Relative Gradient Learning for Independent Subspace Analysis

Heeyoul Choi and Seungjin Choi

***Abstract*— Independent subspace analysis (ISA) is a generalization of independent component analysis (ICA), where multidimensional ICA is incorporated with the idea of invariant feature subspaces, allowing components in the same subspace to be dependent, but requiring independence between feature subspaces. In this paper we present a relative gradient algorithm for ISA, derived in the framework of the relative optimization as well as in a direct manner. Empirical comparison with the gradient ISA algorithm, shows that the relative gradient ISA algorithm achieves faster convergence, compared to the conventional gradient algorithm.**

I. Introduction

Learning a statistical structure (or regularity) of observed data, plays an important role in pattern recognition and machine learning. Independent component analysis (ICA) is a popular method for this, seeking a linear transform decomposing multivariate data into a linear sum of non-orthogonal basis vectors with coefficients (encoding variables, components) being statistically independent.

An interesting generalization of ICA is the independent subspace analysis (ISA) where instead of maximizing the independence between components, the independence between the norms of projection on linear subspaces is maximized [8]. A basic idea in ISA is to embed invariant feature subspaces in multidimensional ICA [3], allowing components in the same subspace to be dependent, but requiring independence between feature subspaces. In the context of complex cells in primary visual cortex, the emergence of phase- and shift-invariant features, was well illustrated using ISA [8]. Successful applications of ISA can be found in face recognition [11], speech processing [5], and bioinformatics [9], [10].

In this paper we present a relative gradient algorithm for ISA, the derivation of which is in the framework of the relative optimization proposed in [12]. We also derive the relative gradient algorithm in a direct manner and compare these two algorithms. Actually these two algorithms are exactly equivalent to each other.

In experimental results for 3 image data set, comparisons with the gradient ISA algorithm show that the relative gradient ISA algorithm achieves faster convergence, compared to the gradient algorithm.

II. Independent Subspace Analysis

Linear data models assume that the data matrix $\boldsymbol{X} = [\boldsymbol{x}(1) \cdots \boldsymbol{x}(N)] \in \mathbb{R}^{m \times N}$ (consisting of m-dimensional data vectors) is of the form

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

where $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ contains basis vectors in its columns and the row vectors of $\boldsymbol{S} = [\boldsymbol{s}(1) \cdots \boldsymbol{s}(N)] \in \mathbb{R}^{n \times N}$ represent the time course of encoding variables (coefficients, components, latent variables).

ICA seeks a representation Eq. (1) with the row vectors of $\boldsymbol{S}$ being statistically independent. In general, this can be achieved by finding a linear transform $\boldsymbol{W} = [\boldsymbol{w}_1 \cdots \boldsymbol{w}_m]^T$ such that the row vectors of $\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{X}$ are statistically independent. Multidimensional ICA [3] is a generalization of ICA, allowing the components in a κ-tuple to be dependent but requiring different κ-tuples to be independent. ISA embeds invariant feature subspaces in multidimensional ICA by considering the probability distributions for the κ-tuples of encoding variables that are spherically symmetric, i.e., depend only on the norm. In contrast to ICA, ISA finds a linear transformation $\boldsymbol{W}$ such that feature subspaces become independent but components in a feature subspace are allowed to be dependent. The feature subspaces are obtained by taking a square root of the sum of energy of responses.

For the sake of simplicity, we consider the case where $m = n$ and $\boldsymbol{s}(t) \in \mathbb{R}^n$ are divided into J number of κ-tuple (where κ represents the dimension of subspace, i.e., $n = \kappa J$). Thus, the data matrix given by $\boldsymbol{X} \in \mathbb{R}^{n \times N}$ and a linear transform that ISA seeks, is given by $\boldsymbol{W} \in \mathbb{R}^{n \times n}$. We also assume an identical dimension, κ, for every feature subspace. The jth feature subspace is denoted by $\mathcal{F}_j$.

ISA finds a linear transform $\boldsymbol{W} \in \mathbb{R}^{n \times n}$ which maximizes the independence of the norms of the projection on linear subspaces. The following normalized log-likelihood was considered in ISA,

$$\begin{aligned} \mathcal{L}(\boldsymbol{W}, \boldsymbol{X}) &= \frac{1}{N} \sum_{t=1}^{N} \sum_{j=1}^{J} \log p \left(\sum_{i \in \mathcal{F}_j} \left(\boldsymbol{w}_i^T \boldsymbol{x}(t) \right)^2 \right) \\ &\quad + \log |\det \boldsymbol{W}| , \end{aligned} \tag{2}$$

where $p\left(\sum_{i \in \mathcal{F}_j} y_i^2(t)\right)$ represents the probability density inside the jth κ-tuple of $y_i(t) = \boldsymbol{w}_i^T \boldsymbol{x}(t)$. The cost function $\mathcal{J}(\boldsymbol{W}, \boldsymbol{X})$ is taken as the negative normalized log-likelihood, which has the form

$$\begin{aligned} \mathcal{J}(\boldsymbol{W}, \boldsymbol{X}) &= -\frac{1}{N} \sum_{t=1}^{N} \sum_{j=1}^{J} \log p \left(\sum_{i \in \mathcal{F}_j} y_i^2(t) \right) \\ &\quad - \log |\det \boldsymbol{W}| . \end{aligned} \tag{3}$$

We define

$$\psi \left(\sum_{i \in \mathcal{F}_j} y_i^2(t) \right) = -\log p \left(\sum_{i \in \mathcal{F}_j} y_i^2(t) \right) , \tag{4}$$

Heeyoul Choi is with the Department of Cognitive and Neural Systems, Boston University, USA (email: heeyoul@gmail.com).

Seungjin Choi is with the Department of Computer Science, Pohang University of Science and Technology, Korea (email: seungjin@postech.ac.kr).

where the probability distribution $p(\cdot)$ is assumed to be super-Gaussian, as in [7], [8], that has the form

$$\log p\left(\sum_{i\in\mathcal{F}_j} y_i^2(t)\right) = \alpha\left[\sum_{i\in\mathcal{F}_j} y_i^2(t)\right]^{\frac{1}{2}} + \beta, \tag{5}$$

where α and β are a scaling and a normalization constant that are determined to be compatible with the constraint of unit variance of $y_i(t)$. The hypothesized distribution $p(\cdot)$ is chosen appropriately, depending on applications. The log-distribution in Eq. (5) is one exemplary distribution that we can consider.

We also define $\boldsymbol{\xi}(t) = [\xi_1(t)\ \cdots\ \xi_n(t)]^T$ where

$$\xi_i(t) = \sum_{l\in\mathcal{F}_{j(i)}} y_l^2(t) = \sum_{l\in\mathcal{F}_{j(i)}} \left(\boldsymbol{w}_l^T\boldsymbol{x}(t)\right)^2, \tag{6}$$

where $\mathcal{F}_{j(i)}$ is the feature subspace to which $\boldsymbol{w}_i$ belongs. With these definitions, applying the gradient descent method, leads to the following updating rule for $\boldsymbol{W}$:

$$\begin{aligned}\triangle\boldsymbol{W} &= \eta\frac{\partial\mathcal{J}(\boldsymbol{W},\boldsymbol{X})}{\partial\boldsymbol{W}} \\ &= \eta\left\{\boldsymbol{W}^{-T} - \sum_{t=1}^{N}\left\{\left[\psi'(\boldsymbol{\xi}(t))\odot\boldsymbol{y}(t)\right]\boldsymbol{x}^T(t)\right\}\right\}\end{aligned} \tag{7}$$

where $\eta > 0$ is a learning rate, $\psi' = -p'/p$ (negative score function), i.e., $\psi'(\xi_i(t)) = \frac{1}{2}\alpha\xi_i^{-\frac{1}{2}}(t)$, and $\odot$ is the Hadamard product (which is the element-wise product). Define a matrix $\boldsymbol{\Phi} = [\boldsymbol{\varphi}(1)\cdots\boldsymbol{\varphi}(N)] \in \mathbb{R}^{n\times N}$ where $\boldsymbol{\varphi}(t) = \psi'(\boldsymbol{\xi}(t))\odot\boldsymbol{y}(t)$. Then, (7) can be written in a compact form,

$$\triangle\boldsymbol{W} = \eta\left\{\boldsymbol{W}^{-T} - \boldsymbol{\Phi}\boldsymbol{X}^T\right\}. \tag{8}$$

For the case where the data matrix $\boldsymbol{X}$ is already whitened, then the linear transform $\boldsymbol{W}$ is constrained to be an orthogonal matrix. In such a case, the cost function is simplified as

$$\mathcal{J}(\boldsymbol{W},\boldsymbol{X}) = -\frac{1}{N}\sum_{t=1}^{N}\sum_{j=1}^{J}\log p\left(\sum_{i\in\mathcal{F}_j} y_i^2(t)\right). \tag{9}$$

The associated gradient descent algorithm is also of a simpler form

$$\triangle\boldsymbol{W} = -\eta\boldsymbol{\Phi}\boldsymbol{X}^T, \tag{10}$$

which was originally proposed in [8]

III. Relative Gradient for ISA

This section describes the relative optimization method for ISA from the viewpoint of Lie group.

A. 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 resulted from the idea of learning in Lie group, which was first investigated by Cardoso [2].

The general linear group of degree n, denoted by $GL(n)$, is a set of invertible (nonsingular) $n\times n$ matrices. The general linear group is an instance of Lie group. In the case of ISA as ICA, the parameter matrix $\boldsymbol{W}$ belongs to the general linear group $GL(n)$. A matrix $\boldsymbol{W}$ in $GL(n)$ gives rise to an invertible linear transformation $\Pi : \mathbb{R}^n \to \mathbb{R}^n$, defined by $\Pi(\boldsymbol{x}) = \boldsymbol{W}\boldsymbol{x}$, and the matrix multiplication in the group corresponds to composition of linear transformations. Learning a demixing matrix $\boldsymbol{W}$, can be carried out by a linear transformation of parameters, which leads to the following learning process

$$\boldsymbol{W}^{(k+1)} = \boldsymbol{E}^{(k)}\boldsymbol{W}^{(k)}, \tag{11}$$

where $\boldsymbol{E}^{(k)}$ is a linear transformation of parameters $\boldsymbol{W}^{(k)}$. It follows from Eq. (11) 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 $\boldsymbol{E}^{(k)}$ in 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 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. (11) reflects a manifold. In fact, the natural gradient in ICA (which is identical to the relative gradient in ICA) was developed in the framework of learning in Riemannian manifold [1].

B. Equivariant Property and Serial Update

An estimator $\mathcal{A}$ for $\boldsymbol{A}$ is said to be equivariant if it satisfies

$$\mathcal{A}(\boldsymbol{M}\boldsymbol{X}) = \boldsymbol{M}\mathcal{A}(\boldsymbol{X}), \tag{12}$$

for any invertible $n\times n$ matrix $\boldsymbol{M}$. An important property induced by an equivariant estimator is the uniform performance which implies that the performance of an estimator does not depend on the mixing matrix $\boldsymbol{A}$ in ICA and ISA. Suppose that source signals are estimated as $\boldsymbol{y}(t) = \hat{\boldsymbol{s}}(t) = \boldsymbol{W}\boldsymbol{x}(t) = \boldsymbol{A}^{-1}\boldsymbol{x}(t)$. Then, we have

$$\begin{aligned}\hat{\boldsymbol{s}}(t) &= (\mathcal{A}(\boldsymbol{X}))^{-1}\boldsymbol{x}(t) = (\mathcal{A}(\boldsymbol{A}\boldsymbol{S}))^{-1}\boldsymbol{A}\boldsymbol{s}(t) \\ &= (\boldsymbol{A}\mathcal{A}(\boldsymbol{S}))^{-1}\boldsymbol{A}\boldsymbol{s}(t) = \mathcal{A}(\boldsymbol{S})^{-1}\boldsymbol{s}(t).\end{aligned} \tag{13}$$

Here, source signals estimated by an equivariant estimator $\mathcal{A}$ are given by $\hat{\boldsymbol{s}}(t) = \mathcal{A}(\boldsymbol{S})^{-1}\boldsymbol{s}(t)$, that is, they depend solely on original source signals $\boldsymbol{s}$. In other words, the performance of an equivariant algorithm does not depend at all (theoretically) on the mixing matrix [2], [4]. This equivariant property can be achieved by the 'serial update'.

A family of adaptive ISA algorithms employs an update rule that has the form

$$\boldsymbol{W}^{(k+1)} = \boldsymbol{W}^{(k)} - \eta^{(k)}\widetilde{G}\left(\boldsymbol{X},\boldsymbol{W}^{(k)}\right), \tag{14}$$

where $\widetilde{G}\left(\boldsymbol{X}, \boldsymbol{W}^{(k)}\right)$ is a matrix-valued function and $\eta^{(k)} > 0$ is a learning rate. Without loss of generality, we let

$$\widetilde{G}\left(\boldsymbol{X}, \boldsymbol{W}^{(k)}\right) = G\left(\boldsymbol{W}^{(k)}\boldsymbol{X}, \boldsymbol{W}^{(k)}\right)\boldsymbol{W}^{(k)},$$

so that the update rule becomes

$$\boldsymbol{W}^{(k+1)} = \left(\boldsymbol{I} - \eta^{(k)} G\left(\boldsymbol{Y}^{(k)}, \boldsymbol{W}^{(k)}\right)\right)\boldsymbol{W}^{(k)}, \quad (15)$$

where $\boldsymbol{Y}^{(k)} = \boldsymbol{W}^{(k)}\boldsymbol{X}$. Note that this is a multiplicative fashion.

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

$$\boldsymbol{W}^{(k+1)} = \left(\boldsymbol{I} - \eta^{(k)} G\left(\boldsymbol{Y}^{(k)}\right)\right)\boldsymbol{W}^{(k)}. \quad (16)$$

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 update is carried out in a multiplicative fashion. Note that the binary operator in this transformation is a left matrix multiplication and that update 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

$$\boldsymbol{W}^{(k+1)} = \boldsymbol{E}^{(k)}\boldsymbol{W}^{(k)} = \boldsymbol{E}^{(k)}\boldsymbol{E}^{(k-1)}\cdots\boldsymbol{E}^{(1)}\boldsymbol{E}^{(0)}\boldsymbol{W}^{(0)}, \quad (17)$$

where $\boldsymbol{W}^{(0)}$ is an initial value of $\boldsymbol{W}$. It follows from Eq. (17) that the serial update rule for $\boldsymbol{W}^{(k)}$ reflects a manifold as mentioned in previous section. When the final convergence is achieved after c iterations, the stationary point $\boldsymbol{W}_*$ also consists of series of multiplications of matrices, $\boldsymbol{W}_* = \boldsymbol{E}_*\boldsymbol{W}^{(0)}$ where $\boldsymbol{E}_* = \boldsymbol{E}^{(c)}\boldsymbol{E}^{(c-1)}\cdots\boldsymbol{E}^{(1)}\boldsymbol{E}^{(0)}$. Even in the case of a different mixing matrix $\boldsymbol{A}'$ being involved, if we set the $\boldsymbol{W}^{(0)\prime}$ as an initial matrix of $\boldsymbol{W}$ such that $\boldsymbol{W}^{(0)}\boldsymbol{A} = \boldsymbol{W}^{(0)\prime}\boldsymbol{A}'$, then $\boldsymbol{W}_* = \boldsymbol{E}_*\boldsymbol{W}^{(0)\prime}$ and updating rules are identical to each other, which implies the uniform performance.

C. Relative Optimization

The relative gradient involves the plugging matrix containing the first-order information (in the sense that the gradient is used). This can be generalized by computing the plugging matrix using other optimization methods (for instance, Newton method). The relative optimization [12] is summarized in Table I.

TABLE I
RELATIVE OPTIMIZATION ALGORITHM.

Start with an initial estimate $\boldsymbol{W}^{(0)}$;
repeat $k = 0, 1, 2, ...,$ until convergence
  $\boldsymbol{Y}^{(k)} = \boldsymbol{W}^{(k)}\boldsymbol{X}$;
  Starting with $\boldsymbol{V}^{(k)} = \boldsymbol{I}$ (identity matrix),
  Compute $\boldsymbol{V}^{(k+1)}$ which significantly decreases the cost function $\mathcal{J}(\boldsymbol{V}, \boldsymbol{Y}^{(k)})$;
  Update $\boldsymbol{W}$ by $\boldsymbol{W}^{(k+1)} = \boldsymbol{V}^{(k+1)}\boldsymbol{W}^{(k)}$
end (repeat)

The relative gradient algorithm can be easily derived in the framework of the relative optimization. We set $\boldsymbol{Y}^{(k)} = \boldsymbol{W}^{(k)}\boldsymbol{X}$ and $\boldsymbol{V}^{(k)} = \boldsymbol{I}$ and define the gradient of $\mathcal{J}_r$, $\nabla\mathcal{J}_r$ as

$$\nabla\mathcal{J}_r = \left.\frac{\partial\mathcal{J}(\boldsymbol{W}, \boldsymbol{X})}{\partial\boldsymbol{W}}\right|_{\boldsymbol{W}=\boldsymbol{I}, \boldsymbol{X}=\boldsymbol{Y}}, \quad (18)$$

where

$$\frac{\partial\mathcal{J}(\boldsymbol{W}, \boldsymbol{X})}{\partial\boldsymbol{W}} = -\boldsymbol{W}^{-T} + \boldsymbol{\Phi}\boldsymbol{X}^T. \quad (19)$$

Then we have $\Delta\boldsymbol{V} = -\eta^{(k)}\nabla\mathcal{J}_r(\boldsymbol{W}^{(k)})$, and the multiplicative gain matrix $\boldsymbol{V}^{(k+1)}$ is obtained by

$$\boldsymbol{V}^{(k+1)} = \boldsymbol{I} - \eta^{(k)}\nabla\mathcal{J}_r(\boldsymbol{W}^{(k)}). \quad (20)$$

Taking $\boldsymbol{W}^{(k+1)} = \boldsymbol{V}^{(k+1)}\boldsymbol{W}^{(k)}$ into account, leads to

$$\boldsymbol{W}^{(k+1)} = (\boldsymbol{I} - \eta^{(k)}\nabla\mathcal{J}_r(\boldsymbol{W}^{(k)}))\boldsymbol{W}^{(k)}. \quad (21)$$

¿From Eq. (21), we can obtain the relative gradient algorithm

$$\begin{aligned}\Delta\boldsymbol{W} &= \Delta\boldsymbol{V}^{(k)}\boldsymbol{W}^{(k)} = -\eta^{(k)}\nabla\mathcal{J}_r(\boldsymbol{W}^{(k)})\boldsymbol{W}^{(k)} \\ &= \left\{\boldsymbol{I} - \boldsymbol{\Phi}^{(k)}\left[\boldsymbol{Y}^{(k)}\right]^T\right\}\boldsymbol{W}^{(k)}. \quad (22)\end{aligned}$$

The relative gradient ISA algorithm (22) can also be derived in a direct manner, following the original idea of the relative gradient for ICA in [4] or the natural gradient in [1]. Next section describes the direct derivation of the relative gradient ISA algorithm and shows that two relative gradient ISA algorithms are exactly equivalent.

D. Relative Gradient

In previous section III-B, we did not define a specific form of the function $G(\cdot)$. Here we illustrate an example of the function $G(\cdot)$ by the relative gradient.

Let $\mathcal{J}(\boldsymbol{W})$ be a real differentiable function, then the relative gradient of $\mathcal{J}$ at $\boldsymbol{W}$, denoted by $\nabla_r\mathcal{J}(\boldsymbol{W})$, constitutes

$$\mathcal{J}(\boldsymbol{W} + \mathcal{E}\boldsymbol{W}) = \mathcal{J}(\boldsymbol{W}) + \mathrm{tr}\left(\nabla_r\mathcal{J}^T(\boldsymbol{W})\mathcal{E}\right) + o(\mathcal{E}), \quad (23)$$

for any square matrix $\mathcal{E}$ where tr denotes the trace operator. Taking $\mathcal{E} = -\lambda\nabla_r\mathcal{J}(\boldsymbol{W})$, leads to the minimization of $\mathcal{J}(\boldsymbol{W} + \mathcal{E}\boldsymbol{W}) - \mathcal{J}(\boldsymbol{W})$. Hence,

$$\Delta\boldsymbol{W} = \mathcal{E}\boldsymbol{W} = -\lambda\nabla_r\mathcal{J}(\boldsymbol{W})\boldsymbol{W}, \quad (24)$$

and the relative gradient algorithm has the form

$$\boldsymbol{W}^{(k+1)} = \left(\boldsymbol{I} - \eta^{(k)}\nabla_r\mathcal{J}\left(\boldsymbol{W}^{(k)}\right)\right)\boldsymbol{W}^{(k)}. \quad (25)$$

In this case, the function $G(\cdot)$ is defined by $G(\cdot) = \nabla_r\mathcal{J}(\boldsymbol{W})$.

In order to see a relation between the relative gradient and the absolute gradient[1], we consider the first-order Taylor

[1]The absolute gradient corresponds to the conventional gradient which is the partial derivative of a cost function with respect to parameters. The term 'absolute' is used to emphasize that it is a counterpart of the relative gradient.

series expansion in terms of the absolute gradient,

$$\begin{aligned}\mathcal{J}(\boldsymbol{W}+\mathcal{E}\boldsymbol{W}) &= \mathcal{J}(\boldsymbol{W}) + \mathrm{tr}\left(\nabla\mathcal{J}^T(\boldsymbol{W})\mathcal{E}\boldsymbol{W}\right) + o(\mathcal{E}) \\ &= \mathcal{J}(\boldsymbol{W}) + \mathrm{tr}\left((\nabla\mathcal{J}(\boldsymbol{W})\boldsymbol{W}^T)^T\mathcal{E}\right) + o(\mathcal{E}), \quad (26)\end{aligned}$$

where

$$\nabla\mathcal{J}(\boldsymbol{W}) = \frac{\partial\mathcal{J}(\boldsymbol{W},\boldsymbol{X})}{\partial\boldsymbol{W}}.$$

Taking $\mathcal{E} = -\lambda\nabla\mathcal{J}(\boldsymbol{W})\boldsymbol{W}^T$, leads to the minimization of $\mathcal{J}(\boldsymbol{W}+\mathcal{E}\boldsymbol{W}) - \mathcal{J}(\boldsymbol{W})$. Comparing Eq. (23) and Eq. (26), leads to a relation

$$\nabla_r\mathcal{J}(\boldsymbol{W}) = \nabla\mathcal{J}(\boldsymbol{W})\boldsymbol{W}^T. \quad (27)$$

Finally, considering Eq. (24) with Eq. (27) leads to a relative gradient updating rule:

$$\Delta\boldsymbol{W} = -\lambda\nabla\mathcal{J}(\boldsymbol{W})\boldsymbol{W}^T\boldsymbol{W} = -\lambda\left\{\boldsymbol{I} - \boldsymbol{\Phi}^{(k)}\left[\boldsymbol{Y}^{(k)}\right]^T\right\}\boldsymbol{W}^{(k)}. \quad (28)$$

Then, Eq. (28) is identical to Eq. (22), which implies that the relative gradient is a special instance of the relative optimization.

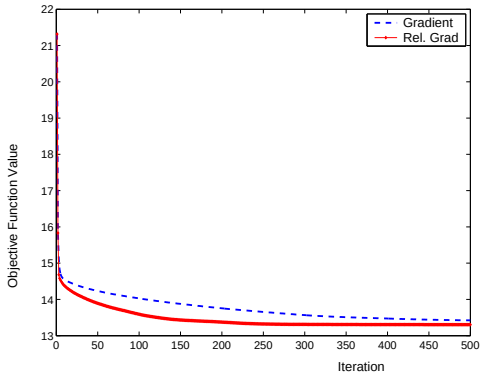
IV. Numerical Experiments

We used patches of natural images as input data $\boldsymbol{X}$ in our experiments. The data was obtained by taking 8×8 and 16×16 pixel image patches at random locations from monochrome photographs depicting wild-life scenes, respectively. The images (animals, meadows, forests, etc.) are available on the World Wide Web[2]. The image patches were then converted into vectors of length 64 and 256, then the mean gray-scale value of each image patch was subtracted. Principal component analysis was applied to reduce the dimension down to 40 and 160, constructing the data matrix $\boldsymbol{X} \in \mathbb{R}^{40\times N}$ and $\boldsymbol{X} \in \mathbb{R}^{160\times N}$, respectively. The subspaces dimension was chosen to be $\kappa = 2, 4$ and the number of invariant feature subspaces was set to $J = 20, 40$. The learning rate was determined by backtracking algorithm [6] for both the gradient ISA algorithm and the relative gradient ISA algorithm.

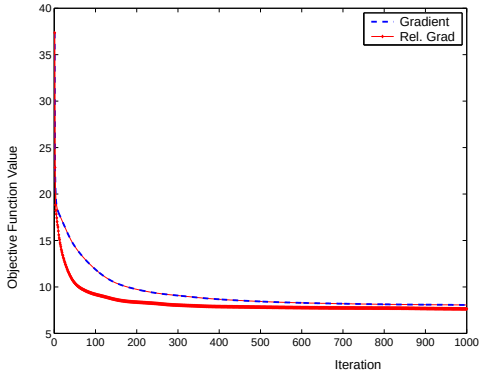
We also used face images with different poses for experiment. Each windowed subimage is normalized into a fixed size of 20×20 pixels. A set of 13,230 face images is used for the relative gradient ISA learning. We experimented with features size $\kappa = 4$ and number of subspaces $J = 20$ and compared the relative gradient method with the conventional gradient method.

Performance comparison is shown in Fig. 1, where the relative gradient ISA algorithm achieved faster convergence, compared to the gradient ISA algorithm. The converged objective value is same to each other method. Fig. 1 shows that the relative gradient method significantly outperforms in the number of iterations. Computational complexity per each iteration in relative gradient is slightly less than gradient method since it does not need the inverse operation.

[2]URL: `http://www.cis.hut.fi/projects/ica/data/images/`

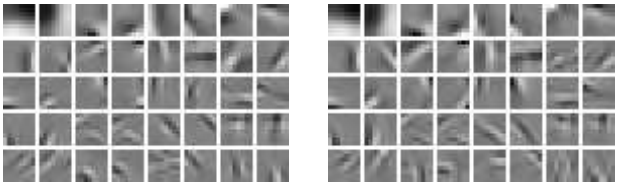


(a)



(b)

Fig. 1. Performance comparison between the gradient ISA algorithm and the relative gradient ISA algorithm for the case of: (a) 8×8 patches of natural scene images; (b) face images with different poses.



(a) (b)

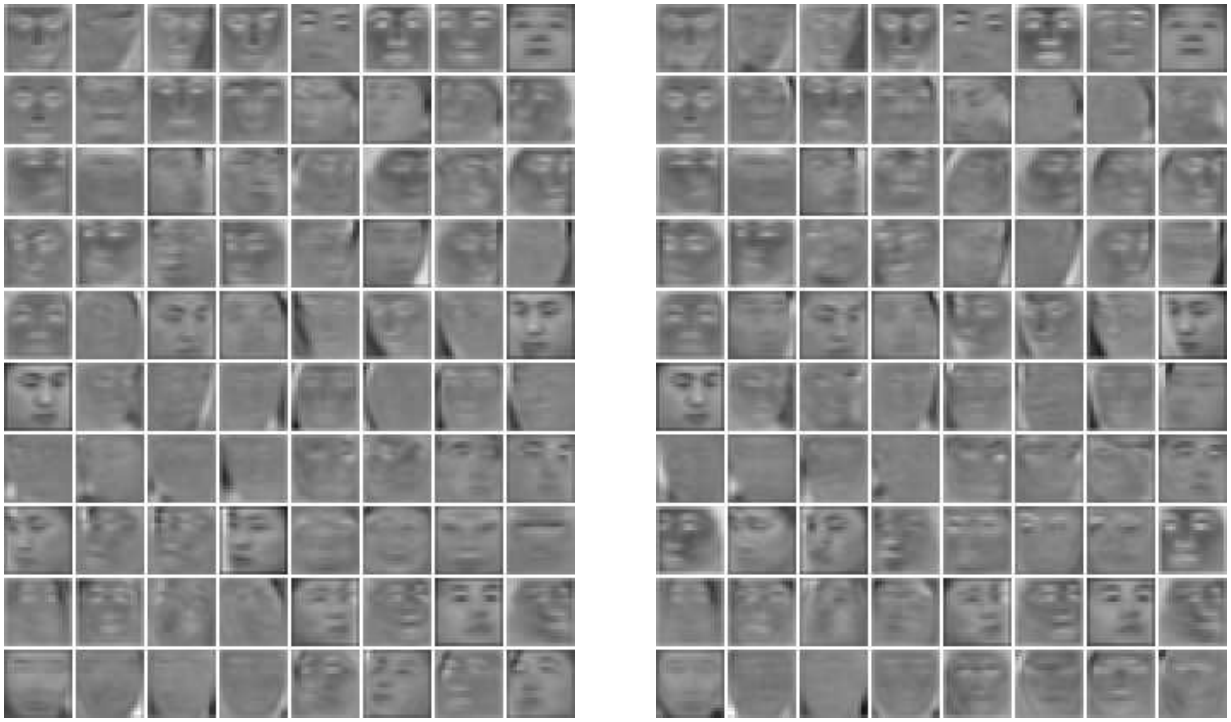
Fig. 2. Basis images computed by ISA for the case of 8×8 patches of natural scence images: (a) gradient ISA; (b) relative Gradient ISA.

The final results of algorithms are similar to each other as expected. In Fig. 2, we can see the 20 subspaces with 2 dimension. Also, Fig. 3 shows that there is no difference in performance between two results from gradient and relative gradient algorithm. Fig. 4 is the results of 16×16 image data which is a high dimensional data set.

These results confirm that our relative gradient algorithm is more faster than gradient algorithm with same performances.

V. Discussions

ISA is a generalization of ICA, which seeks a linear filter which maximizes the independence of the norms of the projection on linear subspaces. The original ISA algorithm in [8] adopted the gradient descent learning, hence it suffered



(a) (b)

Fig. 3. Basis images computed by ISA for the case of face images with different poses: (a) gradient ISA; (b) relative Gradient ISA.

from the slow convergence. We have revisited ISA, providing a compact form of its updating rule that enabled us to derive a relative gradient descent algorithm. The relative gradient ISA algorithm was derived in the frame work of the relative optimization. Empirical comparison between the relative gradient ISA algorithm and the gradient ISA algorithm, has shown that the former achieved faster convergence and lower error, compared to the latter. Currently we are working on faster ISA algorithm, employing the relative trust-region method that was recently proposed in [6] with successful application to ICA.

Acknowledgments: This work was supported by ITEP Brain Neuroinformatics Program, Korea MIC under ITRC support program supervised by the IITA (IITA-2005-C1090-0501-0018), and National Core Research Center for Systems Bio-Dynamics. Heeyoul Choi was also supported by KOSEF Grant funded by Korea government (No. D00115).

REFERENCES

[1] S. Amari, "Natural gradient works efficiently in learning," *Neural Computation*, vol. 10, no. 2, pp. 251–276, 1998.

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

[3] ——, "Multidimensional independent component analysis," in *Proc. IEEE Int'l Conf. Acoustics, Speech, and Signal Processing*, Seattle, WA, 1998.

[4] J. F. Cardoso and B. H. Laheld, "Equivariant adaptive source separation," *IEEE Trans. Signal Processing*, vol. 44, no. 12, pp. 3017–3030, Dec. 1996.

[5] M. Casey, "Separation of mixed audio sources by independent subspace analysis," Mitsubishi Electrical Research Laboratory, Tech. Rep. 31, 2001.

[6] 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.

[7] D. J. Field, "What is the goal of sensory coding?" *Neural Computation*, vol. 6, no. 4, pp. 559–601, 1994.

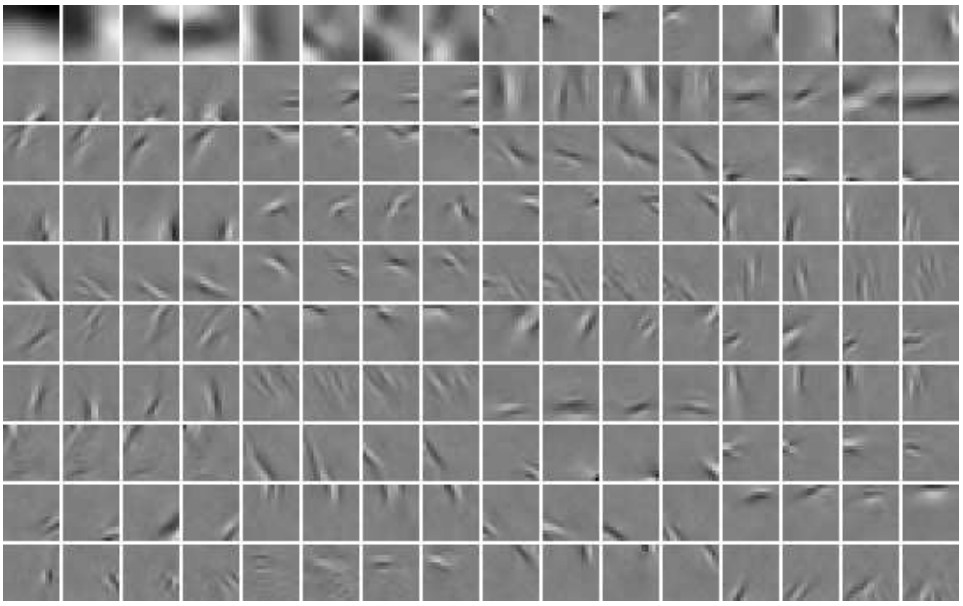
[8] A. Hyvärinen and P. Hoyer, "Emergence of phase- and shift-invariant features by decomposition of natural images into independent feature subspaces," *Neural Computation*, vol. 12, no. 7, pp. 1705–1720, 2000.

[9] H. Kim and S. Choi, "Independent subspaces of gene expression data," in *Proc. IASTED Int'l Conf. Artificial Intelligence and Applications*, Innsbruck, Austria, 2005.

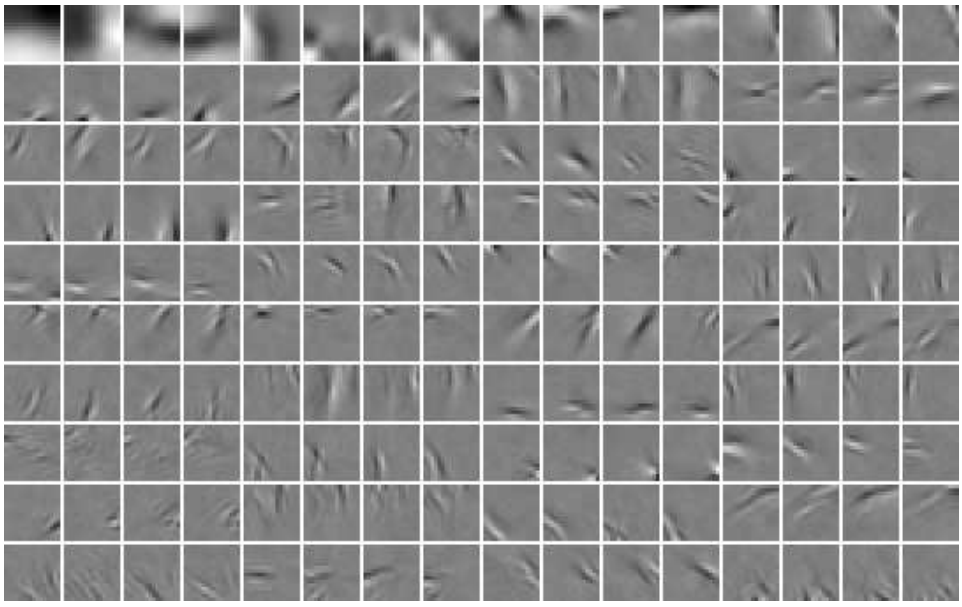
[10] H. Kim, S. Choi, and S. Y. Bang, "Membership scoring via independent feature subspace analysis for grouping co-expressed genes," in *Proc. Int'l Joint Conf. Neural Networks*, Portland, Oregon, 2003.

[11] S. Z. Li, X. Lv, and H. Zhang, "View-based clustering of object appearances based on independent subspace analysis," in *Proc. Int'l Conf. Computer Vision*, Vancouver, Canada, 2001, pp. 295–300.

[12] M. Zibulevsky, "Blind source separation with relative Newton method," in *Proc. ICA*, Nara, Japan, 2003, pp. 897–902.



(a) gradient ISA



(b) relative gradient ISA

Fig. 4. Basis images computed by ISA for the case of 16×16 patches of natural scene images.