RESEARCH ARTICLE

Unique estimation in EEG analysis by the ordering ICA

Yoshitatsu Matsuda[1]*, **Kazunori Yamaguchi**[2]

1 Department of Science and Technology, Seikei University, Tokyo, Japan, **2** Department of General Systems Studies, The University of Tokyo, Tokyo, Japan

* matsuda@st.seikei.ac.jp

OPEN ACCESS

Citation: Matsuda Y, Yamaguchi K (2022) Unique estimation in EEG analysis by the ordering ICA. PLoS ONE 17(10): e0276680. https://doi.org/10.1371/journal.pone.0276680

Editor: Qichun Zhang, University of Bradford, UNITED KINGDOM

Received: March 19, 2022

Accepted: October 11, 2022

Published: October 24, 2022

Data Availability Statement: All resulting data and script files are available from the kaggle database (https://www.kaggle.com/datasets/yoshitatsumatsuda/eeg-analysis-data-for-plos-one-manuscript-2022).

Funding: Yoshitatsu Matsuda. Grant Number JP21K12036. Japan Society for the Promotion of Science KAKENHI. https://kaken.nii.ac.jp/grant/KAKENHI-PROJECT-21K12036/. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Abstract

Independent Component Analysis (ICA) is a method for solving blind source separation problems. Because ICA only needs weak assumptions to estimate the unknown sources from only the observed signals, it is suitable for Electroencephalography (EEG) analysis. A serious disadvantage of the traditional ICA algorithms is that their results often fluctuate and do not converge to the unique and globally optimal solution at each run. It is because there are many local optima and permutation ambiguities. We have recently proposed a new ICA algorithm named the ordering ICA, a simple extension of Fast ICA. The ordering ICA is theoretically guaranteed to extract the independent components in the unique order and avoids the local optima in practice. This paper investigated the usefulness of the ordering ICA in EEG analysis. Experiments showed that the ordering ICA could give unique solutions for the signals with large non-Gaussianity, and the ease of parallelization could reduce computation time.

1 Introduction

Electroencephalography (EEG) is a method of monitoring the electrical activity in the brain [1]. As EEG is a non-invasive and low-cost method without imposing a heavy burden on subjects, it is widely used for measuring brain activity. On the other hand, EEG has only a limited number of noisy channels, and its observed results fluctuate according to the subjects and trials. Therefore, EEG needs advanced and robust techniques for extracting the essential components from such noisy and limited signals. Independent component analysis (ICA) is a widely used technique for analyzing EEG signals [2]. One of the most significant advantages of ICA is its versatility. ICA needs only a simple assumption on the model generating observed signals (the linear mixture of non-Gaussian sources). Therefore, ICA is suitable for EEG analysis where an unknown mixture model changes according to the subjects and trials. An ICA-based analysis tool named EEGLAB has been widely used [3]. One disadvantage of ICA is that it often generates a different solution at each run for the same dataset, unlike principal component analysis (PCA). It reduces the robustness of the analysis. This indeterminateness is caused by the non-linearity of the objective functions of ICA, which makes ICA have many local

Competing interests: The authors have declared that no competing interests exist.

minima. As non-linearity is essential to ICA, it is difficult for the previous ICA methods to solve this problem.

Recently, we proposed a new ICA method named the ordering ICA [4]. The ordering ICA is theoretically guaranteed to find the unique solution of ICA. Though it was much slower than the previous ICA methods, the paper suggested that parallel processing could accelerate the ordering ICA. This paper constructs a parallel implementation on multi-core CPUs of the ordering ICA and applies it to EEG analysis. The experimental results show that our implementation can extract the essential components more robustly than the previous methods with modest computation time.

This paper is organized as follows. In Section 2, ICA and the ordering ICA are briefly explained. In Section 3, the parallel implementation of the ordering ICA is detailed. Section 4 shows the experimental results of our implementation in EEG analysis. Lastly, we conclude this paper in Section 5.

2 Background

2.1 Independent component analysis and EEGLAB

Generally, ICA is used for extracting unknown sources from observed signals in an unsupervised manner [5, 6]. ICA assumes the linear mixture model $\boldsymbol{X} = \boldsymbol{A}\,\boldsymbol{S}$, where $\boldsymbol{X} = (x_{im})$ and $\boldsymbol{S} = (s_{im})$ ($N \times M$ matrices) are the observed signals and the unknown sources, respectively. $\boldsymbol{A} = (a_{ij})$ (an $N \times N$ matrix) is the unknown mixing matrix. N and M are the dimension of signals and the sample size, respectively. ICA extracts the unknown components in $\boldsymbol{A}$ so that the sources in $\boldsymbol{S}$ are statistically independent as much as possible. ICA is solved as an optimization problem, where the objective function is based on some higher-order statistics such as kurtosis. The objective function is derived from an assumption that each source is given independently according to a non-Gaussian distribution. It has been well known that ICA is quite useful for analyzing EEG signals from brain activity. EEGLAB is the most widely used tool implemented in MATLAB [3].

There are many methods of ICA according to the choice of the objective function and the optimization algorithm. The three representative ICA methods are employed in EEGLAB: the extended InfoMax [7] (referred to as InfoMax), Fast ICA [8], and JADE [9]. Many other methods are variations of the above three methods. The third method, JADE, is not applicable to high-dimensional signals. As the EEG channel is generally more than 60, JADE can not extract the essential components from EEG signals. Therefore, we focus on InfoMax and Fast ICA here. InfoMax employs a stochastic gradient algorithm for optimizing one of two hyperbolic tangent based objective functions depending on the currently estimated source distribution. Though InfoMax can find essential components robustly, the convergence of InfoMax is slow. Fast ICA employs Newton-Raphson method in the optimization of kurtosis-based objective functions. The advantage of Fast ICA is the speed of its convergence. The disadvantage is that the succeeding components tend to fluctuate more largely. One reason is that the usual approach of Fast ICA (called the deflation approach) estimates the components one by one under the orthonormality constraint. The errors accumulated by the successive estimation of components cause the fluctuations. On the other hand, such accumulated errors do not occur in InfoMax because InfoMax estimates all the components simultaneously. InfoMax has been preferred to Fast ICA in EEG analysis because of its robustness.

2.2 Ordering ICA

The ordering ICA is our recently-proposed method [4], which is a variation of the deflation approach of Fast ICA. In summary, the ordering ICA finds the globally optimal component at each deflation step by repeating the Newton-Raphson method with many different initializations and selecting the best solution from the multiple candidates. Here, we explain the ordering ICA in brief. See [4] for the details.

First, we employed a novel objective function in the ordering ICA. Let $\boldsymbol{w}_i = (w_{i1}, \ldots, w_{iN})$ be the i-th row of the separating matrix $\boldsymbol{W}$, which is expected to be the inverse of the mixing matrix $\boldsymbol{A}$. $\boldsymbol{Y} = (y_{im}) = \boldsymbol{W}\boldsymbol{X}$ denotes the estimated sources. The objective function of the i-th component in the ordering ICA is given as follows:

$$\Phi_i = \Upsilon(\sum_m y_{im}^4/M - 3) \tag{1}$$

where

$$\Upsilon(\alpha_i) = \alpha_i - 2\log(\alpha_i/2 + 1). \tag{2}$$

Note that $\alpha_i = \sum_m y_{im}^4/M - 3$ is the estimated kurtosis of the i-th row of $\boldsymbol{Y}$ if the row is normalized. Now, the following theorem holds:

Theorem 1 *We assume that* $\boldsymbol{S}$ *consists of normalized independent sources and does not include any uniform Bernoulli variable. We also assume that* $\boldsymbol{X}$ *is given by an invertible linear ICA model* $\boldsymbol{X} = \boldsymbol{AS}$ *in the real domain. Then, all the non-Gaussian sources are extracted in descending order of* $\Upsilon(\kappa_i)$ *if each* Φ_i *is globally maximized subject to* $\sum_m y_{im} = 0$ *and* $\sum_m y_{im} y_{jm}/M = \delta_{ij}$ *for* $j \leq i$ *(the Gram-Schmidt orthonormalization), where* κ_i *is the true kurtosis of the i-th row of* $\boldsymbol{S}$.

In other words, by globally maximizing Φ_i with respect to $\boldsymbol{w}_i$ one by one for each i in the Gram-Schmidt orthonormalization, the true separating matrix $\boldsymbol{W}$ (and the true mixing matrix $\boldsymbol{A} = \boldsymbol{W}^{-1}$) can be estimated uniquely if the assumptions are satisfied. κ_i can be estimated by α_i in practice.

Next, the algorithm for maximizing Φ_i in the Gram-Schmidt orthonormalization is described. It is easily proved from the convexity of $\Upsilon(\alpha_i)$ that the global maximum of $\Phi_i = \Upsilon(\alpha_i)$ is one of local maxima and minima of α_i. Therefore, we can utilize the deflation approach of the Fast ICA algorithm on the kurtosis [8]. As it finds a local minimum or a local maximum in each run by the Newton-Raphson method, we can find the global maximum of Φ_i without fail if the starting point of each run is sampled sufficiently densely from the entire space of $\boldsymbol{w}_i$. Actually, by using Fast ICA to generate many candidates with different random initializations and selecting the best result with the highest Φ_i, the global maximum is expected to be found in most cases. This algorithm estimates $\boldsymbol{W}$ from given $\boldsymbol{X}$ and is called "the ordering ICA" (Algorithm 1). Though the original ordering ICA [4] can estimate the number of independent components adaptively, we fix the number of independent components as N in this paper.

Algorithm 1 Ordering ICA

```
Require: $\boldsymbol{X}$: whitened observed signals. $L$, $K$, $\epsilon$: hyper-parameters.
Ensure: $\boldsymbol{W}$: separating matrix.
 1: Initialize $\boldsymbol{W}$ to the empty matrix
 2: $i \leftarrow 1$
 3: while $i \leq N$ do
 4:   if $i > 1$ then
 5:     $\boldsymbol{E} \leftarrow \boldsymbol{W}^T\boldsymbol{W}$
 6:   else
 7:     $\mathbf{E} \leftarrow 0$
 8:   else if
 9:   Initialize randomly $L$ different candidates $\boldsymbol{w}_i^l$
10:   for all $l \in \{1, \ldots, L\}$ do
11:     $\hat{\boldsymbol{w}}_{il}$, $\boldsymbol{Y} \leftarrow$ FastICA ($\boldsymbol{w}_i^l$, $\boldsymbol{X}$, $\boldsymbol{E}$)
12:     $\alpha_{il} \leftarrow \sum_m y_m^4/M - 3$
```

```
13:   end for
14:   $p \leftarrow \mathrm{argmax}_l\ \Upsilon(\alpha_{il})$
15:   Concatenate $\hat{\boldsymbol{w}}_{ip}^{\top}$ to $\boldsymbol{W}$ as the last row
16:   $i \leftarrow i + 1$
17: end while
18: function FastICA($\boldsymbol{w}$, $\boldsymbol{X}$, $\boldsymbol{E}$)
19: $\boldsymbol{w} \leftarrow \boldsymbol{w} - \boldsymbol{E}\,\boldsymbol{w}$
20: $\boldsymbol{w} \leftarrow \boldsymbol{w}/\sqrt{\boldsymbol{w}^{\top}\boldsymbol{w}}$
21: $t \leftarrow 0$
22:   repeat
23:   $\boldsymbol{W}_{\mathrm{prev}} \leftarrow \boldsymbol{w}$
24:   $\boldsymbol{y} \leftarrow \boldsymbol{w}^{\top}\tilde{\boldsymbol{X}}$
25:   $\boldsymbol{w} \leftarrow \tilde{\boldsymbol{X}}(\boldsymbol{y} \circ \boldsymbol{y} \circ \boldsymbol{y})^{\top}/M - 3\boldsymbol{w}$
26:   $\boldsymbol{w} \leftarrow \boldsymbol{w} - \boldsymbol{E}\,\boldsymbol{w}$
27:   $\boldsymbol{w} \leftarrow \boldsymbol{w}/\sqrt{\boldsymbol{w}^{\top}\boldsymbol{w}}$
28:   $t \leftarrow t + 1$
29: untill $t < K$ and $(||\boldsymbol{w} + \boldsymbol{w}_{\mathrm{prev}}|| > \epsilon$ or $||\boldsymbol{w} - \boldsymbol{w}_{\mathrm{prev}}|| > \epsilon)$
30: $\boldsymbol{y} \leftarrow \boldsymbol{w}^{\top}\boldsymbol{X}$
31: return $\boldsymbol{w}$, $\boldsymbol{Y}$
32: end function
```

In the ordering ICA, many candidates (the number is denoted by L) improve the robustness of the results. Because the total computation time is linear to L, the efficiency advantage of Fast ICA is lost if a naive implementation is employed.

3 Implementation

Here, we describe the parallel implementation of the ordering ICA. The time-consuming part of ordering ICA is multiple Fast ICA runs with different initializations (the lines 9-13 of Algorithm 1). Because the runs are independent, we can execute the runs in parallel. As the widely used tool EEGLAB was implemented on MATLAB, we implemented this using Parallel Computing Toolbox of MATLAB. We refer to this implementation as the parallel-ordering ICA. See the released code (ParallelOrderingICA.m) for the details at https://github.com/yoshitatsumatsuda/orderingICA/blob/master/ParallelOrderingICA.m.

Notably, this implementation uses a simple convergence condition in the FastICA function. If we run the FastICA function just once, we need to estimate $\boldsymbol{w}$ with a different initialization if it fails to converge. The parallel-ordering ICA can skip such a reestimation unless all FastICA runs fail to converge.

The hyper-parameter L (the number of parallel candidates) is set to be R times as large as the number of available cores in the given system. The implemented code is given by `L = feature('numcores') * R;` in Parallel Computing Toolbox. R is the overhead rate. The default value of R is set to 1. By using larger R, we can increase the number of candidates L at the cost of computational overhead. If the convergence threshold ϵ is sufficiently small, a slight difference does not have significant effects on the results. So, we just set ϵ to 10^{-6}. The maximum number K of iterations in each FastICA function is set to 30. K is relatively small because the convergence failures of some FastICA runs are no problems.

4 Experiments on EEG analysis

To verify the usefulness of the proposed method, we used several public EEG datasets and investigated the results from the viewpoint of robustness, the computation time, and the relation between the degree of non-Gaussianity and the success rate of finding a unique solution.

4.1 Datasets

5SUBJECTS consists of the 10 EEG signals syn{02,05,07,08,10}-s{253,254} downloaded at https://sccn.ucsd.edu/eeglab/download/STUDY5subjects.zip. They were observed in a semantic task with five subjects under two conditions. Each signal has 61 channels, namely, $N = 61$. The sample size M depends on the dataset and is about 160,000-190,000.

STERN consists of the 39 EEG datasets S{01-13}-{Ignore,Memorize,Probe} downloaded at https://sccn.ucsd.edu/eeglab/download/STUDYstern.zip. They were observed in Sternberg working memory tasks with 13 subjects under three conditions. Each signal has 69-71 channels, namely, $N = 69$, $N = 70$, or $N = 71$. The sample size M is about 70,000-100,000 under the Ignore condition (I, for short), about 120,000-180,000 under the Memorize one (M), and about 25,000-35,000 under the Probe one (P).

TUH is a dataset of randomly-selected 98 signals from the TUH EEG Corpus downloaded at https://isip.piconepress.com/projects/tuh_eeg/ after registration. The corpus consists of more than 30,000 signals under various conditions, and the number of channels (N) is relatively small (about 30). 100 signals were selected randomly from the corpus (v.1.1.0), and two unreadable signals were removed. In the selected 98 signals, N is 23-33 and M is about 2,000-1,700,000.

4.2 Investigation of the robustness and the computation time

To verify the robustness and efficiency of the parallel-ordering ICA, we investigated the averaged fluctuations of the solutions over the different runs and the averaged computation time in 5SUBJECTS and STERN. MATLAB R2022a with Parallel Computing Toolbox was employed. The implementation of the parallel-ordering ICA (called Ordering ICA) in Section 3 is applied directly to each EEG signal. For comparison, we applied the extended InfoMax of EEGLAB 2021.1 (called InfoMax) at https://github.com/sccn/eeglab and the Fast ICA using the kurtosis (FastICA 2.5) at https://research.ics.aalto.fi/ica/fastica/ to the same signals. We did not apply JADE because N was too large to employ JADE. We conducted experiments on a 32-core server (dual CPU with 16 cores) with 256GB RAM, where each core is an Intel Xeon 2.1GHz processor.

The fluctuation is measured as follows. Each algorithm (Ordering ICA, InfoMax, and Fast ICA) was applied to each EEG signal from different initializations over 10 runs. The number of runs is denoted by $T = 10$. To remove the permutation ambiguities, $\boldsymbol{w}_i$'s were ordered by $\Upsilon(\alpha_i)$, a degree of non-Gaussianity, in all the solutions. Note that the number of $\boldsymbol{w}_i$'s is often less than the number of channels N in Fast ICA. It is because the Fast ICA algorithm often fails to converge if the non-Gaussianity of the signal is low. Now, $\boldsymbol{w}_i$ in the p-th run is denoted by $\boldsymbol{w}_i^p$ ($p = 1, \ldots, 10$). To estimate the fluctuations of the results for the different runs, we use the divergence $\delta(i, p, q)$ between $\boldsymbol{w}_i^p$ and $\boldsymbol{w}_i^q$, defined as

$$\delta(i, p, q) = 1 - \left| \frac{\boldsymbol{w}_i^p \cdot \boldsymbol{w}_i^q}{\|\boldsymbol{w}_i^p\| \|\boldsymbol{w}_i^q\|} \right|. \tag{3}$$

Since the positive or negative sign of $\boldsymbol{w}_i$ is arbitrary in ICA, the absolute value of the cosine similarity is employed. $\delta(i, p, q)$ takes the minimum 0 if and only if $\boldsymbol{w}_i^p$ and $\boldsymbol{w}_i^q$ have the same or opposite directions. When $\boldsymbol{w}_i^p$ is orthogonal to $\boldsymbol{w}_i^q$, $\delta(i, p, q)$ takes the maximum 1. If $\boldsymbol{w}_i^p$ (or $\boldsymbol{w}_i^q$) does not exist due to non-convergence in Fast ICA, $\delta(i, p, q)$ was set to the maximum 1.

Consequently, the fluctuation of $\boldsymbol{w}_i$ over all the runs is evaluated as the average of $\delta(i, p, q)$, defined as

$$\bar{\delta}(i) = \frac{\sum_{p=1}^{T} \sum_{q>p}^{T} \delta(i, p, q)}{T(T-1)/2}. \tag{4}$$

First, the experimental results on 5SUBJECTS are shown. The number of candidates L in Ordering ICA was set from 8 to 128 by increments of 8. Using the 32-core server, the overhead rate R for the 32-core server was set from 0.25 to 4 by increments of 0.25. Fig 1 shows the plot of $\bar{\delta}(i)$ of the 10 signals by InfoMax and Ordering ICA with L = 8, 16, 32, 64, and 128 only. For L = 8 and L = 16, the Ordering ICA results are not so different from InfoMax. For L = 32, the fluctuations of the top-ranked 20 components (with relatively large non-Gaussianity) are quite small. For L = 128, the fluctuations of the top-ranked 40 components are quite small. Fig 2 clearly shows the relations between the number of candidates L and the averaged fluctuation over the components. The fluctuations are averaged over the four different groups of the components: all, the top-ranked 20 ones with large non-Gaussianity, the medium-ranked 20 ones,

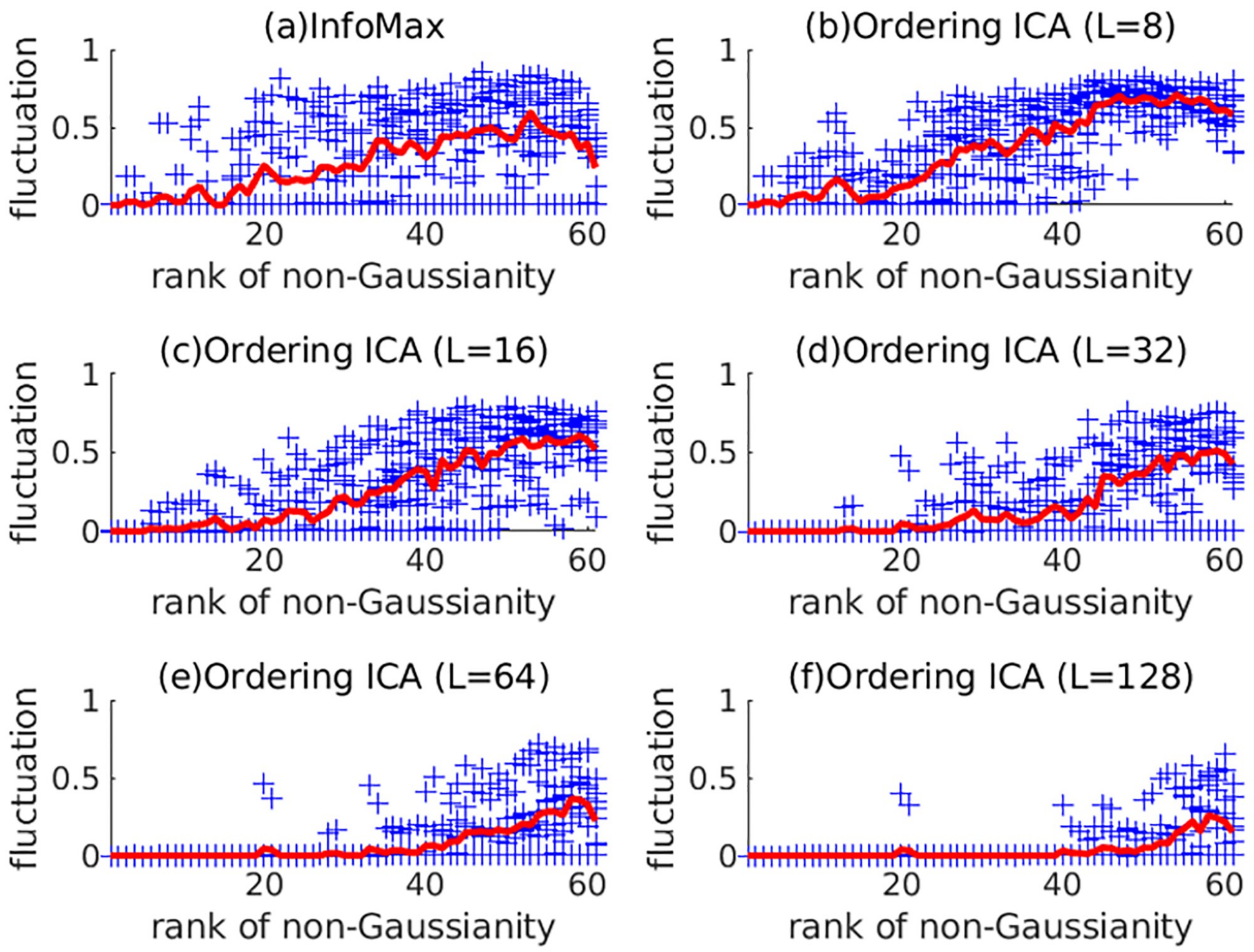

Fig 1. Fluctuation of solutions of ICA in EEG analysis by InfoMax and Ordering ICA with L = 8, 16, 32, 64, and 128 for 5SUBJECTS. Each point (denoted by '+') corresponds to the degree of fluctuations measured as $\bar{\delta}(i)$ in each EEG signal. The horizontal index is sorted by the rank of non-Gaussianity of the estimated components. The solid curve shows the average of the fluctuations over the 10 EEG signals for each index.

https://doi.org/10.1371/journal.pone.0276680.g001

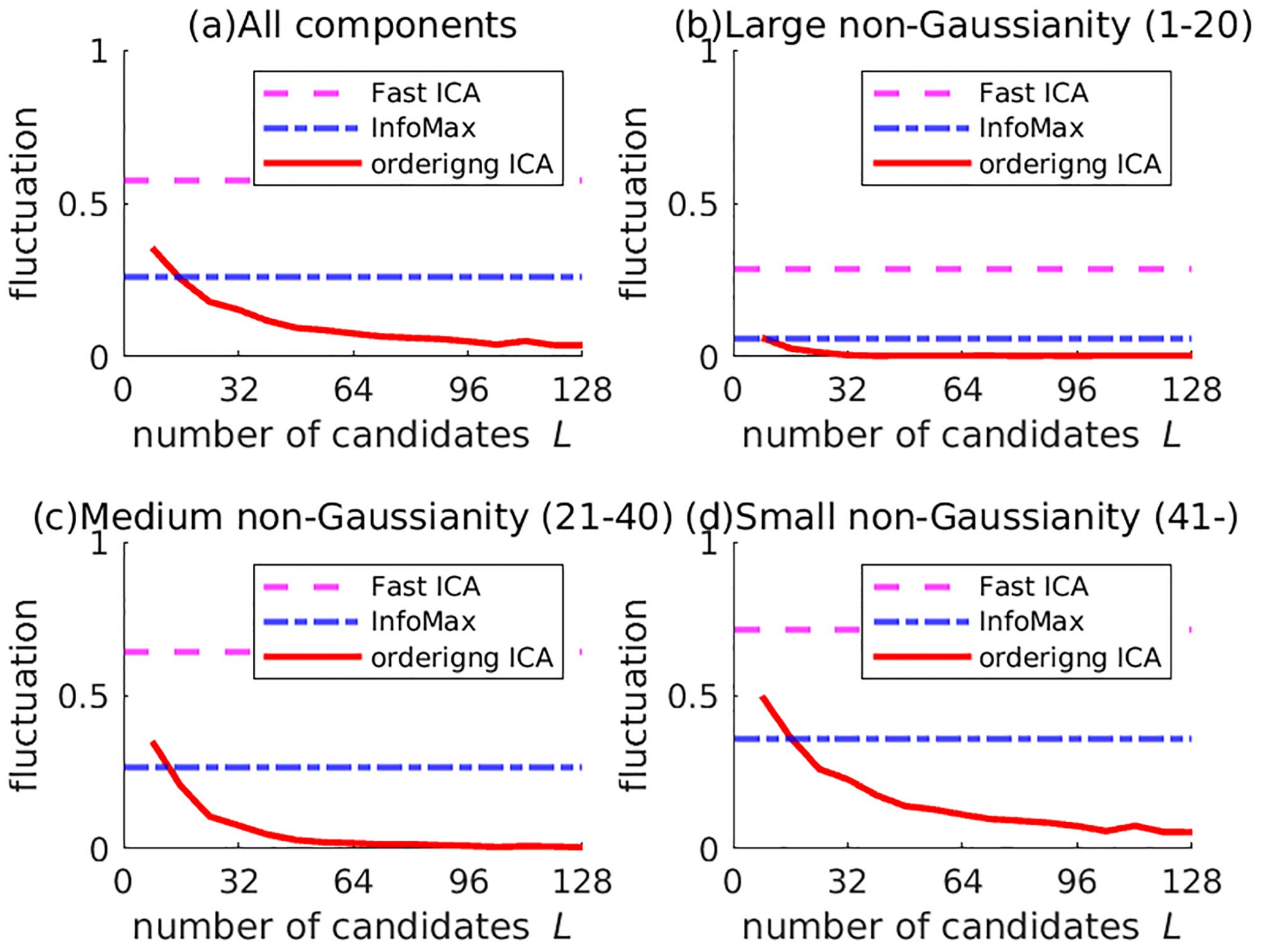

Fig 2. Relation between the number of candidates and the averaged fluctuation over the components for 5SUBJECTS. The solid curve shows the averaged fluctuation over a group of components (all, with large non-Gaussianity (the first-20th ranks), with medium non-Gaussianity (the 21st-40th ranks), with small non-Gaussianity (the rests)). For comparison, the averaged fluctuations of Fast ICA and InfoMax are also shown by the horizontal lines.

https://doi.org/10.1371/journal.pone.0276680.g002

and the rest ones with small non-Gaussianity. The averaged fluctuation of Ordering ICA is smaller than that of Fast ICA and InfoMax around $L = 16$ in every group of components. Regarding the top-ranked 20 components, the averaged fluctuation of Ordering ICA is near 0 around $L = 32$. Regarding the medium-ranked components, it is near 0 around $L = 64$. Though the averaged fluctuation over the rest components does not converge to 0, it decreases as L increases. Fig 3 shows the relation between the computation time of Ordering ICA and the number of candidates L in comparison with Fast ICA and InfoMax. Though the computation time is approximately linear to L, it is slightly shorter if L is a multiple of the number of cores 32. It shows that an integral overhead rate R is efficient. It also shows that the computation time of Ordering ICA is shorter than that of InfoMax if L is less than 64. Even for $L = 128$, the computation time of Ordering ICA is shorter than the double of that of InfoMax.

Next, the experimental results on STERN are shown. The number of candidates L in Ordering ICA was set to 8, 16, 32, 64, or 128 only. The intermediate settings (e.g. $L = 24$ and $L = 40$) were not carried out to save experimental resources. Except for this saving, the experimental results are shown similarly to 5SUBJECTS. If no corresponding channel exists for $N < 71$, the

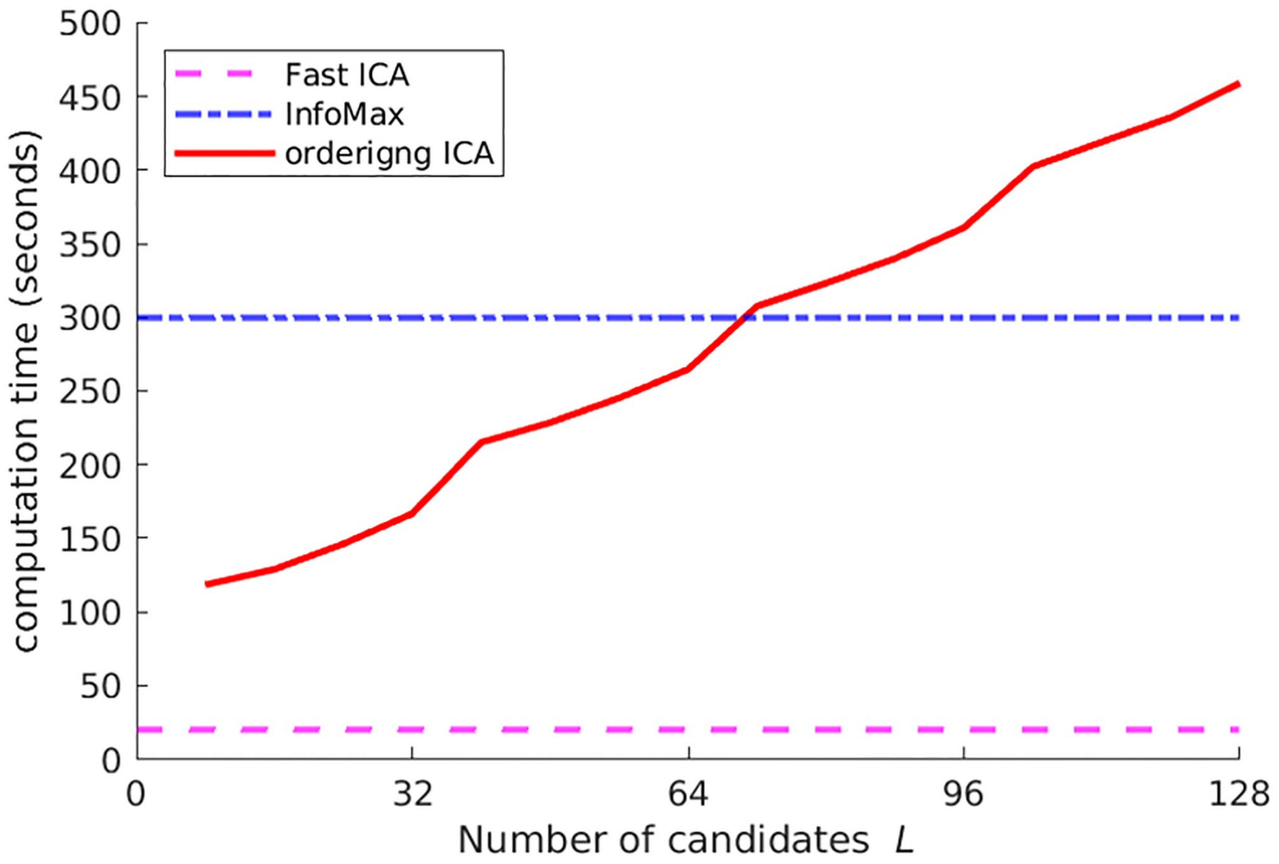

Fig 3. Computation time of Ordering ICA for 5SUBJECTS. The solid curve shows the averaged total computation time over all the signals by the number of candidates *L*. For comparison, the horizontal lines show the averaged computation time of Fast ICA and InfoMax.

https://doi.org/10.1371/journal.pone.0276680.g003

average was calculated over only the existing data. Fig 4 shows the plot of $\bar{\delta}(i)$ of the 39 signals by InfoMax and Ordering ICA with $L = 8, 16, 32, 64$, and 128. Fig 5 shows the relations between the number of candidates $L(= 8, 16, 32, 64, 128)$ and the averaged fluctuation over the components. Note that intermediate numbers of candidates (e.g. $L = 24$) were omitted as opposed to Fig 2. They show that the results of Ordering ICA in STERN are slightly inferior to those in 5SUBJECTS probably because the estimation of the components in STERN is harder than that in 5SUBJECT. The signals of STERN have more channels than those of 5SUBJECTS and the sample size of STERN varies widely. Nevertheless, Ordering ICA is superior to Info-Max around $L = 64$. For $L = 128$, the fluctuation of the top-ranked 20 components and that of the medium-ranked components are near 0. Fig 6 shows the computation time of InfoMax and Ordering ICA ($L = 128$) under STERN's three conditions (I, M, and P). The sample size M is around 100,000 for the condition I, around 150,000 for M, and around 30,000 for P. It shows that the computation time of Ordering ICA with $L = 128$ is at most twice as long as that of InfoMax irrespective of the sample size.

In summary, the ordering ICA algorithm with $L = 64$ is always superior to the widely used InfoMax algorithm from the viewpoint of robustness (measured by the averaged fluctuation). In addition, the computation time of the parallel-ordering ICA algorithm with even $L = 128$ is at most twice as long as that of InfoMax when a 32-core server can be utilized. Though the computation time is roughly linear to L, we can reduce the linear factor by using more cores.

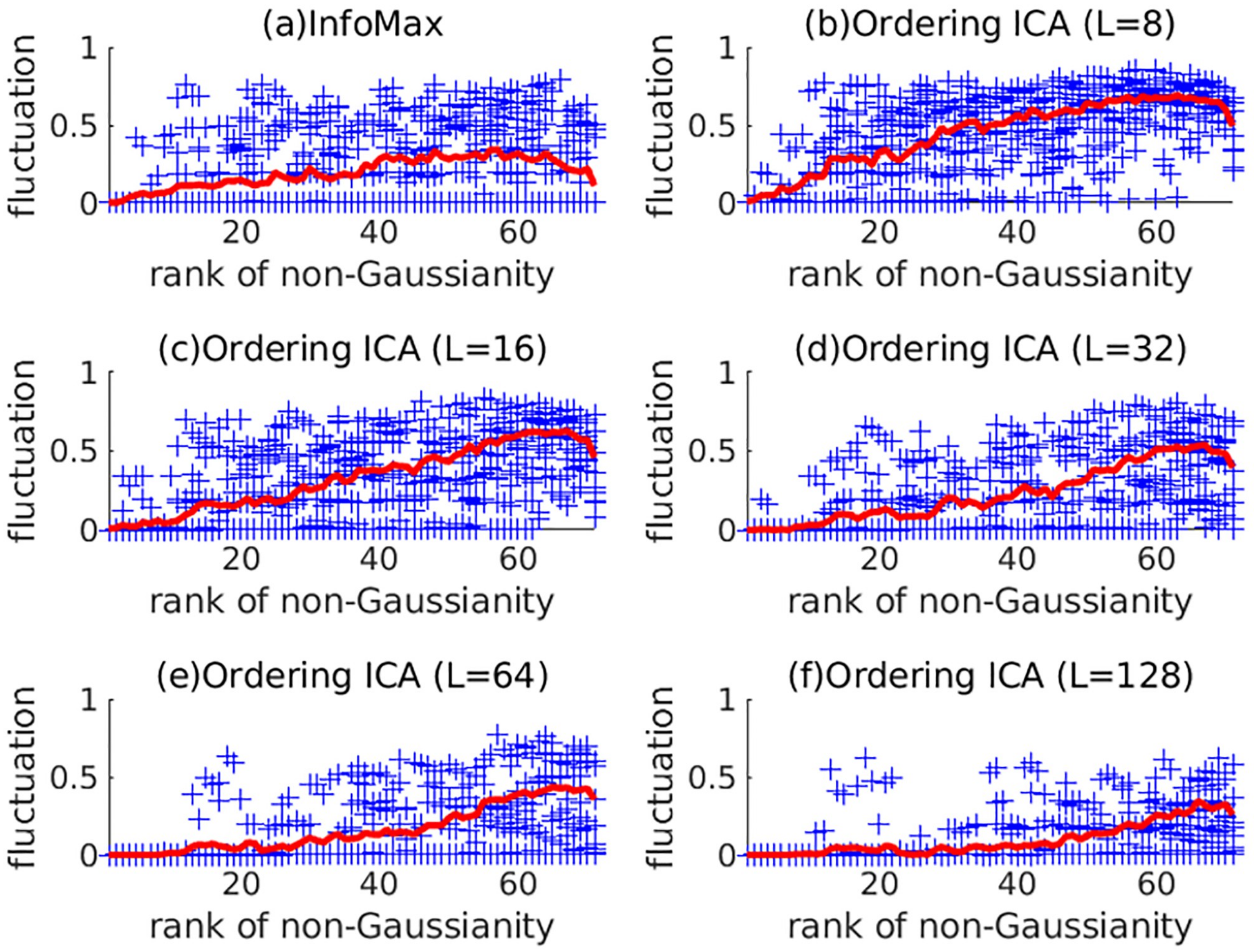

Fig 4. Fluctuation of solutions of ICA in EEG analysis by InfoMax and Ordering ICA with L = 8, 16, 32, 64, and 128 for STERN. The points (denoted by '+') and the curves are displayed in the same way as in Fig 1 for the 39 EEG signals of STERN.

https://doi.org/10.1371/journal.pone.0276680.g004

These results verify the usefulness of the proposed parallel implementation of the parallel-ordering ICA in EEG analysis.

4.3 Relation between non-Gaussianity and success rate of finding unique solution

As shown in the preceding section, the parallel-ordering ICA algorithm can find a unique solution in many cases for the components with large non-Gaussianity. However, it is not guaranteed to succeed in all cases because it is a randomized algorithm. Here, the success rate is evaluated empirically. We also discuss the failure rate of the parallel ordering ICA algorithm and the appropriate choice of the number of candidates L. In addition to 5SUBJECTS and STERN, TUH is employed as a dataset to verify the versatility for various EEG signals.

The success rate of finding the unique solution of a component was empirically evaluated as follows. Ordering ICA with the largest number of candidates ($L = L_{\max}$) is carried out for each EEG signal in a single run. Let $\boldsymbol{w}_i^k$ be the k-th candidate of the i-th components. The estimated i-th components $\boldsymbol{w}_i^{\hat{k}}$ (selected from all the candidates $\boldsymbol{w}_i^k$ ($k = 1, \cdots, L_{\max}$)) is assumed

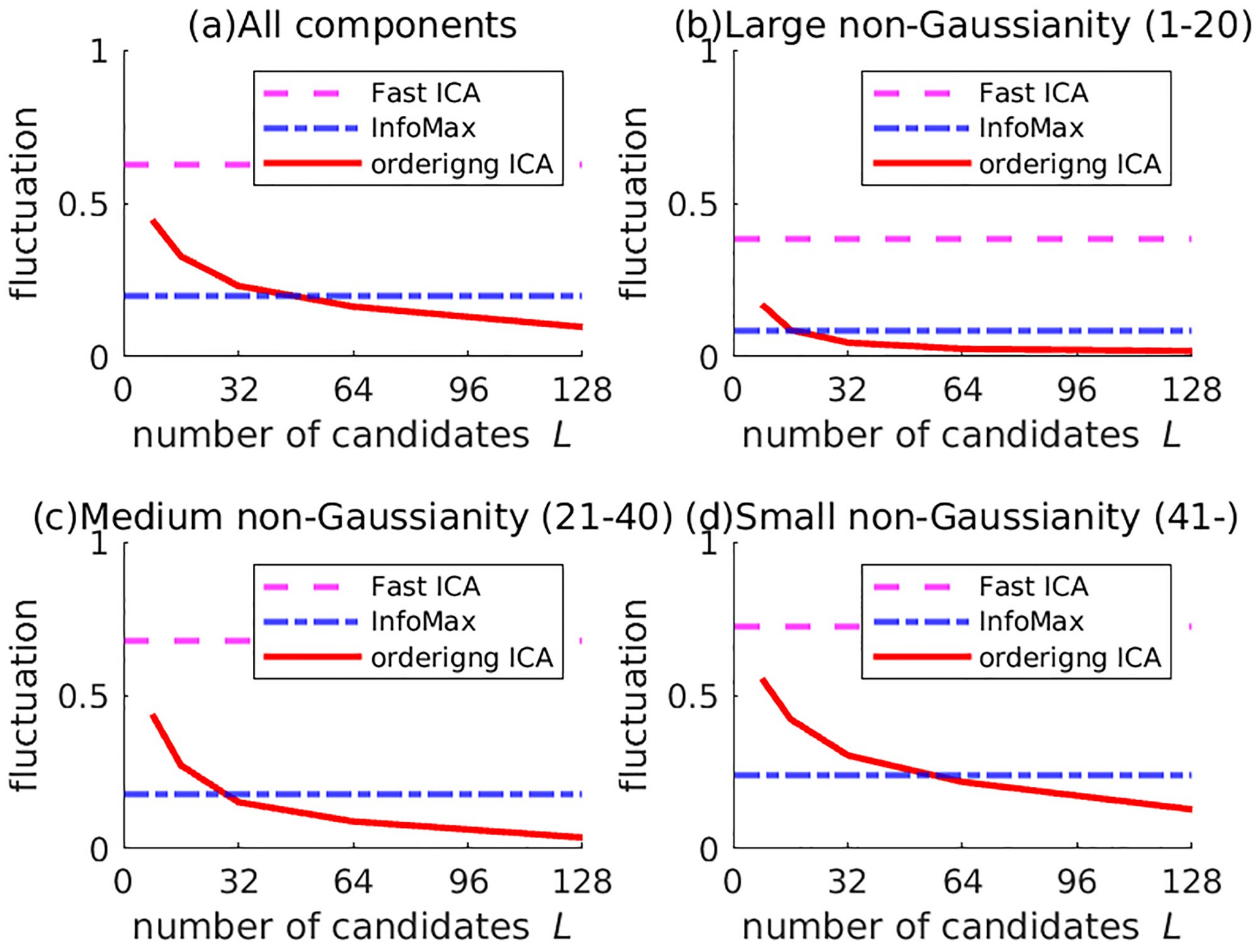

Fig 5. Relation between the number of candidates and the averaged fluctuation over the components for STERN. The curves and the horizontal lines are displayed in the same way as in Fig 2.

https://doi.org/10.1371/journal.pone.0276680.g005

to be the globally optimal and unique solution. The success rate can be evaluated as the rate of candidates sufficiently near the unique solution $\boldsymbol{w}_i^{\hat{k}}$. The set of such candidates is defined as

$$\Xi = \{\boldsymbol{w}_i^k \mid \delta(i, k, \hat{k}) < \epsilon\}, \tag{5}$$

where $\delta(i, k, \hat{k})$ is the divergence between $\boldsymbol{w}_i^k$ and $\boldsymbol{w}_i^{\hat{k}}$ defined by Eq 3. Then, the success rate μ_i of the i-th component is evaluated as

$$\bar{\mu}_i = \frac{\#\Xi}{L_{\max}}, \tag{6}$$

where Ξ is the size of the set Ξ. In this paper, $L_{\max}$ and ϵ were set to 1000 and 0.001, respectively. Note that the minimum of $\bar{\mu}_i$ is $1/L_{\max}$. If every candidate of a component diverges, the component is neglected in the following analysis.

Fig 7 shows the averaged success rate of each component (sorted by non-Gaussianity) for 5SUBJECTS, STERN, and TUH. Here, TUH is higher than 5SUBJECTS and STERN, and 5SUBJECTS is slightly higher than STERN. The difficulty of analyzing each dataset is probably

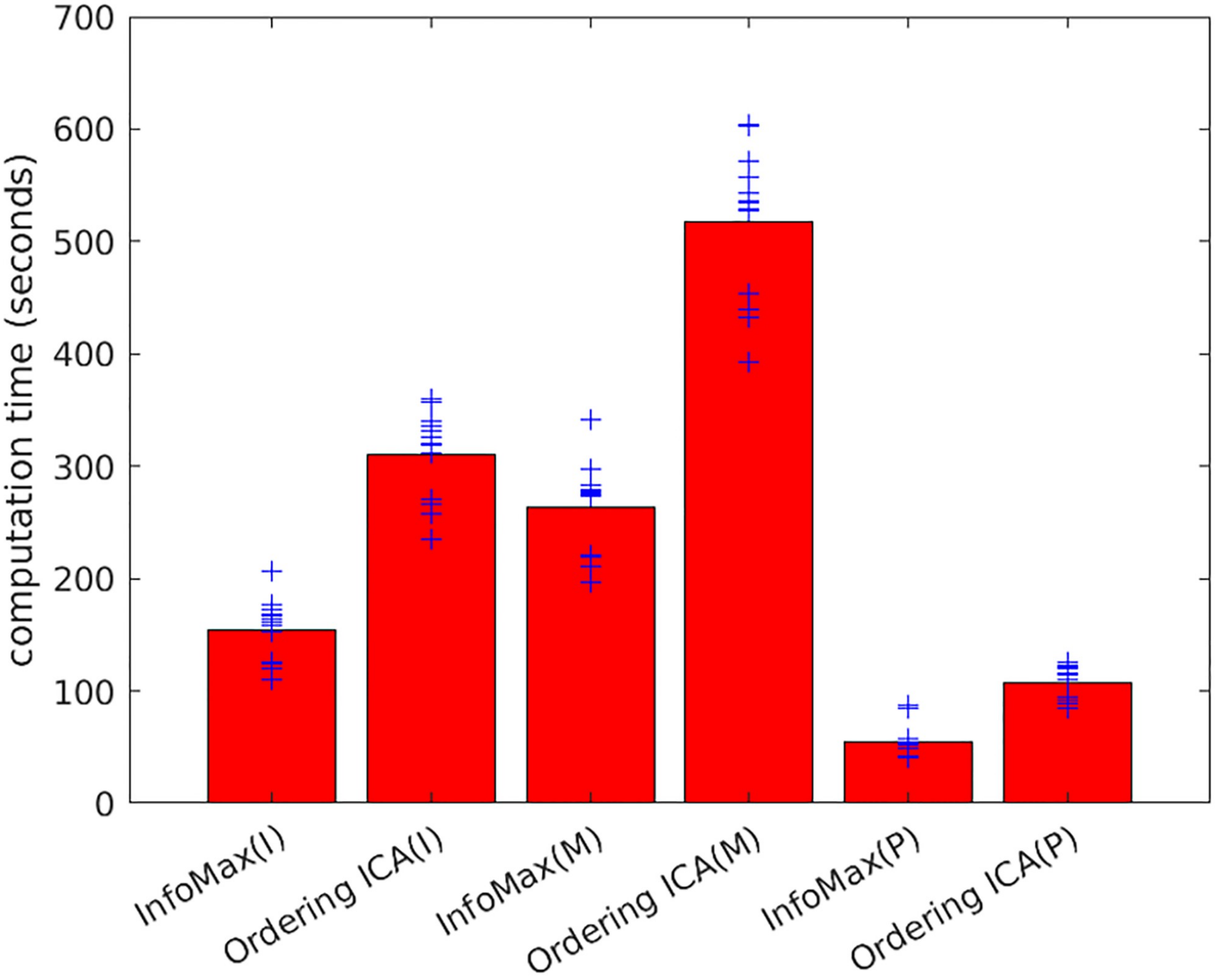

Fig 6. Comparison of computation time of InfoMax and Ordering ICA (L = 128). Each point (denoted by '+') shows the averaged computation time over the runs in each signal of STERN. The results are shown under STERN's three different experimental conditions (I, M, P). The bar shows the computation time averaged over the 13 signals of each condition.

https://doi.org/10.1371/journal.pone.0276680.g006

proportional to the number of channels. It shows that TUH is easier to solve than 5SUBJECTS and STERN. Fig 7 also shows that the success rate decreases as the non-Gaussianity decreases for the upper-ranked components. On the other hand, the success rate tends to increase as the non-Gaussianity decreases for the lower-ranked components. It is probably because the degree of freedom in the deflation approach is lower for the lower-ranked components. For example, the solution of the last component is necessarily unique (namely, $\bar{\mu}_i = 1$) in a single run.

Fig 8 shows the relation between the non-Gaussianity (measured by $\log_{10} \Upsilon(\alpha_i)$ of Eq 2) and the success rate (measured by $\log_{10}\bar{\mu}_i$) for all the upper-ranked components of the three datasets: 5SUBJECTS (the upper-ranked 40 components), STERN (the upper-ranked 40 components), and TUH (the upper-ranked 15 components). It shows that the relation between $\log_{10} \Upsilon(\alpha_i)$ and $\log_{10}\bar{\mu}_i$ is approximately linear. The worst success rate is estimated by

$$\hat{\mu}_i \simeq 0.12\Upsilon(\alpha_i)^{0.21} \qquad (7)$$

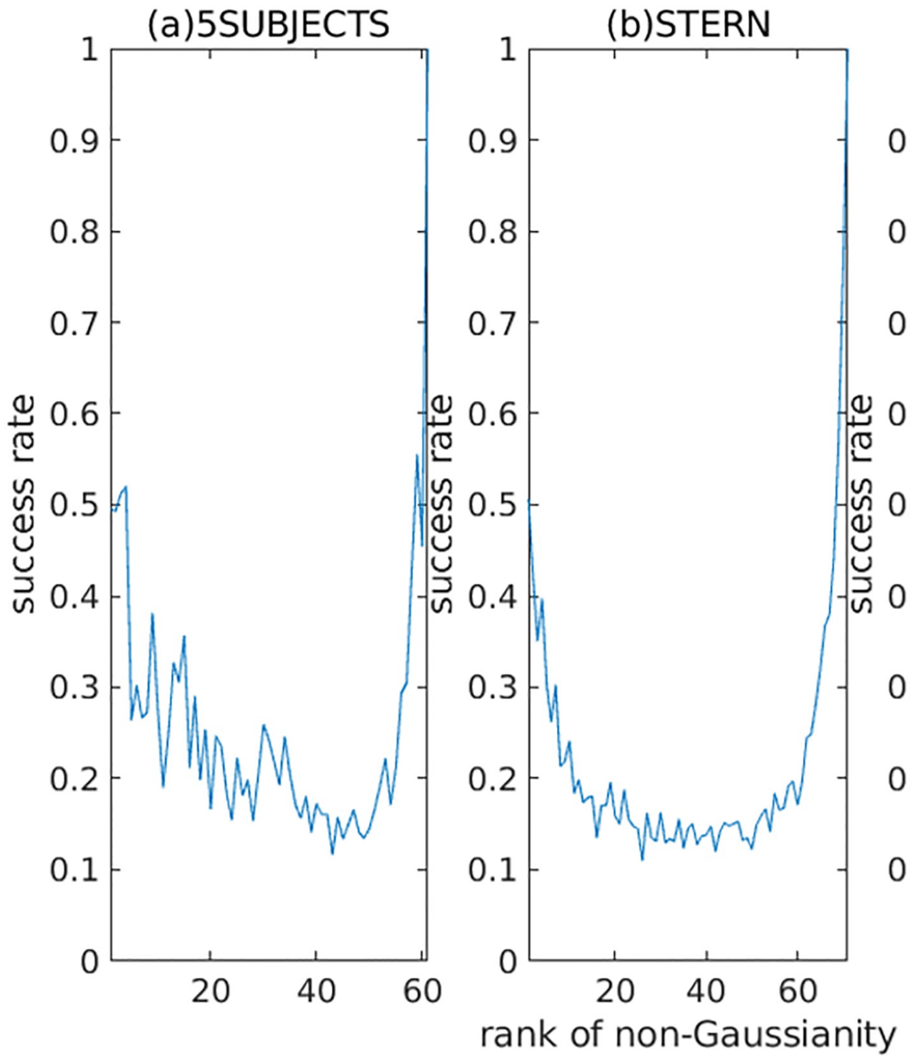

Fig 7. Averaged success rate for 5SUBJECTS, STERN, and TUH. The horizontal index is sorted by the rank of non-Gaussianity of the estimated components. The solid curves show the averages of success rates of finding a unique solution over the signals of each dataset.

https://doi.org/10.1371/journal.pone.0276680.g007

using the worst fitting line for STERN. If the preceding components are estimated global optimally and uniquely, the failure rate v_i^{each} in each i-th component is estimated by

$$v_i^{\text{each}} = (1 - \mu_i)^L. \tag{8}$$

Then, the total failure rate v_i^{total} of failing to find a unique solution is given by the following recurrence relation:

$$v_i^{\text{total}} = v_{i-1}^{\text{total}} + (1 - v_{i-1}^{\text{total}})v_i^{\text{each}}, \tag{9}$$

where $v_0^{\text{total}} = 0$. Therefore, we can approximately evaluate the failure rate of the upper-ranked component by the non-Gaussianity $Y(\alpha_i)$.

Fig 9 plots all the components of all datasets with the non-Gaussianity on the X-axis and the success rate on the Y-axis. It also displays the histograms of each axis. It shows that there

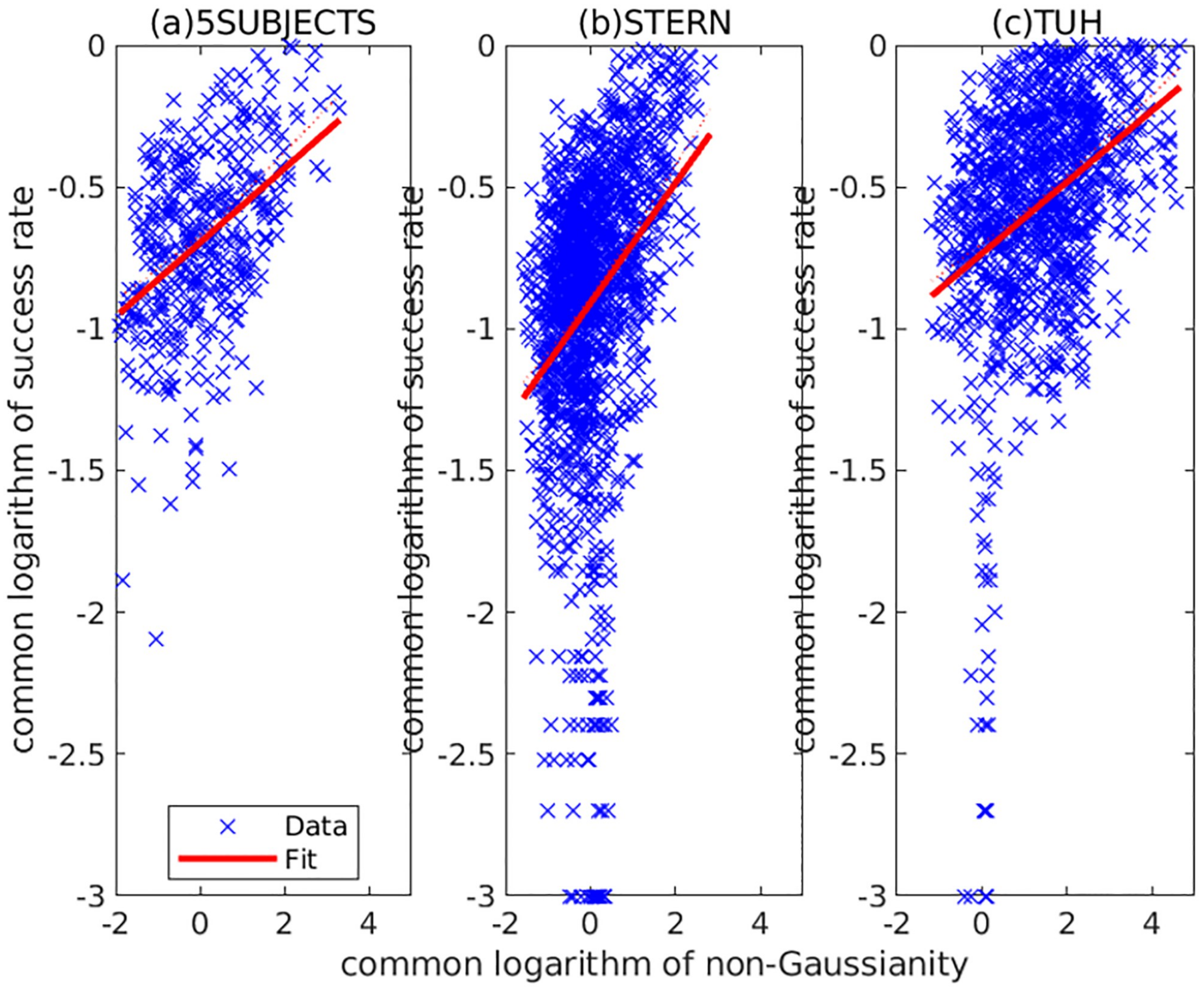

Fig 8. Relation between non-Gaussianity and success rate in upper-ranked components for 5SUBJECTS (upper 40), STERN (upper 40), and TUH (upper 15). Each cross corresponds to $(\log_{10}\Upsilon(\alpha_i), \log_{10}\bar{\mu}_i)$ of a component. The solid line shows the estimated linear regression model $\log_{10}\mu_i = \hat{\beta}_1 + \hat{\beta}_2\log_{10}\Upsilon(\alpha_i)$. $(\hat{\beta}_1, \hat{\beta}_2)$ was estimated as (−0.70, 0.20) for 5SUBJECTS, (−0.91, 0.12) for STERN, and (−0.73, 0.18) for TUH.

https://doi.org/10.1371/journal.pone.0276680.g008

are some bounds. For $\log_{10}\Upsilon(\alpha_i)>1$ (namely, $\Upsilon(\alpha_i)>10$), $\bar{\mu}_i > 10^{-1.5}$ holds in almost all cases. Therefore, the failure rate ν_i^{each} of the components with $\Upsilon(\alpha_i)>10$ is bounded from above by $(1-10^{-1.5})^L \simeq 10^{-0.0139L}$. For example, $\nu_i^{\text{each}} < 0.0164$ for $L = 128$. If we employ a larger L (for example, $L = 256$), ν_i^{each} can be reduced sufficiently ($\nu_i^{\text{each}} < 3 \times 10^{-4}$). The total failure rate also can be reduced ($\nu_i^{\text{total}} < i \times 3 \times 10^{-4}$) if $\Upsilon(\alpha_i)>10$ holds in the top-ranked i components. Because the number of components with $\Upsilon(\alpha_i)>10$ and the total number of components are 1096 and 5713 in the three datasets, the top 20 percent of the components with large non-Gaussianity are expected to be estimated uniquely for $L = 256$. On the other hand, some components have $\bar{\mu}_i < 10^{-1.5}$ if $\log_{10}\Upsilon(\alpha_i)<1$. Finding a unique solution for such components is often difficult. For example, the evaluated $\bar{\mu}_i$ was often the smallest value of 1/1000. It shows that Ordering ICA failed to find a unique solution even for $L = 1000$ in such cases. Because the number of the components with $\bar{\mu}_i < 10^{-1.5}$ is 392 in the three datasets, they are expected to

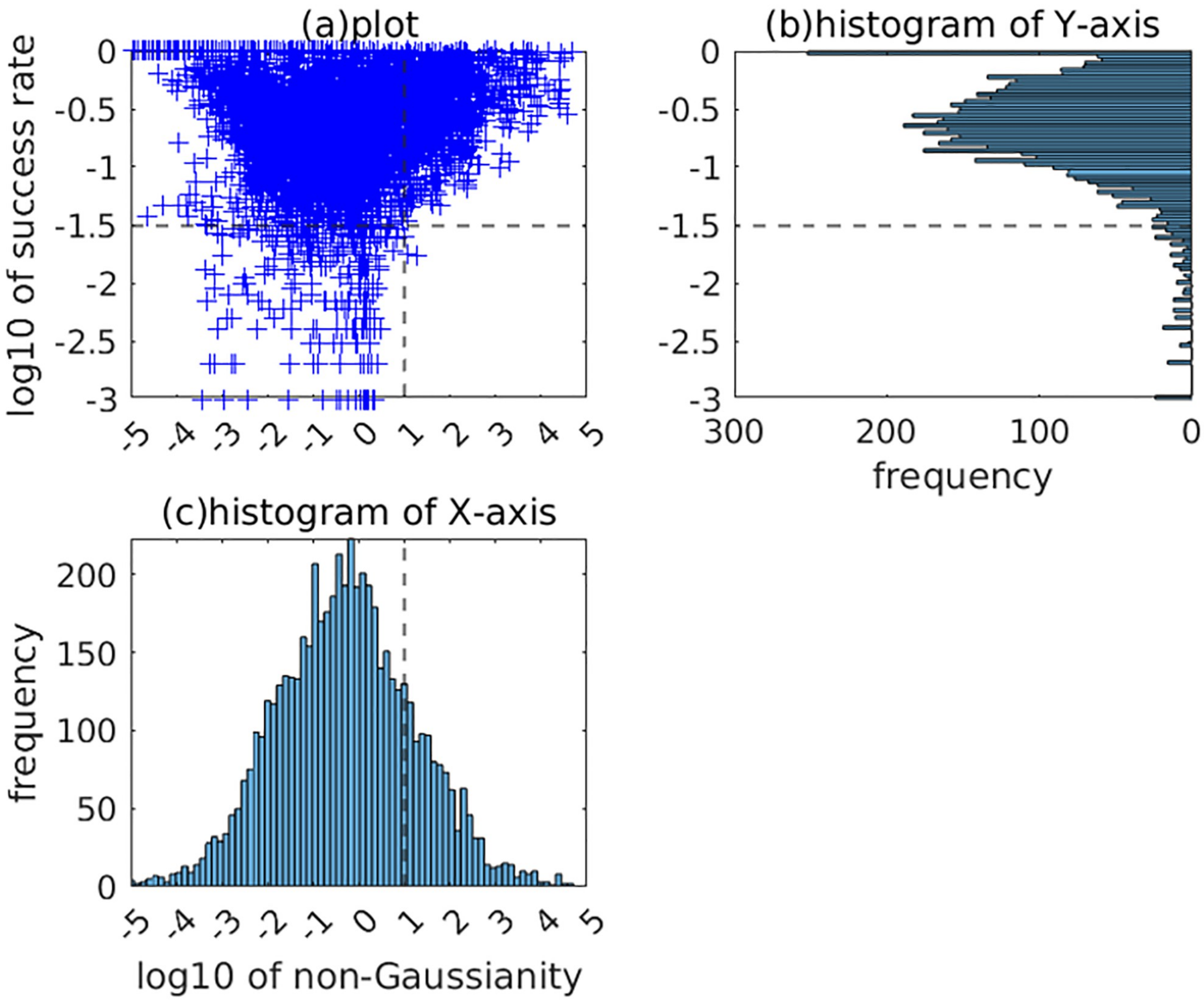

Fig 9. Relation between non-Gaussianity and success rate of all components in all datasets. Each cross in (a) corresponds to $(\log_{10}\Upsilon(\alpha_i), \log_{10}\bar{\mu}_i)$ of a component. (b) and (c) are the histograms of the X-axis and the Y-axis.

https://doi.org/10.1371/journal.pone.0276680.g009

be less than 10 percent of all components. Nevertheless, it should be noted that the total failure rate v_i^{total} can be drastically high if the estimation of a preceding component fails.

In summary, it was observed that there was a linear regression model between the logarithm of the non-Gaussianity and that of the success rate. We can estimate the failure rate of finding a unique solution by the model. In addition, the failure rate was low enough to find a unique solution almost certainly by Ordering ICA with an available number of candidates (for example, $L = 256$) if the non-Gaussianity is relatively large ($\Upsilon(\alpha_i) > 10$).

5 Conclusion

This paper proposed the parallel-ordering ICA algorithm. The experimental results on standard EEG datasets verified that our proposed implementation is superior to the widely-used InfoMax of EEGLAB in both stability and efficiency.

We plan to investigate the usefulness of the parallel-ordering ICA algorithm in practical EEG analysis. In addition, we are planning to accelerate the parallel computation by GPUs and distributed systems. Especially, the utilization of the distributed systems may be promising because the parallelization of the ordering ICA is quite simple.

Author Contributions

Conceptualization: Yoshitatsu Matsuda.

Data curation: Yoshitatsu Matsuda.

Formal analysis: Yoshitatsu Matsuda.

Funding acquisition: Yoshitatsu Matsuda.

Investigation: Yoshitatsu Matsuda.

Methodology: Yoshitatsu Matsuda.

Project administration: Yoshitatsu Matsuda.

Resources: Yoshitatsu Matsuda.

Software: Yoshitatsu Matsuda.

Validation: Yoshitatsu Matsuda.

Visualization: Yoshitatsu Matsuda.

Writing – original draft: Yoshitatsu Matsuda.

Writing – review & editing: Kazunori Yamaguchi.

References

1. Schomer DL, da Silva FHL. Niedermeyer's Electroencephalography: Basic Principles, Clinical Applications, and Related Fields. Oxford, UK: Oxford University Press; 2017.
2. Onton J, Westerfield M, Townsend J, Makeig S. Imaging human EEG dynamics using independent component analysis. Neuroscience & Biobehavioral Reviews. 2006; 30(6):808–822. https://doi.org/10.1016/j.neubiorev.2006.06.007 PMID: 16904745
3. Delorme A, Makeig S. EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics including independent component analysis. Journal of Neuroscience Methods. 2004; 134(1):9–21. https://doi.org/10.1016/j.jneumeth.2003.10.009 PMID: 15102499
4. Matsuda Y, Yamaguch K. A Unifying Objective Function of Independent Component Analysis for Ordering Sources by Non-Gaussianity. IEEE Transactions on Neural Networks and Learning Systems. 2018; 29(11):5630–5642. https://doi.org/10.1109/TNNLS.2018.2806959 PMID: 29993873
5. Cichocki A, Amari S. Adaptive Blind Signal and Image Processing: Learning Algorithms and Applications. Wiley; 2002.
6. Comon P, Jutten C. Handbook of Blind Source Separation: Independent component analysis and applications. Academic press; 2010.
7. Lee TW, Girolami M, Sejnowski TJ. Independent component analysis using an extended infomax algorithm for mixed subgaussian and supergaussian sources. Neural Computation. 1999; 11(2):417–441. https://doi.org/10.1162/089976699300016719 PMID: 9950738
8. Hyvärinen A. Fast and Robust Fixed-Point Algorithms for Independent Component Analysis. IEEE Transactions on Neural Networks. 1999; 10(3):626–634. https://doi.org/10.1109/72.761722 PMID: 18252563
9. Cardoso JF, Souloumiac A. Blind beamforming for non Gaussian signals. IEE Proceedings-F. 1993; 140(6):362–370.