Journal of Signal and Information Processing, **2012, 3, 275-285**
http://dx.doi.org/10.4236/jsip.2012.33037 Published Online August 2012 (http://www.SciRP.org/journal/jsip)

Rapid Algorithm for Independent Component Analysis

Ryota Yokote, Yasuo Matsuyama

Department of Computer Science and Engineering, Waseda University, Tokyo, Japan.
Email: rrryokote@wiz.cs.waseda.ac.jp, yasuo2@waseda.jp

Received March 30th, 2012; revised May 2nd, 2012; accepted May 10th, 2012

ABSTRACT

A class of rapid algorithms for independent component analysis (ICA) is presented. This method utilizes multi-step past information with respect to an existing fixed-point style for increasing the non-Gaussianity. This can be viewed as the addition of a variable-size momentum term. The use of past information comes from the idea of surrogate optimization. There is little additional cost for either software design or runtime execution when past information is included. The speed of the algorithm is evaluated on both simulated and real-world data. The real-world data includes color images and electroencephalograms (EEGs), which are an important source of data on human-computer interactions. From these experiments, it is found that the method we present here, the RapidICA, performs quickly, especially for the demixing of super-Gaussian signals.

Keywords: Independent Component Analysis; Speedup; Past Information; Momentum; Super-Gaussian; Negentropy

1. Introduction

Optimization of the amount of information leads to a rich class of learning algorithms for various types of signal processing. The learning phase is usually iterative, which is often the cause of their effective-but-slow nature. The speed of a learning algorithm is important, because this facilitates the application to real-world problems. Independent component analysis (ICA) [1] is a typical example of such an algorithm. Among the various ICA methods, the FastICA, which is a fixed-point algorithm [2], is the most popular one because it usually out-performs the fastest version of the gradient-style algorithms [3-5]. However, the need for ever faster ICAs has arisen ubiquitously [6,7].

Reflecting this necessity for speedup, this paper presents a class of rapid ICA algorithms. The core idea is the introduction of past information to the fixed-point ICA. This method inherits the idea for the speedup of the gradient-style algorithm using a surrogate function [3-5].

When a new class of learning algorithms is presented, it is necessary to check the following:

1) Is its merit on the performance enough?

2) Is it applicable to substantial real-world problems?

In this paper, we present the RapidICA algorithm and the results of testing it on both artificial and real-world data. As practical data, we adopted color images and electroencephalograms (EEGs) for pattern recognition. After extensive experiments, the effectiveness of the RapidICA over the existing fixed-point ICA can be fully recognized.

The organization of this paper is as follows. In Section 2, preliminaries to the ICA problem are given. Section 3 shows three types of fast versions, starting with the simplest one. Next, a summarized version is given, along with a strategy for stabilization and a measure of the convergence. Section 4 presents the results of experiments comparing the RapidICA with the FastICA for both artificial and real-world data. The practical data were color images and electroencephalograms (EEGs). In Section 5, concluding remarks are given.

2. Preliminaries to ICA

2.1. Formulation of ICA

In this section, we present our notation and the minimum necessary preliminaries for ICA. Given a measured random vector $\boldsymbol{x}$, the problem of ICA starts with the relationship of the superposition.

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

Here,

$$\boldsymbol{x} = \left[x_1, \cdots, x_n\right]^T \tag{2}$$

is zero mean. The matrix $\boldsymbol{A}$ is an $n \times n$ unknown non-singular matrix. The column vector

$$\boldsymbol{s} = \left[s_1, \cdots, s_n\right]^T \tag{3}$$

is also unknown. Its components are assumed to be in-

dependent of each other, and to be non-Gaussian except for one component. The necessity of these assumptions indicates the following:

1) For the consistency of the ICA problem, it is desirable that the x_i are far from a Gaussian distribution.

2) There are two classes that are non-Gaussian. One is super-Gaussian and the other is sub-Gaussian. But by the central limit theorem, the summation of only a few independent sub-Gaussian random variables easily becomes an almost Gaussian one. Therefore, real-world sub-Gaussian mixtures x_i for ICA applications are rare. On the other hand, super-Gaussian mixtures x_i are found in many signal processing sources. Thus, experiments on real-world data will be mainly on super-Gaussian mixtures.

Using the data production mechanism of (1)-(3), the problem of ICA is to estimate both $\boldsymbol{A}$ and $\boldsymbol{s}$ using only $\boldsymbol{x}$. This is, however, a semi-parametric problem where uncertainty remains. Therefore, we will be satisfied if the following $\boldsymbol{y}$ and $\boldsymbol{W}$ are successfully obtained.

$$\boldsymbol{y} = \left[y_1, \cdots, y_n\right]^T = \boldsymbol{W}\boldsymbol{x} = \left[w_1, \cdots, w_n\right]^T \boldsymbol{x} \tag{4}$$

Here, the random vector $\boldsymbol{y}$ is an estimated column vector with independent components. Note that the ordering of components of $\boldsymbol{y}$ is allowed to be permuted from that of $\boldsymbol{s}$. The amplitude of s_i is also uncertain in this generic ICA formulation. Therefore, $\boldsymbol{W}$ is an estimation of $\boldsymbol{\Lambda\Pi A}^{-1}$. Here, $\boldsymbol{\Lambda}$ is a nonsingular diagonal matrix, and $\boldsymbol{\Pi}$ is a permutation matrix.

2.2. Cost Functions for Independence

There are several cost functions that measure the independence of random variables. A popular target function is the minimization of the Kullback-Leibler divergence, or, equivalently, the minimum average mutual information. The methods we present in this paper are related to this information measure.

2.2.1. Minimum Average Mutual Information and Its Extension

As is well known from information theory [8], the average mutual information works as a directed distance between two probability densities of a Kullback-Leibler divergence. By choosing a probability density function (pdf) q to be an independent one, the cost function to be minimized is expressed as follows.

$$\begin{aligned} I\left(\wedge_{i=1}^n Y_i\right) &\equiv D\left(q \| p\right) \\ &= \int_Y \prod_{i=1}^n q_i\left(y_i\right) \log\left(\frac{\prod_{i=1}^n q_i\left(y_i\right)}{p\left(y_1, \cdots, y_n\right)}\right) \mathrm{d}\boldsymbol{y} \geq 0 \end{aligned} \tag{5}$$

This average mutual information becomes zero if and only if the y_i are independent of each other. Thus differentiation with respect to $\boldsymbol{W}$ leads to a gradient descent algorithm. This method can be generalized by using a convex divergence that includes the average mutual information of (5) as a special case.

The convex divergence is an information quantity expressed as follows [9].

$$\begin{aligned} D_f\left(p \| q\right) &\equiv \int_Y p\left(y_1, \cdots, y_n\right) f\left(\frac{\prod_{i=1}^n q_i\left(y_i\right)}{p\left(y_1, \cdots, y_n\right)}\right) \mathrm{d}\boldsymbol{y} \\ &= \int_Y \prod_{i=1}^n q_i\left(y_i\right) g\left(\frac{p\left(y_1, \cdots, y_n\right)}{\prod_{i=1}^n q_i\left(y_i\right)}\right) \mathrm{d}\boldsymbol{y} \\ &\equiv D_g\left(q \| p\right) \geq 0 \end{aligned} \tag{6}$$

Here, $f\left(r\right)$ is convex on $r \in \left(0, \infty\right)$, and $g\left(r\right) = rf\left(1/r\right)$ is a dual convex function. Note that the special case of

$$f\left(r\right) = \log\frac{1}{r} \tag{7}$$

is reduced to the average mutual information (5). The differentiation of $D_g\left(q \| p\right)$ with respect to $\boldsymbol{W}$ leads to a gradient descent update for the demixing matrix [5].

$$\boldsymbol{W}\left(t+1\right) = \boldsymbol{W}\left(t\right) + \tilde{\Delta}_g \boldsymbol{W}\left(t\right) \tag{8}$$

$$\begin{aligned} \tilde{\Delta}_g \boldsymbol{W}\left(t\right) &= \tilde{\Delta}_g \boldsymbol{W}\left(t\right) + \mu_c \tilde{\Delta}_g \boldsymbol{W}\left(t-\tau\right) \\ &= \rho(t)\Big[\left\{\boldsymbol{I} - \varphi\left(\boldsymbol{y}\left(t\right)\right)\boldsymbol{y}\left(t\right)^T\right\}\boldsymbol{W}\left(t\right) \\ &\quad + \mu_c\left\{\boldsymbol{I} - \varphi\left(\boldsymbol{y}\left(t-\tau\right)\right)\boldsymbol{y}\left(t-\tau\right)^T\right\}\boldsymbol{W}\left(t-\tau\right)\Big] \end{aligned} \tag{9}$$

Here, t is the iteration index for the update, and τ is a delay. The symbol $\tilde{\Delta}_g$ represents a natural gradient of g in Equation (6) multiplied by $c\boldsymbol{W}^T\boldsymbol{W}$, where c is a positive constant associated with the Fisher information matrix [10]. $\rho\left(t\right) \geq 0$ is a design parameter. $\boldsymbol{I}$ is the identity matrix. The matrix

$$\begin{aligned} \varphi\left(\boldsymbol{y}\right) &= \left[\varphi_1\left(y_1\right), \cdots, \varphi_n\left(y_n\right)\right]^T \\ &= -\left[\frac{q_1'\left(y_1\right)}{q_1\left(y_1\right)}, \cdots, \frac{q_n'\left(y_n\right)}{q_n\left(y_n\right)}\right] \end{aligned} \tag{10}$$

is unknown in practice since $q_i\left(y_i\right)$ is unknown in the formulation of the ICA. Therefore, in practical ICA execution, the function $\varphi_i\left(y_i\right)$ is assumed to be a nonlinear one, such as $\varphi_i\left(y_i\right) = y_i^3$ or $\varphi_i\left(y_i\right) = \tanh y_i$.

An important point in this preliminary section appears in the last line of Equation (9). If $\mu_c = 0$, then the second term disappears, and it becomes the method of minimum average mutual information using Equation (5). On the other hand, the method of minimum convex divergence generates the momentum term by $\mu_c > 0$. This was the fastest ICA by the gradient descent method [3]. But, it could not beat the fixed-point algorithm of [2],

which is based on the non-Gaussianity maximization.

2.2.2. Non-Gaussianity Maximization Based upon Approximated Negentropy

Non-Gaussianity can be measured by the Kullback-Leibler divergence (5) from a pdf $p(y_1,\cdots,y_n)$ to a Gaussian pdf $p_{\text{Gauss}}(y_1,\cdots,y_n)$, which is called negentropy.

$$\begin{aligned} J(\boldsymbol{y}) &\equiv D\left(p(y_1,\cdots,y_n)\middle\| p_{\text{Gauss}}(y_1,\cdots,y_n)\right) \\ &= H(\boldsymbol{y}_{\text{Gauss}}) - H(\boldsymbol{y}) \end{aligned} \tag{11}$$

A criterion for the ICA of Equation (4) can be set to the maximization of the negentropy (11). It is not possible to maximize the negentropy directly since $p(y_1,\cdots,y_n)$ is unknown. Therefore, in the FastICA [2], the negentropy maximization is approximated. For this approximation, we pose an assumption that the x_i are pre-whitened to be uncorrelated with each other.

$$E\left[\boldsymbol{x}\boldsymbol{x}^T\right] = \boldsymbol{I} \tag{12}$$

Note that this assumption was not needed in the method of Section 2.2.1.

Then, using a predetermined contrast function $G(y)$, the maximization of (11) is approximated by the following optimization with constraint [2].

$$\begin{aligned} &\text{Maximize} && \textstyle\sum_{i=1}^{n}\left\{E\left[G\left(\boldsymbol{w}_i^T\boldsymbol{x}\right)\right] - E\left[G(\nu)\right]\right\}^2 \\ &\text{wrt} && \boldsymbol{w}_i, i=1,\cdots,n \\ &\text{under} && E\left[\left(\boldsymbol{w}_j^T\boldsymbol{x}\right)\left(\boldsymbol{w}_k^T\boldsymbol{x}\right)\right] = \delta_{jk} \end{aligned} \tag{13}$$

Here, ν is a zero-mean, unit-variance Gaussian random variable. By virtue of the above drastic approximation, the FastICA with a couple of variants [2] is obtained.

$$\begin{aligned} &\boldsymbol{w}_i^+ = E\left[\boldsymbol{x}_i\partial G\left(\boldsymbol{w}_i^T\boldsymbol{x}_i\right)\right] - E\left[\partial^2 G\left(\boldsymbol{w}_i^T\boldsymbol{x}_i\right)\right]\boldsymbol{w}_i, \\ &\text{and then } \boldsymbol{w}_i^* = \boldsymbol{w}_i^+ / \left\|\boldsymbol{w}_i^+\right\| \text{ for all } i \end{aligned} \tag{14}$$

Here, ∂G and $\partial^2 G$ are the first and second derivatives of a contract function $G(y)$. Examples of $G(y)$ are y^4, log cosh y, and $\exp\left(-y^2/2\right)$. The update method (14) is a fixed-point algorithm and, because of its speed, it is the current de facto standard for ICA.

2.2.3. Mediation between Gradient Descent and Fixed-Point Algorithms for Speedup

So far, two preliminary sections have been presented. The first one, Section 2.2.2, presents a direct minimization of the convex divergence between the current and independent pdfs. The last term of the last line of Equation (9) is the important one.

1) This term, which depends on the past information, is the core of the speedup [10].

2) The case of $\mu_c = 0$ corresponds to the optimization of the Kullback-Leibler divergence or its subsidiary, the entropy.

The second preliminary, the FastICA derived from the entropy difference, updates Equation (14) in a fixed-point style. Based upon the theoretical foundation of the derivation of Equation (9), one may conjecture that the acceleration methods for Equation (14) would be obtainable by using the strategy of Equation (9). The rest of this paper will show that this conjecture is answered in the affirmative. We comment here in advance that the process we use to obtain the accelerated version of the fixed-point method is not naïve. It requires more artifice than the method of Section 2.2.1 since orthonormalization steps need to be interleaved. But we will see that the result will be easy to code as software.

3. Rapid Methods for ICA

In this section, several steps towards the final rapid version are presented. From this point on, the description will be more software oriented.

3.1. Observed Signal Preprocessing and Additive Update Expression

Instead of the measured vector $\boldsymbol{x}$, its whitened version $\boldsymbol{z}$, obtained in the following way, is used as a target source to be demixed.

$$\boldsymbol{z} = \boldsymbol{V}\boldsymbol{x} \tag{15}$$

$$\boldsymbol{V} = \boldsymbol{D}^{-1/2}\boldsymbol{E}^T = \boldsymbol{D}^{-1/2}\left[\boldsymbol{e}_1,\cdots,\boldsymbol{e}_m\right]^T \tag{16}$$

Here, $\boldsymbol{D}$ is the diagonal matrix constructed from the largest $m \le n$ eigenvalues of the covariance matrix $E\left[\boldsymbol{x}\boldsymbol{x}^T\right]$, and $\boldsymbol{e}_i$, $i=1,\cdots,m$, are the corresponding eigen-vectors. Then, the update and orthonormalization steps of the fixed-point method are expressed as follows:

Update step: $\boldsymbol{W}$ is updated by the following computation.

$$\boldsymbol{W} \leftarrow E\left[\partial G(\boldsymbol{y})\boldsymbol{z}^T\right] - \text{diag}\left(E\left[\partial^2 G(y_i)\right]\right)\boldsymbol{W} \tag{17}$$

Orthonormalization step: $\boldsymbol{W}$ is orthonormalized by using the eigenvalue decomposition.

$$\boldsymbol{W} \leftarrow \left(\boldsymbol{W}\boldsymbol{W}^T\right)^{-1/2}\boldsymbol{W} \tag{18}$$

Since we will use the speedup method of Section 2.2.1, it is necessary to rewrite (17) as an additive form.

$$\boldsymbol{W} \leftarrow \boldsymbol{W} - \text{diag}\left(1/E\left[\partial^2 G(y_i)\right]\right)E\left[\partial G(\boldsymbol{y})\boldsymbol{z}^T\right] \tag{19}$$

Equation (19) can be derived by dividing the right-hand side of Equation (14) by $-E\left[\partial^2 G\left(\boldsymbol{w}_i^T\boldsymbol{x}\right)\right]$. This is allowed since the orthonormalization step for $\boldsymbol{W}$ follows.

Then the basic method, which is the FastICA, is described as follows:

[Basic Method]

Step 1: $\boldsymbol{y} \leftarrow \boldsymbol{W}\boldsymbol{z}$

Step 2: $\boldsymbol{W} \leftarrow \boldsymbol{W} - \mathrm{diag}\left(1/E\left[\partial^2 G(y_i)\right]\right) E\left[\partial G(\boldsymbol{y})\boldsymbol{z}^T\right]$

Step 3: $\boldsymbol{W} \leftarrow \left(\boldsymbol{W}\boldsymbol{W}^T\right)^{-1/2}\boldsymbol{W}$

Beginning in the next subsection, a series of speedup versions is presented. Findings in each version are supported by extensive experiments. Except for the destination algorithm, intermediate experimental results are omitted in order to save space.

3.2. High-Speed Version I

For the basic method, which is the FastICA, it has been pointed out that the orthonormalization of Step 3 sometimes hinders the learning of non-Gaussianity by causing a slow convergence. Therefore, we start by considering an algorithm that simply increases the update amount.

[Method 0: Naïve version]

Step 1: $\boldsymbol{y} \leftarrow \boldsymbol{W}\boldsymbol{z}$

Step 2: $\Delta\boldsymbol{W} \leftarrow -\mathrm{diag}\left(1/E\left[\partial^2 G(y_i)\right]\right) E\left[\partial G(\boldsymbol{y})\boldsymbol{z}^T\right]$

Step 3: $\boldsymbol{W} \leftarrow \boldsymbol{W} + (1+\eta)\Delta\boldsymbol{W}$

Step 4: $\boldsymbol{W} \leftarrow \left(\boldsymbol{W}\boldsymbol{W}^T\right)^{-1/2}\boldsymbol{W}$

This naïve method is too simple, in that convergence will frequently fail for unadjusted $\eta > 0$, even for small figures. Therefore, we make this increase usable by inserting an orthonormalization step.

[Method 1: Opening version]

Step 1: $\boldsymbol{y} \leftarrow \boldsymbol{W}\boldsymbol{z}$

Step 2: $\Delta\boldsymbol{W} \leftarrow -\mathrm{diag}\left(1/E\left[\partial^2 G(y_i)\right]\right) E\left[\partial G(\boldsymbol{y})\boldsymbol{z}^T\right]$

Step 3: $\boldsymbol{W} \leftarrow \boldsymbol{W} + \Delta\boldsymbol{W}$

Step 4: $\boldsymbol{W} \leftarrow \left(\boldsymbol{W}\boldsymbol{W}^T\right)^{-1/2}\boldsymbol{W}$

Step 5: $\boldsymbol{W} \leftarrow \boldsymbol{W} + \eta\Delta\boldsymbol{W}$

Step 6: $\boldsymbol{W} \leftarrow \left(\boldsymbol{W}\boldsymbol{W}^T\right)^{-1/2}\boldsymbol{W}$

One might think that this version simply computes the same update twice, but the following observations lead us to better versions:

1) Step 2 requires much more computational power than do the others.

2) Without Step 4, the increase by $\eta > 0$ could cause oscillations.

3.3. High-Speed Version II

By integrating Method 1 and the usage of the past information described in Section 2.2.1, the following method is derived.

[Method 2: Second-order version]

Step 1: $\boldsymbol{y} \leftarrow \boldsymbol{W}\boldsymbol{z}$

Step 2: $\boldsymbol{W}_{\mathrm{old}} \leftarrow \boldsymbol{W}$

Step 3: $\boldsymbol{W} \leftarrow \boldsymbol{W} - \mathrm{diag}\left(1/E\left[\partial^2 G(y_i)\right]\right) E\left[\partial G(\boldsymbol{y})\boldsymbol{z}^T\right]$

Step 4: $\boldsymbol{W} \leftarrow \left(\boldsymbol{W}\boldsymbol{W}^T\right)^{-1/2}\boldsymbol{W}$

Step 5: $\Delta\boldsymbol{W} \leftarrow \boldsymbol{W} - \boldsymbol{W}_{\mathrm{old}}$

Step 6: $\boldsymbol{W} \leftarrow \boldsymbol{W} + \eta\Delta\boldsymbol{W}$

Step 7: $\boldsymbol{W} \leftarrow \left(\boldsymbol{W}\boldsymbol{W}^T\right)^{-1/2}\boldsymbol{W}$

The key point of this method is that there are two types of increments. The first type is in Step 3 and the second one is in Step 6. This method has the following properties:

1) Each increment uses different time information. This is equivalent to the usage of a higher-order strategy.

2) The computed increments $\Delta\boldsymbol{W}$ of Step 5 gradually converge to the null matrix $\boldsymbol{O}$. Therefore, this method is more stable than Method 1.

3) The computational load of Step 5 is negligible compared to that of Step 3. Therefore, the reduction of iterations will be directly reflected in the runtime speedup.

3.4. High-Speed Version III

In High-speed Version II, the adjusting parameter $\eta > 0$ of the increment was a scalar, such as 0.1. The next step is to find reasonable values of $\eta_i > 0$ that depend on the row indices *i* of $\boldsymbol{W}$.

[Method 3: Variable step-size version]

Step 1: $\boldsymbol{y} \leftarrow \boldsymbol{W}\boldsymbol{z}$

Step 2: $\boldsymbol{W}_{\mathrm{old}} \leftarrow \boldsymbol{W}$

Step 3: $\boldsymbol{W} \leftarrow \boldsymbol{W} - \mathrm{diag}\left(1/E\left[\partial^2 G(y_i)\right]\right) E\left[\partial G(\boldsymbol{y})\boldsymbol{z}^T\right]$

Step 4: $\boldsymbol{W} \leftarrow \left(\boldsymbol{W}\boldsymbol{W}^T\right)^{-1/2}\boldsymbol{W}$

Step 5: $\Delta\boldsymbol{W} \leftarrow \boldsymbol{W} - \boldsymbol{W}_{\mathrm{old}}$

Step 6: $\boldsymbol{W} \leftarrow \boldsymbol{W} + \mathrm{diag}(\eta_i)\Delta\boldsymbol{W}$

Step 7: $\boldsymbol{W} \leftarrow \left(\boldsymbol{W}\boldsymbol{W}^T\right)^{-1/2}\boldsymbol{W}$

In this method, the main issue is how to find an effective $\eta_i > 0$. Here, we adopt the idea of jointly using cur-

rent and past information.

$$\eta_i = \max\left(\left\langle \frac{\Delta \boldsymbol{w}_i}{|\Delta \boldsymbol{w}_i|}, \frac{\Delta \boldsymbol{w}_{i,\text{old}}}{|\Delta \boldsymbol{w}_{i,\text{old}}|} \right\rangle, 0\right) \quad (20)$$

Equation (20) has the property that a large change in the direction of $\Delta \boldsymbol{w}_i$ from $\Delta \boldsymbol{w}_{i,\text{old}}$ causes the value of $\eta_i > 0$ to be small. This property is useful in helping to avoid oscillations during the intermediate steps. However, the magnitude of $\Delta \boldsymbol{w}_i$ is not taken into account. Also, near to the convergence maximum, the value of $\eta_i > 0$ might become numerically unstable since division by zero could occur. We now get to the final form.

$$\eta_i = \beta \frac{\max\left(\left\langle \Delta \boldsymbol{w}_i, \Delta \boldsymbol{w}_{i,\text{old}} \right\rangle, 0\right)}{\max\left(\left\|\Delta \boldsymbol{w}_i\right\|^2, \left\|\Delta \boldsymbol{w}_{i,\text{old}}\right\|^2\right) + \gamma^2} \quad (21)$$

Here, $\beta > 0$ is a constant, but it can be omitted (*i.e.*, $\beta = 1$). The constant $\gamma > 0$ serves the purpose of providing numerical stability by preventing division by zero ($\gamma^2 = 10^{-6}$ for 32-bit machines). Equation (21) has the following properties:

1) When $\Delta \boldsymbol{w}_i$ and $\Delta \boldsymbol{w}_{i,\text{old}}$ are close together, the value of $\eta_i > 0$ is large since this direction needs to be emphasized. On the other hand, if the directions of $\Delta \boldsymbol{w}_i$ and $\Delta \boldsymbol{w}_{i,\text{old}}$ are considerably different (the extreme case is anti-parallel), the value of $\eta_i > 0$ is close to zero.

2) Because of γ^2, a small $\Delta \boldsymbol{w}_i$ generates a small $\eta_i > 0$.

Properties 1) and 2) give us hope that considerable speedup can be obtained with very little increase in computational complexity. There is one more property on which to comment.

3) It should be noted that every ICA algorithm has the possibility of non-convergence. The FastICA [2] is not an exception [6]. There are many sources that cause this, and one can easily generate such a case by using data that is non-Gaussian but almost Gaussian. In such a case, a little bit of a slowdown greatly helps the stability.

$$\boldsymbol{W} \leftarrow \boldsymbol{W} - \alpha \operatorname{diag}\left(1 \Big/ E\left[\partial^2 G(y_i)\right]\right) E\left[\partial G(\boldsymbol{y}) \boldsymbol{z}^T\right] \quad (22)$$

Here, $\alpha \in (0,1)$ is a slowdown constant. The case of $\alpha = 1.0$ is the FastICA, which may not converge on some data. In our preliminary experiments with Method 3, $0.98 \le \alpha < 1.0$ worked well in achieving convergence without too much slowdown.

3.5. Total Algorithm: The RapidICA

Up to this point, we have presented steps that increase the speedup and stabilization. Integrating all these steps is expected to produce an even faster ICA than the existing ones. The following procedure summarizes the RapidICA.

[RapidICA]

Step 1: $\boldsymbol{y} \leftarrow \boldsymbol{W}\boldsymbol{z}$

Step 2: $\boldsymbol{W}_{\text{old}} \leftarrow \boldsymbol{W}$

Step 3: $\boldsymbol{W} \leftarrow \boldsymbol{W} + \Delta \boldsymbol{W}_1$

where

$$\Delta \boldsymbol{W}_1 = -\alpha \operatorname{diag}\left(1 \Big/ E\left[\partial^2 G(y_i)\right]\right) E\left[\partial G(\boldsymbol{y}) \boldsymbol{z}^T\right]$$

Step 4: $\boldsymbol{W} \leftarrow \left(\boldsymbol{W}\boldsymbol{W}^T\right)^{-1/2} \boldsymbol{W}$

Step 5: $\Delta \boldsymbol{W}_{2,\text{old}} \leftarrow \Delta \boldsymbol{W}_2$

Step 6: $\Delta \boldsymbol{W}_2 \leftarrow \boldsymbol{W} - \boldsymbol{W}_{\text{old}}$

Step 7: $\boldsymbol{W} \leftarrow \boldsymbol{W} + \operatorname{diag}(\eta_i) \Delta \boldsymbol{W}_2$

with η_i of Equation (21).

Step 8: $\boldsymbol{W} \leftarrow \left(\boldsymbol{W}\boldsymbol{W}^T\right)^{-1/2} \boldsymbol{W}$

In the appendix, we will give the source code of the RapidICA in the programming language R.

It is important to emphasize the following:

1) The algorithm of the RapidICA utilizes past information over three steps in a single cycle. This is illustrated in **Figure 1**. Let the iteration index be denoted by *t*. Given the demixing matrix $\boldsymbol{W}(t)$, an estimation of the independent component $\boldsymbol{y}(t)$ is computed. This step requires the most computation. This $\boldsymbol{y}(t)$ is then used to compute $\Delta \boldsymbol{W}_1(t)$. By orthonormalization, $\boldsymbol{W}(t)$ proceeds to $\boldsymbol{W}(t+1)$. Then, this $\boldsymbol{W}(t+1)$ together with $\boldsymbol{W}(t)$ is used in the computation of $\Delta \boldsymbol{W}_2(t)$. Finally, $\boldsymbol{W}(t+2)$ is computed by using $\boldsymbol{W}(t+1)$, $\Delta \boldsymbol{W}_2(t+1)$, and $\Delta \boldsymbol{W}_2(t-1)$, which was saved as $\Delta \boldsymbol{W}_{\text{old}}$. Thus, the RapidICA uses the update information from up to three prior steps. Note that the RapidICA of Method 2 (constant step coefficient) lacks the arrow from $\Delta \boldsymbol{W}_2(t-1)$ to $\boldsymbol{W}(t+2)$. Therefore, the RapidICA of Method 2 makes use of past information of no more than two prior steps.

2) From **Figure 1**, one may be reminded of the use of a momentum variable with a dynamical system to turn it into a learning system [11]. The use of a momentum term appears in various iterative methods. The FastICA is not

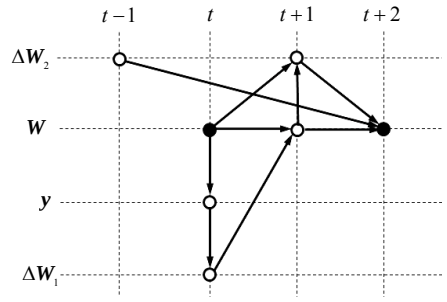

Figure 1. RapidICA diagram for information flow.

an exception, since acceleration methods for the natural gradient ICA have already been presented [3-5] based upon the idea of surrogate optimization of the likelihood ratio [10]. In fact, the RapidICA presented in this paper is an embodiment of the momentum term of the α-ICA applied to the fixed-point ICA expressed by the additive form (19). Besides the RapidICA of this paper, further variants are possible, some of which might show performances comparable to that of the RapidICA. We also note here that the surrogate optimization used in the alpha-HMM [12] originates in the same place [10] as does the RapidICA.

3) Computationally, the heaviest parts are Steps 1 and 3 of the High-speed Versions II and III. Let n be the number of independent components and T be the number of samples. Then the order of the computation is $O(n^2T)$. The computation of $\mathrm{diag}(\eta_i)$, however, is only $O(n^2)$. Since $T \gg n$ holds for most cases, the computational overhead for realizing the RapidICA from the FastICA remains small. Therefore a reduction in the number of iterations leads directly to a speedup in CPU time.

In the next section, the speedup effects of the RapidICA will be measured by the number of iterations and the CPU time.

4. Comparison of Convergence

We evaluated the performance of the RapidICA with respect to simulated and real-world data. Because of the semi-parametric nature of the ICA formulation, methods to measure convergence are different between simulated and real-world data.

4.1. Error Measures for Simulated and Real-World Data

The evaluation of simulated data is important since the mixing matrix $\boldsymbol{A}$ is specified in advance. Note that this matrix is assumed to be unknown in the ICA setting.

For simulated data, the first measure of error counts how close $\boldsymbol{W}^{-1}$ is to $\boldsymbol{A}$. There is also the permutation uncertainty. Therefore, the error measure is based on the matrix

$$\boldsymbol{P} = \boldsymbol{WVA} \tag{23}$$

Here, $\boldsymbol{V}$ is a transformation matrix of Equation (16). The error measure is then defined as follows [13].

$$\begin{aligned}\text{error} = &\frac{1}{n^2}\sum_{i=1}^{n}\left[\left(\sum_{j=1}^{n}\frac{|p_{ij}|}{\max_k |p_{ik}|}\right)-1\right] \\ &+\frac{1}{n^2}\sum_{j=1}^{n}\left[\left(\sum_{i=1}^{n}\frac{|p_{ij}|}{\max_k |p_{kj}|}\right)-1\right]\end{aligned} \tag{24}$$

Here, p_{ij} is the element of the matrix $\boldsymbol{P}$. If $\boldsymbol{V} = \boldsymbol{I}$, *i.e.*, the source signal is regarded as a preprocessed one, then the matrix $\boldsymbol{P}$ becomes a permutation matrix.

The second error measure reflects the independence by using $G(y)$.

$$J(y) = \frac{1}{n}\sum_{i=1}^{n} E\left[G(y_i)\right] \tag{25}$$

This is applicable to both simulated and real-world data.

The third error criterion measures the convergence of the demixing matrix. Let $\boldsymbol{w}_i$ and $\boldsymbol{w}_{i,\text{old}}$ be the row vectors of $\boldsymbol{W}$ and $\boldsymbol{W}_{\text{old}}$, respectively. Here, $\boldsymbol{W}$ is the currently orthonormalized version. Thus the convergence measure is defined as follows.

$$\text{conv} = 1-(1/n)\sum_{i=1}^{n}\left|\langle \boldsymbol{w}_i, \boldsymbol{w}_{i,\text{old}}\rangle\right| \tag{26}$$

This is the most important convergence measure, since it can be used as a stop criterion for the iteration.

$$\text{conv} < \varepsilon \tag{27}$$

A typical value for ε is 10^{-6} to 10^{-4}.

4.2. Experiments on Simulated Data

First, we generated a super-Gaussian source from Gaussian random numbers, as follows:

Step 1: A mixture matrix $\boldsymbol{A}$ was selected.

Step 2: We drew $N(0, 1)$ Gaussian pseudo-random numbers.

Step 3: For each random number r, $\mathrm{pow}(r,4)$ was applied to obtain an $\boldsymbol{s}$.

Step 4: A total of 2000 such $\boldsymbol{s}$ were generated.

Step 5: The time series of $\boldsymbol{s}$ was renormalized to have zero mean and unit variance.

Step 6: A total of $n = 20$ of such super-Gaussian sources were generated.

Step 7: The mixture signal $\boldsymbol{x}$ was generated by Equation (1).

Figure 2 illustrates a super-Gaussian component generated by the above method. For this class of subsources, simulations were performed by using the following contrast functions.

$$G(y) = \log\cosh y \tag{28}$$

$$\partial G(y) = \tanh y \tag{29}$$

Figure 3 illustrates the resulting speedup comparison of three methods: the FastICA, the RapidICA with a constant $\mathrm{diag}(\eta_i)$, and the RapidICA with the $\mathrm{diag}(\eta_i)$ of Equation (21). The last method of the three types is simply called the RapidICA. In **Figure 3**, the horizontal axis indicates the number of iterations, and the vertical axis indicates the error of Equation (24) using a logarithmic scale. As can be observed from this figure, the RapidICA outperformed the FastICA.

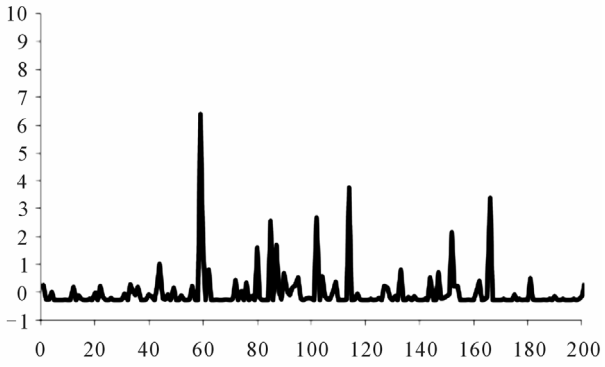

Figure 2. A super-Gaussian signal for simulations.

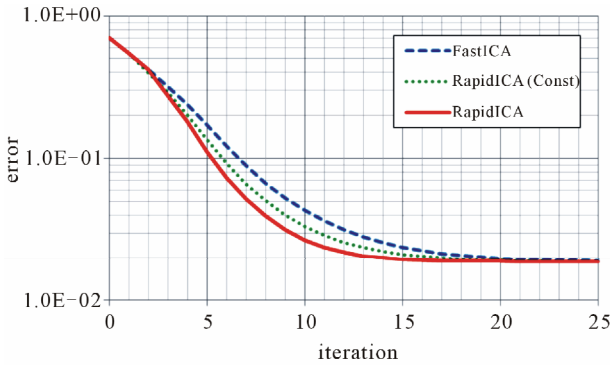

Figure 3. Convergence speed check by error measure.

Figure 4 illustrates the trend of iterations versus the log cosh measure of Equation (25), which was used to check if the converged result had enough independence. This trend was compatible with that of the convergence of **Figure 3**.

In **Figures 3** and **4**, both types of RapidICAs outperformed the FastICA. Also, the RapidICA, with $\mathrm{diag}(\eta_i)$ of Equation (21), outperformed the version with η_i = const. This means that the diagonal matrix $\mathrm{diag}(\eta_i)$ appropriately changed its elements. Since there are n = 20 elements, we computed their average so that a general trend could be more easily seen. **Figure 5** illustrates the course of the average:

$$\bar{\eta} = \frac{1}{n}\sum_{i=1}^{n}\eta_i \tag{30}$$

From **Figure 5**, we observed the following properties:

1) During the first two iterations, η_i was set to zero since there was no past information.

2) Once the adjustment of η_i started, iterations used this information to adjust the step size and the direction of $\Delta \boldsymbol{W}_2$. The average $\bar{\eta}$ became close to zero as the iteration proceeded. In **Figure 5**, this phenomenon could be observed after the 17th iteration. This coincided with the convergence of **Figures 3** and **4**. Note that at the 17th iteration, the FastICA and the RapidICA with a constant set of η_i were still in the course of learning updates.

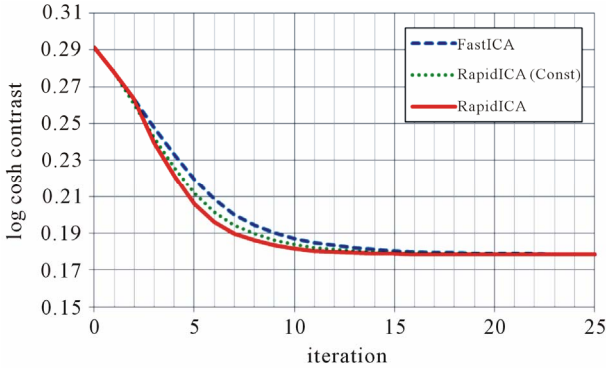

Figure 4. Convergence speed check using the contrast function.

Figure 5. Trend of $\bar{\eta}$ in the course of the convergence.

4.3. ICA Comparison on Real-World Data: Obtaining Bases for Images

Digital color images are popular real-world data for ICA benchmarking. First, we checked to see the trend of the convergence by using a single typical natural image. We first present this, and then follow with the average performance on 200 images.

4.3.1. Convergence Comparison on a Typical Natural Image

We applied the ICAs to obtain image bases. In this case, the true mixing matrix $\boldsymbol{A}$ was unknown.

Step 1: Image patches of $\boldsymbol{x}$ were collected directly from RGB source images. First, we drew a collection of 8 × 8 size patches. The total number was 15,000.

Step 2: Each patch $\boldsymbol{x}$ was regarded as a 192-dimensional vector (192 = 8 × 8 × 3).

Step 3: By the whitening of Equation (15), the dimension was reduced to 64.

In this experiment, the contrast function of Equations (28) and (29) was used.

Figure 6 compares the convergence of three ICA methods: the FastICA, the RapidICA with fixed η_i, and

the RapidICA with adjusting η_i. The performance measure was Equation (25). Here, the horizontal axis is the CPU time. It can be observed that the RapidICA again outperformed the FastICA.

It is important for ICA users to understand that there are difficulties with using log cosh as a performance measure. This is because its value cannot be known in advance because of the semi-parametric formulation of the ICA problem. We can understand this by comparing the vertical axis levels of **Figures 4** and **6**. For this reason, the convergence criterion (26) is the most practical one. **Figure 7** compares this convergence criterion for three ICA methods; the FastICA, the RapidICA with fixed η_i, and the RapidICA with adjusting η_i. In this figure, the horizontal axis is CPU time, and the vertical axis is the convergence measure of Equation (26). The convergence speed of the RapidICA obviously outperformed that of the FastICA. Moreover, the RapidICA achieved a better convergence region (on the vertical axis) than that which the FastICA could attain. One can easily see this in **Figure 7** by drawing a horizontal line at $\text{conv} = 1.0\text{E}-05$.

By examining **Figure 7**, one realizes that the iteration-truncated matrices for the demixing $\boldsymbol{W}$ are different

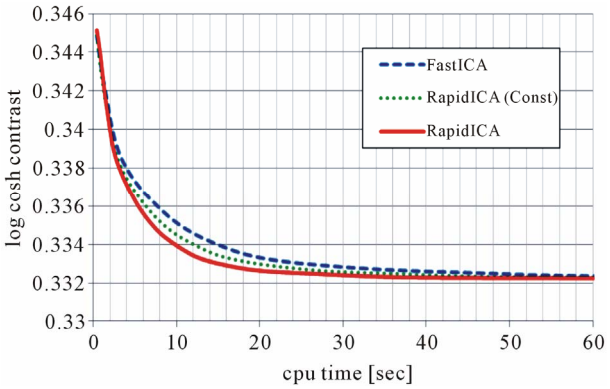

Figure 6. Convergence check by the log cosh contrast function.

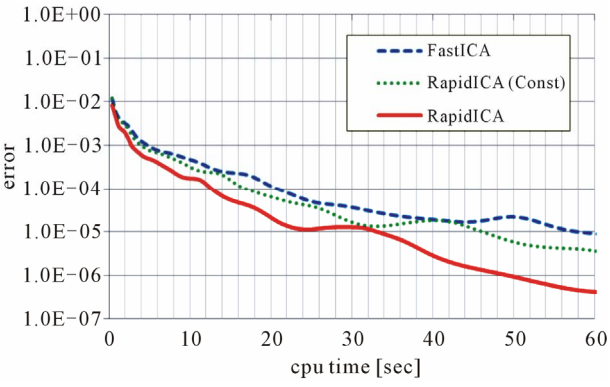

Figure 7. Convergence check by the basis direction similarity.

among the three ICA methods, but they are expected to be similar to each other except for the ordering permutation. **Figures 8** and **9** are sets of bases obtained by the RapidICA and the FastICA at the 300th iteration. These are raster-scan visualizations of the column vectors of the matrix

$$\boldsymbol{V}^{*}\boldsymbol{W}^{T} = \hat{\boldsymbol{A}} = \left[\hat{\boldsymbol{a}}_1, \cdots, \hat{\boldsymbol{a}}_n\right] \tag{31}$$

Here, $\boldsymbol{V}^{*}$ is a Moor-Penrose generalized inverse of $\boldsymbol{V}$. $\hat{\boldsymbol{A}}$ is an estimation of the unknown mixing matrix $\boldsymbol{A}$.

By considering the permutation, we compare two sets of bases

$$\hat{\boldsymbol{A}}^{\text{RapidICA}} = \left[\hat{\boldsymbol{a}}_1^{\text{RapidICA}}, \cdots, \hat{\boldsymbol{a}}_n^{\text{RapidICA}}\right] \tag{32}$$

and

$$\hat{\boldsymbol{A}}^{\text{FastICA}} = \left[\hat{\boldsymbol{a}}_1^{\text{FastICA}}, \cdots, \hat{\boldsymbol{a}}_n^{\text{FastICA}}\right] \tag{33}$$

by the computation procedure described in **Figure 10**.

If $S \geq 0.8$, we can regard the two basis sets as having a similar role. By the computation of S in **Figure 10** for the basis sets of **Figures 8** and **9**, we found that $S = 0.95$. Therefore, the basis sets of **Figure 8** (RapidICA) and **Figure 9** (FastICA) essentially play the same role. In other words, both the RapidICA and the FastICA converged to the sound and similar local optimum.

4.3.2. Convergence Comparison by 200 Images: Super-Gaussian Data Set

We prepared a data set of 200 color images as follows.

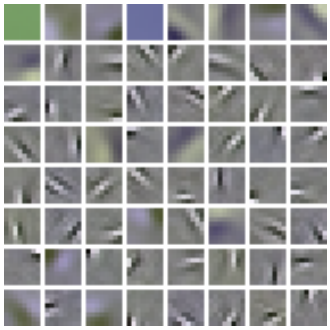

Figure 8. ICA bases by the RapidICA.

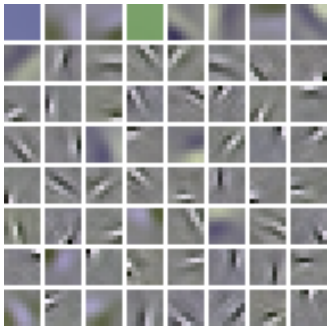

Figure 9. ICA bases by the FastICA.

do $k = 1, n$

$c_k = \max_{i,j} \left| \left\langle \hat{\boldsymbol{a}}_1^{\text{RapidICA}}, \hat{\boldsymbol{a}}_1^{FastdICA} \right\rangle \right|$

$i_k, j_k = \arg\max_{i,j} \left| \left\langle \hat{\boldsymbol{a}}_1^{\text{RapidICA}}, \hat{\boldsymbol{a}}_1^{FastdICA} \right\rangle \right|$

$\hat{\boldsymbol{a}}_{i_k}^{\text{RapidICA}} \leftarrow 0$

$\hat{\boldsymbol{a}}_{j_k}^{FastICA} \leftarrow 0$

end do

$S = 1/n \sum_k c_k$

Figure 10. Procedure to compute a basis similarity.

As will be seen, the image data are mostly super-Gaussian.

Step 1: Each color image was resized to 150 × 112 pixels.

Step 2: From each image, a set of 16,800 patches of 8 × 8 pixels was drawn by overlapped sampling.

Step 3: Each set of patches was normalized to have zero mean.

Step 4: By the whitening method of Equations (15) and (16), the patch vector's dimension of 192 = 8 × 8 × 3 was reduced to 64.

For the color image data set prepared as described above, we conducted an experiment to compare the RapidICA (Method 3) and the FastICA. The adopted contrast function was log cosh. **Figure 11** is a histogram of the iteration speedup by the RapidICA. Here, the horizontal axis is the speedup ratio, where the truncation criterion of (27) is $\varepsilon = 10^{-4}$.

$$\text{speedup ratio} = \frac{\text{necessary iterations for FastICA}}{\text{necesary iterations for RapidICA}} \quad (34)$$

The vertical axis is the number of occurrences. Obviously, the RapidICA outperformed the FastICA in all 200 cases. For the total of 200 ICA trials, the RapidICA required 5349 iterations with a CPU time of 1837 seconds by a conventional PC. On the other hand, the FastICA required 9848 iterations with a CPU time of 3031 seconds. Therefore, the average iteration speedup ratio was 1.67, and the CPU speedup ratio was 1.65. The most frequent speedup ratio for the iterations was 1.80, and its CPU speedup ratio was 1.78. In all of the 200 cases, the RapidICA outperformed the FastICA.

Accompanied by the 200 image experiments, we used the measure of –log cosh to check the non-Gaussianity. This is illustrated in **Figure 12**. The horizontal axis is the non-Gaussianity, and the vertical axis is the number of occurrences. The heavy black arrow is the position of the Gaussian distribution:

$$J(y) \approx -0.375 \quad (35)$$

As can be understood from **Figure 12**, the color image data are super-Gaussian.

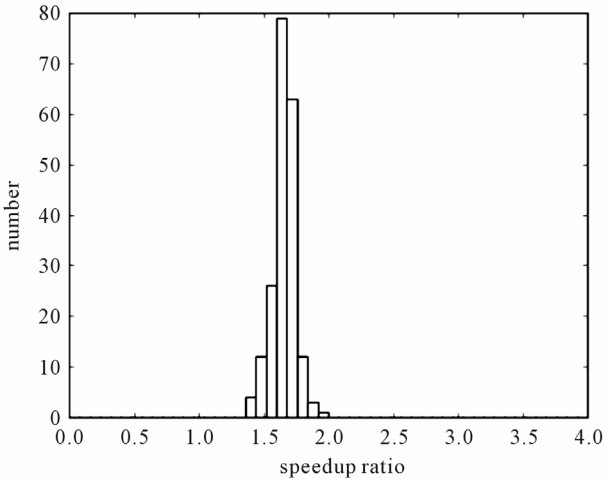

Figure 11. Iteration speedup ratio of RapidICA over FastICA on image data.

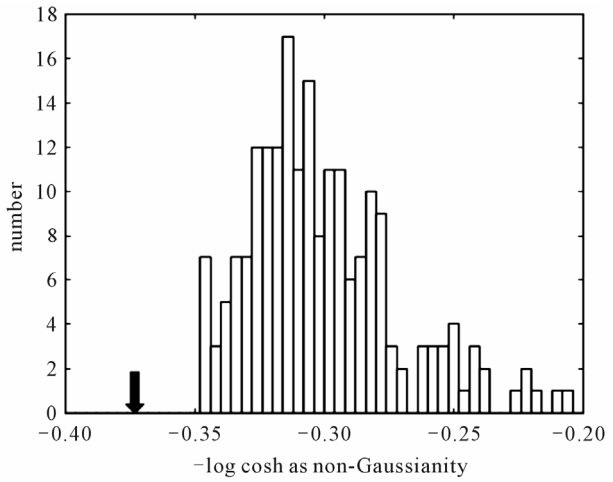

Figure 12. Non-Gaussianity measure of –log cosh on image data.

4.3.3. Convergence Comparison by 200 EEG Data: Almost Gaussian Data Set

We prepared a data set of 200 EEG vector time series. As will be seen, EEG data are weakly super-Gaussian. That is, they are almost Gaussian, which can cause difficulties in ICA decomposition, as was stated in the ICA formulation of Section 2.1. The EEG data sets were prepared and preprocessed in the following way:

Step 1: 200 EEG time series were drawn at random from the data of [14]. Each time series had 59 channels.

Step 2: Each EEG time series was divided into chunks that were 8 seconds in length. This generated 8000 samples (1 KHz sampling).

Step 3: Each EEG time series was normalized to have zero mean.

Step 4: By the whitening method of Equations (15) and (16), the EEG channel dimension of 59 was reduced to 32.

Figure 13 illustrates the histogram of the speedup ratio of Equation (34), where the convergence truncation is $\varepsilon = 10^{-4}$. For the 200 data sets, the Rapid ICA again outperformed the FastICA in 181 cases. For the total of 200 ICA trials, the RapidICA required a total of 7305 iterations with a CPU time of 613 seconds by a conventional PC. On the other hand, the FastICA required 8256 iterations with a CPU time of 656 seconds. Therefore, the iteration speedup was 1.13, and the CPU speedup ratio was 1.07. The most frequent speedup ratio on the iteration was 1.30, and its CPU speedup ratio was 1.23. Upon obtaining this result, we checked the degree of the non-Gaussianity.

Figure 14 illustrates the non-Gaussianity by using the measure of –log cosh. The position of the heavy black arrow is that of the Gaussian distribution of Equation (35). As can be observed, sub-Gaussian EEGs were mixed in the test data set. By comparing **Figures 12** and **14**, one finds that the EEG data drawn from [14] are very weak super-Gaussian. Some data sets contained multiple Gaussian signals. Such cases violate the assumption of the ICA formulation of Section 2.1. This indicates that appropriate preprocessing is desirable for EEGs before the ICA is applied. We will consider this point in the next section.

5. Concluding Remarks

In this paper, a class of speedy ICA algorithms was presented. This method, called the RapidICA, outperformed the FastICA. Since the increase in computation is very light, the reduction of iterations directly realizes the CPU speedup. The speedup is drastic if the sources are super-Gaussian, which is the case when the source data are natural images. Since ICA bases have the roles of expressing texture information for data compression and measuring the similarity between different images, they can be used in this system for the similar-image retrieval [7]. In such a case, the speedup of the RapidICA is quite good.

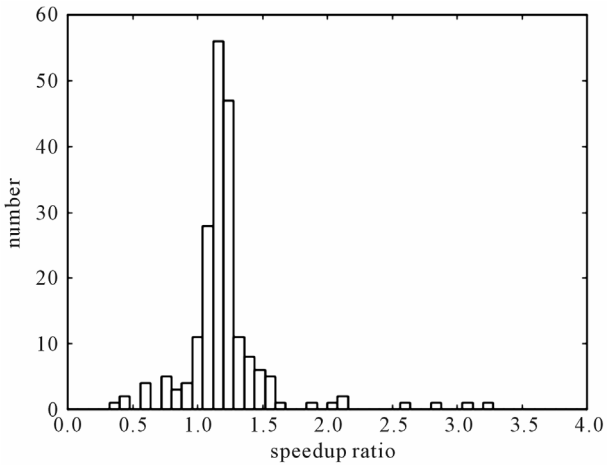

Figure 13. Iteration speedup ratio of RapidICA over FastICA on EEG data.

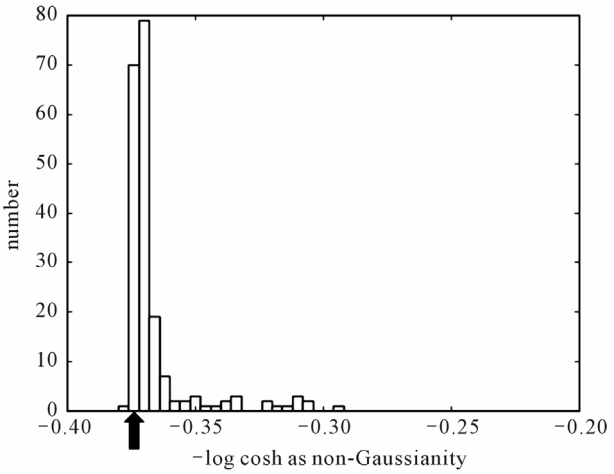

Figure 14. Non-Gaussianity measure of –log cosh on EEG data with a magnified horizontal axis.

For the case of signals that are nearly Gaussian, such as the EEGs, the RapidICA again outperformed the FastICA in terms of speed. But the margin of improvement is less than that in the case of images. This is because cases of multiple Gaussian sources need to be excluded from the ICA formulation, including both the RapidICA and the FastICA. For such cases, if the ICA is applied, it is necessary to transform raw EEG data into good super-Gaussian data. This is possible on the EEGs since the purpose of using these data in only for the detection of event changes in source signals. This approach will be presented in future papers.

6. Acknowledgements

This study was supported by the Ambient SoC Global COE Program of Waseda University from MEXT Japan. Waseda University Grant for Special Research Projects No. 2010B and the Grant-in-Aid for Scientific Research No. 22656088 are also acknowledged.

REFERENCES

[1] P. Common and C. Jutten, Eds., "Handbook of Blind Source Separation: Independent Component Analysis and Applications," Academic Press, Oxford, 2010.

[2] A Hyvärinen, "Fast and Robust Fixed-Point Algorithms for Independent Component Analysis," *IEEE Transactions on Neural Networks*, Vol. 10, No. 3, 1999, pp. 626-634.

[3] Y. Matsuyama, N. Katsumata, Y. Suzuki and S. Imahara, "The α-ICA Algorithm," *Proceedings of* 2*nd International Workshop on ICA and BSS*, Helsinki, 2000, pp. 297-302.

[4] Y. Matsuyama, N. Katsumata and S. Imahara, "Convex

Divergence as a Surrogate Function for Independence: The f-Divergence ICA," *Proceedings of 4th International Workshop on ICA and BSS*, Nara, 2003, pp. 173-178.

[5] Y. Matsuyama, N. Katsumata and R. Kawamura, "Independent Component Analysis Minimizing Convex Divergence," *Lecture Notes in Computer Science*, No. 2714, 2003, pp. 27-34.

[6] V. Zarzoro, P. Common and M. Kallel, "How Fast Is FastICA?" 14*th European Signal Processing Conference* (*EUSIPCO*), 4-8 September 2006, pp. 4-8.

[7] N. Katsumata and Y. Matsuyama, "Database Retrieval from Similar Images Using ICA and PCA Bases," *Engineering Applications of Artificial Intelligence*, Vol. 18, No. 6, 2005, pp. 705-717. doi:10.1016/j.engappai.2005.01.002

[8] T. Cover and J. Thomas, "Elements of Information Theory," John Wiley and Sons, New York, 1991. doi:10.1002/0471200611

[9] I. Csiszár, "Information-Type Measures of Difference of Probability Distributions and Indirect Observations," *Studia Scientiarum Mathematicarum Hungarica*, Vol. 2, 1967, pp. 299-318.

[10] Y. Matsuyama, "The α-EM Algorithm: Surrogate Likelihood Maximization Using α-Logarithmic Information Measures," *IEEE Transactions on Information Theory*, Vol. 49, No. 3, 2003, pp. 692-706. doi:10.1109/TIT.2002.808105

[11] C. M. Bishop, "Neural Networks for Pattern Recognition," Oxford University Press, Oxford, 1995.

[12] Y. Matsuyama, "Hidden Markov Model Estimation Based on Alpha-EM Algorithm: Discrete and Continuous Alpha-HMMs," *Proceedings of International Joint Conference on Neural Networks*, San Jose, 7 July-5 August 2011, pp. 808-816.

[13] H. H. Yang and S. Amari, "Adaptive Online Learning Algorithm for Blind Separation: Maximum Entropy and Minimum Mutual Information," *Neural Computation*, Vol. 9, No. 7, 1997, pp. 1457-1482. doi:10.1162/neco.1997.9.7.1457

[14] B. Blankertz, G. Dornhege, M. Lrauledat, K. R. Muller and G. Curio, "The Non-Invasive Brain-Computer Interface: Fast Acquisition of Effective Performance in Untrained Subjects," *Nueroimage*, Vol. 37, No. 2, 2007, pp. 539-550. doi:10.1016/j.neuroimage.2007.01.051

Appendix: R Code of RapidICA

```
# settings
iter.max <- 100
eps <- 1e-5
p.alpha <- 1.0
p.beta <- 1.0
p.gamma <- 1e-6

# initialization
W <- diag(1, d, d)
dW1 <- matrix(0, d, d)
dW2 <- matrix(0, d, d)
eta <- matrix(0, d, d)

# iterations
for (p in 1:iter.max) {

  # (learning non-gaussianity)
  Wold <- W
  y <- W %*% z
  dW1 <-
    -p.alpha *
    diag(1 / rowSums(G2(y))) %*%
    (G1(y) %*% t(z))
  W <- W + dW1
  W <- orth(W)

  # (convergence test)
  conv <-
    1 - sum(abs(diag(W %*% t(Wold)))) / d
  if (conv < eps) break

  # (acceleration steps)
  dW2old <- dW2
  dW2 <- W - Wold
  for (i in 1:d) {
    eta.i.num <-
      max(t(dW2[i,]) %*% dW2old[i,], 0)
    eta.i.den <-
      max(t(dW2[i,]) %*%
      dW2[i,], t(dW2old[i,]) %*%
      dW2old[i,])
    eta[i,i] <- p.beta *
      eta.i.num / (eta.i.den + p.gamma)
  }
  W <- W + eta * dW2
  W <- orth(W)
}
```

