

7 Relative Newton and Smoothing Multiplier Optimization Methods for Blind Source Separation*

Michael Zibulevsky

Department of Computer Science, Technion, Haifa 32000, Israel
E-mail: mzib@cs.technion.ac.il

Abstract. We study a relative optimization framework for quasi-maximum likelihood blind source separation and relative Newton method as its particular instance. The structure of the Hessian allows its fast approximate inversion. In the second part we present Smoothing Method of Multipliers (SMOM) for minimization of sum of pairwise maxima of smooth functions, in particular sum of absolute value terms. Incorporating Lagrange multiplier into a smooth approximation of max-type function, we obtain an extended notion of non-quadratic augmented Lagrangian. Our approach does not require artificial variables, and preserves the sparse structure of Hessian. Convergence of the method is further accelerated by the Frozen Hessian strategy. We demonstrate efficiency of this approach on an example of blind separation of sparse sources. The non-linearity in this case is based on the absolute value function, which provides super-efficient source separation.

7.1 Introduction

In this chapter we study quasi-maximum likelihood blind source separation (quasi-ML BSS) [1,2] in batch mode, without orthogonality constraint. This criterion provides improved separation quality [3,4], and is particularly useful in separation of sparse sources. We will present optimization methods, which produce quasi-ML BSS efficiently.

7.1.1 Quasi-ML blind source separation (BSS)

Consider the BSS problem, where an N -channel sensor signal $x(t)$ arises from N unknown scalar source signals $s_i(t)$, $i = 1, \dots, N$, linearly mixed together by an unknown $N \times N$ matrix A

$$x(t) = As(t). \tag{7.1}$$

We wish to estimate the mixing matrix A and the N -dimensional source signal $s(t)$. In the discrete time case $t = 1, 2, \dots, T$ we use matrix notation

* Chapter in the book: S. Makino, T.W. Lee and H. Sawada (Eds.) *Blind Speech Separation*, Springer Series: Signals and Communication Technology XV, 2007

$X = AS$, where X and S are $N \times T$ matrices with the signals $x_i(t)$ and $s_i(t)$ in the corresponding rows. We also denote the unmixing matrix $W = A^{-1}$.

When the sources are *i.i.d.*, stationary and white, the normalized minus-log-likelihood of the observed data X is (see for example [4])

$$L(W; X) = -\log |\det W| + \frac{1}{T} \sum_{i,t} h(W_i x(t)), \quad (7.2)$$

where W_i is i -th row of W , $h(\cdot) = -\log f(\cdot)$, and $f(\cdot)$ is the probability density function (pdf) of the sources. Consistent estimator can be obtained by minimization of (7.2), also when $h(\cdot)$ is not exactly equal to $-\log f(\cdot)$. Such *quasi-ML estimation* is practical when the source pdf is unknown, or is not well-suited for optimization. For example, when the sources are sparse or sparsely representable, the absolute value function or its smooth approximation is a good choice for $h(\cdot)$ [5–10]. Here we will use a family of convex smooth approximations to the absolute value

$$h_1(c) = |c| - \log(1 + |c|) \quad (7.3)$$

$$h_\lambda(c) = \lambda h_1(c/\lambda) \quad (7.4)$$

with λ a proximity parameter: $h_\lambda(c) \rightarrow |c|$ as $\lambda \rightarrow 0^+$. Widely accepted natural gradient method does not work well when the approximation of the absolute value becomes too sharp. In this work we consider the relative Newton method, which overcomes this obstacle.

The Newton equations considered in this work are similar in part to those obtained by Pham and Garat [1], using different considerations. However, the algorithm given in [1], is not used in practice, because of a possibility of convergence to spurious solutions. We overcome this difficulty using line search and forcing positive definiteness of the Hessian.

Several other Newton-like BSS methods have been studied in the literature. They are based on negentropy approximation with orthogonality constraint [11], cumulant model [12,13] and joint diagonalization of correlation matrices [14–17].

The relative Newton method presented here is dedicated to quasi-ML BSS in general (not only to the sparse source case).

7.1.2 Smoothing Method of Multipliers (SMOM) for Sum-Max problems

In the second part we present a method for minimization of a sum of pairwise maxima of smooth functions, in particular sum of absolute value terms, arising in quasi-ML BSS.

Methods of multipliers, involving *nonquadratic augmented Lagrangians* [18–27] successfully compete with the interior-point methods in non-linear and semidefinite programming. They are especially efficient when a very high

accuracy of solution is required. This success is explained by the fact, that due to iterative update of multipliers, the penalty parameter does not need to become extremely small in the neighborhood of solution.

Direct application of the augmented Lagrangian approach to the sum-max problem requires introduction of artificial variables, one per element of the sum, that significantly increases the problem size and the computational burden. Alternatively, one can use a smooth approximation of the max-type function [28,29], which will keep the size of the problem unchanged, but will require the smoothing parameter to become extremely small in order to get accurate solution.

In this work we incorporate multiplier into a smooth approximation of the max-type function, obtaining an extended notion of augmented Lagrangian. This allows us to keep the size of the problem unchanged, and achieve a very accurate solution under a moderate value of the smoothing parameter. Convergence of the method is further accelerated by the Frozen Hessian strategy.

We demonstrate the efficiency of this approach on an example of blind separation of sparse sources with the absolute value non-linearity. It preserves sparse structure of the Hessian, and achieves 12 – 15 digits of source separation accuracy.

7.2 Relative Optimization Algorithm

We consider the following algorithm for minimization of the quasi-ML function (7.2)

- Start with an initial estimate W_1 of the separation matrix;
- **For** $k = 1, 2, \dots$, until convergence
 1. Compute the current source estimate $U_k = W_k X$;
 2. Starting with $V=I$, get V_{k+1} by one or few steps of a conventional optimization method, decreasing sufficiently $L(V; U_k)$;
 3. Update the estimated separation matrix $W_{k+1} = V_{k+1} W_k$;
- **End**

The relative (natural) gradient method [30–32] is a particular instance of this approach, when a standard gradient descent step is used in Item 2. The following remarkable property of the relative gradient is also preserved in general: *given current source estimate U , the progress of the method does not depend on the original mixing matrix.* This means that even nearly ill-conditioned mixing matrix influences the convergence of the method not more than a starting point. Convergence analysis of the Relative Optimization algorithm is presented in Appendix A. In the following we will use a Newton step in Item 2 of the method.

7.3 Hessian evaluation

The likelihood $L(W; X)$ is a function of a matrix argument W . The corresponding gradient is also a matrix

$$G(W) = \nabla L(W; X) = -W^{-T} + \frac{1}{T} h'(WX) X^T, \quad (7.5)$$

where $h'(WX)$ is a matrix with the elements $h'((WX)_{ij})$. The Hessian of $L(W; X)$ is a linear mapping \mathcal{H} defined via the differential of the gradient

$$dG = \mathcal{H}dW. \quad (7.6)$$

We can also express the Hessian in standard matrix form converting W into a long vector $w = \text{vec}(W)$ using row stacking. We will denote the reverse conversion $W = \text{mat}(w)$. Let

$$\hat{L}(w, X) \equiv L(\text{mat}(w), X), \quad (7.7)$$

so that the gradient

$$g(w) = \nabla \hat{L}(w; X) = \text{vec}(G(W)) \quad (7.8)$$

Then

$$dg = Hdw, \quad (7.9)$$

where H is an $N^2 \times N^2$ Hessian matrix. We also have

$$dg = \text{vec}(dG) \quad (7.10)$$

7.3.1 Hessian of $-\log \det W$

Using the expression

$$d(W^{-1}) = -W^{-1}(dW)W^{-1},$$

which follows from the equality

$$0 = d(WW^{-1}) = (dW)W^{-1} + Wd(W^{-1}),$$

we obtain the differential of the first term in (7.5)

$$dG = d(-W^{-T}) = A^T(dW^T)A^T, \quad (7.11)$$

where $A = W^{-1}$. A particular element of the differential is determined using A^i and A_j , i -th column and j -th row of A respectively:

$$dG_{ij} = A^{iT} dW^T A_j^T = \text{Trace} A^{iT} dW^T A_j^T = \text{Trace} A_j^T A^{iT} dW^T, \quad (7.12)$$

or

$$dG_{ij} = \text{Trace} A^i A_j dW = \langle (A^i A_j)^T, dW \rangle \quad (7.13)$$

On the other hand, (7.9) and (7.10) gives us

$$dG_{ij} = dg_k = \langle H_k, dw \rangle, \quad (7.14)$$

where H_k – k -th row of H , $k = (i-1)N + j$. Comparing the last two equations, we conclude that the k -th row of H contains the matrix $(A^i A_j)^T$ stacked row-wise

$$H_k = (\text{vec}(A^i A_j)^T)^T. \quad (7.15)$$

7.3.2 Hessian of $\frac{1}{T} \sum_{m,t} h(W_m x(t))$

It is easy to see that the Hessian of the second term in $\hat{L}(w, X)$ is a block-diagonal matrix with the following $N \times N$ blocks

$$B^m = \frac{1}{T} \sum_t h''(W_m x(t)) x(t) x^T(t), \quad m = 1, \dots, N \quad (7.16)$$

7.4 Newton Method

Newton method is an efficient tool of unconstrained optimization. It often converges fast and provides quadratic rate of convergence. However, its iteration may be costly, because of the necessity to compute the Hessian matrix and solve the corresponding system of equations. In the next section we will see that this difficulty can be overcome using the relative Newton method.

First, let us consider the standard Newton approach, in which the direction is given by solution of the linear equation

$$Hy = -\nabla \hat{L}(w; X) \quad (7.17)$$

where $H = \nabla^2 \hat{L}(w; X)$ is the Hessian of (7.7). In order to guarantee descent direction in the case of nonconvex objective function, we use modified Cholesky factorization¹ [33], which automatically finds a diagonal matrix R such that the matrix $H + R$ is positive definite, providing a solution to the modified system

$$(H + R)y = -\nabla \hat{L}(w; X) \quad (7.18)$$

After the direction y is found, the new iterate w^+ is given by

$$w^+ = w + \alpha y \quad (7.19)$$

¹ We use the MATLAB code of modified Cholesky factorization by Brian Borchers, available at <http://www.nmt.edu/~borchers/ldlt.html>

where the step size α is determined by exact line search

$$\alpha = \arg \min_{\alpha} \hat{L}(w + \alpha y; X) \quad (7.20)$$

or by a backtracking line search [33]:

$\alpha := 1$

While $\hat{L}(w + \alpha y; X) > \hat{L}(w; X) + \beta \alpha \nabla \hat{L}(w; X)^T d$

$\alpha := \gamma \alpha$

end

where $0 < \alpha < 1$ and $0 < \gamma < 1$. The use of line search guarantees monotone decrease of the objective function at every Newton iteration. In our computations the line search constants were $\beta = \gamma = 0.3$. It may also be reasonable to give β a small value, like 0.01.

Computational complexity. The Hessian is a $N^2 \times N^2$ matrix; its computation requires N^4 operations in (7.15) and $N^3 T$ operations in (7.16). Solution of the Newton system (7.18) using modified Cholesky decomposition, requires $N^6/6$ operations for decomposition and N^4 operations for back/forward substitution. All together, we need

$$2N^4 + N^3 T + N^6/6$$

operations for one Newton step. Comparing this to the cost of the gradient evaluation (7.5), which is equal to $N^2 T$, we conclude that the Newton step costs about N gradient steps when the number of sources is small (say, up to 20). Otherwise, the third term become dominating, and the complexity grows as N^6 .

7.5 Relative Newton Method

In order to make the Newton algorithm invariant to the value of mixing matrix, we use the relative Newton method, which is a particular instance of the Relative Optimization algorithm. This approach simplifies the Hessian computation and the solution of the Newton system.

7.5.1 Basic relative Newton step

The optimization in Item 2 of the Relative Optimization algorithm is produced by a single Newton-like iteration with exact or backtracking line search. The Hessian of $L(I; U)$ has a special structure, which permits fast solution of the Newton system. First, the Hessian of $-\log \det W$ given by (7.15), becomes very simple and sparse, when $W = A = I$: each row of H

$$H_k = \text{vec}^T(e_i e_j^T), \quad (7.21)$$

contains only one non-zero element, which is equal to 1. Here e_j is an N -element standard basis vector, containing 1 at j -th position. The remaining part of the Hessian is block-diagonal. There are various techniques for solving sparse symmetric systems. For example, one can use sparse modified Cholesky factorization for direct solution, or alternatively, conjugate gradient-type methods, possibly preconditioned by incomplete Cholesky factor, for iterative solution. In both cases, the Cholesky factor is often not as sparse as the original matrix, but it becomes sparser, when appropriate matrix permutation is applied before factorization (see for example MATLAB functions CHOLINC and SYMAMD.)

7.5.2 Fast relative Newton step

Further simplification of the Hessian is obtained by considering its structure at the solution point $U_k = S$. The elements of the m -th block of the second term of $\nabla^2 L(I; S)$ given by (7.16), are equal to

$$B_{ij}^m = \frac{1}{T} \sum_t h''(s_m(t)) s_i(t) s_j(t), \quad i, j = 1, \dots, N.$$

When the sources are independent and zero mean, we have the following zero expectation

$$E\{h''(s_m(t)) s_i(t) s_j(t)\} = 0, \quad m, i \neq j,$$

hence the off-diagonal elements B_{ij}^m converge to zero as sample size grows. Therefore we use a diagonal approximation of this part of the Hessian

$$B_{ii}^m = \frac{1}{T} \sum_t h''(u_m(t)) u_i^2(t), \quad i = 1, \dots, N; \quad m = 1, \dots, N, \quad (7.22)$$

where $u_m(t)$ are current estimates of the sources. In order to solve the simplified Newton system, let us return to the matrix-space form (7.6) of the Hessian operator. Let us pack the diagonal of the Hessian given by (7.22) into $N \times N$ matrix D , row-by-row. Taking into account that $A = I$ in (7.11), we will obtain the following expression for the differential of the gradient

$$dG = \mathcal{H}dW = dW^T + D \odot dW, \quad (7.23)$$

where “ \odot ” denotes element-wise multiplication of matrices. For an arbitrary matrix Y ,

$$\mathcal{H}Y = Y^T + D \odot Y. \quad (7.24)$$

In order to solve the Newton system

$$Y^T + D \odot Y = G, \quad (7.25)$$

we need to solve $N(N-1)/2$ systems of size 2×2 with respect to Y_{ij} and Y_{ji}

$$\begin{aligned} D_{ij}Y_{ij} + Y_{ji} &= G_{ij}, \quad i = 1, \dots, N; \quad j = 1, \dots, i-1 \\ D_{ji}Y_{ji} + Y_{ij} &= G_{ji}. \end{aligned} \quad (7.26)$$

The diagonal elements Y_{ii} can be found directly from the set of single equations

$$D_{ii}Y_{ii} + Y_{ii} = G_{ii}. \quad (7.27)$$

In order to guarantee a descent direction and avoid saddle points, we modify the Newton system (7.26), changing the sign of the negative eigenvalues [33]. Namely, we compute analytically the eigenvectors and the eigenvalues of 2×2 matrices

$$\begin{pmatrix} D_{ij} & 1 \\ 1 & D_{ji} \end{pmatrix},$$

invert the sign of the negative eigenvalues, and force small eigenvalues to be above some threshold (say, 10^{-8} of the maximal one in the pair). Then we solve the modified system, using the eigenvectors already obtained and the modified eigenvalues.

Computational complexity. Computing the diagonal of the Hessian by (7.22) requires N^2T operations, which is equal to the cost of the gradient computation. Solution cost of the set of 2×2 linear equations (7.26) is about $15N^2$ operations, which is negligible compared to the gradient cost.

7.6 Sequential Optimization

When the sources are sparse, the quality of separation greatly improves with reduction of smoothing parameter λ in the absolute value approximation (7.4). On the other hand, the optimization of the likelihood function becomes more difficult for small λ . Therefore, we use sequential optimization with gradual reduction of λ . Denote

$$L(W; X, \lambda) = -\log |\det W| + \frac{1}{T} \sum_{i,t} h_\lambda(W_i x(t)), \quad (7.28)$$

where $h_\lambda(\cdot)$ is given by (7.3–7.4).

Sequential optimization algorithm

1. Start with λ_1 and W_1 ;
2. **For** $k = 1, 2, \dots, K$,
 - (a) Compute current source estimate $U_k = W_k X$;

- (b) Find $V_{k+1} = \arg \min_V L(V, U_k, \lambda_k)$, using $V = I$ as a starting point;
- (c) Update the separation matrix $W_{k+1} = V_{k+1}W_k$;
- (d) Update the smoothing parameter $\lambda_{k+1} = \mu\lambda_k$;

3. **End**

In our computations we choose the parameters $\lambda_1 = 1$ and $\mu = 0.01$. Note that step (b) includes the whole loop of unconstrained optimization, which can be performed, for example, by the relative Newton method.

7.7 Smoothing Method of Multipliers (SMOM)

Even gradual reduction of the smoothing parameter may require significant number of Newton steps after each update of λ . More efficient way to achieve an accurate solution of a problem involving a sum of absolute value functions is to use SMOM method presented in this section. This method is an extension of augmented Lagrangian technique [19,20,25] used in constrained optimization. It allows to obtain accurate solution without forcing the smoothing parameter λ to go to zero. In this work we combine the SMOM with relative optimization.

Consider non-smooth optimization problem

$$\min_w \left\{ F(w) = f_0(w) + \sum_{i=1}^m \max[\alpha_i f_i(w), \beta_i f_i(w)] \right\}, \quad (7.29)$$

where $f_i(w)$, $i = 0, \dots, m$ are smooth functions, $\alpha_i < \beta_i$ are certain constants. In particular, when $\alpha_i = -1$ and $\beta_i = 1$, we get sum of absolute value terms, like in the quasi-ML BSS:

$$F(w) = f_0(w) + \sum_{i=1}^m |f_i(w)|. \quad (7.30)$$

This kind of problem arises in many other areas of signal and image processing, in the context of sparse representations, total variation regularization, *etc.* (see for example [5]).

7.7.1 Smoothing the max-function

Consider a maximum function shown in Figure 7.1,

$$r(t) = \max(\alpha t, \beta t).$$

We introduce its smooth approximation $\varphi(t; \mu, \lambda)$, $\alpha < \mu < \beta$, $\lambda > 0$, with two parameters: μ and λ . The parameter λ defines the accuracy of the approximation of the max-function $r(\cdot)$, becoming perfect as $\lambda \rightarrow 0$. The parameter μ determines the derivative of φ at $t = 0$; it will serve as a Lagrange multiplier. The graph of the linear function μt is tangent to the plot of $\varphi(\cdot; \mu, \lambda)$ at the origin.

The function φ possesses the following properties:

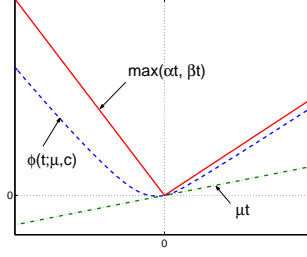


Fig. 7.1. $\text{Max}(\alpha t, \beta t)$ – solid line; smoothing function $\varphi(t; \mu, \lambda)$ – dashed line; linear support μt – dot-dashed line

- $\varphi(t; \mu, \lambda)$ is convex in t ;
- $\varphi(0; \mu, \lambda) = 0$;
- $\varphi'_t(0; \mu, \lambda) = \mu$;
- $\lim_{t \rightarrow -\infty} \varphi'_t(t; \mu, \lambda) = \alpha$;
- $\lim_{t \rightarrow +\infty} \varphi'_t(t; \mu, \lambda) = \beta$;
- $\lim_{\lambda \rightarrow 0} \varphi(t; \mu, \lambda) = r(t)$.

The particular form of the function φ we introduce and prefer to use in our computations consists of three smoothly connected branches:

$$\varphi(t, \mu, \lambda) = \begin{cases} \alpha t - p_1 \log \frac{t}{\tau_1} + s_1, & t < \tau_1 \leq 0 \\ \frac{t^2}{2\lambda} + \mu t, & \tau_1 \leq t \leq \tau_2 \\ \beta t - p_2 \log \frac{t}{\tau_2} + s_2, & t > \tau_2 \geq 0. \end{cases} \quad (7.31)$$

The coefficients $p_1, p_2, s_1, s_2, \tau_1, \tau_2$ are chosen to make the function φ continuous and twice differentiable at the joint points τ_1 and τ_2 :

$$\begin{aligned} \tau_1 &= \frac{\lambda(\alpha - \mu)}{2}; & \tau_2 &= \frac{\lambda(\beta - \mu)}{2}; \\ p_1 &= \tau_1^2/\lambda; & p_2 &= \tau_2^2/\lambda; \\ s_1 &= \frac{\tau_1^2}{2\lambda} + (\mu - \alpha)\tau_1; \\ s_2 &= \frac{\tau_2^2}{2\lambda} + (\mu - \beta)\tau_2. \end{aligned}$$

When $\tau_i = 0$, we put $p_i \log \frac{t}{\tau_i} = \lambda \tau_i^2 \log \frac{t}{\tau_i} = 0$. One can note, that when $\alpha \rightarrow 0$ and $\beta \rightarrow \infty$, $\varphi(\cdot)$ becomes a *quadratic-logarithmic* penalty for inequality constraint in nonquadratic augmented Lagrangian [25]. On the other hand, when $\alpha \rightarrow -\infty$ and $\beta \rightarrow \infty$, $\varphi(\cdot)$ becomes a quadratic penalty for equality constraint in a standard quadratic augmented Lagrangian [34]. In this way our approach generalizes known augmented Lagrangian techniques.

7.7.2 Generalized Lagrangian and Augmented Lagrangian

A standard way to introduce augmented Lagrangian for the sum-max problem would be to reformulate it as a smooth constrained optimization problem using artificial variables, which will increase the problem size significantly. Instead we introduce extended notions of Lagrangian and augmented Lagrangian, which keep the problem size unchanged, preserving at the same time many important classical properties of these functions. We propose the following extended notion of Lagrangian

$$L(w, u) = f_0(w) + \sum_{i=1}^m u_i f_i(w), \quad \alpha \leq u_i \leq \beta, \quad (7.32)$$

and corresponding extended notion of augmented Lagrangian

$$M(w, u, \lambda) = f_0(w) + \sum_{i=1}^m \varphi(f_i(w), u_i, \lambda), \quad \alpha \leq u_i \leq \beta. \quad (7.33)$$

As we show in [35], the Lagrangian saddle point theorem and duality theory can be extended to our case. As an important consequence of this, *there exists a vector of multipliers u^* , such that an unconstrained minimizer w^* of the augmented Lagrangian provides an optimal solution to Problem (7.29), i.e. we do not need to force the smoothing parameter λ toward zero in order to solve the problem. This property serves as a basis for the method of multipliers presented below.*

7.7.3 Smoothing Method of Multipliers (SMOM)

We introduce the method of multipliers, which performs the following steps at each outer iteration:

1. Minimize augmented Lagrangian in w , starting from the point w^k given by the previous iteration

$$w^{k+1} = \arg \min_w M(w, u^k, \lambda_k); \quad (7.34)$$

2. Update the multipliers using the derivative of φ with respect to the first argument

$$u_i^{k+1} = \varphi'(f_i(w^{k+1}), u_i^k, c_k), \quad i = 1, \dots, m; \quad (7.35)$$

3. Update the smoothing parameter (optionally)

$$\lambda_{k+1} = \gamma \lambda_k, \quad 0 < \gamma < 1. \quad (7.36)$$

The multiplier update rule (7.35) is motivated by the fact that in this way w^{k+1} becomes a minimizer of the Lagrangian: $\nabla_w L(w^{k+1}, u^{k+1}) = 0$. Therefore the optimal solution (w^*, u^*) is a fixed point of the algorithm. Similar considerations are used in standard augmented Lagrangian algorithm (see for example [25]).

In practical implementation we restrict the relative change of the multipliers to some bounds in order to stabilize the method:

$$\gamma_1 < \frac{u_i^{k+1} - \alpha}{u_i^k - \alpha} < \gamma_2 \quad (7.37)$$

$$\gamma_1 < \frac{\beta - u_i^{k+1}}{\beta - u_i^k} < \gamma_2 \quad (7.38)$$

$$\alpha + \delta < u_i^{k+1} < \beta - \delta. \quad (7.39)$$

We also restrict the smoothing parameter to remain above some minimal value λ_{min} . We usually put $\gamma = 0.5$, $\gamma_1 = \frac{1}{\gamma_2} = 2$, $\delta = 10^{-6}$, $\lambda_{min} = 10^{-3}$. In general, the algorithm is rather insensitive to changes in the parameters in order of magnitude or more. Convergence analysis of the method in convex case is presented in [35]. In practice, it works well also with non-convex problems, as we will see on example of quasi-ML BSS. In the later case we use the relative Newton method at the inner optimization stage (7.34).

7.7.4 Frozen Hessian Strategy

A useful fact is that changes in the Hessian of the augmented Lagrangian become very small at late outer iterations. This happens because changes in primal variables and multipliers become small toward convergence to solution, while the smoothing parameter λ remains constant. Therefore we can reuse the inverse Hessian (or its Cholesky factor) from the previous iterations [26], unless the number of steps in the current unconstrained optimization exceeds a predefined limit (say, 3 – 10 steps). Often 5–7 last outer iterations require only one Newton step each, without recomputing Hessian at all (see the experimental section).

7.8 Computational Experiments

Two data sets were used. The first group of sources was artificial sparse data with Bernoulli-Gaussian distribution

$$f(s) = p\delta(s) + (1 - p) \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-s^2/2\sigma^2),$$

generated by the MATLAB function SPRANDN. We used the parameters $p = 0.5$ and $\sigma = 1$. The second group of sources were four natural images from [36]. The mixing matrix was generated randomly with uniform *i.i.d.* entries.

7.8.1 Relative Newton method

In all experiments we used a backtracking line search with the constants $\beta = \gamma = 0.3$. Figure 7.2 shows the typical progress of different methods applied to the artificial data with 5 mixtures of 10k samples. The fast relative Newton method converges in about the same number of iterations as the relative Newton with exact Hessian, but significantly outperforms it in time. Natural gradient in batch mode requires much more iterations, and has a difficulty to converge when the smoothing parameter λ in (7.4) becomes too small.

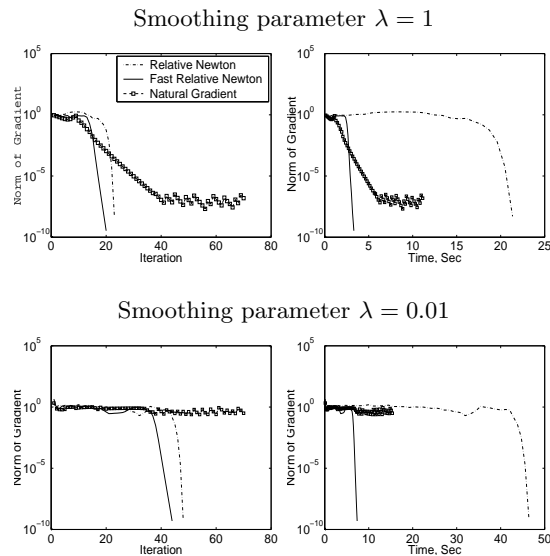


Fig. 7.2. Separation of artificial sparse data with 5 mixtures by 10k samples. Relative Newton with exact Hessian – dashed line, fast relative Newton – continuous line, natural gradient in batch mode – squares.

In the second experiment, we demonstrate the advantage of the batch-mode quasi-ML separation, when dealing with sparse sources. We compared the the fast relative Newton method with stochastic natural gradient [30–32], Fast ICA [11] and JADE [37]. All three codes were obtained from public web sites [38–40]. Stochastic natural gradient and Fast ICA used tanh nonlinearity. Figure 7.3 shows separation of artificial stochastic sparse data: 5 sources of 500 samples, 30 simulation trials. The quality of separation is measured by interference-to-signal ratio (ISR) in amplitude units. As we see, fast relative Newton significantly outperforms other methods, providing practically ideal separation with the smoothing parameter $\lambda = 10^{-6}$ (sequential update of the smoothing parameter was used here). Timing is of about the same order for

all the methods, except of JADE, which is known to be much faster with relatively small matrices.

7.8.2 SMOM combined with the Relative Newton method

In the third experiment we have used the first stochastic sparse data set: 5 mixtures, 10k samples. Figure 7.5 demonstrates advantage of the SMOM combined with the frozen Hessian strategy. As we see, the last six outer iterations does not require new Hessian evaluations. At the same time the sequential smoothing method without Lagrange multipliers, requires 3 to 8 Hessian evaluations per outer iterations toward end. As a consequence the method of multipliers converges much faster.

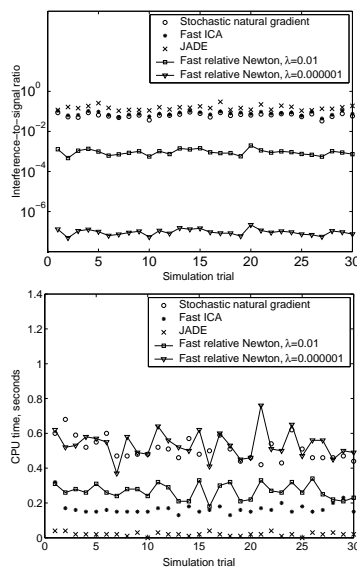


Fig. 7.3. Separation of stochastic sparse data: 5 sources of 500 samples, 30 simulation trials. Top – interference-to-signal ratio, bottom – CPU time.

In the fourth experiment, we separated four natural images [36], presented in Figure 7.6. Sparseness of images can be achieved via various wavelet-type transforms [8–10], but even simple differentiation can be used for this purpose, since natural images often have sparse edges. Here we used the stack of horizontal and vertical derivatives of the mixture images as an input to separation algorithms. Figure 7.7 shows the separation quality achieved by stochastic natural gradient, Fast ICA, JADE, the fast relative Newton method with $\lambda = 10^{-2}$ and the the SMOM. Like in the previous experiments, SMOM provides practically ideal separation with ISR of about 10^{-12} . It outperforms the other methods by several orders of magnitude.

In the fifth experiment we separated five musical sound recordings (one second, 10k samples), presented in Figure 7.8. Preprocessing (sparsification) was produced by the short time Fourier transform (STFT) of the mixtures using time windows of 2048 samples. Real and imaginary parts of the coefficients were concatenated into long vectors. As we see from Figure 7.9, the use of STFT significantly improves quality of separation, but even then the Relative Newton method with $\lambda = 10^{-3}$ and SMOM provide much better quality than the other methods.

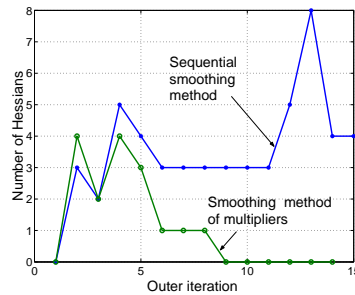


Fig. 7.4. Relative Newton method with frozen Hessian: number of Hessian evaluations per outer iteration (the first data set)

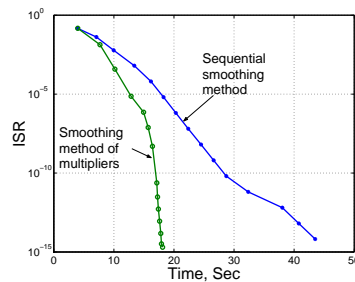


Fig. 7.5. Interference-to-signal ratio: progress with iterations / CPU time. Dots correspond to outer iterations.

7.9 Conclusions

We have presented the relative optimization framework for quasi-ML BSS, and studied the relative Newton method as its particular instance. Gradient-type computational cost of the Newton iteration makes it especially attractive.

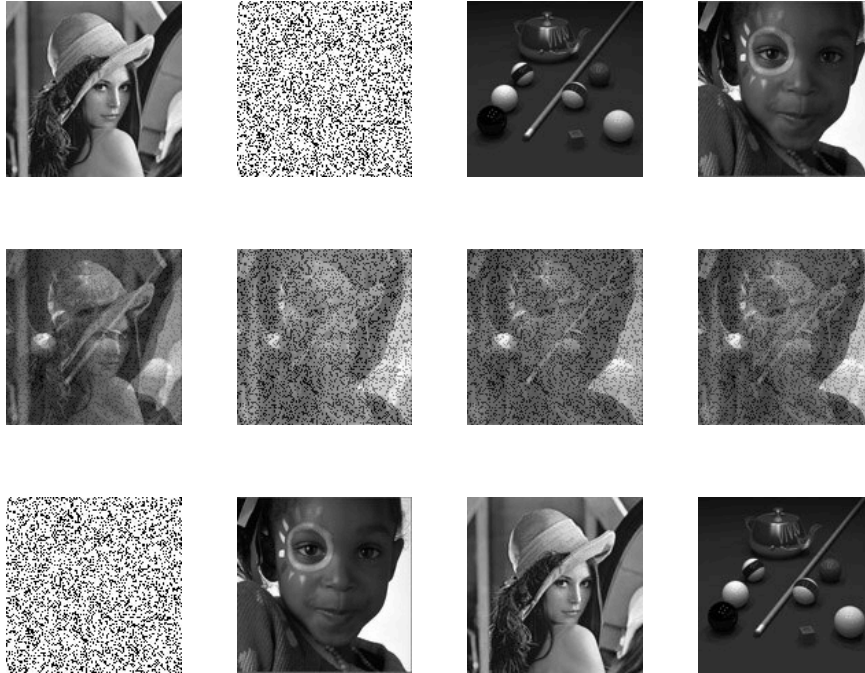


Fig. 7.6. Separation of images with preprocessing by differentiation. Top – sources, middle – mixtures, bottom – separated.

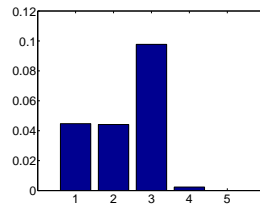


Fig. 7.7. Interference-to-signal ratio (ISR) of image separation: 1 – stochastic natural gradient; 2 – Fast ICA; 3 – JADE; 4 – Relative Newton with $\lambda = 10^{-2}$; 5 – SMOM (bar 5 is not visible because of very small ISR, of order 10^{-12} .)

We also presented SMOM method for minimization of sum of pairwise maxima of smooth functions (in particular sum of absolute value terms, like used in quasi-ML separation of sparse sources.) Incorporating Lagrange multiplier into a smooth approximation of max-type function, we obtained an extended notion of non-quadratic augmented Lagrangian. This approach does not require artificial variables, and preserves sparse structure of Hessian.

We apply the Frozen Hessian strategy, using the fact that changes in the Hessian of the augmented Lagrangian become very small at late outer



Fig. 7.8. Separation of musical sounds (one second, 10k samples) using Relative Newton method: Left – sources, middle – mixtures, right – separated.

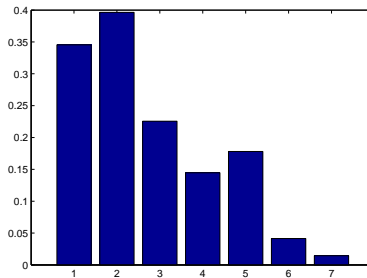


Fig. 7.9. Interference-to-signal ratio (ISR) of sound separation: 1 – JADE; 2 – Fast ICA; 3 – stochastic natural gradient; The remaining bars – separation using STFT transform of the mixed signals: 4 – JADE; 5 – Fast ICA; 6 – stochastic natural gradient; 7 – Relative Newton with $\lambda = 10^{-3}$ (SMOM gives similar results)

iterations of SMOM. In our experiments 5–7 last outer iterations require only one Newton step each, without recomputing Hessian at all.

Experiments with sparsely representable artificial data, natural images and sounds show that quasi-ML separation is almost perfect when the non-linearity approaches the absolute value function.

Currently we are conducting more experiments with non-sparse source distributions and various kinds of non-linearities. Preliminary results confirm fast convergence of the relative Newton method.

Appendix A

Convergence Analysis of Relative Optimization algorithm

Definition 1 We say that a function f sufficiently decreases at iteration k , if for any $\epsilon > 0$ there exists $\delta > 0$ such that from $\|x_k - x^*\| > \epsilon$ it follows that $f(x_k) - f(x_{k+1}) > \delta$. Here x^* is a local minimum closest to x_k .

Suppose the following properties of the function $h(\cdot)$ in (7.2)

- (h1) $h(\cdot)$ is bounded below;
- (h2) $h(\cdot)$ grows faster than $\log(|\cdot|)$ toward $\pm\infty$

$$\lim_{\alpha \rightarrow \pm\infty} h(\alpha)/\log(|\alpha|) = \infty \quad (7.40)$$

Proposition 1 The sequence $L(W_k; X)$ generated by Relative Optimization algorithm is monotone decreasing at each step by the value

$$L(W_k; X) - L(W_{k+1}; X) = L(I; U_k) - L(V_k; U_k) > 0 \quad (7.41)$$

Proof Optimization step 2 reduces the function value

$$L(V_k; U_k) < L(I; U_k).$$

Taking into account that by (7.2)

$$L(W_k; X) = -\log |\det W_k| + L(I; U_k) \quad (7.42)$$

$$L(W_{k+1}; X) = -\log |\det W_k| + L(V_k; U_k) \quad (7.43)$$

we get (7.41). \square

Proposition 2 The likelihood function (7.2) is bounded from below and has bounded level sets.

Proof is based on the properties (h1 – h2). We need to show that the function $L(W; X)$ in (7.2) has an infinite growth along any radial direction

$$\lim_{\alpha \rightarrow \infty} L(\alpha W_0) = \infty$$

for any invertible W_0 . This is an obvious consequence of (7.40). \square

Lemma 1 The sequence W_k generated by the Relative Optimization algorithm, has limit point[s]; any limit point belongs to a local minimum of the likelihood function (7.2)

Proof The sequence of the function values $L(W_k; X)$ generated by the Relative Optimization algorithm is monotone decreasing (by Proposition 1), so all iterates W_k belong to the level set $\{W : L(W; X) < L(W_0; X)\}$, which is bounded according to Proposition 2. Therefore the sequence of the iterates W_k has limit point[s].

The second part of the proof is continued by contradiction. Let \bar{W} be a limit point, which is not equal to the closest local minimum W^* , *i.e.* for any point W_k from a small neighborhood of \bar{W} , $W_k \in \mathcal{N}(\bar{W})$,

$$\|I - W^*W_k^{-1}\| > \epsilon > 0 \quad (7.44)$$

Let V_k^* be a local minimizer of $L(\cdot; U_k)$, so that $W^* = V_k^*W_k$. It follows from (7.44) that

$$\|I - V_k^*\| > \epsilon \quad (7.45)$$

therefore step 2 of the Relative Optimization algorithm provides significant decrease of the objective function (see Definition 1)

$$L(I; U_k) - L(V_k; U_k) > \delta \quad (7.46)$$

Since \bar{W} is a concentration point, there are infinite number of iterates $W_k \in \mathcal{N}(\bar{W})$ satisfying (7.44 – 7.46). Taking into account (7.41), we conclude that the function $L(W_k; X)$ will decrease infinitely, which contradicts its below boundedness, stated by Proposition 2. \square

References

1. D. Pham and P. Garat, “Blind separation of a mixture of independent sources through a quasi-maximum likelihood approach,” *IEEE Transactions on Signal Processing*, vol. 45, no. 7, pp. 1712–1725, 1997.
2. A. J. Bell and T. J. Sejnowski, “An information-maximization approach to blind separation and blind deconvolution,” *Neural Computation*, vol. 7, no. 6, pp. 1129–1159, 1995.
3. J.-F. Cardoso, “On the performance of orthogonal source separation algorithms,” in *EUSIPCO*, Edinburgh, Sept. 1994, pp. 776–779.
4. —, “Blind signal separation: statistical principles,” *Proceedings of the IEEE*, vol. 9, no. 10, pp. 2009–2025, Oct. 1998. [Online]. Available: <ftp://sig.enst.fr/pub/jfc/Papers/ProcIEEE.us.ps.gz>
5. S. S. Chen, D. L. Donoho, and M. A. Saunders, “Atomic decomposition by basis pursuit,” *SIAM J. Sci. Comput.*, vol. 20, no. 1, pp. 33–61, 1998.
6. B. A. Olshausen and D. J. Field, “Sparse coding with an overcomplete basis set: A strategy employed by v1?” *Vision Research*, vol. 37, pp. 3311–3325, 1997.
7. M. S. Lewicki and B. A. Olshausen, “A probabilistic framework for the adaptation and comparison of image codes,” *Journal of the Optical Society of America*, vol. 16, no. 7, pp. 1587–1601, 1999, in press.

8. M. Zibulevsky and B. A. Pearlmutter, "Blind source separation by sparse decomposition in a signal dictionary," *Neural Computations*, vol. 13, no. 4, pp. 863–882, 2001.
9. M. Zibulevsky, B. A. Pearlmutter, P. Bofill, and P. Kisilev, "Blind source separation by sparse decomposition," in *Independent Components Analysis: Principles and Practice*, S. J. Roberts and R. M. Everson, Eds. Cambridge University Press, 2001.
10. M. Zibulevsky, P. Kisilev, Y. Y. Zeevi, and B. A. Pearlmutter, "Blind source separation via multinode sparse representation," in *Advances in Neural Information Processing Systems 12*. MIT Press, 2002.
11. A. Hyvärinen, "Fast and robust fixed-point algorithms for independent component analysis," *IEEE Transactions on Neural Networks*, vol. 10, no. 3, pp. 626–634, 1999.
12. T. Akuzawa and N. Murata, "Multiplicative nonholonomic Newton-like algorithm," *Chaos, Solitons and Fractals*, vol. 12, p. 785, 2001.
13. T. Akuzawa, "Extended quasi-Newton method for the ICA," Laboratory for Mathematical Neuroscience, RIKEN Brain Science Institute, Tech. Rep., 2000, <http://www.mns.brain.riken.go.jp/~akuzawa/publ.html>.
14. D. Pham, "Joint approximate diagonalization of positive definite matrices," *SIAM J. on Matrix Anal. and Appl.*, vol. 22, no. 4, pp. 1136–1152, 2001.
15. D. Pham and J.-F. Cardoso, "Blind separation of instantaneous mixtures of non stationary sources," *IEEE Transactions on Signal Processing*, vol. 49, no. 9, pp. 1837–1848, 2001.
16. M. Joho and K. Rahbar, "Joint diagonalization of correlation matrices by using Newton methods with application to blind signal separation," *SAM 2002*, 2002, http://www.phonak.uiuc.edu/~joho/research/publications/sam_2002.2.pdf.
17. A. Ziehe, P. Laskov, G. Nolte, and K.-R. Mueller, "A fast algorithm for joint diagonalization with non-orthogonal transformations and its application to blind source separation," *Journal of Machine Learning Research*, vol. 5(Jul), pp. 801–818, 2004.
18. B. Kort and D. Bertsekas, "Multiplier methods for convex programming," *Proc 1073 IEEE Conf. Decision Control, San-Diego, Calif.*, pp. 428–432, 1973.
19. D. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*. New York: Academic Press, 1982.
20. R. Polyak, "Modified barrier functions: Theory and methods," *Math. Programming*, vol. 54, pp. 177–222, 1992.
21. A. Ben-Tal, I. Yuzefovich, and M. Zibulevsky, "Penalty/barrier multiplier methods for min-max and constrained smooth convex programs," Opt. Lab., Dept. of Indust. Eng., Technion, Haifa, Israel, Tech. Rep. 9, 1992.
22. P. Tseng and D. Bertsekas, "Convergence of the exponential multiplier method for convex programming," *Math. Programming*, vol. 60, pp. 1–19, 1993.
23. M. G. Breitfeld and D. Shanno, "Computational experience with penalty/barrier methods for nonlinear programming," *Annals of Operations Research*, vol. 62, pp. 439–464, 1996.
24. M. Zibulevsky, "Penalty/barrier multiplier methods for large-scale nonlinear and semidefinite programming," Ph.D. dissertation, Technion – Israel Institute of Technology, 1996, <http://ie.technion.ac.il/~mcib/>.
25. A. Ben-Tal and M. Zibulevsky, "Penalty/barrier multiplier methods for convex programming problems," *SIAM Journal on Optimization*, vol. 7, no. 2, pp. 347–366, 1997.

26. L. Mosheev and M. Zibulevsky, "Penalty/barrier multiplier algorithm for semi-definite programming," *Optimization Methods and Software*, vol. 13, no. 4, pp. 235–261, 2000.
27. M. Kocvara and M. Stingl, "PENNON – a code for convex nonlinear and semi-definite programming," *Optimization Methods and Software*, vol. 18(3), pp. 317–333, 2003.
28. A. Ben-Tal and M. Teboulle, "A smoothing technique for nondifferentiable optimization problems," *Fifth French German Conference, Lecture Notes in Math. 1405*, Springer-Verlag, New York, pp. 1–11, 1989.
29. C. Chen and O. L. Mangasarian, "A class of smoothing functions for nonlinear and mixed complementarity problems," *Computational Optimization and Applications*, vol. 5, pp. 97–138, 1996.
30. A. Cichocki, R. Unbehauen, and E. Rummert, "Robust learning algorithm for blind separation of signals," *Electronics Letters*, vol. 30, no. 17, pp. 1386–1387, 1994.
31. S. Amari, A. Cichocki, and H. H. Yang, "A new learning algorithm for blind signal separation," in *Advances in Neural Information Processing Systems 8*. MIT Press, 1996. [Online]. Available: <http://www.cs.cmu.edu/Groups/NIPS/NIPS95/Papers.html>
32. J.-F. Cardoso and B. Laheld, "Equivariant adaptive source separation," *IEEE Transactions on Signal Processing*, vol. 44, no. 12, pp. 3017–3030, 1996.
33. P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*. New York: Academic Press, 1981.
34. R. Rockafellar, *Convex Analysis*. Princeton, NJ: Princeton University Press, 1970.
35. M. Zibulevsky, "Smoothing method of multipliers for sum-max problems," Dept. of Elec. Eng., Technion, Tech. Rep., 2003, <http://ie.technion.ac.il/~mcib/>.
36. A. Cichocki, S. Amari, and K. Siwek, "ICALAB toolbox for image processing – benchmarks," 2002, <http://www.bsp.brain.riken.go.jp/ICALAB/ICALABImageProc/benchmarks/>.
37. J.-F. Cardoso, "High-order contrasts for independent component analysis," *Neural Computation*, vol. 11, no. 1, pp. 157–192, 1999.
38. S. Makeig, "ICA toolbox for psychophysiological research," Computational Neurobiology Laboratory, the Salk Institute for Biological Studies, 1998, <http://www.cnl.salk.edu/~ica.html>.
39. A. Hyvärinen, "The Fast-ICA MATLAB package," 1998, <http://www.cis.hut.fi/~aapo/>.
40. J.-F. Cardoso, "JADE for real-valued data," 1999, <http://sig.enst.fr:80/~cardoso/guidesepsou.html>.