2(t). \tag{16}$$

From the supposed condition (i) of Theorem 1, the second-order moment of the sparse component $s_r(t)$ is σ_s^2. Thus

$$\begin{aligned}\lim_{T\to+\infty}\frac{1}{T}\sum_{t\in\Omega_i}\boldsymbol{a}_r\boldsymbol{a}_r^{\mathrm{T}}\cdot s_r^2(t)&=\lim_{T\to+\infty}\frac{1}{T}\sum_{t=1}^{T}\boldsymbol{a}_r\boldsymbol{a}_r^{\mathrm{T}}\cdot s_r^2(t)\\&=\boldsymbol{a}_r\boldsymbol{a}_r^{\mathrm{T}}\cdot\lim_{T\to+\infty}\frac{1}{T}\sum_{t=1}^{T}s_r^2(t)\\&=\boldsymbol{a}_r\boldsymbol{a}_r^{\mathrm{T}}\cdot\sigma_s^2.\end{aligned} \tag{17}$$

By analogy, we can obtain $\sum_{t\in\Omega_i}\boldsymbol{e}^{\mathrm{T}}(t)\cdot s_r(t)=\sum_{t=1}^{T}\boldsymbol{e}^{\mathrm{T}}(t)\cdot s_r(t)$ because $s_r(t)=0$ if $t\notin\Omega_i$. From the supposed conditions (ii) and (iii), $s_r(t)$ is independent to noise vector $\boldsymbol{e}(t)$ and the mean of $\boldsymbol{e}(t)$ is $\boldsymbol{0}$, i.e., $E[\boldsymbol{e}^{\mathrm{T}}(t)]=\boldsymbol{0}^{\mathrm{T}}$, where $E(\cdot)$ denotes mathematical expectation.[1] So $E[\boldsymbol{e}^{\mathrm{T}}(t)\cdot s_r(t)]=E[\boldsymbol{e}^{\mathrm{T}}(t)]\cdot E[s_r(t)]=0$. So we have

$$\begin{aligned}\lim_{T\to+\infty}\frac{1}{T}\sum_{t\in\Omega_i}\boldsymbol{a}_r\cdot\boldsymbol{e}^{\mathrm{T}}(t)\cdot s_r(t)&=\boldsymbol{a}_r\cdot\lim_{T\to+\infty}\frac{1}{T}\sum_{t=1}^{T}\boldsymbol{e}^{\mathrm{T}}(t)\cdot s_r(t)\\&=\boldsymbol{a}_r\cdot E[\boldsymbol{e}^{\mathrm{T}}(t)\cdot s_r(t)]=\boldsymbol{0}.\end{aligned} \tag{18}$$

From Eqs. (17) and (18), (15) can be simplified as

$$\lim_{T\to+\infty}\frac{1}{T}\sum_{t\in\Omega_i}\tilde{\boldsymbol{x}}(t)\cdot\tilde{\boldsymbol{x}}^{\mathrm{T}}(t)=\boldsymbol{a}_r\boldsymbol{a}_r^{\mathrm{T}}\cdot\sigma_s^2+\lim_{T\to+\infty}\frac{1}{T}\sum_{t\in\Omega_i}[\boldsymbol{e}(t)\cdot\boldsymbol{e}^{\mathrm{T}}(t)]. \tag{19}$$

From the supposed condition (ii), we have

$$\lim_{T\to+\infty}\frac{1}{T}\sum_{t=1}^{T}\boldsymbol{e}(t)\cdot\boldsymbol{e}^{\mathrm{T}}(t)=E[\boldsymbol{e}(t)\cdot\boldsymbol{e}^{\mathrm{T}}(t)]=\boldsymbol{I}_{m\times m}\cdot\sigma_e^2. \tag{20}$$

[1] For simplicity, the mathematical expectation $E(\cdot)$ of stochastic process is calculated as the arithmetic mean of the samples in this paper. For example, $E[e(t)\cdot e^{\mathrm{T}}(t)]=\lim_{T\to+\infty}(1/T)\sum_{t=1}^{T}\boldsymbol{e}(t)\cdot\boldsymbol{e}^{\mathrm{T}}(t)$.

From (20), we obtain

$$\begin{aligned}0&\leqslant\frac{\boldsymbol{u}^{\mathrm{T}}\cdot\lim_{T\to+\infty}[\sum_{t\in\Omega_i}\boldsymbol{e}(t)\cdot\boldsymbol{e}^{\mathrm{T}}(t)/T]\cdot\boldsymbol{u}}{\boldsymbol{u}^{\mathrm{T}}\boldsymbol{u}}\\&\leqslant\frac{\boldsymbol{u}^{\mathrm{T}}\cdot\lim_{T\to+\infty}[\sum_{t=1}^{T}\boldsymbol{e}(t)\cdot\boldsymbol{e}^{\mathrm{T}}(t)/T]\cdot\boldsymbol{u}}{\boldsymbol{u}^{\mathrm{T}}\boldsymbol{u}}\\&=\sigma_e^2.\end{aligned} \tag{21}$$

From (19), (12) can lead to the following inequalities:

$$\begin{aligned}f_i&=\max_{\boldsymbol{u}\neq0}\frac{\boldsymbol{u}^{\mathrm{T}}\cdot\lim_{T\to\infty}[\sum_{t\in\Omega_i}\tilde{\boldsymbol{x}}(t)\cdot\tilde{\boldsymbol{x}}^{\mathrm{T}}(t)/T]\cdot\boldsymbol{u}}{\boldsymbol{u}^{\mathrm{T}}\boldsymbol{u}}\\&=\max_{\boldsymbol{u}\neq0}\frac{\boldsymbol{u}^{\mathrm{T}}\lim_{T\to\infty}\{\boldsymbol{a}_r\boldsymbol{a}_r^{\mathrm{T}}\cdot\sigma_s^2+[\sum_{t\in\Omega_i}\boldsymbol{e}(t)\cdot\boldsymbol{e}^{\mathrm{T}}(t)]/T\}\boldsymbol{u}}{\boldsymbol{u}^{\mathrm{T}}\boldsymbol{u}}\\&\leqslant\max_{\boldsymbol{u}\neq0}\left\{\sigma_s^2\cdot\frac{\boldsymbol{u}^{\mathrm{T}}\cdot[\boldsymbol{a}_r\boldsymbol{a}_r^{\mathrm{T}}]\cdot\boldsymbol{u}}{\boldsymbol{u}^{\mathrm{T}}\boldsymbol{u}}+\frac{\boldsymbol{u}^{\mathrm{T}}\cdot\lim_{T\to\infty}[\sum_{t=1}^{T}\boldsymbol{e}(t)\cdot\boldsymbol{e}^{\mathrm{T}}(t)/T]\cdot\boldsymbol{u}}{\boldsymbol{u}^{\mathrm{T}}\boldsymbol{u}}\right\}.\end{aligned} \tag{22}$$

Since $\boldsymbol{a}_r\boldsymbol{a}_r^{\mathrm{T}}$ is a rank-1 matrix and $\|\boldsymbol{a}_r\|_2=1$ (see the assumption of model (1) in the Introduction section),

$$\boldsymbol{a}_r=\arg\max_{\boldsymbol{u}\neq0}\frac{\boldsymbol{u}^{\mathrm{T}}\cdot[\boldsymbol{a}_r\boldsymbol{a}_r^{\mathrm{T}}]\cdot\boldsymbol{u}}{\boldsymbol{u}^{\mathrm{T}}\boldsymbol{u}}$$

and

$$\max_{\boldsymbol{u}\neq0}\frac{\boldsymbol{u}^{\mathrm{T}}\cdot[\boldsymbol{a}_r\boldsymbol{a}_r^{\mathrm{T}}]\cdot\boldsymbol{u}}{\boldsymbol{u}^{\mathrm{T}}\boldsymbol{u}}=1$$

from [37, Chapter 4]. So, (22) can be re-written as

$$f_i\leqslant\sigma_s^2+\max_{\boldsymbol{u}\neq0}\frac{\boldsymbol{u}^{\mathrm{T}}\cdot\lim_{T\to\infty}[\sum_{t=1}^{T}\boldsymbol{e}(t)\cdot\boldsymbol{e}^{\mathrm{T}}(t)/T]\cdot\boldsymbol{u}}{\boldsymbol{u}^{\mathrm{T}}\boldsymbol{u}}. \tag{23}$$

From Eqs. (21) and (23), we have

$$\begin{aligned}\sigma_s^2&\leqslant f_i\\&\leqslant\sigma_s^2+\max_{\boldsymbol{u}\neq0}\frac{\boldsymbol{u}^{\mathrm{T}}\cdot\lim_{T\to\infty}[\sum_{t=1}^{T}\boldsymbol{e}(t)\cdot\boldsymbol{e}^{\mathrm{T}}(t)/T]\cdot\boldsymbol{u}}{\boldsymbol{u}^{\mathrm{T}}\boldsymbol{u}}\\&=\sigma_s^2+\sigma_e^2.\end{aligned}$$

(2) Proof of the second part: if $L(\boldsymbol{l}_j)$ is a spurious estimation, i.e., $\boldsymbol{l}_j\notin\{\boldsymbol{a}_1,\ldots,\boldsymbol{a}_n\}$, then $f_j\leqslant\sigma_e^2$.

Since $L(\boldsymbol{l}_j)$ is a spurious estimation ($\boldsymbol{l}_j\notin\{\boldsymbol{a}_1,\ldots,\boldsymbol{a}_n\}$), so we have $s_\tau(t)=0,\ \tau=1,\ldots,n,\ t\in\Omega_j$. Then

$$\tilde{\boldsymbol{x}}(t)=\sum_{\tau=1}^{n}\boldsymbol{a}_\tau\cdot s_\tau(t)+\boldsymbol{e}(t)=\boldsymbol{e}(t),\quad t\in\Omega_j \text{ and } \boldsymbol{l}_j\notin\{\boldsymbol{a}_1,\ldots,\boldsymbol{a}_n\}. \tag{24}$$

Therefore,

$$\begin{aligned}f_j=\lambda_{1j}&=\max_{\boldsymbol{u}\neq0}\frac{\boldsymbol{u}^{\mathrm{T}}\cdot\lim_{T\to+\infty}[\tilde{\boldsymbol{X}}_j\cdot(\tilde{\boldsymbol{X}}_j)^{\mathrm{T}}/T]\cdot\boldsymbol{u}}{\boldsymbol{u}^{\mathrm{T}}\boldsymbol{u}}\\&=\max_{\boldsymbol{u}\neq0}\frac{\boldsymbol{u}^{\mathrm{T}}\cdot\lim_{T\to+\infty}[\sum_{t\in\Omega_j}\tilde{\boldsymbol{x}}(t)\cdot\tilde{\boldsymbol{x}}^{\mathrm{T}}(t)/T]\cdot\boldsymbol{u}}{\boldsymbol{u}^{\mathrm{T}}\boldsymbol{u}}\\&=\max_{\boldsymbol{u}\neq0}\frac{\boldsymbol{u}^{\mathrm{T}}\cdot\lim_{T\to+\infty}[\sum_{t\in\Omega_j}\boldsymbol{e}(t)\cdot\boldsymbol{e}^{\mathrm{T}}(t)/T]\cdot\boldsymbol{u}}{\boldsymbol{u}^{\mathrm{T}}\boldsymbol{u}}\\&\leqslant\max_{\boldsymbol{u}\neq0}\frac{\boldsymbol{u}^{\mathrm{T}}\cdot\lim_{T\to+\infty}[\sum_{t=1}^{T}\boldsymbol{e}(t)\cdot\boldsymbol{e}^{\mathrm{T}}(t)/T]\cdot\boldsymbol{u}}{\boldsymbol{u}^{\mathrm{T}}\boldsymbol{u}}.\end{aligned} \tag{25}$$

Combining Eqs. (20) and (25), we derive

$$f_j \leqslant \sigma_e^2.$$

References

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

[2] Z. He, S. Xie, S. Ding, A. Cichocki, Convolutive blind source separation in frequency domain based on sparse representation, IEEE Transactions on Audio, Speech and Language Processing 15 (5) (2007) 1551–1563.

[3] Z. He, S. Xie, L. Zhang, A. Cichocki, A note on Lewicki–Sejnowski gradient for learning overcomplete representation, Neural Computation 20 (3) (2008) 636–643.

[4] B.A. Olshausen, D.J. Field, Emergence of simple-cell receptive field properties by learning a sparse code for natural images, Nature 381 (6583) (1996) 607–609.

[5] M. Zibulevsky, B.A. Pearlmutter, Blind source separation by sparse decomposition in a signal dictionary, Neural Computation 13 (2001) 863–882.

[6] A. Cichocki, S. Amari, Adaptive Blind Signal and Image Processing: Learning Algorithms and Applications, Wiley, New York, 2003.

[7] Y.Q. Li, A. Cichocki, S. Amari, Blind estimation of channel parameters and source components for EEG signals: a sparse factorization approach, IEEE Transactions on Neural Networks 17 (2) (2006) 419–431.

[8] I.F. Gorodnitsky, B.D. Rao, Sparse signal reconstruction from limited data using focuss: a recursive weighted minimum norm algorithm, IEEE Transactions on Signal Processing 45 (3) (1997) 600–616.

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

[10] P. Bofill, M. Zibulevsky, Underdetermined blind source separation using sparse representations, Signal Processing 81 (2001) 2353–2362.

[11] T.W. Lee, M.S. Lewicki, M. Girolami, T.J. Sejnowski, Blind source separation of more sources than mixtures using overcomplete representation, IEEE Signal Processing Letter 6 (4) (1999) 87–90.

[12] M.S. Lewicki, T.J. Sejnowski, Learning overcomplete representations, Neural Computation 12 (2) (2000) 337–365.

[13] M. Girolami, A variational method for learning sparse and overcomplete representations, Neural Computation 13 (11) (2001) 2517–2532.

[14] Y.Q. Li, S. Amari, A. Cichocki, D.W.C. Ho, S.L. Xie, Underdetermined blind source separation based on sparse representation, IEEE Transactions on Signal Processing 54 (2) (2006) 423–437.

[15] Ö. Yilmaz, S. Rickard, Blind separation of speech mixtures via time–frequency masking, IEEE Transactions on Signal Processing 52 (7) (2004) 1830–1847.

[16] F. Abrard, Y. Deville, A time–frequency blind signal separation method applicable to underdetermined mixtures of dependent sources, Signal Processing 85 (7) (2005) 1389–1403.

[17] A. Jourjine, S. Rickard, Ö. Yilmaz, Blind separation of disjoint orthogonal signals: demixing n sources from 2 mixtures, in: Proceeding of IEEE Conference on Acoustics, Speech, and Signal Processing (ICASSP2000), vol. 5, Istanbul, Turkey, 2000, pp. 2985–2988.

[18] S. Chen, D.L. Donoho, M.A. Saunders, Atomic decomposition by basis pursuit, SIAM Journal on Scientific Computing 20 (1) (1998) 33–61.

[19] D.L. Donoho, M. Elad, Maximal sparsity representation via ℓ_1 minimization, Proceedings of the National Academy of Sciences 100 (2003) 2197–2202.

[20] I. Takigawa, M. Kudo, J. Toyama, Performance analysis of minimum ℓ_1-norm solutions for underdetermined source separation, IEEE Transactions on Signal Processing 52 (3) (2004) 582–591.

[21] S.F. Cotter, B.D. Rao, K. Engan, K. Kreutz-Delgado, Sparse solutions to linear inverse problems with multiple measurement vectors, IEEE Transactions on Signal Processing 53 (7) (2005) 2477–2488.

[22] K. Kreutz-Delgado, J.F. Murry, B.D. Rao, et al., Dictionary learning algorithms for sparse representation, Neural Computation 15 (2003) 349–396.

[23] J.A. Hartigan, M.A. Wong, A K-means clustering algorithm, Applied Statistics 28 (1) (1979) 100–108.

[24] F. Theis, C. Puntonet, E. Lang, Median-based clustering for underdetermined blind signal processing, IEEE Signal Processing Letters 13 (2) (2006) 96–99.

[25] F. Theis, A. Jung, C. Puntonet, E. Lang, Linear geometric ICA: fundamentals and algorithms, Neural Computation 15 (2003) 419–439.

[26] P.D. O'Grady, B.A. Pearlmutter, Soft-LOST: EM on a mixture of oriented lines, in: International Conference on Independent Component Analysis 2004, 2004, pp. 428–435.

[27] Z.S. He, S.L. Xie, Y.L. Fu, Sparse representation and blind source separation of ill-posed mixtures, Science in China Series F-Information Sciences 49 (5) (2006) 639–652.

[28] S. Arberet, R. Gribonval, F. Bimbot, A robust method to count and locate audio sources in a stereophonic linear instantaneous mixture, in: Proceedings of ICA 2006, Charleston SC, USA, 2006, pp. 536–543.

[29] Z.S. He, A. Cichocki, K-EVD clustering and its applications to sparse component analysis, in: Proceedings of ICA 2006, Charleston SC, USA, 2006, pp. 90–97.

[30] M. Babaie-Zadeh, C. Jutten, A. Mansour, Sparse ICA via cluster-wise PCA, Neurocomputing 69 (13–15) (2006) 1458–1466.

[31] A. Dempster, N. Laird, D. Rubin, Maximum likelihood from incomplete data via the EM algorithm, Journal of the Royal Statistical Society, Series B 39 (1) (1977) 1–38.

[32] G. McLachlan, T. Krishnan, The EM Algorithm and Extensions, in: Wiley Series in Probability and Statistics, Wiley, New York, 1997.

[33] A. Cichocki, R. Zdunek, Multilayer nonnegative matrix factorization, Electronics Letters 42 (16) (2006) 947–948.

[34] D. Neill, A. Moore, et al., Detecting significant multidimensional spatial clusters, Advances in Neural Information Processing Systems 17 (2005) 969–976.

[35] A. Cichocki, D. Erdogmus, MLSP 2005 competition: large scale, ill-conditioned blind source separation with limited number of samples, 2005, website: ⟨http://mlsp2005.conwiz.dk/fileadmin/Cichocki.pdf⟩.

[36] J.A. Tropp, Greed is good: algorithmic results for sparse approximation, IEEE Transactions on Information Theory 50 (10) (2004) 2231–2242.

[37] R.A. Horn, C.R. Johnson, Matrix Analysis, Cambridge University Press, Cambridge, 1999.