Distributed sparse BSS for large-scale datasets

Tobias I. Liaudat, Jérôme Bobin, and Christophe Kervazo
CEA, IRFU, SEDI/Service d’Astrophysique
91191 Gif-sur-Yvette Cedex, France
Email: tobisliaudat@gmail.com

Abstract—Blind Source Separation (BSS) [1] is widely used to analyze multichannel data stemming from origins as wide as astrophysics to medicine. Forthcoming projects like the SKA telescope will substantially increase the amount of data to process and existent BSS methods do not efficiently handle large datasets. In this work, we propose a new method coined DGMCA (Distributed Generalized Morphological Component Analysis) in which the original BSS problem is decomposed into subproblems that can be tackled in parallel, alleviating the large-scale issue. We propose to use the RCM (Riemannian Center of Mass – [6][7]), during the iterative process, to aggregate the estimations yielded by the different subproblems. The approach is made robust both by the chosen issue. We propose to use the RCM (Riemannian Center of Mass – [6][7]), Component Analysis) in which the original BSS problem is decomposed into subproblems of different size and that each column belongs to more than one of these subproblems. To tackle Eq. (1), the GMCA [4] algorithm has been a success, as it not only handles large-scale datasets but also substantially accelerates the computation of the estimations (cf. Fig. 3).

I. LARGE-SCALE BLIND SOURCE SEPARATION

Given m row observations of size t stacked in a matrix Y assumed to follow a linear model Y = AS + N, the objective of BSS [1] is to estimate the matrices A (size m x n) and S (size n x t) up to a mere permutation and scaling indeterminacy. In this model, A mixes the n row sources in S, the observations being disrupted by some unknown noise N (size m x t). We will assume that n ≤ m. While ill-posed, this problem can be regularized assuming the sparsity of S [2] resulting in a good separation quality [5]. The estimation will then turn into the minimization of:  
\[ \hat{A}, \hat{S} = \arg \min_{A,S} \frac{1}{2} \|Y - AS\|_F^2 + \|A \odot S\|_1 + i_\xi(X_k)\|_{\infty} = 1, \forall k(A), \]  
(1)

with \( \|\cdot\|_F \) the Frobenius norm, A the regularization parameters and \( i_\xi(\cdot) \) the indicator function of the set C. The first term is a data fidelity one, the second enforces the sparsity and the last avoids degenerated solutions with \( \|A\|_F^2 \rightarrow 0 \) by enforcing unit norm columns. To tackle Eq. (1), the GMCA [4] algorithm has been a success, partly due to an automatic decreasing parameter strategy making it robust to local minima. However, we will assume that the data Y are large-scale in the sense that t can attain huge values (e.g. up to \( 10^9 \) samples), which makes the treatment of Y as a whole intractable. In this context, using GMCA is prohibitive.

II. PROPOSED METHOD

This difficulty motivates the construction of J subproblems \( (j) \) of the type \( Y_j = A S_j + N_j \), where \( j \) denotes a subset of \( t_j \) columns of the corresponding matrices. We use disjoints sets with \( \sum_j |t_j| = t \). A natural idea is then the extension of GMCA to work in parallel on tractable smaller subproblems to minimize Eq. (1). While this approach is reminiscent to mini-batch approaches in machine learning [9], it raises two issues in the context of solving BSS through GMCA: i) each subproblem \( (j) \) yields a full estimate \( \hat{A}_{(j)} \) of A. Is it possible to aggregate them to get a better final estimate?; ii) is it possible to extend the automatic parameter choice of GMCA (that made its success) to a parallel implementation? A naive approach would be to independently solve each subproblem \( (j) \) and aggregate the different final results. However, since GMCA is an iterative algorithm, aggregating the estimations \( \hat{A}_{(j)} \) of the different subproblems \( (j) \) during the iterations should reduce the error propagation, as the estimations are shared throughout the subproblems. More specifically, our DGMCA algorithm performs the aggregation through the weighted RCM [6] of the different columns \( \hat{A}_{(j)} \) of the estimations \( \hat{A}_{(j+1)} \) yielded by the different \( (j) \) subproblems at iteration \( k+1 \), which enables to take into account the geometry of the problem, the fact that each column belongs to the unit hypersphere. Its calculation is done following a gradient descent whose convergence is assured by [7]. Roughly speaking, the RCM can be understood as a weighted angular mean on the hypersphere. To robustify this process, we further propose to compute the weights based on an estimation of the Signal-to-Noise Ratio (SNR) of the corresponding estimated sources \( \hat{s}_{i,(k+1)} \) to penalize noisy estimations (cf. Fig. 3).

Concerning question ii), the parameter choice of GMCA needs to access the whole distribution of the sources at each iteration, which is intractable in the large-scale regime. We propose a new strategy that can be used in a parallel implementation. It is based on a parametrized exponential decay which adapts to the signal statistics by using the maximum value of the estimated sources. The \( \|\cdot\|_\infty \) can be taken of each mini-batch and then the maximum of these values will give the global \( \|\cdot\|_\infty \). The expression is the following one:  
\[ \lambda_{i,(k+1)} = \lambda_{i,(k)} e^{-k \alpha_i}, \]  
(2)

where \( S_i^{(k)} \) is the i row of the source matrix, \( \sigma_{S_i,(k)} \) is the noise standard deviation of the i source (it can be estimated using the Median Absolute Deviation) and \( K \) is a constant depending on the desired noise thresholding, generally set to 3. The threshold decay rapidity is regulated by the parameter \( \alpha_i \) and can be adjusted in the first iterations by fitting a generalized Gaussian to the sources.

As hypothesis, we assume that the different sources are not correlated and that each source signal is stationary throughout the columns of S. In practice, to partially overcome the first assumption, the penalization idea in [5] can be introduced in DGMCA. Concerning the second one, each subset \( j \) of columns can be randomly chosen in each iteration aiming to have identically distributed subproblems.

III. EMPIRICAL RESULTS AND CONCLUSIONS

Numerical experiments can be found in Fig. 1 and 2, while the algorithm overview in Fig. 3. In brief, our method paves the way for distributed approaches of BSS problems with automatic parameter tuning. It not only handles large-scale datasets but it enables a linear acceleration. Furthermore, it does not lower the separation quality compared to GMCA and outperforms methods like the optimized ODL [9].

ACKNOWLEDGEMENTS

This work is supported by the European Community through the grant LENA (ERC StG - contract no. 678282).
Fig. 1. Performance comparison of the GMCA, the ODL (Online Dictionary Learning [9]), and the DGMCA and its variants. Note that the parameters of the ODL algorithm have been optimized for this experiment by an exhaustive search. The x-axis corresponds to the size $t_j$ of each subproblem (j), which is set for all $j$ to be $t_j = t/J$. The y-axis represents the separation quality, measured by a mixing matrix criterion [5] defined as the average value of the $\ell_2$ norm of the columns of $PA^* - I$, where $A^*$ is the ground truth, $P$ accounts for the correction of the permutations and $I$ is the identity matrix.

To generate the experiments, the source matrix was randomly sampled from a $\mathbb{I}_D \times \mathbb{R}^I$ matrix consisting of solving the $\ell_0$ sparse problem $(1)$, which was expected due to the lack of statistics for the algorithm to work. The experiment was run using a C++ parallelized version of the DGMCA algorithm and the ODL with its hyperparameters optimized. The computer cluster used had the setup of the experience is similar to the one in Fig 1, with a $\beta$ parameter of 0.5, having $n = 5$ sources, $t = 10000$ samples and a $SNR$ of 40dB. The linear trend of the time gain was predicted by the complexity analysis of the algorithms, and now confirmed by the numerical experiment. In brief, the time spent by the DGMCA is approximately the one spent in the GMCA algorithm divided by the number of subproblems used.

Fig. 2. Computational time gain between the parallelized DGMCA and the GMCA algorithms against the data reduction ratio, which is calculated as the problem total size, $t$, divided by the size of the mini-batch, $t/t_j = J$. Each point on the figure represents the mean over 10 problems. The experiment was run using a C++ parallelized version of the DGMCA algorithm and the maximum number of mini-batches used is 40 as it is the number of cores the computer cluster used had. The setup of the experience is similar to the one in Fig 1, with a $\beta$ parameter of 0.5, having $n = 5$ sources, $t = 10000$ samples and a $SNR$ of 40dB. The linear trend of the time gain was predicted by the complexity analysis of the algorithms, and now confirmed by the numerical experiment. In brief, the time spent by the DGMCA is approximately the one spent in the GMCA algorithm divided by the number of subproblems used.

Fig. 3. DGMCA. The operator $(\cdot)\dagger$ is the pseudo-inverse, $S_A(\cdot)$ is the soft-thresholding operator with the threshold $\lambda$, $a_i$ denotes the ith column of $A$ and the subscript (j) denotes the estimation of the j subproblem.

REFERENCES


