IADIS International Journal on Computer Science and Information Systems
Vol. 4, No. 1, pp. 21-29
ISSN: 1646-3692

Acceleration of the gradient-type methods in blind source separation[1]

David Blanco*, Diego P. Ruiz*, María C. Carrion*, and Carlos García-Puntonet+

*Department of Applied Physics, University of Granada, Spain
+Department of Computer Architecture and Computer Technology , University of Granada, Spain

Abstract

The steepest descendent is the non-linear optimiza-tion method most used in ICA algorithms. The method is used to find the unmixing matrix, which solves the problem and is a minimum of a non-linear cost function. In this paper the use of quasi-Newton optimization methods, instead of gradient-type ICA methods, is studied. These methods can increase the speed of the method what it is corroborated by simulations.

Keywords: radar target recognition, natural resonances, principal component analysis, neural network, matrix pencil method.

1 Introduction

Blind Source Separation (BSS) is usually solved by the Independent Component Analysis (ICA) through the optimization of a cost function, which measures the statistical dependence between of the outputs of a linear transformation of the data. This is based in the fact that the outputs are statistically independent when the linear transformation is the inverse of the original mixture, which is called the unmixing matrix. Then, this transformation forms part of the unknowns of the const function, which is built such as its minimum or maximum is reached when the linear transformation is the unmixing matrix. As a maximization of a cost function can be easily turned into a minimization problem, just changing the sign of the cost function, it is assumed in the rest of the paper that the optimization is a minimization problem, without losing generality.

The cost function is a non-linear function, therefore it is necessary to resort to non-linear minimization algorithm to find its minimum. The most used family are the gradient algorithms, which use the gradient of the cost function in each point to find the

1 This work was supported in part by the Spanish Ministry of Science and Technology under Project TEC2007-68030-C02-02/TCM.

direction of maximum variation in the space of the unknowns, moving to smaller values of the cost function and reaching, in this way, a minimum in iterative steps. If the cost function, the unmixing matrix and the gradient of the cost function are denoted as J, W and ∇J, respectively, the unmixing matrix is updated in each step as $W_{k+1}=W_k-\mu_k\nabla J_k$, where μ_k is the size of the k step (the subscript k indicates the step where the variable is calculated). The step size can be found using a linear search, be fixed or change following a predetermined pattern. This method, called the steepest descendant method, can be really slow, especially near the solution, and other methods are generally preferred to speed up the convergence.

In ICA, the method preferred to increase the speed consists in using the natural gradient [1] instead the standard gradient. This method studies the geometry of the space of unknowns, which is generally non-Euclidean, to find the direction of maximum variation, that could be viewed as a "geodesic". In general, this scheme is a very complicate one, needing Riemann geometry analysis and does not produce simple results; but in ICA, as the unknowns posses a special structure (they are a multiplicative group), the natural gradient can be found and has a very simple and attractive form. Specifically, the update formula is $W_{k+1}=W_k-\mu(\nabla J_k)W_k^H W_k$, where the step size is usually kept fixed to a certain value. The convergence of the natural gradient method is much faster than in the standard gradient method, although the value of step size is a key, and sometimes problematic, issue in the speed. A too big step size can make the method to diverge or oscillate around the minimum, and a too small one makes the method too slow. The problem has been solved with the approach of fixed-point algorithms, as FastICA [2], leading to much faster algorithm without the use of step size. Some problems appear though in the fixed-point algorithm, as the necessity of a prewhitening in the data and that they are offline algorithms. Although the prewhitening is a general preprocessing step in many ICA methods, there are situations, as the existence of non negligible additive noise, where it is not an optimum procedure, and the fix-point approach is not the best one. On the other part, when algorithms have to work online, due for example to a fast variation in the system, gradient algorithms are preferable. In any case, this paper is concerning to gradient-type algorithms and how they can be accelerated, without going into if they are preferred to fixed-point ones.

There are other options to minimize the cost function, although, they are not generally used by the ICA community. For example, linear searching methods that do not use information about the derivates, as the simplex algorithm, can be used. However, their convergence near the solution is slow and the speed decreases dramatically when the number of unknowns and the complexity of the cost function increase. In general, these methods are much slower than the natural gradient and are not used.

Other option is to resort to second-order learning, instead to first-order as in the gradient algorithm. The gradient methods, natural or standard, assume a linear approximation in the proximity of each point to find the smallest value of the cost function in this proximity. The second-order learning assumes a quadratic approximation and therefore can find the smallest value of the cost function in a bigger neighborhood of each point. The best known one of these algorithms is the Newton method. For a cost function J(b) depending on the vector of unknowns b (it can be, for example, $b_{i+(j-1)N}=W_{ij}$ for a N×N unmixing matrix), the Newton method's update formula is $b_{k+1}=b_k-\mu_k(\nabla^2 J_k)^{-1}J_k$, where

∇J_k is the gradient evaluated in the point b_k, i.e. $[\nabla J_k]_i = \partial J / \partial b_i \,|_{\mathbf{b}=\mathbf{b}_k}$, and the matrix $\nabla^2 J_k$ is the Hessian of the cost function evaluated in the same point, i.e. $[\nabla^2 J_k]_{ij} = \partial^2 J /(\partial b_i \partial b_j)\,|_{\mathbf{b}=\mathbf{b}_k}$. When this algorithm is applicable, it is usually much faster than any gradient algorithm, however it suffers of serious technical problems. One is the necessity of computing the Hessian and then inverting it, in each iteration, which increases the complexity of each step. But the biggest problems are the appearance of spurious local minimum in the convergence (see, for example, [1]) and the possibility for the inverse to not be definite at a point, which is the result of a non-full rank Hessian matrix. When any of this happens, the method stops in the local minima or diverges. As the existence of a minimum in the cost function only assures a full Hessian at the minimum, non full rank Hessian will, in general, appear.

On the other hand, the quasi-Newton methods are also second-order learning methods but, instead the compute the Hessian at each point and then invert it, they estimate directly the inverse of the Hessian using the value of two consecutive points and the gradient at these two points, constraining the matrix to be symmetric and full rank. The algorithms so built are faster than the gradient type and they do not posses the drawbacks of the Newton method. The complexity of each step is still bigger than in the gradient methods, but can be compensated with the increasing in the speed, which makes them interesting alternatives to the gradient methods. This is precisely what this paper proposes: to study the use of quasi-Newton methods in ICA. In Sect. 2, the model and a general overview of the most used cost functions are stated. In Sect. 3, the quasi-Newton methods are presented. In Sect. 4, the theoretical analysis is verified in simulations, where the quasi-Newton methods are compared with well known gradient methods, to test if, in fact, the speed is increased.

2 ICA model and cost functions

Although the substitution of the natural gradient method by a quasi-Newton method, can be valid in multichannel deconvolution, non-linear ICA or underdetemined ICA problems, if they obtain their solutions by the minimization of a nonlinear cost function, this paper is restricted to instantaneous ICA, such as the data are the instantaneous linear mixture of independent sources. This restriction is done to test the ideas in widely known algorithms, but the study can be easily extended to more sophisticated models and problems. Then, the $N\times 1$ vector of data $\mathbf{x}$ is assumed to be

$$\mathbf{x} = \mathbf{A}\mathbf{s} \tag{1}$$

where the $N\times N$ matrix $\mathbf{A}$ is the mixing matrix and $\mathbf{s}$ are the original independent sources. Real data and same number of sources and data are also assumed for simplicity, but the generalization to complex data and more data than sources is straightforward.

It is clear that the entries of $\mathbf{y}=\mathbf{C}\mathbf{s}$ are independent if the matrix C, called the *global transformation*, is a scaled-permutated version of the identity matrix, i.e. the entries $\mathbf{y}$

are equal to the original sources except for a scale or permutation.

This is used by the ICA methods, since they search for the linear transformation **W** of the data **x** such as the outputs are independent, and the obtained matrix is the inverse of the mixing matrix, except for a possible permutation or scale in their rows. It is only necessary a measure of the independence between the components of **y**. This is usually done by a cost function $J[\mathbf{y}]$ that can be function only of **W** or also of the Probability Density Function (PDF) of the original sources. The first case arises when a PDF of the sources is assumed or when higher-order approximations are done in the PDF-dependant cost functions. In the second case, the PDFs are not supposed known, but, in general, some model with free parameters is assumed to be followed by the PDFs, such as the free parameters are also unknowns of the problem. In each step, the parameters of the PDF are estimated using the information of the previous step, and with them fixed, $\mathbf{W}_k$ is updated with the natural gradient. This scheme is suboptimal, because does not minimize the cost function respect all the parameters, but is needed if we want to use the natural gradient, since the inclusion of all the unknowns together would break the multiplicative group structure of the non-singular matrices. The simplest example in the parametrization of the PDFs can be found in [3], where a binary parameter is used to model the PDFs. The cost function is a real function that possesses the property $J[\mathbf{Cs}]\leq J[\mathbf{s}]$ with the equality only when **C** is a permutated-scaled version of the identity. The most common cost functions are the likelihood, the Kullback divergence, the mutual information and the negentropy, or their approximations using higher-order statistic (see [4] for an overview of the most common cost functions and the relation between them).

In any case, the natural gradient method is used to minimize the cost function respect to the unmixing matrix, although other unknowns may be involved and are estimated suboptimally in each step.

3 Quasi-Newton Methods

The gradient algorithm are based in a Taylor expansion up to first order of J, such as $J(\mathbf{b}_k+\mathbf{c})\approx J(\mathbf{b}_k)+(\nabla J_k)^T\mathbf{c}$. Then, the direction of maximum variation of Jin the step k is $-\nabla J_k/\|\nabla J_k\|$, so the adaptation rule is selected as $\mathbf{b}_{k+1}=\mathbf{b}_k-\mu_k\nabla J_k$. In the Newton method, on the other hand, a Taylor expansion up to second order is done, such as the cost function near a point b_k is $J(\mathbf{b}_k+\mathbf{c})\approx J(\mathbf{b}_k)+(\nabla J_k)^T\mathbf{c}+\mathbf{c}^T\nabla^2 J_k\mathbf{c}/2$. The value of **c** that produces the maximum descent in the step k is $-(\nabla^2 J_k)^{-1}\nabla J_k$, and the vector of unknowns is updated as $\mathbf{b}_{k+1}=\mathbf{b}_k-\mu_k(\nabla^2 J_k)^{-1}\nabla J_k$, where the step size has been added, as in the previous case, to take into account that the approximation is not exact. As it was pointed before, the Newton method converges if it can find a decreasing direction in each point it passes through, and this happens if the Hessian is positive definite at all these points. But if the Hessian is not positive definite at any point, the Newton direction either is not definite or does not provide a decrease in the cost function. It is possible to modify the method to overcome this limitation (generally adding to the Hessian a matrix that follows some properties when it is not positive definite) but still

the method needs to compute the second derivates to built the Hessian and then invert it in each step, which increases notably the complexity of each step.

The quasi-Newton methods follow also a quadratic approximation of the cost function, but using a symmetric positive definite matrix instead of the Hessian, i.e. $J(\mathbf{b}_k+\mathbf{c})\approx J(\mathbf{b}_k)+(\nabla J_k)^T\mathbf{c}+\mathbf{c}^T\mathbf{B}_k\mathbf{c}/2$, where the matrix $\mathbf{B}_k$ is updated in each iteration imposing some conditions, constrained to be symmetric and positive definite. The first of this conditions is that the gradient of the cost function built with the quadratic approximation should be equal to the true gradient in two consecutive points, this is hold if $\mathbf{B}_k\mathbf{p}_k=\mathbf{q}_k$, with $\mathbf{p}_k=\mathbf{b}_k-\mathbf{b}_{k-1}$and $\mathbf{q}_k=\nabla J_k-\nabla J_{k-1}$, which is called the *secant condition*. The other condition is that the two matrices in consecutive steps have to be the closest in some sense. Different measures of the distance between the matrices produce different quasi-Newton method. This same scheme can be done over the inverse of $\mathbf{B}_k$, which is noted $\mathbf{H}_k$ and is also forced to be symmetric a positive definite. Specifically, the secant condition for this matrix is $\mathbf{H}_k\mathbf{p}_k=\mathbf{q}_k$. This is what is done in the most extended quasi-Newton method, the BFGS (Broyden-Fletcher-Goldfarb-Shannon), where the condition of two consecutives matrices being the closest is imposed over $\mathbf{H}_k$ and the resulting update formula for $\mathbf{H}_k$ is [5]:

$$\mathbf{H}_k = (\mathbf{I}-u_k\mathbf{p}_k\mathbf{q}_k^T)\mathbf{H}_{k-1}(\mathbf{I}-u_k\mathbf{p}_k\mathbf{q}_k^T)+u_k\mathbf{p}_k\mathbf{p}_k^T \tag{2}$$

with $u_k=1/(\mathbf{q}_k^T\mathbf{p}_k)$ and the update formula for $\mathbf{b}_k$ is:

$$\mathbf{b}_{k+1}=\mathbf{b}_k-\mu_k\mathbf{H}_k\nabla J_k \tag{3}$$

The step size μ_k is found through a linear search along the direction $-\mathbf{H}_k\nabla J_k$. As it can be seen, the BFGS method is very easy to program and the complexity of each step is small, as it does not need the inversion of matrices nor the computation of second derivates. On the other hand, it keeps a big part of the speed of the Newton method, due to the quadratic scheme it follows. In the next section, the BFGS method is compared with natural gradient methods, both offline and on-line, to demonstrate that they are faster in practical situations

4 Results

In this section the comparison of the Infomax method for supergaussian sources [6] with the BFGS method is done, in order to study the behavior of the quasi-Newton methods, working both off-line and on-line.

The classical Infomax method proposes a cost function as:

$$J = -\log(|\det(W)|) + 2\sum_{i=1}^{N} E\{\cosh(y_i)\} \tag{4}$$

so, with the natural gradient, the updated expression for $\mathbf{W}_k$ is:

$$\mathbf{W}_{k+1} = \mathbf{W}_k - \mu(\mathbf{I} - 2\tanh(\mathbf{y}^T)\mathbf{y})\mathbf{W}_k \tag{5}$$

where μ is kept fixed in all the iterative process. The BGFS is used as a quasi-Newton method to minimize the same cost function, where μ_k is obtained through a linear search along the direction $-\mathbf{H}_k\nabla J_k$. When the algorithm works off-line, an inexact linear search is preferred, since the times the cost function has to be evaluated along the line is reduced and, therefore, the speed increased. On the other hand, when the algorithms works on-line this would not be the better option and an exact linear search would be more adequate. As the behavior of the algorithm depends on the off-line or on-line character, both situations are studied separately. The data will consist in the linear mixture of six laplacian sources with mean zero, variance one and different shape. The number of data per source is M=100,000. The mixing matrix is generated randomly with the only condition to be non-singular.

4.1 Off-line study

In this case all the M data of signals are assumed known and the algorithm works with the whole set. The cost function J only depends on the unmixing matrix, since the data $\mathbf{x}$ are fixed through all the iterative minimization process. In this situation, the time spent by BFGS and Infomax to converge is compared. The estimation of the method in the step k is characterized by the minimum distance of the global transformation in that step $\mathbf{C}_k$=$\mathbf{W}_k\mathbf{A}$, to the identity matrix or any permutated scaled version of it. This is measure by a parameter E, whose explicit expression can be found in [3] as:

$$E = \sum_{i=1}^{N}\left(\sum_{i=1}^{N}\frac{|C_{ij}|}{\max|C_{ik}|} - 1\right) + \sum_{j=1}^{N}\left(\sum_{i=1}^{N}\frac{|C_{ij}|}{\max|C_{kj}|} - 1\right) \tag{6}$$

To finish the BFGS method it is necessary to say how the search for μ_k is done along the line $-\mathbf{H}_k\nabla J_k$, in each step. Two different approaches are studied. The first is to do an inexact search along the line, only satisfying some weak conditions (the Wolfe's conditions [5]) to assure the convergence, so J and ∇J are evaluated few times along the line and the speed is increased. The second approach is to do an exact linear search, using the golden rule algorithm, for example. This means evaluate J much more times, slowing down the method but needing fewer steps. The parameter E as function of the time for Infomax, BFGS with inexact linear search and BFGS with exact search, are shown in Figure 1. Although BFGS with exact linear search converges in the smallest number of iterations, the time consumed in each iteration is much bigger than the time

used by Infomax, and it results that, in time, Infomax algorithm is faster. Of course, if the comparison would have been done in iterations instead of time, BFGS with exact search would have shown that converges in much less iterations than Infomax, although it is the time and not the number of iteration what it is interesting to reduce working off-line. In any case, BFGS with inexact search converges in much less time than the two others, showing that it is quite faster than the Infomax methods working off-line. The number of iterations necessary for BFGS with exact and with inexact linear search is similar.

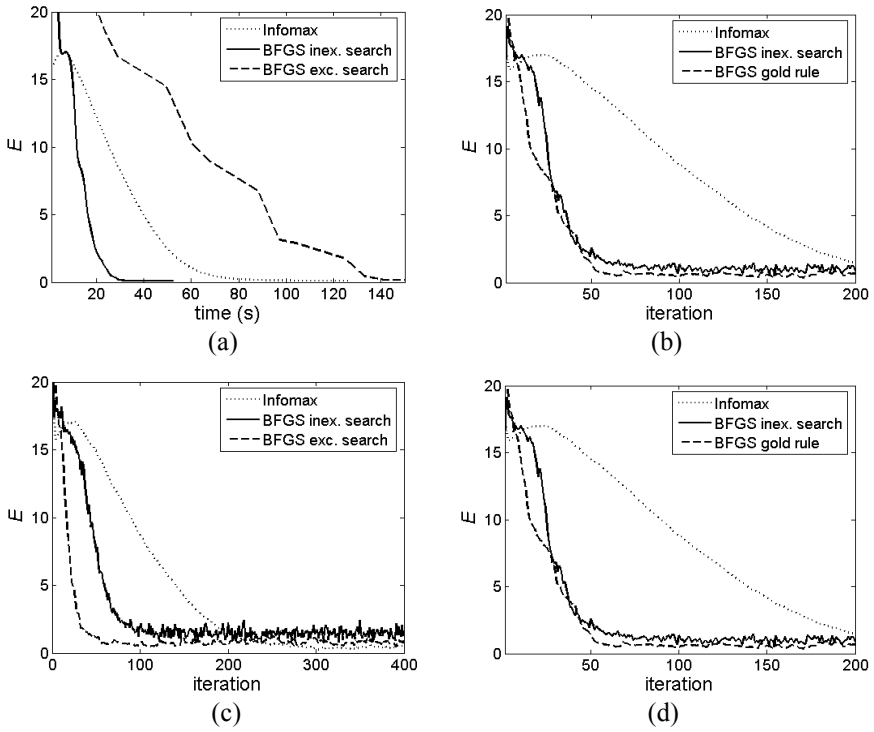

Fig. 1. Parameter *E* as function of time (a) working off-line (a) working off line and (d) working on-line for *P*=500; as function of iteration for (b) *P*=500 and (c) *P*=250, working on-line.

4.2 On-line study

In this case the *M* vectors of data are assumed to appear in segments of *P* data, for a total of *M*/*P* segments. The unmixing matrix is estimated in the step k using the data in the *k* segment. In this case the cost function J changes from iteration to iteration, not only due to the change in unmixing matrix, but also to the change of **x** in each segment.

As it is well known, this change in the data in each segment contributes to increase the value of E in the steady-state. This increase is bigger for BFGS than for Infomax, since the first uses two consecutive steps while the second only uses one step in the update process, and therefore it would be more sensible to difference between segments.

In this case, it is wanted the methods to converge in few iterations, since then they will be able to deal with faster change in non-stationary signals or will need a smaller amount of data to solve a problem. In Figure 1(a), the two BFGS options and Infomax are compared for P=500 as function of the number of iteration. It can be seen that both BFGS methods converge in quite less iterations, with BFGS with exact linear search usually converging in a bit less iterations although the time consumed in each iteration by this method is much bigger than for BFGS with inexact search. This is explicitly shown in Figure 1(c), where the parameter E is shown in the same case, but as function of the time, instead of the iteration. In this last figure, it can also be seen how the time consuming by BFGS with inexact linear search is similar to the time of Infomax. The bigger the number of data per segment the greater the difference between the iterations needed by the BFGS methods and Infomax.

The principal problem of BFGS with inexact linear search is that the steady state presents a bigger value of E and bigger variance than for Infomax. In BFGS with exact search, the value and the variance of E in the steady-state is a bit reduced, as it can be in Figure 1(a), and more clearly in Figure 1(b), where the on-line study has been done with P=250. The smaller the number of data per segment, the bigger the value of E in the steady-state for the BFGS algorithms and its variance, the bigger the difference between BFBS with exact and inexact search, and smaller the difference between the number of iterations needed by the BFGS methods and Infomax.

This study has been repeated for different cost functions, numbers of data in the off-line case and numbers of data per segment in the on-line case, and the results can be considered general. Broadly, the more complicate the cost function, the bigger the acceleration the convergence gets when the natural gradient minimization method is substituted by a quasi-Newton method. The fact is very much accentuated in the case that natural gradient can not be used, as is the case of [7].

5 Conclusion

In this paper has been shown that the quasi-Newton method can be an option to the gradient minimization methods in BBS and ICA. If the method works off-line the quasi-Newton methods converge faster than the natural gradient algorithms, and if it does on-line, the quasi-Newton methods converge in a minor number of iterations. The time necessary in this second case is similar, although the quasi-Newton methods present a worse behaviour in the steady state

References

[1] S. Douglas and S. Amari, "Natural-gradient adaptation," in *Unsupervised adaptative filtering*, S. Haykin, Ed. John Wiley & Sons, 1998, vol. I Blind source separation, ch. 1.

[2] A. Hayvärinen and E. Oja, "A fast fixed-point algorithm for independent component analysis," *Neural Computation*, vol. 9, pp. 1483–1492, 1997.

[3] T. W. Lee, M. Girolami, and T. J. Sejnowski, "Independent component analysis using an extended Infomax algorithm for mixed subgaussian and supergaussian sources," *Neural Computation*, vol. 11, pp. 417–441, 1999.

[4] J.-F.Cardoso, "Blind signal separation: statistical principles," *Proceedings of the IEEE*, vol. 86, pp. 2009–2025, 1998.

[5] J. Nocedal and S. J. Wright, *Numerical optimization*. Springer, 2000.

[6] A. J. Bell and T. J. Sejnowski, "An information-maximization approach to blind separation and blind deconvolution," *Neural Computation*, vol. 37, pp. 1129–1159, 1997.

[7] D. Blanco, B. Mulgrew, D. P. Ruiz, and M. C. Carrion, "ICA in signals with multiplicative noise using fourth-order statistic," in *Proc.of EUSIPCO*, 2005.