A Fast Method for Sparse Component Analysis Based on Iterative Detection-Estimation

Arash Ali Amini*, Massoud Babaie-Zadeh[1*] and Christian Jutten†

*Electrical engineering department, Sharif University of Technology, Tehran, Iran.
†Laboratoire des Images et des Signaux (LIS), Institut National Polytechnique de Grenoble (INPG), Grenoble, France. email

Abstract. We introduce a new iterative algorithm for Sparse Component Analysis (SCA). The algorithm, which we call Iterative Detection-Estimation (IDE), is essentially a method to £nd suf£ciently sparse solutions of underdetermined linear systems of equations. In the SCA context, this solves the source separation part of the problem. Each iteration of IDE consists of two steps. In the detection step, starting with a previously known estimate of the sparse solution vector, we detect which components of the solution are (possibly) active, i.e., having a considerable value. Then, in the estimation step, we compute the new estimate by £nding a solution of the system which is the closest to the subspace speci£ed by the detection step. This is called projection into the activity subspace. We will compare the solution obtained by the proposed algorithm against the minimum 1-norm solution obtained by Linear Programming (LP). It is shown by experiment that, with proper choice of parameters, the proposed algorithm is about two orders of magnitude faster than state-of-the-art interior-point LP solvers, while providing the same (or better) accuracy. The algorithm may then be considered as an alternative to the LP approach for large-scale problems.

Keywords: sparse component analysis, atomic decomposition, sparse representation, overcomplete signal representation, sparse source separation, blind source separation.

INTRODUCTION

Obtaining sparse solutions of underdetermined linear systems of equations is of fundamental importance in signal processing and statistics. Despite recent theoretical developments [1, 2], the computational cost of the methods has remained as the main restriction. In this article, we will propose an approach which provides a considerable reduction in complexity. To introduce the problem in more details, we will use the context of SCA. The discussions, however, may be easily followed in other contexts in which the problem arises. For example, it can also be used to £nd 'sparse decomposition' of a signal in an overcomplete dictionary, which is the goal of the so-called 'atomic decomposition' [1].

SCA can be viewed as a method to achieve separation of sparse sources. The general Blind Source Separation (BSS) problem is to recover m unknown (statistically independent) sources from n observed mixtures of them, where little or no information is available about the sources (except their statistical independence) and the mixing system. In linear instantaneous (noise-free) model, it is assumed that $\mathbf{x}(t) = \mathbf{A}\mathbf{s}(t)$ in which $\mathbf{x}(t)$ and $\mathbf{s}(t)$ are the $n \times 1$ and $m \times 1$ vectors of sources and mixtures and $\mathbf{A}$ is the $n \times m$

[1] This work has been partially funded by French Embassy in Tehran, and by Center for International Research and Collaboration (ISMO).

mixing matrix. The goal of BSS is then to £nd $\mathbf{s}(t)$ only by observing $\mathbf{x}(t)$. The general BSS problem is not easy for the case $m > n$. However, if the sources are sparse (i.e., not a totally blind situation), then the problem can be solved [2, 3] in two steps: (1) estimating the mixing matrix, (2) with $\mathbf{A}$ being known, estimating the sources. For sparse sources, the £rst step may be done by a simple clustering [2, 3]. However, for the case $m > n$ (that is, where there are fewer sensors than sources), determining $\mathbf{A}$ is not equivalent to source separation: for each instant of time (t), an underdetermined linear system of equations ($\mathbf{x}(t) = \mathbf{A}\mathbf{s}(t)$) should be solved under the sparsity assumption [2, 3, 4, 5].

To obtain the sparsest solution we may search for a solution of $\mathbf{x} = \mathbf{A}\mathbf{s}$ having minimal l^0 norm, i.e., minimum number of nonzero components. This, however, requires combinatorial search which quickly becomes intractable as the dimension increases. The l^0 norm is also sensitive to noise. However, it has been shown £rst empirically [6] and then theoretically [1, 7, 8] that the minimal l^0 norm solution is (almost always) the same as minimal l^1 norm solution (that is a solution which minimizes $\sum_{i=1}^{m} |s_i|$). This minimization can be done using the Linear Programming (LP) techniques. By using fast LP algorithms, speci£cally interior-point LP solvers, large-scale problems with thousands of sources and mixtures will become tractable.

In this article, we will introduce a new method for sparse decomposition which we call Iterative Detection-Estimation (IDE). The method performs typically two order of magnitudes faster than interior-point LP and, with the proper choice of parameters, produces results as accurate as those of LP.

MOTIVATION FOR ITERATIVE DETECTION-ESTIMATION

The idea of the IDE algorithm is as follows. At each iteration: (1) Starting with a previous estimate (or initial guess) of the source vector, $\hat{\mathbf{s}}$, £rst (roughly) detect which sources are 'active' and (2) Having the indices of 'active' sources, obtain a new estimate of $\mathbf{s}$ by £nding a solution of $\mathbf{x} = \mathbf{A}\mathbf{s}$ whose active indices better coincide with those predicted by the detection step. This two step iteration continues until no further improvement is possible (i.e. until a given level of accuracy is reached). The term 'active' is used to refer to sources having 'considerable values'.

To provide motivation for the detection step, we use a Mixtures of Gaussians (MoG) to model sparsity. More speci£cally, let π_0 be the probability of s_i being *inactive* ($\pi_0 \lessapprox 1$ to insure sparsity) . Then, the value of an inactive source is modeled by $\mathcal{N}(0, \sigma_0^2)$, and an active source by $\mathcal{N}(0, \sigma_1^2)$, where $\sigma_0^2 \ll \sigma_1^2$. The probability π_0 is not used in the development of the algorithm, but is a measure of sparsity in the experimental results.

The test for activity is carried for one source at time. To be speci£c, consider the problem of detecting the activity of s_1. We may formulate the problem in terms of a binary hypothesis testing [9]. The vector $\mathbf{x} = s_1\mathbf{a}_1 + \sum_{i=2}^{m} s_i\mathbf{a}_i$ is observed, and we are to decide which of the following two hypotheses has occurred :

$$\begin{aligned} H_0 : &\quad s_1 \sim \mathcal{N}(0, \sigma_0^2) \\ H_1 : &\quad s_1 \sim \mathcal{N}(0, \sigma_1^2). \end{aligned}$$

It may be justi£ed that the scalar quantity $\mathbf{a}_1^T\mathbf{x} \triangleq t$ has the same information as the

vector $\mathbf{x}$, regarding the discrimination of the two hypotheses, i.e., t is the suf£cient statistics [9]. De£ning $\mu \triangleq \sum_{i=2}^{m} s_i \mathbf{a}_1^T \mathbf{a}_i$ we will have $t = \mathbf{a}_1^T \mathbf{x} = s_1 + \mu$. We consider $\{s_i\}_{i=2}^m$ to be unknown (nuisance) parameters. Also, no prior probabilities are assigned to the hypotheses. In other words, the detection problem will be approached in a Neyman-Pearson (NP) framework. The equivalent test in terms of the suf£cient statistics, t, may now be stated as $H_i : t \sim \mathcal{N}(\mu, \sigma_i^2)$ for $i = 0, 1$.

The optimal test (in the NP sense) is the likelihood-ratio test. Its critical region can be simpli£ed to $|t - \mu| > \tau'$, with known parameters being absorbed into the threshold. Since the critical region depends on the value of μ (an unknown parameter), there is no Uniformly Most Powerful (UMP) test. That is, no single test is optimal for all the values of μ. In fact, the performance of the test with the above critical region is highly dependent on the value of μ. Recalling the de£nition of μ, it is seen that implementing the optimal test for activity of s_1 requires the knowledge of all the other sources. Since we don't know the actual values, we will replace them with their estimates (obtained from a previous iteration or from an initial guess). The resulting sub-optimal test is

$$\left| \mathbf{a}_1^T \mathbf{x} - \sum_{i=2}^{m} \hat{s}_i \, \mathbf{a}_1^T \mathbf{a}_i \right| > \epsilon.$$

This test is conducted separately for each of the n sources (with obvious modi£cations). After determining the activity status, we try to obtain (i.e., £nd a new estimate of) the actual values of the sources. For the sake of discussion, assume that the £rst k sources, $\{s_i\}_{i=1}^k$, have been found to be *inactive* by the detection step. The approximate source vector is then obtained as the solution to the following optimization problem

$$\text{minimize} \ \sum_{i=1}^{k} s_i^2 \ \text{ subject to } \ \mathbf{x} = \mathbf{A}\mathbf{s}. \tag{1}$$

The solution set of $\mathbf{x} = \mathbf{A}\mathbf{s}$, in variable $\mathbf{s}$, de£nes an af£ne set in $\mathbb{R}^m$. The cost function in (1) may be viewed as the distance between this af£ne set and the subspace generated by the last $m - k$ standard unit vectors of $\mathbb{R}^m$, i.e., $\{\mathbf{e}_i\}_{i=k+1}^m$. This subspace, which we call activity subspace, is where the sparse solution of the system is believed to lie. By minimizing the cost function of (1) we suppress those sources believed to be negligible while leaving the active sources to change freely so that the overall vector is still a solution of $\mathbf{x} = \mathbf{A}\mathbf{s}$. This may also be viewed as a form of *projection into the activity subspace*. It is intuitively plausible that if the actual active sources are among those detected to be active, and if (1) has a unique solution, this solution coincides with the actual (sparse) source vector. In the next section we will provide a solution for (1).

For now, we summarize an iteration of the IDE algorithm. To simplify notation, de£ne

$$g_i(\mathbf{x}, \hat{\mathbf{s}}) \triangleq \left| \mathbf{a}_i^T \mathbf{x} - \sum_{j \neq i} \hat{s}_j \, \mathbf{a}_i^T \mathbf{a}_j \right|$$

for $1 \leq i \leq m$. This may be called the activity function of the i-th source given the estimate $\hat{\mathbf{s}}$. Also let $\hat{\mathbf{s}}^{(k)}$ denotes the estimate of the source vector found at the k-th iteration. The DE-iteration may be summarized as

Detection Step : Find indices of the inactive sources

$$\mathcal{I} = \{1 \le i \le m : g_i(\mathbf{x}, \hat{\mathbf{s}}^{(k)}) < \epsilon^{(k+1)}\}$$

Projection Step : Find the new estimate as

$$\hat{\mathbf{s}}^{(k+1)} = \arg\min_{\mathbf{s}} \sum_{i \in \mathcal{I}} s_i^2 \ \ (\text{s.t. } \mathbf{x} = \mathbf{A}\mathbf{s})$$

Note that we have allowed the threshold ($\epsilon^{(k+1)}$) to vary across iterations. The thresholds are usually chosen to form a decreasing sequence. We will use $\hat{\mathbf{s}}^{(0)} = \mathbf{0}$ as the initial estimate when no (other) prior information is available, since the most probable value for a source is zero.

SOLUTION OF THE ESTIMATION STEP

The optimization problem (1) is a special case of Quadratic Programming (QP) which has been extensively studied in the literature [10]. In fact, the cost function in (1) may be stated as the quadratic form $\mathbf{s}^T\mathbf{H}\mathbf{s}$ with $\mathbf{H} = \left(\begin{smallmatrix} \mathbf{I}_k & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{smallmatrix}\right)$ where $\mathbf{I}_k$ is the $k \times k$ identity matrix.

Among many numerically ef£cient approaches available [10, 11], here we consider the direct solution of the so-called Karush-Kuhn-Tucker (KKT) system of equations as a necessary condition for optimality [10], i.e., the optimal solution should satisfy

$$\begin{pmatrix} \mathbf{H} & \mathbf{A}^T \\ \mathbf{A} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{s} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{x} \end{pmatrix}$$

where $\boldsymbol{\lambda}$ is the $n \times 1$ vector of Lagrange multipliers. For solving this system of equations without calculating the inverse of an $(m+n) \times (m+n)$ matrix, let $p \triangleq m - k$ be the number of sources detected active (it is assumed that $p < n$) and partition vectors and matrices into 'inactive/active' parts

$$\begin{pmatrix} \mathbf{I}_k & \mathbf{0} & \mathbf{A}_\iota^T \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_\alpha^T \\ \mathbf{A}_\iota & \mathbf{A}_\alpha & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{s}_\iota \\ \mathbf{s}_\alpha \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{x} \end{pmatrix} \quad \text{or} \quad \begin{cases} \mathbf{s}_\iota + \mathbf{A}_\iota^T\boldsymbol{\lambda} = \mathbf{0} \\ \mathbf{A}_\alpha^T\boldsymbol{\lambda} = \mathbf{0} \\ \mathbf{A}_\iota\mathbf{s}_\iota + \mathbf{A}_\alpha\mathbf{s}_\alpha = \mathbf{x} \end{cases} \tag{2}$$

Calculating $\mathbf{s}_\iota$ from the £rst equation and putting it into the two others, we obtain:

$$\begin{pmatrix} -\mathbf{A}_\iota\mathbf{A}_\iota^T & \mathbf{A}_\alpha \\ \mathbf{A}_\alpha^T & \mathbf{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{\lambda} \\ \mathbf{s}_\alpha \end{pmatrix} = \begin{pmatrix} \mathbf{x} \\ \mathbf{0} \end{pmatrix} \quad \text{or} \quad \begin{cases} -\mathbf{A}_\iota\mathbf{A}_\iota^T\boldsymbol{\lambda} + \mathbf{A}_\alpha\mathbf{s}_\alpha = \mathbf{x} \\ \mathbf{A}_\alpha^T\boldsymbol{\lambda} = \mathbf{0} \end{cases} \tag{3}$$

The second equation shows that $\boldsymbol{\lambda}$ is in the null space of $\mathbf{A}_\alpha^T$. Let $\mathbf{Z}$ be an $n \times (n-p)$ matrix whoes columns provide a basis for this null space (in MATLAB, `Z=null(Aa')`). Letting $\boldsymbol{\lambda} = \mathbf{Z}\boldsymbol{\beta}$ (where $\boldsymbol{\beta}$ is an unknown $(n-p) \times 1$ vector), and putting it in the £rst equation of (3), we have $-\mathbf{A}_\iota\mathbf{A}_\iota^T\mathbf{Z}\boldsymbol{\beta} + \mathbf{A}_\alpha\mathbf{s}_\alpha = \mathbf{x}$, which is a system of n equations and

n unknowns ($\boldsymbol{\beta}$ and $\mathbf{s}_\alpha$), and can be solved by inverting an $n \times n$ matrix. However, multiplying both sides of it by $\mathbf{Z}^T$ and noting $\mathbf{Z}^T\mathbf{A}_\alpha = (\mathbf{A}_\alpha^T\mathbf{Z})^T = \mathbf{0}$, gives a closed form formula for $\boldsymbol{\beta}$ and hence $\mathbf{s}_\iota$. Then, multiplying both sides of the third equation of (2) by $\mathbf{A}_\alpha^T$ results in a closed form fomula for $\mathbf{s}_\alpha$. Finally:

$$\begin{cases} \mathbf{s}_\iota = \mathbf{B}_\iota^T(\mathbf{B}_\iota\mathbf{B}_\iota^T)^{-1}\mathbf{Z}^T\mathbf{x} \\ \mathbf{s}_\alpha = (\mathbf{A}_\alpha^T\mathbf{A}_\alpha)^{-1}\mathbf{A}_\alpha^T(\mathbf{x} - \mathbf{A}_\iota\mathbf{s}_\iota) \end{cases}$$

where $\mathbf{B}_\iota \triangleq \mathbf{Z}^T\mathbf{A}_\iota$. For the case $p \leq \min\{n, m-n\}$, we can also write [12]:

$$\begin{cases} \mathbf{s}_\alpha = (\mathbf{A}_\alpha^T\mathbf{P}\mathbf{A}_\alpha)^{-1}\mathbf{A}_\alpha^T\mathbf{P}\mathbf{x} \\ \mathbf{s}_\iota = \mathbf{A}_\iota^T\mathbf{P}\left(\mathbf{x} - \mathbf{A}_\alpha\mathbf{s}_\alpha\right) \end{cases} \tag{4}$$

where $\mathbf{P} \triangleq (\mathbf{A}_\iota\mathbf{A}_\iota^T)^{-1}$. In the experiments of this article, we use (4).

COMMENTS ON THE CHOICE OF THRESHOLDS

As mentioned earlier, if *the actual active sources are among those detected to be active* and if *the solution to (1) is unique*, then an IDE iteration will yield a close estimate. This is best observed in the extreme case where actual inactive sources are (exactly) zero, in which case the actual source vector will be the unique solution to (1), with a cost function value of zero. The observation suggests that within the limits of the two assumptions only a rough detection is suf£cient. In other words, there are implicit upper/lower bounds on the threshold. To ensure the £rst assumption, the threshold value should be taken low enough so that the actual active sources are detected correctly. It should however be high enough so that at least $m - n$ number of sources are detected to be inactive. This ensures the second assumption. These two bounds are not tight for most problems. In fact, it is easy to £nd proper threshold values for families of problems, as shown by the experimental results of the following section. A family of problems is determined by the n/m ratio and the sparsity level π_0.

Another notable observation is that when $\mathbf{s}$ obtained in a previous iteration is an exact solution of $\mathbf{x} = \mathbf{A}\mathbf{s}$, then the activity function is simply the absolute value of the respective source, i.e., $g_i(\mathbf{x}, \mathbf{s}) = |s_i|$. If we have bounds on the absolute values of the sources, it is natural to choose a threshold satisfying those bounds.

EXPERIMENTAL RESULTS

We will use the Gaussian mixture model introduced earlier, with $\sigma_0/\sigma_1 = 0.01$, to create (original) source vectors used in the £rst experiment. The model is fairly general and lets us control the sparsity level through the choice of π_0. We normalize the source vector to unit l^∞ norm. This restricts the absolute values of the sources, and consequently the thresholds, to $[0, 1]$ interval. The $n \times m$ mixing matrix $\mathbf{A}$ is taken to be random. It will be constructed column-wise by drawing m, $n \times 1$ vectors from the uniform distribution

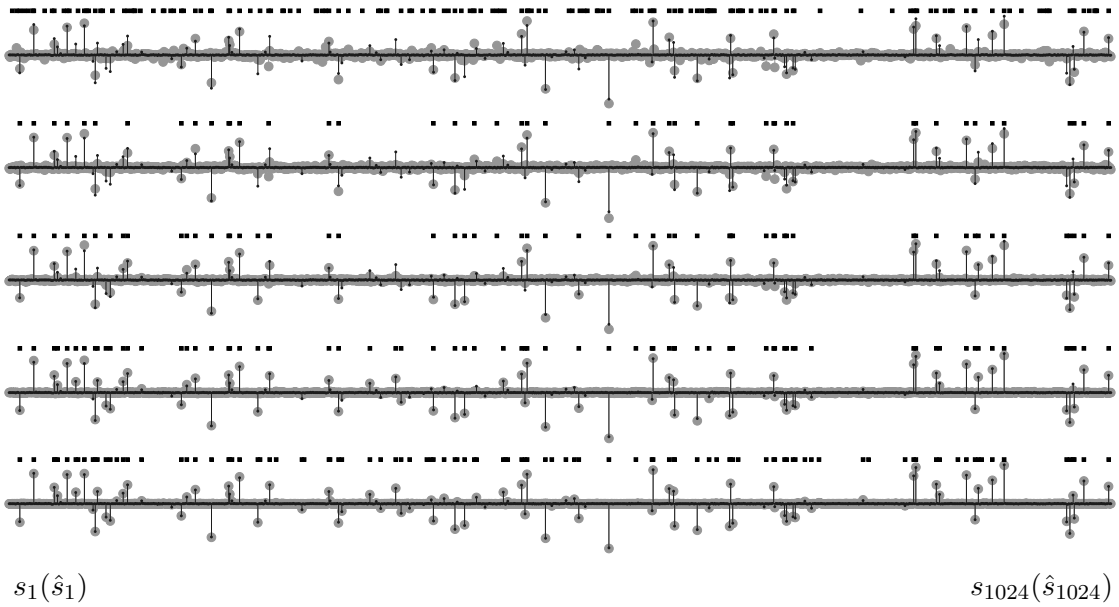

FIGURE 1. The evolution of IDE toward the solution (Experiment 1) : $m = 1024$, $n = \lfloor 0.4m \rfloor = 409$, $\pi_0 = 0.9$. Each plot shows the original source vector (black) and its estimate obtained after an iteration (gray). The sources detected to be active are marked with a black square above each plot. Six iterations were used with threshold values (form top to bottom) $\epsilon = 0.3, 0.2, 0.1, 0.05, 0.02, 0.01$. The top plot corresponds to the £rst iteration.

on the unit sphere in $\mathbb{R}^n$. For the purpose of performance evaluation, Mean Square Error (MSE) or Signal-to-Noise Ratio (SNR) will be used to judge accuracy and the CPU time (in sec.) will be used as a measure of complexity. The MSE is de£ned as $(1/m)\|\mathbf{s} - \hat{\mathbf{s}}\|_2^2$. The CPU time is measured on a P4 PC under the MATLAB environment. The IDE performance will be compared against that of LP. The LP algorithm used is the interior-point solver implemented in the MATLAB's optimization toolbox called LIPSOL.

Fig. 1 shows the evolution of the IDE algorithm toward the £nal solution. We have taken $m = 1024$ and $n = \lfloor 0.4m \rfloor = 409$ which, at sparsity level $\pi_0 = 0.9$, produces a relatively challenging problem. Six IDE iterations with threshold values $\epsilon = 0.3, 0.2, 0.1, 0.05, 0.02, 0.01$ were used to obtain an MSE comparable to that obtained by LP. The plots show the estimated source vectors, $\hat{\mathbf{s}}$, obtained at each iteration along with the original $\mathbf{s}$. Also, the sources detected to be active are marked with a square (above them). Note how the algorithm corrects the initial guess for active sources and produces progressively better approximations. The threshold (ϵ), the number of sources detected active (detected #act), the CPU time taken in seconds (ΔT), and the MSE obtained at each iteration are summarized in Table 1. Comparison of the total simulation times clearly illustrates the reduced complexity of IDE relative to that of LP.

The data used in Fig. 1 and Table 1 are for a single (typical) realization of the experiment. The set of thresholds used, however, is found to produce similar results (i.e., MSE of the order of 10^{-5}) for other realizations of the problem as long as we have $n/m \approx 0.4$ and $\pi_0 \approx 0.9$. For easier problems, where n/m is much closer to unity or

TABLE 1. Progress of the IDE for a problem with $m = 1024$, $n = \lfloor 0.4m \rfloor = 409$.

itr. #	ϵ	detected #act	ΔT	MSE
1	0.3	158	0.377	$3.36e{-3}$
2	0.2	47	0.297	$2.22e{-3}$
3	0.1	58	0.292	$9.66e{-4}$
4	0.05	73	0.293	$2.21e{-4}$
5	0.02	105	0.310	$4.31e{-5}$
6	0.01	107	0.315	$1.39e{-5}$

algorithm	total time	MSE	SNR (dB)
IDE (6 itr.)	1.88 seconds	$1.39e{-5}$	30.28
LP interior	123 seconds	$3.51e{-5}$	26.25

sparsity (π_0) is much higher, even fewer iterations may be used. For example, at the same sparsity level $\pi_0 = 0.9$, but with $n/m \approx 0.5$, the threshold sequence $\epsilon = 0.3, 0.05, 0.02$ is found to be suf£cient to reach MSE of 10^{-5}. For $n/m \approx 0.6$, even the two-iteration IDE with $\epsilon = 0.3, 0.05$ yields the desired MSE.

On the other hand, a relatively long threshold sequence may always be used when problem dif£culty is unknown. For example, the sequence $\epsilon = 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1, 0.07, 0.05, 0.02$ is found experimentally to produce slightly more accurate results than those of LP over a wide range of sparsity. To show this, we use a different type of sparse source. More speci£cally, given the number of active sources, $\#act$, a source vector is generated with exactly $\#act$ of its components randomly selected to be unity. The rest of the components, which represent inactive sources, are drawn from a zero-mean Gaussian with variance 0.01. This allows us to control sparsity more accurately. To ensure the 'uniqueness of sparsest solution' property, the $\#act$ should be less than $n/2$ [1]. Thus, for this type of source, we may use $\#act/(n/2)$ as a normalized measure of sparsity. To conduct the experiment (with $m = 500, n = \lfloor 0.4m \rfloor = 200$), we select 10 values of $\#act/(n/2)$ in the range $[0.1, 1]$. For each $\#act$, both IDE and LP methods are applied to $N = 10$ realizations of the problem and the average SNR (over the N experiments) obtained by each method is determined. Fig. 2 illustrates the result. Note that the horizontal axis shows $\#act/(n/2)$ in a logarithmic scale. It is observed that both methods perform very well before $\#act \approx n/4$ and poorly after $\#act \approx 3n/10$. The IDE exhibits slightly superior performance over the range $\#act < n/4$. On the other hand, LP has the advantage of a more gradual degradation, leaving it applicable over a (slightly) wider range of sparsity.

CONCLUSION

We developed a method of sparse decomposition based on a two-step iterative procedure, namely: detection of active sources and estimation by projecting into the activity subspace. For the detection step, comparison of an activity function against a threshold was proposed. It was found by experiment that the algorithm is not too sensitive to the choice of threshold sequence and (that) proper choices may be found for families of

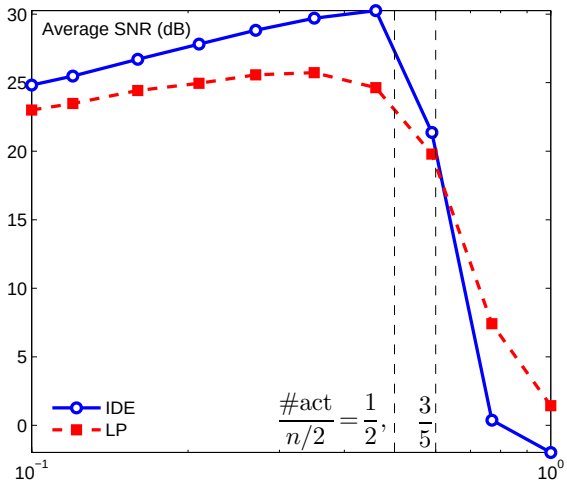

FIGURE 2. Average SNR versus normalized number of active sources, $\#act/(n/2)$ (in a logarithmic scale), as a measure of sparsity. For each value of $\#act$, the average SNR is computed over $N = 10$ realizations. In this experiment $m = 500$ and $n = \lfloor 0.4\,m \rfloor = 200$.

problems. The overall algorithm was shown to be nearly two orders of magnitude faster than LP interior-point solvers, while providing the same (or better) accuracy. The authors believe that the algorithm is also an evidence that the sparse decomposition problem is not computationally as *hard* as it is suggested by the LP approach. There are still issues deserving further investigation, e.g., is it possible to (devise an algorithm to) systematically choose a threshold sequence? Are there other activity functions that may be used in the detection step? How the algorithm will perform if we try to lower the complexity by skipping some of the computationally intensive parts? For example, is it possible that a low-complexity variant of the estimation step only computes the active portion of the solution and estimate the inactive part to be zero? These issues are currently under study.

REFERENCES

1. D. L. Donoho, For most large underdetermined systems of linear equations the minimal l^1-norm solution is also the sparsest solution, Tech. rep. (2004), URL `http://www-stat.stanford.edu/~donoho/Reports/2004/`.
2. P. Bo£ll, and M. Zibulevsky, *Signal Processing* **81**, 2353–2362 (2001).
3. R. Gribonval, and 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.
4. Y. Li, A. Cichocki, and S. Amari, *Neural Computation* **16**, 1193–1234 (2004).
5. M. Zibulevsky, and B. A. Pearlmutter, *Neural Computation* **13**, 863–882 (2001).
6. S. S. Chen, D. L. Donoho, and M. A. Saunders, *SIAM Journal on Scienti£c Computing* **20**, 33–61 (1999).
7. D. L. Donoho, and X. Huo, *IEEE Trans. Inform. Theory* **47**, 2845–2862 (2001).
8. M. Elad, and A. Bruckstein, *IEEE Trans. Inform. Theory* **48**, 2558–2567 (2002).
9. L. L. Scharf, *Statistical Signal Processing: Detection, Estimation, and Time Series Analysis*, Addison-Wesley, 1991.
10. J. Nocedal, and S. Wright, *Numerical Optimization*, Springer, New York, 1999.
11. N. Gould, M. Hribar, and J. Nocedal, *SIAM J. Sci. Computing* **23**, 1375–1394 (2001).
12. A. A. Amini, M. Babie-Zadeh, and C. Jutten, *to appear in EUSIPCO'06* (2006).