Recursive Generalized Eigendecomposition for Independent Component Analysis

Umut Ozertem[1], Deniz Erdogmus[1,2], and Tian Lan[2]

[1] CSEE Department, OGI, Oregon Health & Science University, Portland, OR, USA
{ozertemu, deniz}@csee.ogi.edu
[2] BME Department, OGI, Oregon Health & Science University, Portland, OR, USA
lantian@bme.ogi.edu

Abstract. Independent component analysis is an important statistical tool in machine learning, pattern recognition, and signal processing. Most of these applications require on-line learning algorithms. Current on-line ICA algorithms use the stochastic gradient concept, drawbacks of which include difficulties in selecting the step size and generating suboptimal estimates. In this paper a recursive generalized eigendecomposition algorithm is proposed that tracks the optimal solution that one would obtain using all the data observed.

1 Introduction

Independent component analysis (ICA) has now established itself as an essential statistical tool in signal processing and machine learning, both as a solution to problems (such as blind source separation) [1,2] and as a preprocessing instrument that complements other pieces of a more comprehensive solution (such as dimensionality reduction and feature extraction) [3-5]. All of these applications of ICA require on-line learning algorithms that can operate in real-time on contemporary digital signal processors (DSP).

Currently, the on-line ICA solutions are obtained using algorithms designed using the stochastic gradient concept (e.g., Infomax [6]), similar to the well-known least-mean squares (LMS) algorithm [7]. The drawbacks of stochastic gradient algorithms in on-line learning include difficulty in selecting the step size for optimal speed misadjustment trade-off and suboptimal estimates of the weights given the information contained in all the samples seen at any given iteration.

Recursive least squares (RLS) is an on-line algorithm for supervised adaptive filter training, which has the desirable property that the estimated weights correspond to the optimal least squares solution that one would obtain using all the data observed so far, provided that initialization is done *properly* [7]. This benefit, of course comes at the cost of additional computational requirements compared to LMS. Nevertheless, certain applications where an on-line ICA algorithm that tracks the optimal solution one would have obtained using all samples observed up to that point in time would be beneficial. To this end, we derive a recursive generalized eigendecomposition (GED) based ICA algorithm that is similar to RLS in principle, but solves the simultaneous diagonalization problem using second and fourth order joint statistics of the observed mixtures.

The joint diagonalization of higher order statistics have been known to solve the ICA problem under the assumed linear mixing model and have led to popular algorithms (e.g., JADE [8]). The joint diagonalization problem in ICA is essentially a GED problem, a connection which has been nicely summarized in a recent paper by Parra and Sajda [9] for various signal models in linear instantaneous BSS; others have pointed out this connection earlier as well. The algorithm we develop here is based on the non-Gaussian independent sources assumption, with independent and identically distributed samples of mixtures (the latter assumption eliminates the need for weighted averaging for minimum bias estimation of the expectations).

In Section 2, we derive the recursive update equations for the required higher order statistics and the corresponding *optimal* ICA solution. In Section 3, we demonstrate using Monte Carlo simulations that the algorithm tracks the optimal ICA solution exactly when all matrices are initialized to their ideal values and that the algorithm converges to the optimal ICA solution when the matrices are initialized to arbitrary small matrices (whose bias on the solution should diminish as more samples are observed and utilized).

2 Recursive ICA Algorithm

The square linear ICA problem is expressed in (1), where $\mathbf{X}$ is the $n\times N$ observation matrix, $\mathbf{A}$ is the $n\times n$ mixing matrix, and $\mathbf{S}$ is the $n\times N$ independent source matrix.

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

Each column of $\mathbf{X}$ and $\mathbf{S}$ represents one sample of data. If we consider each column as a sample in time, (1) becomes:

$$\mathbf{x}_t = \mathbf{A}\mathbf{s}_t \tag{2}$$

The joint diagonalization of higher order cumulant matrices can be compactly formulated in the form of a generalized eigendecomposition problem that gives the ICA solution in an analytical form [9]. According to this formulation, under the assumption of independent non-Gaussian source distributions the separation matrix $\mathbf{W}$ is the solution to the following generalized eigendecomposition problem:

$$\mathbf{R}_{\mathbf{x}}\mathbf{W} = \mathbf{Q}_{\mathbf{x}}\mathbf{W}\boldsymbol{\Lambda} \tag{3}$$

where $\mathbf{R}_x$ is the covariance matrix and $\mathbf{Q}_x$ is the cumulant matrix estimated using sample averages. While any order of cumulants could have been employed, lower orders are more robust to outliers and small sample sizes, therefore we focus on the fourth order cumulant matrix: $\mathbf{Q}_x$=E[$\mathbf{x}^\mathbf{T}\mathbf{x}\mathbf{x}\mathbf{x}^\mathbf{T}$]-$\mathbf{R}_x$tr($\mathbf{R}_x$)-E[$\mathbf{x}\mathbf{x}^\mathbf{T}$]E[$\mathbf{x}\mathbf{x}^\mathbf{T}$]-$\mathbf{R}_x\mathbf{R}_x$. Given the sample estimates for these matrices, the ICA solution can be easily determined using efficient generalized eigendecomposition algorithms (or the *eig* command in Matlab®). With the assumption of iid samples, expectations reduce to simple sample averages, and the estimates of covariance and cumulant matrices are given by (for real-valued mixtures)

$$\mathbf{R}_{\mathbf{x}} = \sum_i \mathbf{x}_i\mathbf{x}_i^T \qquad \mathbf{Q}_{\mathbf{x}} = \sum_i \left(\mathbf{x}_i^T\mathbf{x}_i\mathbf{x}_i\mathbf{x}_i^T\right) - \mathbf{R}_{\mathbf{x}}tr(\mathbf{R}_{\mathbf{x}}) - 2\mathbf{R}_{\mathbf{x}}^2 \tag{4}$$

2.1 The Update Equations

Given the estimates in (4), one can define recursive update rules for the estimates of the covariance and cumulant matrices **R** and **Q**, as well as $\mathbf{R}^{-1}$ and $\mathbf{R}^{-1}\mathbf{Q}$ for further computational savings. The recursive update for the covariance matrix is

$$\mathbf{R}_t = \frac{t-1}{t}\mathbf{R}_{t-1} + \frac{1}{t}\mathbf{x}_t\mathbf{x}_t^T \tag{5}$$

and the update rule for the cumulant matrix is given by

$$\mathbf{Q}_t = \mathbf{C}_t - \mathbf{R}_t tr(\mathbf{R}_t) - 2\mathbf{R}_t^2 \tag{6}$$

where the matrix **C** is defined as $\mathbf{C} = E\{\mathbf{x}^T\mathbf{x}\,\mathbf{x}\,\mathbf{x}^T\}$, and estimating the expectation using sample averages as before, it becomes

$$\mathbf{C} = \sum_i \left(\mathbf{x}_i^T\mathbf{x}_i\ \mathbf{x}_i\ \mathbf{x}_i^T\right) \tag{7}$$

Now, we can define the update rules for **C** and $\mathbf{R}^2$ to obtain the recursive update for the cumulant matrix. The update rule for **C** is given by

$$\mathbf{C}_t = \frac{t-1}{t}\mathbf{C}_{t-1} + \frac{1}{t}\left(\mathbf{x}_t^T\mathbf{x}_t\right)\mathbf{x}_t\mathbf{x}_t^T \tag{8}$$

The recursive update of $\mathbf{R}^2$ can be derived from (5) by squaring both sides. Hence, the update rule for $\mathbf{R}^2$ becomes

$$\mathbf{R}_t^2 = \frac{(t-1)^2}{t^2}\mathbf{R}_{t-1}^2 + \frac{1}{t^2}\left(\mathbf{x}_t^T\mathbf{x}_t\right)\mathbf{x}_t\mathbf{x}_t^T + \frac{(t-1)}{t^2}[\mathbf{v}_t\mathbf{x}_t^T + \mathbf{x}_t\mathbf{v}_t^T] \tag{9}$$

where for further computational savings we introduce the vector $\mathbf{v}_t$ as

$$\mathbf{v}_t = \mathbf{R}_{t-1}\mathbf{x}_t \tag{10}$$

Finally, the update rule for the cumulant matrix **Q** can be obtained by substituting (5), (8), and (9) into (6). Further computational savings can be obtained by iterating $\mathbf{R}^{-1}$ and $\mathbf{R}^{-1}\mathbf{Q}$ to avoid matrix multiplications and inversions, each having an $O(n^3)$ computational load. The reason why we need these two matrices will be clear as we proceed to the fixed-point algorithm that solves for the generalized eigendecomposition. Employing the matrix inversion lemma, the recursion rule for $\mathbf{R}^{-1}$ becomes

$$\mathbf{R}_t^{-1} = \frac{t}{t-1}\mathbf{R}_{t-1}^{-1} - \frac{t}{(t-1)\,\alpha_t}\mathbf{u}_t\mathbf{u}_t^T \tag{11}$$

where α_t and $\mathbf{u}_t$ are defined as

$$\alpha_t = (t-1) + \mathbf{x}_t^t\mathbf{u}_t \qquad \mathbf{u}_t = \mathbf{R}_{t-1}^{-1}\mathbf{x}_t \tag{12}$$

Here we define the matrix **D**, the update equation of whom can easily be defined by substituting the previously given update equations for $\mathbf{R}^{-1}$ and **Q**, using (11) and (6).

$$\mathbf{D}_t = \mathbf{R}_t^{-1}\mathbf{Q}_t \tag{13}$$

2.2 Deflation Procedure

Having the update equations, the aim is to find the optimal solution for the eigendecomposition for the updated correlation and cumulant matrices in each iteration. Recall the original problem given in (3); we need to solve for the weight matrix **W**. We will employ the deflation procedure to determine each generalized eigenvector sequentially. Every generalized eigenvector $\mathbf{w}_d$ that is a column of the weight matrix **W** is a stationary point of the function

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{R}\,\mathbf{w}}{\mathbf{w}^T \mathbf{Q}\,\mathbf{w}}. \tag{14}$$

This fact can be easily seen by taking the derivative and equating it to zero:

$$\begin{aligned} \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} &= \frac{\mathbf{w}^T \mathbf{Q}\,\mathbf{w}(2\,\mathbf{R}\,\mathbf{w}) - \mathbf{w}^T \mathbf{R}\,\mathbf{w}(2\,\mathbf{Q}\,\mathbf{w})}{\left(\mathbf{w}^T \mathbf{Q}\,\mathbf{w}\right)^2} = 0 \\ \mathbf{R}\mathbf{w} &= \frac{\mathbf{w}^T \mathbf{R}\,\mathbf{w}}{\mathbf{w}^T \mathbf{Q}\,\mathbf{w}} \mathbf{Q}\mathbf{w} \end{aligned} \tag{15}$$

This is nothing but the generalized eigenvalues equation where the eigenvalues are the values of the objective criterion $J(\mathbf{w})$ given in (14) evaluated at its stationary points. Hence, the fixed-point algorithm becomes

$$\mathbf{w} \leftarrow \frac{\mathbf{w}^T \mathbf{R}\,\mathbf{w}}{\mathbf{w}^T \mathbf{Q}\,\mathbf{w}} \mathbf{R}^{-1}\mathbf{Q}\mathbf{w}. \tag{16}$$

This fixed-point optimization procedure converges to the largest generalized eigenvector[1] of **R** and **Q**, and the deflation procedure is employed to manipulate the matrices such that the obtained matrices have the same generalized eigenvalue and eigenvector pairs except the ones that have been determined previously. The larger eigenvalues are replaced by zeros in each deflation step. Note that in this subsection the time index is implicit and omitted for notational convenience. While *d* represents the dimension index, the deflation procedure employed while iterating the dimensions is given by

$$\mathbf{Q}_d = \left[\mathbf{I} - \frac{\mathbf{Q}_{d-1}\mathbf{w}_{d-1}\mathbf{w}_{d-1}^T}{\mathbf{w}_{d-1}^T \mathbf{Q}_{d-1}\mathbf{w}_{d-1}}\right]\mathbf{Q}_{d-1} \qquad \mathbf{R}_d = \mathbf{R}_{d-1}. \tag{17}$$

The deflated matrices are initialized to $\mathbf{Q}_1=\mathbf{Q}$ and $\mathbf{R}_1=\mathbf{R}$. Obtaining the new matrices using deflation, we will employ the same fixed-point iteration procedure given in (16) to find the corresponding eigenvector.

Investigating the fixed-point algorithm in (16), it is clear that iterating $\mathbf{R}^{-1}$ and **D** as suggested earlier will result in computational savings. The deflation rules for these matrices can be deduced from (17) easily. The deflation of $\mathbf{R}^{-1}$ is

$$\mathbf{R}_d^{-1} = \mathbf{R}_{d-1}^{-1}. \tag{18}$$

[1] The largest eigenvector is the one that corresponds to the greatest eigenvalue.

Similarly, the deflation rule for **D** can be obtained by combining (17) and (18) as

$$\mathbf{D}_d = [\mathbf{I} - \mathbf{w}_{d-1}\mathbf{w}_{d-1}^T\mathbf{Q}_{d-1}]\,\mathbf{D}_{d-1} \tag{19}$$

For each generalized eigenvector, the corresponding fixed-point update rule then becomes as follows:

$$\mathbf{w}_d \leftarrow \frac{\mathbf{w}_d^T\mathbf{R}_d\,\mathbf{w}_d}{\mathbf{w}_d^T\mathbf{Q}_d\mathbf{w}_d}\mathbf{D}_d\,\mathbf{w}_d \tag{20}$$

Employing this fixed-point algorithm for each dimension and solving for the eigenvectors sequentially, one can update the **W** matrix and proceed to the next time update step. In the following section we will present results comparing the original GED-ICA algorithm [9] with the results of the proposed recursive GED-ICA algorithm.

3 Experimental Results

In this section, the results provided by the proposed recursive algorithm will be compared with those of the original GED-ICA algorithm. The experiments are done on a synthetic dataset, which is simply generated by a linear mixture of independent uniform sources. Experiments using mixing matrices with varying condition numbers are employed to test the dependency of the tracking performance on the mixture conditioning.

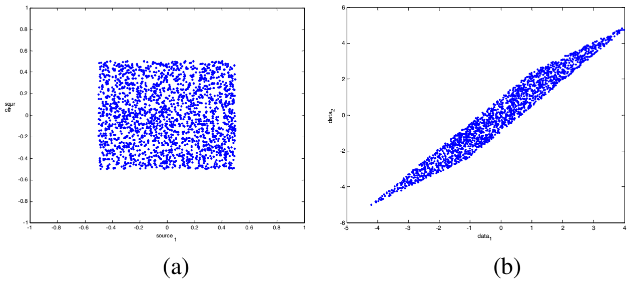

Fig. 1. An illustration of the samples (a) from independent uniform sources (b) after the linear mixing

The joint distribution of the sources is presented in Figure 1 for a two-dimensional case. Mixing matrices with condition numbers 10 and 100 are employed for the mixing and the corresponding results are presented for two cases. In the first case, the original GED-ICA is employed on a small initialization data set to obtain ideal initial values for all matrices involved, including the eigenvectors. The expected result for the first simulation is to observe that the recursive algorithm is capable of tracking the result of the original algorithm within a range of small numerical error. In second case, these values are initialized to arbitrary small matrices. As increasing number of samples are utilized for the matrix updates, the bias of these arbitrary initial conditions is expected to decay. The second experiment will allow us to investigate this decay process by comparing the *biased* solution to that of the original GED-ICA procedure.

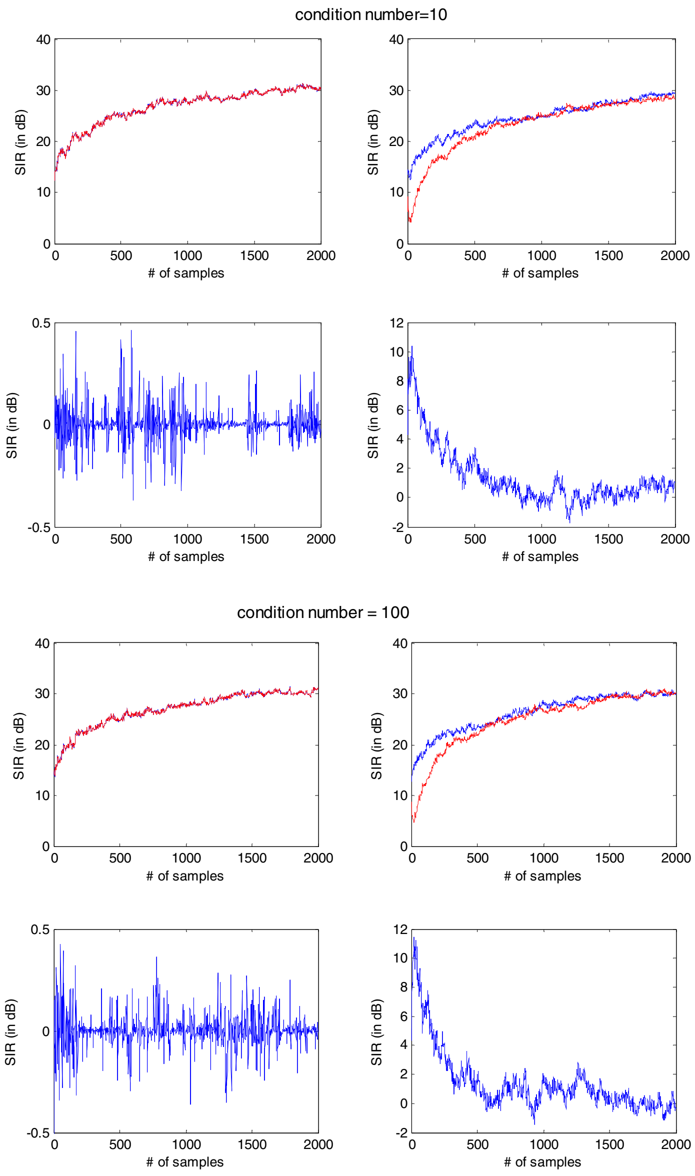

Fig. 2. The performances of recursive and the original methods are compared for a mixing of condition numbers 10 and 100. Performances and performance differences for exact initialization (top and bottom left, accordingly) and random initialization (top and bottom right) are shown.

In the simulations, the 20 samples have been used for initialization the ideal solution, and for the arbitrary initialization identity matrices with a small variances are employed (note that once **R** and **C** are initialized to values at the order of 10^{-6} all other matrices can be determined consistently with the equations). The corresponding average tracking results for 2000 samples are shown in Figure 2 for mixing matrix condition numbers of 10 and 100. These results are averaged over 100 Monte Carlo runs keeping the condition number of the mixture and the joint source distribution fixed and randomizing the right and left eigenvectors of the mixing matrix as well as the actual source samples using in the sample averages.

4 Conclusions

On-line ICA algorithms are essential for may signal processing and machine learning applications, where the ICA solution acts as a front-end preprocessor, a feature extractor, or a major portion of the solution itself. Stochastic gradient based algorithm motivated by various ICA criteria have been utilized successfully in such situations and they have the advantage of yielding computationally simple weight update rules. On the other hand, they are not able to offer optimal solutions at every iteration.

In this paper, we derived a recursive ICA algorithm based on the joint diagonalization of covariance and fourth order cumulants. The derivation employs the use of the matrix inversion lemma and the sample update rules for expectations approximated by sample averages. Since the proposed method is the recursive version of the algorithm proposed in [9], and it is tracking the optimal solution given by this algorithm in a recursive manner, the experimental results section is limited to the comparisons between the proposed recursive method and the original algorithm.

The resulting algorithm, of course, is computationally more expensive than its stochastic gradient counterpart. However, it has the ability to converge to and track the optimal solution based on this separation criterion in a small number of samples, even when initialized to arbitrary matrices.

Acknowledgments

This work was supported by DARPA under contract DAAD-16-03-C-0054 and by NSF under grant ECS-0524835.

References

1. Hyvarinen A., Karhunen J., Oja E.: Independent Component Analysis, Wiley, New York (2001)
2. Cichocki A., Amari S.I.: Adaptive Blind Signal and Image Processing: Learning Algorithms and Applications, Wiley, 2002.
3. Hyvärinen A., Oja E., Hoyer P., Hurri J.: Image Feature Extraction by Sparse Coding and Independent Component Analysis, Proceedings of ICPR'98, (1998) 1268-1273

4. Lan T., Erdogmus D., Adami A., Pavel M.: Feature Selection by Independent Component Analysis and Mutual Information Maximization in EEG Signal Classification, Proceedings IJCNN 2005, Montreal, (2005) 3011-3016
5. Everson R., Roberts S.: Independent Component Analysis: A Flexible Nonlinearity and Decorrelating Manifold Approach, Neural Computation, vol. 11. (2003) 1957-1983
6. Bell A., Sejnowski T.: An Information-Maximization Approach to Blind Separation and Blind Deconvolution, Neural Computation, vol. 7. (1195) 1129-1159
7. Haykin S.: Adaptive Filter Theory, Prentice Hall, Upper Saddle River, New Jersey, (1996)
8. Cardoso J.: Bind signal separation: Statistical principles, Proc. of the IEEE, vol. 86. (1998)
9. Parra L., Sajda P.: Blind Source Separation via Generalized Eigenvalue Decomposition, Journal of Machine Learning Research, vol. 4. (2003) 1261-1269