A MOVING WINDOW ALGORITHM FOR l_∞ NORM BASED BLIND SOURCE SEPARATION APPROACH

Alper T. Erdogan

EE Department, Koc University
Rumelifeneri Yolu, 34450, Istanbul, Turkey
phone: + (90) 212-338 1490, fax: + (90) 212 338 1548, email: alperdogan@ku.edu.tr
web: http://aspc.ku.edu.tr

ABSTRACT

An l_∞ norm based Source Separation Method has recently been introduced [1, 2], with the corresponding subgradient based algorithm that has a simple update rule. In this article, we propose a moving window modification to this algorithm, where instead of a long fixed window, a shorter length moving window is used. The simulation results for different blind source separation setups suggest that a significant level of complexity reduction can be achieved via use of the proposed approach.

1. INTRODUCTION

In the area of Blind Source Separation (BSS), there exists various approaches that are based on different criteria. Among these, we can list Maximum Likelihood based methods (e.g., [3, 4]), methods that are based on maximization of Non-Gaussianity (e.g., [5, 6, 7] and mutual information minimization based methods (e.g., [8, 9, 10]) as most popular approaches.

Recently, an l_∞ norm based geometric BSS has been proposed for source signals with bounded magnitudes [2]. By this approach finding the independent components from whitened observations has been posed as a geometric problem of finding a rotation/reflection that minimizes the maximum value of the transformer output over the ensemble. The corresponding cost function is

$$J(\Theta) = \sup \|\Theta \mathbf{x}\|_\infty, \tag{1}$$

where, the supremum is over the ensemble of outputs, $\mathbf{x}$ is the whitened mixture vectors and Θ is the unitary separator. With this approach, separation can be achieved in relatively short bursts of data and the algorithm is robust against the unknown correlations in the input[2].

The cost function in (1) is non-smooth (but convex), and therefore, a subgradient based algorithm has been proposed for this cost function. Although the update rule turns out to be really simple, as in any batch algorithm (such as FastICA [11]), the proposed algorithm requires the computation of a window of output values for each iteration, which turns out to be the main source of complexity. Therefore, the complexity can effectively be reduced only if the complexity of output computations can be reduced. As the complexity of output computations itself is linearly proportional to the size of the window, the simplest approach can be considered as the reduction of the window size. However, using a small fixed-window means inadequate sampling of the observations processes which may result in the misrepresentation of the ensemble behavior. A better approach is the use of a small moving-window, since it is more likely to capture the ensemble behavior in multiple windows, which is in fact the approach that we propose with this article.

The organization of the article is as follows: In Section 2, we provide the BSS setup assumed and information on l_∞ norm based approach of [2]. In Section 3, the moving window modification to this algorithm is provided. The simulation examples for the proposed approach are given in Section 4. Finally, Section 5 is the conclusion.

This work is supported in part by TUBITAK Career Award, Contract No:104E073

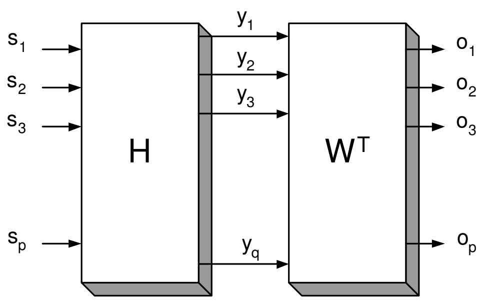

Figure 1: Blind Source Separation Setup

2. BLIND SOURCE SEPARATION SETUP AND l_∞ NORM BASED ALGORITHM

The instantaneous blind source separation setup that we consider throughout the article is shown in Figure 1. In this figure,

- $s_1(k), s_2(k), \ldots, s_p(k)$ are source signals. It is assumed that they are all i.i.d with zero mean and unity variance (without loss of generality), and mutually independent of each other. Furthermore, the source signals are considered to be bounded and complex symmetric in the sense that

$$\begin{aligned} \max \Re e\{s_l\} &= \max Im\{s_l\} = -\min \Re e\{s_l\} \\ &= -\min \mathscr{I}m\{s_l\} = M. \end{aligned} \tag{2}$$

- The source signals are mixed by a MIMO system with a $q \times p$ transfer matrix $\mathbf{H}$, which has q outputs denoted by

$y_1(k), y_2(k), \ldots, y_q(k)$. Therefore, we can write

$$\underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_q \end{bmatrix}}_{\mathbf{y}} = \mathbf{H} \underbrace{\begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_p \end{bmatrix}}_{\mathbf{s}}. \tag{3}$$

- The goal in BSS is to come up with a separator matrix $\mathbf{W}$ such that its outputs are "close to" the original sources, without any knowledge of the channel. We assume that the separator is composed of two factors, i.e.,

$$\mathbf{W} = \mathbf{W}_{pre}^T \mathbf{\Theta} \tag{4}$$

where $\mathbf{W}_{pre}$ is the pre-whitening matrix to whiten the mixtures and $\mathbf{\Theta}$ is the unitary separator. The whitening transformation is obtained through second-order statistics based approaches (such as linear prediction), and the most critical part is to come up with a scheme to obtain the unitary separator $\mathbf{\Theta}$ which requires higher order information.
In [2], the following problem is introduced as an implicit higher-order statistics approach to obtain $\mathbf{\Theta}$:

$$\begin{aligned} &\text{minimize } f(\mathbf{\Theta}) \\ &\text{s.t. } \mathbf{\Theta}^H \mathbf{\Theta} = I, \end{aligned}$$

where

$$f(\mathbf{\Theta}) = \max_{k \in \{0,1,\ldots,\Omega-1\}} \|\mathscr{R}e\{\mathbf{o}(k)\}\|_\infty, \tag{5}$$

and Ω is the length of the fixed data window. If we define $\mathbf{X} = [\ \mathbf{x}(0) \quad \mathbf{x}(1) \quad \ldots \mathbf{x}(\Omega-1)\ ]$, as the matrix of input values in the window of interest, then for a given $\mathbf{\Theta}$ the corresponding outputs can be placed in a matrix $\mathbf{O}$:

$$\begin{aligned} \mathbf{O} &= [\ \mathbf{o}(0) \quad \mathbf{o}(1) \quad \ldots \quad \mathbf{o}(\Omega-1)\ ] && (6) \\ &= \mathbf{\Theta}^T \mathbf{X}. && (7) \end{aligned}$$

Based on these definitions, the subgradient based BSS algorithm of [2] can be written as

$$\underline{\mathbf{\Theta}}^{(i+1)} = \mathbf{\Theta}^{(i)} - \mu^{(i)} \frac{\Re e\{\mathbf{O}^{(i)}_{m^{(i)},n^{(i)}}\}}{|\Re e\{\mathbf{O}^{(i)}_{m^{(i)},n^{(i)}}\}|} \bar{\mathbf{X}}_{:,n^{(i)}} \mathbf{e}^T_{m^{(i)}} \tag{8}$$

$$\mathbf{\Theta}^{(i+1)} = \mathscr{P}_U\{\underline{\mathbf{\Theta}}^{(i+1)}\}, \tag{9}$$

where

- $\mathbf{\Theta}^{(i)}$ is the value of $\mathbf{\Theta}$ at the i^{th} iteration,
- $\mathbf{O}^{(i)}$ is the output matrix calculated based on $\mathbf{\Theta}^{(i)}$,
- $(m^{(i)}, n^{(i)})$ is the index for a maximum real component magnitude entry of $\mathbf{O}^{(i)}$,
- $\mu^{(i)}$ is the step size at the i^{th} iteration. We use the relaxation rule for fast convergence. (see [2] for the details).
- $\underline{\mathbf{\Theta}}^{(i+1)}$ is the unprojected version of the updated $\mathbf{\Theta}$,
- $\mathscr{P}_U$ is the projection operator to the unitary matrix set where we use the minimum-distance projection operator to the set of unitary matrices [2] which is defined as

$$\begin{aligned} \mathbf{\Theta}^{(i+1)} &= \mathscr{P}_U\{\underline{\mathbf{\Theta}}^{(i+1)}\} \\ &= \mathscr{P}_U\{\mathbf{U}^{(i+1)}\mathbf{\Sigma}^{(i+1)}\mathbf{V}^{(i+1)^H}\} \\ &= \mathbf{U}^{(i+1)}\mathbf{V}^{(i+1)^H}, \end{aligned}$$

where $\mathbf{U}^{(i+1)}\mathbf{\Sigma}^{(i+1)}\mathbf{V}^{(i+1)^H}$ is the SVD of $\underline{\mathbf{\Theta}}^{(i+1)}$. These matrices can be conveniently computed using a Gram-Schmidt based algorithm [2] which exploits the special update structure in (8).

3. THE MOVING WINDOW APPROACH

The fixed-window algorithm summarized in the previous section has a simple update rule with a low computational requirement. However, it is a batch algorithm and each iteration requires computation of Ω output vectors, which dominates the overall computational requirement. In order to capture the ensemble behavior, Ω needs to be chosen as large as possible.

To reduce the complexity, we can choose to use a shorter window length. However, as the ensemble behavior is reflected by the use of larger number of independent samples, we need to use a moving window approach.

For the moving window approach, the window is proposed to be cascade of two sub-windows:

$$\mathbf{X}^{(i)} = [\ \mathbf{P}^{(i)} \quad \mathbf{D}^{(i)}\ ], \tag{10}$$

where

- $\mathbf{D}^{(i)}$ is a $p \times \Omega_D$ matrix, which contains Ω_D new input data vectors:

$$\mathbf{D}^{(i)} = [\ \mathbf{x}(i\Omega_D) \quad \mathbf{x}(i\Omega_D+1) \quad \ldots \quad \mathbf{x}((i+1)\Omega_D-1)\ ], \tag{11}$$

- $\mathbf{P}^{(i)}$ is a $p \times \Omega_P$ matrix, which contains the window of Ω_P past input vectors used in search vector computation.

In a typical moving window based adaptive algorithm, only $\mathbf{D}^{(i)}$ matrix is used. We include $\mathbf{P}^{(i)}$ as a memory element and the inclusion of this component can be justified by the fact that an input vector that created maximum (real) component output in the previous windows is likely to cause the maximum real output in a new window. The reasoning for this can be given as follows: the input vector causing the maximum real output given in a given window has the maximum alignment with the error vector $\mathbf{w} - \mathbf{w}_{opt}$, and this alignment can continue for several iterations especially when $\mathbf{w}$ is not in the vicinity of $\mathbf{w}_{opt}$ and the step size is small. The inclusion of the memory element in a window is also intuitive in terms of spreading infinity-norm evaluation to a longer time span.

The output data matrix will be given by

$$\mathbf{O}^{(i)} = \mathbf{\Theta}^{(i)^T} \mathbf{X}^{(i)}, \tag{12}$$

and the only change in update expression of (8) would be the inclusion of the super index for the input vector:

$$\underline{\mathbf{\Theta}}^{(i+1)} = \mathbf{\Theta}^{(i)} - \mu^{(i)} \frac{\Re e\{\mathbf{O}^{(i)}_{m^{(i)},n^{(i)}}\}}{|\Re e\{\mathbf{O}^{(i)}_{m^{(i)},n^{(i)}}\}|} \bar{\mathbf{X}}^{(i)}_{:,n^{(i)}} \mathbf{e}^T_{m^{(i)}} \tag{13}$$

$$\Theta^{(i+1)} = \mathscr{P}_U\{\underline{\Theta}^{(i+1)}\}. \tag{14}$$

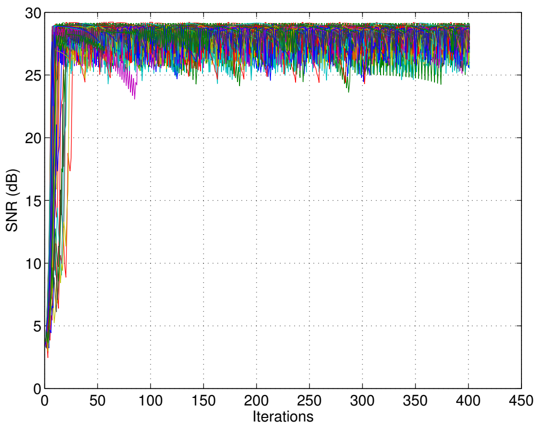

Figure 2: SNR convergence curves for the fixed window algorithm with window length 1000.

As illustrated by the examples in the next section, when we use the moving window approach to replace the fixed window version, window size can be reduced by a factor of 50-100 whereas the required number of iterations increases by a factor of 2-3. Hence, this approach can provide a remarkable reduction in the overall complexity.

4. SIMULATION EXAMPLES

In the first example, we assume the simulation setup in [6] where $\mathbf{H}$ is defined as

$$\mathbf{H} = \left[\begin{array}{cc} -0.307+0.071i & -0.844+0.379i \\ -0.616+0.691i & 0.1798-0.3315i \end{array}\right] \tag{15}$$

and x_i's are the identically distributed subgaussian random variables with the probability mass function (pmf)

$$p(x) = \left\{\begin{array}{cc} \frac{3}{4} & x=0, \\ \frac{1}{16} & x=1+i, 1-i, -1+i, -1-i. \end{array}\right. \tag{16}$$

The output of $\mathbf{H}$ is corrupted by a Gaussian noise and $SNR = 30dB$.

In Figure 2, the fixed window algorithm's SNR convergence curves (for different runs of the algorithm) are shown, where the window length is equal to 1000. The algorithm converges in about 15 iterations on average, which corresponds to a requirement of 15000 output computations.

In Figure 3, the convergence curves for the moving window algorithm are shown. In the moving-window version both Ω_D and Ω_P are taken as 15. According to this figure, 50 iterations on average is needed. Therefore, we need about $(\Omega_D + \Omega_P) * 50 = 30 * 50 = 1500$ output computations. Comparing this with the fixed window version, it can be seen that a factor of 10 complexity reduction is achieved.

In the second example, we consider a scenario with 5 sources and 6 mixtures, where the channel is arbitrarily selected. The source signals are zero mean $16-$QAM i.i.d. sequences. We assume that the output of the channel is corrupted by an additive white Gaussian noise. The average power of the noise is such that the overall SNR is equal to 40dB.

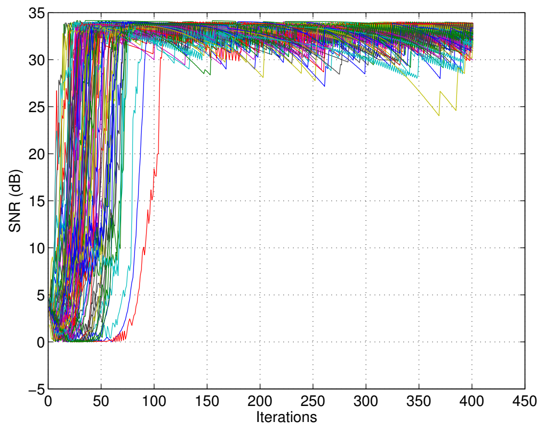

Figure 3: SNR convergence curves for the moving window algorithm.

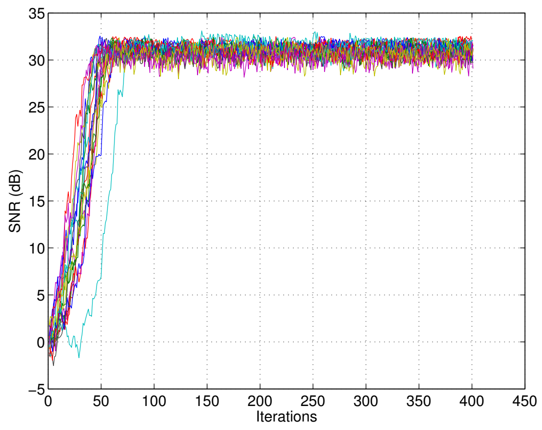

Figure 4: SNR convergence curves for the fixed window algorithm with window length 1000.

In Figure 4, SNR convergence curves (for different runs) for the algorithm with fixed window are shown. Here the window length is chosen as $\Omega = 1000$. According to this figure, the algorithm converges in 50 iterations on average.

The convergence curves (for different runs) for the moving window version of the algorithm are shown in Figure 5. In this case , Ω_D is selected as 14 and $\Omega_P = 0$. According to this figure, the algorithm converges in 100 iterations on average. The worst case convergence occurs for 200 iterations.

If we compare the complexities of the fixed window and the moving window algorithms for the above example: we first note that the window size is reduced by a factor of 70 where as the required number of iterations only increased by a factor of 2 by the use of moving window algorithm. If we calculate the required number of output computations:

- Fixed window source separation algorithm requires $50 * 1000 = 50000$ output computations.
- Moving window algorithm requires $100 * 14 = 1400$ output computations. We should note in this case that we only need to do one output vector computation per input

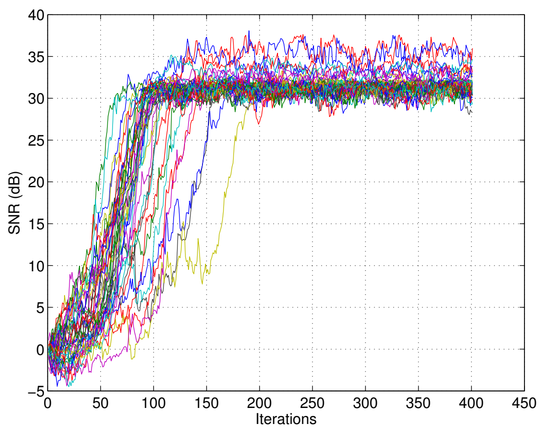

Figure 5: SNR convergence curves for the moving window algorithm with window length 14.

vector and only about 100 updates.

Therefore, since both algorithms' complexities are dominated by the output computations, complexity is reduced by a factor more than 30 through use of moving window algorithm.

5. CONCLUSION

Blind source separation algorithm based on l_∞ norm minimization has the feature that its update rule is very simple and therefore the complexity is mostly dominated by the output computations. We introduced a moving window approach to significantly reduce the number of output computations per iteration (at the expense of slight increase in the number of iterations). As illustrated by the examples, by the moving window approach, the overall computational requirement can be lowered to a level that is suitable for real-time implementations.

REFERENCES

[1] Alper T. Erdogan, "A blind separation approach for magnitude bounded sources," *IEEE ICASSP-2005, Philadelphia, USA*, March 2005.

[2] Alper T. Erdogan, "A simple geometric blind source separation method for bounded magnitude sources," *IEEE Trans. on Signal Processing, to appear*, 2004.

[3] S.I. Amari, A. Cichocki, and H. H. Young, "A new learning algorithm for blind source separation," *In Advances in Neural Information Processing Systems, MIT Press Cambridge*, vol. 8, pp. 757–763, 1997.

[4] Dinh Tuan Pham, P. Garat, and C. Jutten, "Separation of a mixture of independent sources through a maximum likelihood approach," *Proceedings of EUSIPCO*, pp. 771–774, 1992.

[5] Nathalie Delfosse and Philippe Loubaton, "Adaptive blind separation of independent sources: a deflation approach," *Signal Processing*, vol. 45, pp. 59–83, July 1995.

[6] Constantinos Papadias, "Globally convergent blind source separation based on a multiuser kurtosis maximization criterion," *IEEE Trans. on Signal Processing*, vol. 48, pp. 3508–3519, December 2000.

[7] Aapo Hyvärinen, "Fast and robust fixed-point algorithms for independent component analysis," *IEEE Trans. on Neural Networks*, vol. 10, no. 3, pp. 626–634, 1999.

[8] P. Common, "Independent component analysis- a new concept?," *Signal Processing*, vol. 36, pp. 287–314, April 1994.

[9] H. H. Yang and S. I. Amari, "Adaptive on-line learning algorithms for blind separation: Maximum entropy and minimum mutual information," *Neural Computaion*, vol. 9, pp. 1457–1482, 1997.

[10] Dinh Tuan Pham, "Mutual information approach to blind separation of stationary sources," *IEEE Trans. on Signal Processing*, vol. 48, pp. 1935 – 1946, July 2002.

[11] Aapo Hyvärinen and Erkki Oja, "A fast fixed-point algorithm for independent component analysis," *Neural Computation*, vol. 9, no. 7, pp. 1483–1492, October 1997.