A Relative Trust-Region Algorithm for Independent Component Analysis

Heeyoul Choi, Seungjin Choi *

Department of Computer Science
Pohang University of Science and Technology
San 31 Hyoja-dong, Nam-gu
Pohang 790-784, Korea

Abstract

In this paper we present a method of parameter optimization, *relative trust-region learning*, where the trust-region method and the relative optimization [21] are jointly exploited. The relative trust-region method finds a direction and a step size with the help of a quadratic model of the objective function (as in the conventional trust-region methods) and updates parameters in a multiplicative fashion (as in the relative optimization). We apply this relative trust-region learning method to the problem of independent component analysis (ICA), which leads to the *relative TR-ICA* algorithm which turns out to possess the equivariant property (as in the relative gradient) and to achieve faster convergence than the relative gradient and even Newton-type algorithms. Empirical comparisons with several existing ICA algorithms, demonstrate the useful behavior of the relative TR-ICA algorithm, such as the equivariant property and fast convergence.

Key words: Blind source separation, Gradient-descent learning, Independent component analysis, Relative optimization, Trust-region methods.

1 Introduction

Independent component analysis (ICA) is a statistical method that decomposes a multivariate data into a linear sum of non-orthogonal basis vectors

* Corresponding author. Tel.: +82-54-279-2259; Fax: +82-54-279-2299
Email: hychoi@postech.ac.kr (H. Choi), seungjin@postech.ac.kr (S. Choi)
URL: http://www.postech.ac.kr/∼seungjin (S. Choi)

with basis coefficients being statistically independent. A variety of approaches to ICA have been developed. These include maximum likelihood estimation, mutual information minimization, output entropy maximization (infomax), and negentropy maximization (for example, see [17, 16, 13, 12] and references therein). All these approaches lead to an identical objective function in ICA [6, 18]. A popular implementation in these approaches, is the gradient-descent learning (including the natural gradient). Although gradient-based algorithms are simple and guarantee the local stability, but they are relatively slow and require a careful choice of a learning rate, which are cumbersome in practical applications. In order to overcome these drawbacks, Newton-type algorithms were proposed [2, 21]. Even though FastICA is a fixed-point algorithm, it can be also interpreted as a kind of Newton-type algorithm [15].

A trust-region method is an optimization technique, where a direction and a step size are determined in a reliable manner with the help of a quadratic model of the objective function [20]. Trust-region methods define a region around the current iterate within which they trust the model to be an adequate representation of the objective function, and then choose the step to be the approximate minimizer of the model in this trust region. In effect, they choose the direction and length of the step simultaneously. If a step is not acceptable to minimize the objective function while the step minimizes the model, they reduce the size of the region and find a new minimizer. They are, in general, faster than gradient descent methods and their learning rate (or step size) is determined automatically by the size of the trust-region, whereas gradient-based methods require a careful choice of a learning rate. Their convergence is between linear and quadratic rate and its stability is always guaranteed, in contrast to the Newton method. The stability of FastICA is investigated in [15].

The relative gradient [8] or the natural gradient [3] was shown to be efficient in learning the parameters when the parameter space belongs to the Riemannian manifold. It was also shown that the relative gradient leads to algorithms having the equivariant property which produces the uniform performance, regardless of the condition of the mixing matrix in the task of blind source separation (BSS) or independent component analysis (ICA). Especially when the mixing matrix is ill-conditioned, the relative gradient ICA algorithms outperform other-type algorithms. The relative gradient is a particular instance of the relative optimization [21], where one determines the modified differential matrix corresponding to the search direction from the identity matrix via conventional optimization methods and updates the parameters in a multiplicative fashion. The relative Newton method [21] is another exemplary technique of the relative optimization method, where the modified differential matrix is learned through Newton-type updates.

In this paper we present a relative trust-region learning method [1], where we incorporate the relative optimization into the trust-region method. In the relative trust-region optimization, we determine a search direction and a step size with the help of a quadratic model in a similar way to conventional trust-region methods. The parameters are updated in a multiplicative fashion as in the relative optimization. We apply the relative trust-region learning method to the problem of ICA, which leads to the relative TR-ICA algorithm. The relative TR-ICA algorithm inherits various useful properties, such as the fast convergence, stability, and the equivariant property, from both conventional trust-region methods and the relative optimization. Moreover we exploit a special structure of the Hessian matrix for memory-efficiency in the relative TR-ICA algorithm, so that the algorithm is useful, especially for the case of high-dimensional data.

The rest of this paper is organized as follows. Next section briefly summarizes the ICA formulation and introduces necessary notations that will be used throughout this paper. Sec. 3 reviews the trust-region method to make this paper self-contained. The relative optimization, including the relative gradient, is explained, in Sec. 4, following earlier work in [8, 7, 21]. The main contribution of this paper, i.e., the relative trust-region method and the relative TR-ICA algorithm, are illustrated in Sec. 5. Numerical experiments and results are shown in Sec. 6, with comparison to several existing ICA algorithms. Finally conclusions are drawn in Sec. 7.

2 Independent Component Analysis

Given N data points, the simplest form of ICA considers the noise-free linear generative model where the observation data $\boldsymbol{x}(t) \in \mathbb{R}^n$ is assumed to be generated by

$$\boldsymbol{X} = \boldsymbol{A}\boldsymbol{S}, \tag{1}$$

where $\boldsymbol{X} = [\boldsymbol{x}(1), \ldots, \boldsymbol{x}(N)]$, $\boldsymbol{S} = [\boldsymbol{s}(1), \ldots, \boldsymbol{s}(N)]$ and $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ contains n basis vectors $\boldsymbol{a}_i \in \mathbb{R}^n$, $i = 1, \ldots, n$ in its columns. $\boldsymbol{s}(t) \in \mathbb{R}^n$ is a latent variable vector whose elements $s_i(t)$ are mutually independent.

The maximum likelihood estimation leads to the objective function that has the form

[1] The earlier work was presented in [9]

$$f(\boldsymbol{W}, \boldsymbol{X}) = -\log|\det \boldsymbol{W}| + \frac{1}{N}\sum_{t=1}^{N}\sum_{i=1}^{n}\psi_i(y_i(t)), \tag{2}$$

where $\boldsymbol{W} = \boldsymbol{A}^{-1}$, $\boldsymbol{y}(t) = \boldsymbol{W}\boldsymbol{x}(t)$, and $\psi_i(y_i(t)) = -\log p_i(y_i(t))$. The matrix $\boldsymbol{W}$ is referred to as a *demixing matrix* and the estimate of latent variable vector, $\boldsymbol{y}$, is restored up to the scaled and re-ordered version of the original hidden variable vector $\boldsymbol{s}$.

Throughout this paper, we will often use a parameter vector $\boldsymbol{w} \in \mathbb{R}^{n^2} = \mathrm{vec}\left(\boldsymbol{W}^T\right)$ where $\mathrm{vec}(\cdot)$ is the *vec-function* which stacks the columns of the given matrix into one long vector. On the contrary, $\boldsymbol{W} = \mathrm{mat}^T(\boldsymbol{w})$. We also abuse notations such as

$$\begin{aligned} f(\boldsymbol{W}, \boldsymbol{X}) &= f(\boldsymbol{W}) = f(\boldsymbol{w}), \\ f(\boldsymbol{I}, \boldsymbol{Y}) &= f_r(\boldsymbol{W}) = f_r(\boldsymbol{w}), \end{aligned} \tag{3}$$

where $\boldsymbol{I} \in \mathbb{R}^{n\times n}$ is the identity matrix and the subscript r is used to emphasize that it involves the *relative mode* which will be described in detail later. In the relative mode, the objective function involves $\boldsymbol{I}$ instead of $\boldsymbol{W}$, hence $\boldsymbol{Y}$ instead of $\boldsymbol{X}$. The relative optimization is based on the serial updating where the parameters are learned in a multiplicative fashion. At every iterations, the relative optimization searches the parameters, starting from the identity matrix $\boldsymbol{I}$.

Popular ICA algorithms are based on the gradient or the natural gradient method [5, 4]. Although gradient-based algorithms are simple and guarantee the local stability, but they are relatively slow and require a careful choice of a learning rate, which are cumbersome in practical applications.

3 Trust-Region Methods

In this section we overview a trust-region method (see [20] for more details), in order to help readers to understand how parameters are estimated through the trust-region optimization.

Let us consider an objective function $f(\boldsymbol{w}) : \mathbb{R}^{n^2} \to \mathbb{R}$ to be minimized with respect to the parameter vector $\boldsymbol{w} \in \mathbb{R}^{n^2}$. A quadratic model function $m^{(k)}$ which has elliptical contours, is based on the objective function and its derivative information at $\boldsymbol{w}^{(k)}$. The search direction $\boldsymbol{p} \in \mathbb{R}^{n^2}$ is determined by solving the following subproblem:

$$\arg\min_{\|\boldsymbol{p}\| \leq \triangle^{(k)}} m^{(k)}(\boldsymbol{p}) = f^{(k)} + \left[\nabla f^{(k)}\right]^T \boldsymbol{p} + \frac{1}{2}\boldsymbol{p}^T \boldsymbol{B}^{(k)} \boldsymbol{p}, \tag{4}$$

where $\triangle^{(k)} > 0$ is the trust-region radius, $\|\cdot\|$ is the Euclidean norm and

$$\begin{aligned} f^{(k)} &= f(\boldsymbol{w}^{(k)}), \\ \nabla f^{(k)} &= \left.\frac{\partial f}{\partial \boldsymbol{w}}\right|_{\boldsymbol{w}=\boldsymbol{w}^{(k)}}. \end{aligned} \tag{5}$$

Here, $\boldsymbol{B}^{(k)} \in \mathbb{R}^{n^2 \times n^2}$ is some symmetric matrix. In our work, we used the true Hessian matrix as $\boldsymbol{B}^{(k)}$. The solution $\boldsymbol{p}_*^{(k)}$ of Eq. (4) is the minimizer of $m^{(k)}$ in the ball with its radius being $\triangle^{(k)}$. In regards to finding $\boldsymbol{p}_*^{(k)}$, there are several approximate solutions. In our relative trust-region optimization, we employ the dogleg method (see [20] for more details). A direct application of the trust-region method to ICA, can be found in [10] where the algorithm was referred to as TR-ICA.

An important issue in a trust-region method is a strategy for choosing a trust-region radius $\triangle^{(k)}$ at each iteration. The choice of $\triangle^{(k)}$ is based on the agreement between the model function $m^{(k)}$ and the objective function f at previous iteration. Given a direction $\boldsymbol{p}^{(k)}$, this agreement measure $\rho^{(k)}$ is defined as the ratio of *actual reduction* to *predicted reduction*, i.e.,

$$\rho^{(k)} = \frac{f\left(\boldsymbol{w}^{(k)}\right) - f\left(\boldsymbol{w}^{(k)} + \boldsymbol{p}^{(k)}\right)}{m^{(k)}(\boldsymbol{0}) - m^{(k)}\left(\boldsymbol{p}^{(k)}\right)}. \tag{6}$$

The convergence of the trust-region method is between linear and quadratic rate, hence it is, in general, faster than gradient methods. It also guarantees the stability, regardless of initial conditions, whereas the Newton method requires an extra technique for stability. Especially, trust-region is much better than Newton method when the quadratic model does not approximate the objective function well.

4 Relative Optimization

This section describes the relative optimization method which is an extension of the relative gradient [21]. The method is illustrated in the framework of learning in Lie group, following some results in [7]. We also revisit the relative gradient in the context of the relative optimization.

4.1 Learning in a Group

The serial updating where the parameters are learned in a multiplicative fashion, keeps the parameters in a group structure (especially Lie group). The relative gradient is based on the idea of learning in Lie group, which was first investigated by Cardoso [7]. See also [14] for recent review of learning in orthogonal group.

The general linear group of degree n, denoted by $GL(n)$, is an instance of Lie group. In the case of ICA, the parameter matrix $\boldsymbol{W}$ belongs to the general linear group $GL(n)$, and learning a demixing matrix $\boldsymbol{W}$, can be carried out by a linear transformation of parameters, which leads to the following learning process

$$\boldsymbol{W}^{(k+1)} = \boldsymbol{E}^{(k)} \boldsymbol{W}^{(k)}, \tag{7}$$

where $\boldsymbol{E}^{(k)}$ is a linear transformation of parameters $\boldsymbol{W}^{(k)}$. It follows from Eq. (7) that the updating rule consists of series of multiplications of matrices where the multiplication is the binary operator of the group $GL(n)$. This implies that $\boldsymbol{W}^{(k)}$ and $\boldsymbol{E}^{(k)}$ at every iteration are the elements of $GL(n)$ which form a Lie group. The linear transformation $\boldsymbol{E}^{(k)}$ is computed such that an objective function (for instance, Eq. (2) in the case of ICA) is minimized on the Lie group. Moreover, a Lie group is a differentiable manifold obeying the group properties. Therefore, the multiplicative learning rule of $\boldsymbol{W}^{(k)}$ in Eq. (7) reflects a manifold. In fact, the natural gradient (which is identical to the relative gradient in ICA) was developed in the framework of learning in Riemannian manifold [3].

4.2 Equivariant Property and Serial Updating

An important property induced by an equivariant estimator is the uniform performance [7, 8], implying that the performance of an estimator does not depend on the mixing matrix $\boldsymbol{A}$ in ICA. This equivariant property is a consequence of the 'serial updating'.

Without loss of generality, a family of adaptive ICA algorithms employs the updating rule that has the form

$$\boldsymbol{W}^{(k+1)} = \left(\boldsymbol{I} - \eta^{(k)} G\left(\boldsymbol{Y}^{(k)}, \boldsymbol{W}^{(k)}\right)\right) \boldsymbol{W}^{(k)}, \tag{8}$$

where $\boldsymbol{Y}^{(k)} = \boldsymbol{W}^{(k)} \boldsymbol{X}$, $\widetilde{G}\left(\boldsymbol{X}, \boldsymbol{W}^{(k)}\right)$ is a matrix-valued function and $\eta^{(k)} > 0$

is a learning rate. A special case where the function $G(\cdot)$ does not rely on $\boldsymbol{W}^{(k)}$, leads to an updating rule that has the form

$$\boldsymbol{W}^{(k+1)} = \left(\boldsymbol{I} - \eta^{(k)} G\left(\boldsymbol{Y}^{(k)}\right)\right) \boldsymbol{W}^{(k)}. \tag{9}$$

The current parameter matrix $\boldsymbol{W}^{(k)}$ is transformed by a 'plugging' matrix $\left(\boldsymbol{I} - \eta^{(k)} G\left(\boldsymbol{Y}^{(k)}\right)\right)$, in order to produce an updated parameter matrix $\boldsymbol{W}^{(k+1)}$. This serial updating is carried out in a multiplicative fashion. Note that updating rule depends only on the current output. In this sense, this serial updating is consistent with equivariance.

If we denote the 'plugging' matrix $\left(\boldsymbol{I} - \eta^{(k)} G\left(\boldsymbol{Y}^{(k)}\right)\right)$ by $\boldsymbol{E}^{(k)}$, then the parameter matrix $\boldsymbol{W}^{(k+1)}$ can be decomposed into

$$\boldsymbol{W}^{(k+1)} = \boldsymbol{E}^{(k)} \boldsymbol{W}^{(k)} = \boldsymbol{E}^{(k)} \boldsymbol{E}^{(k-1)} \cdots \boldsymbol{E}^{(1)} \boldsymbol{E}^{(0)} \boldsymbol{W}^{(0)}, \tag{10}$$

where $\boldsymbol{W}^{(0)}$ is an initial value of $\boldsymbol{W}$. It follows from Eq. (10) that the serial updating rule for $\boldsymbol{W}^{(k)}$ reflects a manifold as mentioned in previous section.

The relative gradient [8] is an exemplary instance of the serial updating where the function $G(\cdot)$ is determined by a search direction $\mathcal{E}$ that minimizes

$$f(\boldsymbol{W} + \mathcal{E}\boldsymbol{W}) = f(\boldsymbol{W}) + \mathrm{tr}\left(\nabla_r f^T(\boldsymbol{W}) \mathcal{E}\right) + o(\mathcal{E}), \tag{11}$$

where $\nabla_r f(\boldsymbol{W})$ denotes the relative gradient of f at $\boldsymbol{W}$ and $\mathrm{tr}(\cdot)$ denotes the trace operator.

4.3 Relative Optimization

The relative gradient involves the plugging matrix containing the first-order information (in the sense that the gradient is used). The relative optimization [21] is a generalization of the relative gradient, where the plugging matrix is determined by well-developed optimization methods (for example, Newton method) in the framework of the serial updating.

The relative gradient algorithm can be easily derived in the framework of the relative optimization. The gradient in the relative mode, $\nabla f_r(\boldsymbol{W})$, is defined by

$$\nabla f_r(\boldsymbol{W}, \boldsymbol{X}) = \left.\frac{\partial f(\boldsymbol{W}, \boldsymbol{X})}{\partial \boldsymbol{W}}\right|_{\boldsymbol{W}=\boldsymbol{I}, \boldsymbol{X}=\boldsymbol{Y}} \tag{12}$$

where $\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{X}$.

The increment for $\boldsymbol{V}$ is given by $\Delta\boldsymbol{V} = -\eta\nabla f_r(\boldsymbol{W}, \boldsymbol{X})$, leading to $\boldsymbol{V}^{(k+1)} = \boldsymbol{I} - \eta^{(k)}\nabla f_r(\boldsymbol{W}^{(k)}, \boldsymbol{X})$, where $\boldsymbol{V}^{(k)}$ is set as $\boldsymbol{I}$. Taking the updating rule for $\boldsymbol{W}$, $\boldsymbol{W}^{(k+1)} = \boldsymbol{V}^{(k+1)}\boldsymbol{W}^{(k)}$ into account, leads to

$$\boldsymbol{W}^{(k+1)} = \left\{\boldsymbol{I} - \eta^{(k)}\nabla f_r\left(\boldsymbol{W}^{(k)}, \boldsymbol{X}\right)\right\}\boldsymbol{W}^{(k)}. \tag{13}$$

One can easily see that Eq. (13) is identical to the original relative gradient ICA algorithm [8] where the updating rule is given by

$$\boldsymbol{W}^{(k+1)} = \left\{\boldsymbol{I} - \eta^{(k)}\nabla_r f\left(\boldsymbol{W}^{(k)}, \boldsymbol{X}\right)\right\}\boldsymbol{W}^{(k)}. \tag{14}$$

It can be easily verified that $\nabla_r f(\boldsymbol{W}, \boldsymbol{X}) = \nabla f_r(\boldsymbol{W}, \boldsymbol{X})$ in the case of ICA where the objective function Eq. (2) is considered.

5 Relative Trust-Region Optimization

This section contains the main contribution of this paper where we explain the relative trust-region optimization followed by the illustration of the relative TR-ICA algorithm.

5.1 Gradient and Hessian

The objective function that we are concerned about, is given in Eq. (2) that has the form

$$f(\boldsymbol{w}) = f(\boldsymbol{W}, \boldsymbol{X}) = -\log|\det \boldsymbol{W}| + \frac{1}{N}\sum_{t=1}^{N}\sum_{i=1}^{n}\psi_i(y_i(t)). \tag{15}$$

Here we use the parameter vector $\boldsymbol{w} \in \mathbb{R}^{n^2} = \mathrm{vec}\left(\boldsymbol{W}^T\right)$, in illustrating the relative optimization method and the relative TR-ICA algorithm. The row vectors and column vectors of $\boldsymbol{W}$ are denoted by $\vec{\boldsymbol{w}}_i$ and $\boldsymbol{w}_i$ for $i = 1, \ldots, n$, respectively. We calculate the gradient and Hessian of Eq. (15) with respect to $\boldsymbol{w}$, instead of $\boldsymbol{W}$. The gradient of Eq. (15), $\nabla f(\boldsymbol{w}) \in \mathbb{R}^{n^2}$, is given by

$$\begin{aligned}\nabla f(\boldsymbol{w}) &= \left[\frac{\partial f}{\partial \vec{\boldsymbol{w}}_1} \cdots \frac{\partial f}{\partial \vec{\boldsymbol{w}}_n}\right]^T \\ &= \mathrm{vec}\left(-\boldsymbol{W}^{-1} + \frac{1}{N}\sum_{t=1}^{N} \boldsymbol{x}(t)\left[\psi'(\boldsymbol{y}(t))\right]^T\right), \end{aligned} \tag{16}$$

where $\psi'(\boldsymbol{y}(t))$ is the n-dimensional vector, whose ith element is $\psi_i'(y_i(t)) = \frac{d\,\psi_i(y_i(t))}{d\,y_i(t)}$.

The Hessian matrix of the objective function Eq. (15), $\nabla^2 f(\boldsymbol{w}) \in \mathbb{R}^{n^2 \times n^2}$, is computed by

$$\begin{aligned}\nabla^2 f(\boldsymbol{w}) &= \frac{\partial}{\partial \boldsymbol{w}}\left[\frac{\partial f}{\partial \boldsymbol{w}}\right]^T \\ &= \boldsymbol{H} + \boldsymbol{D}, \end{aligned} \tag{17}$$

where $\boldsymbol{D} \in \mathbb{R}^{n^2 \times n^2}$ is a block-diagonal matrix, consisting of $\boldsymbol{D}_l \in \mathbb{R}^{n \times n}$, $l = 1, \ldots, n$, that is of the form

$$\boldsymbol{D}_l = \frac{1}{N}\sum_{t=1}^{N} \psi_l''(y_l(t))\boldsymbol{x}(t)\boldsymbol{x}^T(t), \tag{18}$$

and $\boldsymbol{H} \in \mathbb{R}^{n^2 \times n^2}$ consists of n^2 row vectors, $\vec{\boldsymbol{h}}_m$, that is given by

$$\vec{\boldsymbol{h}}_m = \mathrm{vec}^T\left(\boldsymbol{a}_i \vec{\boldsymbol{a}}_j\right), \quad m = (i-1)n + j, \tag{19}$$

for $i = 1, \ldots, n$ and $j = 1, \ldots, n$. $\boldsymbol{a}_j$ and $\vec{\boldsymbol{a}}_i$ denote the jth column vector and the ith row vector of $\boldsymbol{A} = \boldsymbol{W}^{-1}$.

The matrix $\boldsymbol{D}$ in Eq. (17) corresponds to the Hessian matrix of $\frac{1}{N}\sum_{t=1}^{N}\sum_{i=1}^{n}\psi_i(y_i(t))$ that is the second term in Eq. (15). This is derived in a straightforward manner. The matrix $\boldsymbol{H}$ in Eq. (17) is the Hessian matrix of $-\log|\det \boldsymbol{W}|$. Its detailed derivation is explained below.

We define

$$\boldsymbol{G}_w = \frac{\partial}{\partial \boldsymbol{W}}\left[-\log|\det \boldsymbol{W}|\right] = -\boldsymbol{W}^{-T}. \tag{20}$$

The Hessian matrix $\boldsymbol{H}$ is derived from the relation

$$\begin{aligned}\boldsymbol{g}_w &= \mathrm{vec}\left(d\boldsymbol{G}_w^T\right) \\ &= \boldsymbol{H}\, d\boldsymbol{w}, \end{aligned} \tag{21}$$

where $\boldsymbol{g}_w \in \mathbb{R}^{n^2}$. Note that

$$\begin{aligned} d\boldsymbol{G}_w &= -\left(d\boldsymbol{W}^{-1}\right)^T \\ &= \boldsymbol{W}^{-T} d\boldsymbol{W}^T \boldsymbol{W}^{-T} \\ &= \boldsymbol{A}^T d\boldsymbol{W}^T \boldsymbol{A}^T. \end{aligned} \tag{22}$$

Using the identity, $\mathrm{tr}\,(\boldsymbol{AB}) = \left[\mathrm{vec}\left(\boldsymbol{A}^T\right)\right]^T \mathrm{vec}(\boldsymbol{B})$, the (i,j)-element of $d\boldsymbol{G}_w$ is computed by

$$\begin{aligned} [d\boldsymbol{G}_w]_{ij} &= \boldsymbol{a}_i^T d\boldsymbol{W}^T \vec{\boldsymbol{a}}_j^T \\ &= \mathrm{tr}\left(\vec{\boldsymbol{a}}_j^T \boldsymbol{a}_i^T d\boldsymbol{W}^T\right) \\ &= \left[\mathrm{vec}\left(\boldsymbol{a}_i \vec{\boldsymbol{a}}_j\right)\right]^T \mathrm{vec}\left(d\boldsymbol{W}^T\right) \\ &= \mathrm{vec}^T\left(\boldsymbol{a}_i \vec{\boldsymbol{a}}_j\right) d\boldsymbol{w}. \end{aligned} \tag{23}$$

It follows from Eq. (21) and Eq. (23) that $\boldsymbol{H}$ is the matrix consisting of n^2 row vectors, $\vec{\boldsymbol{h}}_m$, given in Eq. (19).

5.2 Relative TR-ICA Algorithm

The relative trust-region optimization exploits jointly the conventional trust-region method and the relative optimization. The main motivation of the relative trust-region optimization is from the relative Newton ICA [21],

The relative trust-region method consists of two modes: (1) relative mode; (2) absolute mode. The term 'absolute mode' is used to emphasize that in the algorithm the actual reduction is calculated with real objective function as in conventional trust-region method, while the step $\boldsymbol{p}^{(k)}$ and the predicted reduction are calculated in relative mode. In the relative mode, we consider an modified quadratic model, $m_r^{(k)}$ where the subscript r represents the relative mode and find a direction $\boldsymbol{p}^{(k)}$ which solves the following subproblem modified from Eq. (4),

$$\underset{\|\boldsymbol{p}\| \le \triangle^{(k)}}{\arg\min}\, m_r^{(k)}(\boldsymbol{p}) = f_r^{(k)} + \left[\nabla f_r^{(k)}\right]^T \boldsymbol{p} + \frac{1}{2}\boldsymbol{p}^T \nabla^2 f_r^{(k)} \boldsymbol{p}, \tag{24}$$

where $f_r(\boldsymbol{w}), \nabla f_r(\boldsymbol{w})$, and $\nabla^2 f_r(\boldsymbol{w})$ are the relative counterpart of $f(\boldsymbol{w}), \nabla f(\boldsymbol{w})$, and $\nabla^2 f(\boldsymbol{w})$, where the calculation is followed by replacing $\boldsymbol{W} = \boldsymbol{I}$ and $\boldsymbol{X} = \boldsymbol{Y}$.

For the case of ICA where the objective function Eq. (15) is considered, the objective function, gradient, and Hessian, in the relative mode, are given by

$$f_r(\boldsymbol{w}) = \frac{1}{N}\sum_{t=1}^{N}\sum_{i=1}^{n}\psi_i(y_i(t)), \tag{25}$$

$$\nabla f_r(\boldsymbol{w}) = \mathrm{vec}\left(-\boldsymbol{I} + \frac{1}{N}\sum_{t=1}^{N}\boldsymbol{y}(t)\left[\psi'(\boldsymbol{y}(t))\right]^T\right), \tag{26}$$

$$\nabla^2 f_r(\boldsymbol{w}) = \widetilde{\boldsymbol{H}} + \widetilde{\boldsymbol{D}}, \tag{27}$$

where $\widetilde{\boldsymbol{D}} \in \mathbb{R}^{n^2\times n^2}$ is a block-diagonal matrix, consisting of $\widetilde{\boldsymbol{D}}_l \in \mathbb{R}^{n\times n}$, $l = 1,\ldots,n$, that is of the form

$$\widetilde{\boldsymbol{D}}_l = \frac{1}{N}\sum_{t=1}^{N}\psi_l''(y_l(t))\boldsymbol{y}(t)\boldsymbol{y}^T(t), \tag{28}$$

and $\widetilde{\boldsymbol{H}} \in \mathbb{R}^{n^2\times n^2}$ consists of n^2 row vectors, $\widetilde{\vec{\boldsymbol{h}}}_m$, that is given by

$$\widetilde{\vec{\boldsymbol{h}}}_m = \mathrm{vec}^T\left(\boldsymbol{e}_i\boldsymbol{e}_j^T\right), \quad m = (i-1)n + j, \tag{29}$$

for $i = 1,\ldots,n$ and $j = 1,\ldots,n$ and $\boldsymbol{e}_i \in \mathbb{R}^n$ is the unit vector where the ith element is 1 and the rest of it are zero.

The basic idea in modifying the quadratic model comes from the fact that the relative optimization considers $f(\boldsymbol{I},\boldsymbol{Y})$ instead of $f(\boldsymbol{W},\boldsymbol{X})$. The predicted reduction is also calculated with equations in relative mode, considering the modified quadratic model $m_r^{(k)}$. The actual reduction, however, should be calculated in the original objective function $f(\boldsymbol{w})$ instead of $f_r(\boldsymbol{w})$. Therefore, in the relative trust-region method, the agreement measure $\rho^{(k)}$ in Eq. (6) is modified, which leads to $\rho_r^{(k)}$ that has the form

$$\rho_r^{(k)} = \frac{f\left(\boldsymbol{W}^{(k)}\right) - f\left(\boldsymbol{W}^{(k+1)}\right)}{m_r^{(k)}(\boldsymbol{0}) - m_r^{(k)}\left(\boldsymbol{p}^{(k)}\right)}, \tag{30}$$

where $\boldsymbol{W}^{(k+1)} = \left(\mathrm{mat}^T\left(\boldsymbol{p}^{(k)}\right) + \boldsymbol{I}\right)\boldsymbol{W}^{(k)}$.

That is, in the relative trust-region algorithm, the direction $\boldsymbol{p}^{(k)}$ and the predicted reduction $m_r^{(k)}(\boldsymbol{0}) - m_r^{(k)}\left(\boldsymbol{p}^{(k)}\right)$ are computed with equations in the relative mode and the actual reduction $f\left(\boldsymbol{w}^{(k)}\right) - f\left(\boldsymbol{w}^{(k+1)}\right)$ is computed with equations in the absolute mode. Then trust-region method determines the size of the region and finally parameters $\boldsymbol{W}$ are serially updated (like the relative optimization). The relative trust-region is summarized in Table 1.

Table 1
Relative TR-ICA Algorithm.

Given $\hat{\triangle} > 0$, $\triangle^{(0)} \in (0, \hat{\triangle})$, and $\zeta \in [0, \frac{1}{4})$:
for $k = 0, 1, 2, \ldots$
  $\boldsymbol{Y}^{(k)} = \boldsymbol{W}^{(k)}\boldsymbol{X}$
  Obtain $\boldsymbol{p}^{(k)}$ by solving Eq. (24);
  Calculate the agreement measure in Eq. (30);

  Evaluate $\rho_r^{(k)}$ in Eq. (30)
  if $\rho_r^{(k)} < \frac{1}{4}$, then $\triangle^{(k+1)} = \frac{1}{4}\|\boldsymbol{p}^{(k)}\|$
  else
    if $\rho_r^{(k)} > \frac{3}{4}$ and $\|\boldsymbol{p}^{(k)}\| = \triangle^{(k)}$, then $\triangle^{(k+1)} = \min(2\triangle^{(k)}, \hat{\triangle})$
    else, then $\triangle^{(k+1)} = \triangle^{(k)}$;

  if $\rho_r^{(k)} > \zeta$, then $\boldsymbol{W}^{(k+1)} = \left(\mathrm{mat}^T\left(\boldsymbol{p}^{(k)}\right) + \boldsymbol{I}\right)\boldsymbol{W}^{(k)}$
  else, then $\boldsymbol{W}^{(k+1)} = \boldsymbol{W}^{(k)}$
end (for)

5.3 A Trick for Memory-Efficiency

In the relative mode, the direction $\boldsymbol{p}^{(k)}$ is computed by solving the modified subproblem Eq. (24) which is involved with the calculation of the Hessian matrix $\nabla^2 f_r(\boldsymbol{w})$ in order to compute $\left[\nabla^2 f_r(\boldsymbol{w})\right]^{-1}\nabla f_r(\boldsymbol{w})$ and $\boldsymbol{v}^T\nabla^2 f_r(\boldsymbol{w})$ where $\boldsymbol{v}$ is a gradient or other learning directions. In the case of high-dimensional data, the Hessian matrix takes a huge memory space. As in the fast relative Newton method [21], we use the modified Newton direction which is found at low computational complexity, in order to take care of $\left[\nabla^2 f_r(\boldsymbol{w})\right]^{-1}\nabla f_r(\boldsymbol{w})$. Now we show that the term $\boldsymbol{v}^T\nabla^2 f_r(\boldsymbol{w})$ in Eq. (24) can be easily computed, due to a special structure of $\widetilde{\boldsymbol{H}}$ and $\widetilde{\boldsymbol{D}}$ in Eq. (27).

Let us consider

$$\boldsymbol{v}^T\nabla^2 f_r(\boldsymbol{w}) = \boldsymbol{v}^T\widetilde{\boldsymbol{H}} + \boldsymbol{v}^T\widetilde{\boldsymbol{D}}. \tag{31}$$

The second term $\boldsymbol{v}^T\widetilde{\boldsymbol{D}}$ can be efficiently computed by $\left[\boldsymbol{v} \odot \widetilde{\boldsymbol{d}}\right]^T$ where $\widetilde{\boldsymbol{d}} \in \mathbb{R}^{n^2}$ is a vector that keeps only diagonal elements of $\widetilde{\boldsymbol{D}}$, i.e., $[\widetilde{\boldsymbol{d}}]_i = [\widetilde{\boldsymbol{D}}]_{ii}$, $i = 1, \ldots, n^2$ and $\odot$ denotes the Hadamard product (element-wise multiplication). The first term $\boldsymbol{v}^T\widetilde{\boldsymbol{H}}$ in Eq. (31) can also be easily computed, considering that $\widetilde{\boldsymbol{H}}$ is nothing but a permutation matrix in the relative mode. In other words, $\boldsymbol{v}^T\widetilde{\boldsymbol{H}}$ requires just re-ordering the elements of $\boldsymbol{v}$, i.e.,

$$[\boldsymbol{v}^T \widetilde{\boldsymbol{H}}]_{(i-1)\times n+j} = [\boldsymbol{v}]_{i+j-1}, \quad i,j = 1,2,...n. \tag{32}$$

Therefore, the computation of the Hessian matrix is not required, which saves dramatically memory space. The trust-region, which our proposed algorithm uses, effects the learning direction and controls the step length. This gives us the better direction and proper step length at lower computational complexity, whereas fast relative Newton method uses just the Newton direction and employs the backtracking line search method for learning rate. Hence, the relative TR-ICA algorithm shows faster convergence, compared to the fast relative Newton ICA algorithm.

The computational cost to calculate the Hessian is the same as the relative Newton method, requiring $\mathcal{O}(n^2N)$. However, the total cost per iteration is much less than the relative Newton method because the step size is determined in different ways. Relative trust-region method determines the step size within the trust region whereas the relative Newton uses the backtracking line search method for each iteration (several iterations of the backtracking are required). The computational cost to calculate the trust region is $\mathcal{O}(n^2)$, which is negligible compared to $\mathcal{O}(n^2N)$. That is, the computational difference is mainly based on the backtracking line search method, where each iteration requires the calculation of function value including matrix inversion and determinant. Notice that the function evaluation is much more complex than other simple operations. See Table. 2 for the details. In Table. 2, (*) and (**) might be $32n^2+n+11$ and $29n^2+n-5$, respectively, when the step is determined on the trust-region boundary.

Table 2
Complexity Comparison between Newton and relative TR method in ICA with n-dimensional data.

Operations	Newton	relative TR	Backtracking
Multiplication, (x)	$20n^2+n$	$28n^2+n+3$ (*)	$4n^2$
Addition, (+)	$16n^2-3$	$23n^2+n-4$ (**)	$3n^2-n$
Square Root, $(\cdot)^{1/2}$	$n+2$	$n+3$	0
max/min/abs	1/0/1	1/1/1	0/0/0
Function Evaluation	1	1	1
Gradient (∇f)	1	0	0
Gradient (∇f_r)	1	1	0
Hessian	1	1	0

6 Numerical Experiments

We used 4 different data sets for our numerical experiments. The first data set contains the mixtures of two speech signals and one music signal, all of them were sampled at 8 kHz. The second set of data is the USPS data which contains handwritten digits that are converted to 256-dimensional data ($16 \times 16 = 256$) of length 4000. The third data set is DNA microarray data which contains the expression of 4026 human genes in 95 samples of normal and malignant lymphocytes [19], which are converted to the data matrix $\boldsymbol{X} \in \mathbb{R}^{95 \times 4026}$. The last data set consists of linear instantaneous mixtures of n independent binary sources (either +1 or -1) with 3000 data points for each source, where $n = 2, ..., 7$.

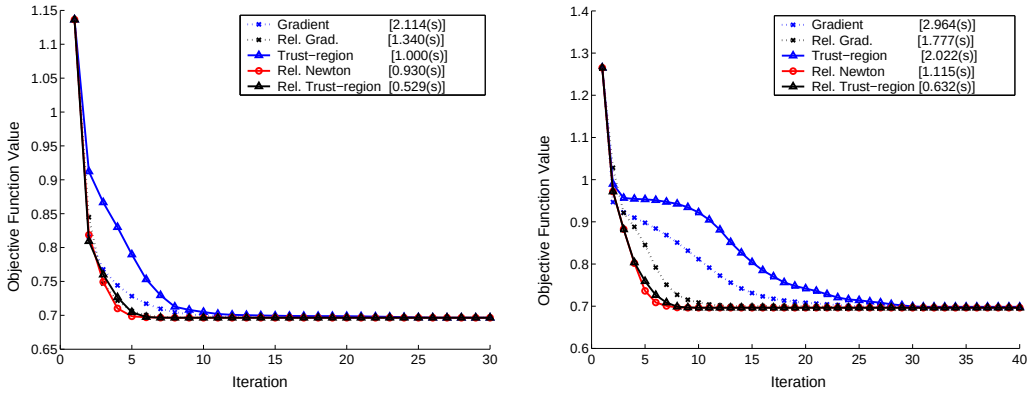
The optimization methods in ICA that we compared our relative TR-ICA with, include (1) the gradient; (2) the relative (or natural) gradient; (3) TR-ICA (with the dogleg implementation); (4) the fast relative Newton [2] [21]. In the gradient, the relative gradient, and Newton's methods, an optimal learning rate was determined by the backtracking line search method.

6.1 Comparison according to Mixing Matrix

For 3-dimensional sound data, three mixture signals were generated using the mixing matrix $\boldsymbol{A}$ where its condition number is 11.12 (well-conditioned). We executed every optimization method 50 times with randomly initialized demixing matrix $\boldsymbol{W}$. Fig. 1 (a) shows the average convergence comparison of several numerical optimization methods in ICA. The relative TR-ICA algorithm outperforms other methods in terms of the CPU time. The fast relative Newton method took the same number of iterations as the relative trust-region method, but ate up more CPU time due to its high time complexity. This complexity comes from the backtracking method that is used to find an optimal learning rate as in the gradient methods. Even using a pre-selected constant learning rate in the gradient-based ICA algorithms, the relative TR-ICA algorithm takes similar time complexity per iteration.

Relative optimization (or the natural gradient) was shown to have the equivariant property [8, 3, 11, 21]. In the second experiment, sound signals were artificially mixed using an ill-conditioned mixing matrix $\boldsymbol{A}$ where its condition number is 525.44. Fig. 1 (b) shows the average convergence behavior of various ICA algorithms. The relative algorithms (both relative Newton and relative trust-region) made convergence much faster than the gradient-based

[2] The matlab code is available at `http://ie.technion.ac.il/~mcib/relnwt021203.zip`



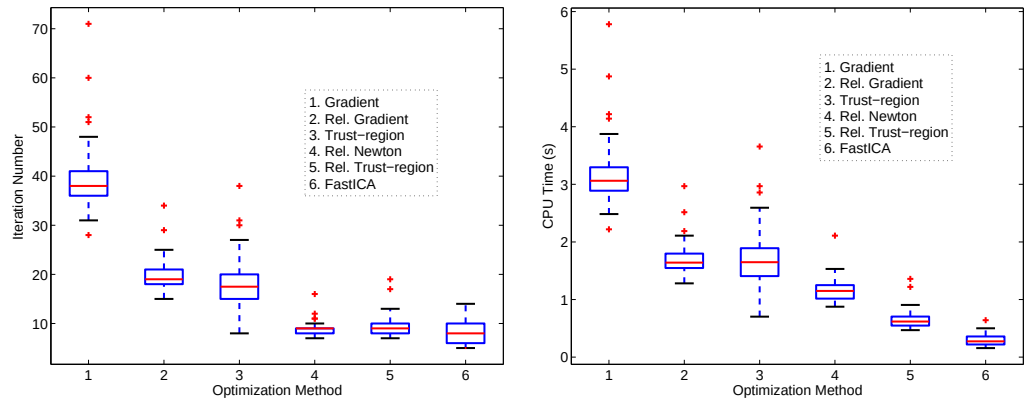
(a) (b)

Fig. 1. Convergence comparison of several numerical optimization methods in the quasi maximum likelihood ICA for a set of sound data: (a) for well-conditioned mixing matrix (b) for ill-conditioned mixing matrix.

algorithms. Once again, the relative trust-region algorithm was the fastest one among all algorithms that we tested.

6.2 Statistical Comparison

To compare our proposed method with other optimization methods, we used 3-dimensional sound data again. We executed every optimization method including FastICA 50 times with random mixing matrix $\boldsymbol{A}$. Fig. 2 shows the statistical comparison of iteration numbers and CPU times until convergence, respectively.



(a) (b)

Fig. 2. Statistical comparison of several optimization methods: (a) Comparison of iteration numbers (b) Comparison of CPU times.

In Fig. 2 (a) which compares the iteration numbers, Newton method including FastICA is slightly better than others. Notice that the variances of relative

methods are smaller than the variances of others. This shows the equivariant property of relative optimization where the performances are consistent disregarding the mixing matrix. Comparing (a) with (b), we can see other interesting points. Even though relative Newton converges with slightly less number of iterations than relative trust-region, relative trust-region is faster than relative Newton. As stated above, Newton method requires higher time complexity per iteration. Actually, FastICA which is also a kind of Newton-type algorithm, is best in both of iteration numbers and CPU time. In (a), however, the variance of FastICA is larger than relative methods, which means it has no equivariance property. In addition, when we have insufficient data samples, FastICA could be unstable in contrast to other methods. In next sections, we will see the weak points of FastICA.

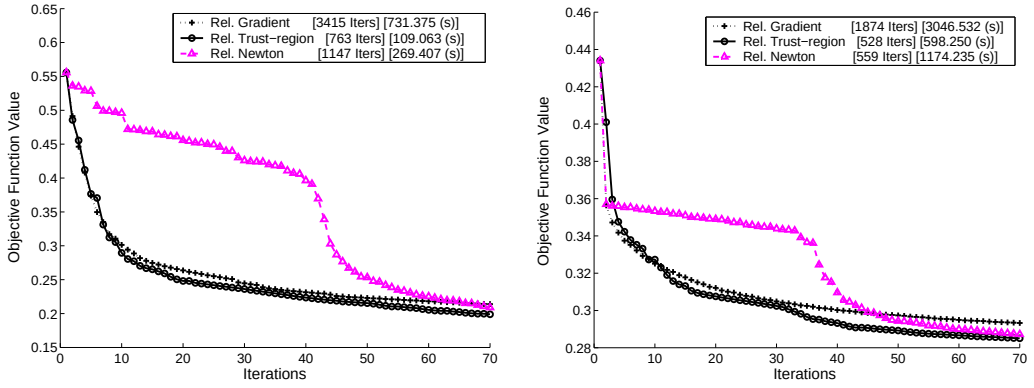
6.3 High-Dimensional Data

This experiment was conducted with high-dimensional data sets which are 256-dimensional USPS handwritten data set and 95-dimensional DNA microarray data set. For the case of these data sets, the computation of Hessian matrix at each iteration is very expensive and the Hessian matrix is not positive definite at some iteration steps. Moreover, if dimension is very high (more than 30), then conventional trust-region method cause a memory problem. However, the relative TR-ICA algorithm overcomes these limits by using a trick for memory-efficiency.

Fig. 3 confirms that the relative TR-ICA algorithm method works well, even with high-dimensional data. Our proposed algorithm is faster than any other methods in high dimensional data set. In this figure, even the iteration number of relative TR-ICA is much less than fast relative Newton's, as well as the CPU time. This means that the objective functions for high-dimensional real data have difficulty to be modelled with quadratic equation, so the recommended learning direction of trust-region method is better than Newton method's. In (a), one can observe that the objective value in the relative trust-region method decreased much faster than the relative Newton method at the first iteration. This stresses out the characteristics of the relative trust-region method, which is, the trust-region method is much better than the Newton method when the quadratic model does not approximate the objective function well.

In the numerical experiment with USPS data, we chose the portion associated with the digit '2' and reduced the dimension by PCA, which produced 100-dimensional data of length 379. In such a case, FastICA [15] had difficulty in convergence because the number of data points were not enough. The small number of data points does not guarantee the objective function to be a

contraction map which is a necessary condition for the fixed point algorithm. That is, our proposed algorithm is the best optimization method when the dimension of data set is very high and the number of data points is not enough to guarantee FastICA to be stable. In addition, we will discuss another weak point of FastICA in next experiment.



(a) (b)

Fig. 3. Convergence comparison of several numerical optimization methods in the quasi maximum likelihood ICA: (a) for digit '2' among USPS data (Dimension is reduced into 100 by PCA) (b) for DNA micro array data.

6.4 Comparison with FastICA

In this experiment, the equivariant property of relative trust-region method is shown clearly compared with FastICA. The binary sources were artificially mixed using an Hilbert matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ given by

$$\boldsymbol{A}(i,j) = 1/(i+j), \tag{33}$$

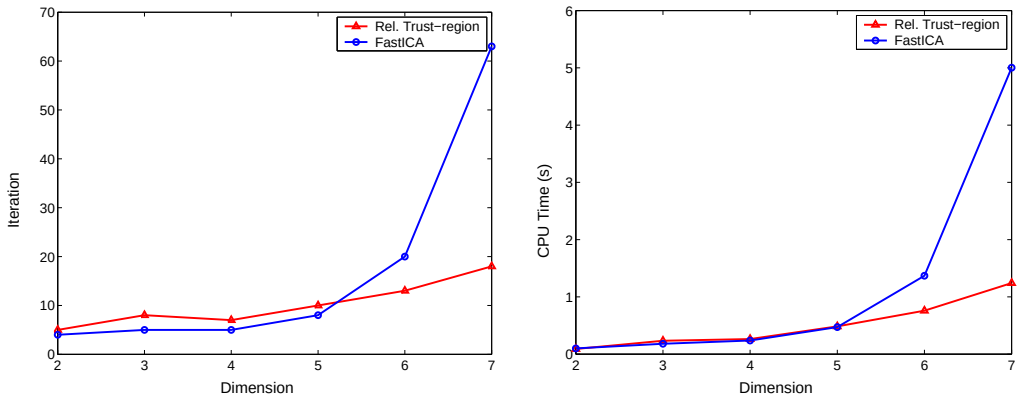
which is an ill-conditioned mixing matrix. As the dimension of binary sources increases, the condition number of the Hilbert matrix $\boldsymbol{A}$ also increases (See Table. 3).

Table 3
Condition numbers of Hilbert matrix $\boldsymbol{A}$ corresponding to dimension

Dimension n	2	3	4	5	6	7
Cond($\boldsymbol{A}$)	3.84e+1	1.35e+3	4.58e+4	1.53e+6	5.10e+7	1.69e+9

For each dimension, we executed both of relative trust-region and FastICA 5 times and Fig. 4 shows the graph of iteration numbers and average CPU times for two algorithm until convergence. When the dimension is greater than 7, both do not converge. In relative trust-region, iteration number and CPU time increase linearly which means the equivariant property, whereas FastICA

rises exponentially. So, in ill-conditioned high-dimensional data, relative trust-region converges much faster than FastICA.



(a) (b)

Fig. 4. Equivariant property of Relative TR-ICA compared with FastICA: (a) Iteration numbers until convergence (b) Average CPU time on 5 experiments

7 Discussions

We have introduced a relative trust-region method which jointly exploited the trust-region method and the relative optimization. In the relative trust-region method, a direction and a step size were searched with the help of a quadratic model (like the trust-region method) and the parameters were serially updated (relative optimization). The trust-region algorithm took much less number of iterations for convergence, compared to the gradient algorithms and required less CPU time, compared to the Newton method. In the relative optimization, the Riemannian structure of the parameter space was exploited.

We have applied this relative trust-region method to the problem of ICA, which led to the relative TR-ICA algorithm. The relative TR-ICA algorithm enjoyed fast convergence, compared to most of existing ICA algorithms and showed the equivariant property like the natural or the relative gradient. Moreover exploiting a special structure of the Hessian matrix in the relative mode led to a memory-efficient relative TR-ICA algorithm which could handle high-dimensional data. Especially in the case of an ill-conditioned mixing matrix or in the case of high-dimensional data, the useful behavior and the high performance of the relative TR-ICA algorithm were observed in terms of the iteration numbers and CPU time.

FastICA [15] was also considered in comparison with the relative TR-ICA algorithm. In fact, FastICA seemed a little bit faster than our proposed algorithm. However, in the case of small size of data, FastICA did not work

well, whereas our algorithm worked fine. Moreover, in ill-conditioned high-dimensional data, relative trust-region converges much faster than FastICA. Although we focused on ICA, the relative trust-region method could be applied to other machine learning problems.

It has been brought to the notice of the authors during the review process that trust-region methods on Riemannian manifolds (which share similar spirit with our method) have been independently developed and applied to numerical linear algebra problems [1] .

Acknowledgments

We appreciate anonymous reviewers' and associate editor's critical comments, which improved the clarity of our paper. This work was supported by Korea Ministry of Commerce, Industry, and Energy under Brain Neuroinformatics Program, KOSEF International Cooperative Research Program, Korea MIC under ITRC support program supervised by the IITA (IITA-2005-C1090-0501-0018), and Basic Research Fund (2004) in POSTECH. Heeyoul Choi was also supported by KOSEF Grant funded by Korea government (No.D00115).

References

[1] P. -A. Absil, C. G. Baker, and K. A. Gallivan. Trust-region methods on Riemannian manifolds. Technical Report FSU-CSIT-04-13, School of Computational Science, Florida State University, 2004.

[2] T. Akuzawa. Extended quasi-Newton method for the ICA. In *Proc. ICA*, pages 521–525, Helsinki, Finland, 2000.

[3] S. Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998.

[4] S. Amari and A. Cichocki. Adaptive blind signal processing - neural network approaches. *Proc. of the IEEE, Special Issue on Blind Identification and Estimation*, 86(10):2026–2048, Oct. 1998.

[5] S. Amari, A. Cichocki, and H. H. Yang. A new learning algorithm for blind signal separation. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 757–763. MIT press, 1996.

[6] J. -F. Cardoso. Infomax and maximum likelihood for source separation. *IEEE Signal Processing Letters*, 4(4):112–114, Apr. 1997.

[7] J. -F. Cardoso. Learning in manifolds: The case of source separation. In *Proc. SSAP*, Portland, Oregon, 1998.

[8] J. -F. Cardoso and B. H. Laheld. Equivariant adaptive source separation. *IEEE Trans. Signal Processing*, 44(12):3017–3030, Dec. 1996.

[9] H. Choi and S. Choi. Relative trust-region learning for ICA. In *Proc. IEEE Int'l Conf. Acoustics, Speech, and Signal Processing*, Philadelphia, PA, 2005.

[10] H. Choi, S. Kim, and S. Choi. Trust-region learning for ICA. In *Proc. Int'l Joint Conf. Neural Networks*, pages 41–46, Budapest, Hungary, 2004.

[11] S. Choi, A. Cichocki, and S. Amari. Equivariant nonstationary source separation. *Neural Networks*, 15(1):121–130, 2002.

[12] S. Choi, A. Cichocki, H. -M. Park, and S. -Y. Lee. Blind source separation and independent component analysis: A review. *Neural Information Processing - Letters and Review*, 6(1):1–57, 2005.

[13] A. Cichocki and S. Amari. *Adaptive Blind Signal and Image Processing: Learning Algorithms and Applications*. John Wiley & Sons, Inc., 2002.

[14] S. Fiori. Quasi-geodesic neural learning algorithms over the orthogonal group: A tutorial. *Journal of Machine Learning Research*, 6:743–781, 2005.

[15] A. Hyvärinen. Fast and robust fixed-point algorithms for independent component analysis. *IEEE Trans. Neural Networks*, 10(3):626–634, 1999.

[16] A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. John Wiley & Sons, Inc., 2001.

[17] T. -W. Lee. *Independent Component Analysis: Theory and Applications*. Kluwer Academic Publishers, 1998.

[18] T. -W. Lee, M. Girolami, A. Bell, and T. Sejnowski. A unifying information-theoretic framework for independent component analysis. *International Journal on Mathematical and Computer Modeling*, 39(11):1–21, 2000.

[19] W. Liebermeister. Linear modes of gene expression determined by independent component analysis. *Bioinformatics*, 18(1):51–60, 2002.

[20] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 1999.

[21] M. Zibulevsky. Blind source separation with relative Newton method. In *Proc. ICA*, pages 897–902, Nara, Japan, 2003.