An Experimental Comparison of Neural ICA Algorithms

Xavier Giannakopoulos, Juha Karhunen, and Erkki Oja
Lab. of Computer and Information Science, Helsinki Univ. of Technology
P.O. Box 2200, 02015 HUT, Espoo, Finland

Abstract

Several neural algorithms for Independent Component Analysis (ICA) have been introduced lately, but their computational properties have not yet been systematically studied. In this paper, we compare the accuracy, convergence speed, computational load, and other properties of five prominent neural or semi-neural ICA algorithms. The comparison reveals some interesting differences between the algorithms.

1 Introduction

Independent Component Analysis (ICA) [1, 2] is an unsupervised technique which tries to represent the data in terms of statistically independent variables. Recently, efficient new neural learning algorithms [3, 4, 5, 6, 7, 8] have been developed for ICA and applied to the closely related blind source separation (BSS) and other problems [3]. However, a serious experimental comparison of ICA algorithms is still lacking. In this paper, we present first results on such a comparison, reported in detail in [9].

We consider the standard linear data model used in ICA and BSS [2, 7, 10]:

$$\mathbf{x}(t) = \mathbf{A}\mathbf{s}(t) = \sum_{i=1}^{m} s_i(t)\mathbf{a}_i. \tag{1}$$

Here the components $s_i(t)$, $i = 1, \ldots, m$, of the column vector $\mathbf{s}(t)$ are the m unknown, mutually statistically independent components (or source signals) at time or index value t. For simplicity, they are assumed to be zero mean and stationary. The components of the m-dimensional data vector $\mathbf{x}(t)$ are some linear mixtures of these independent components or sources. The $m \times m$ mixing matrix $\mathbf{A}$ is an unknown full rank constant matrix. Its columns $\mathbf{a}_i$ are the basis vectors of ICA. At most one of the independent components $s_i(t)$ is allowed to be Gaussian.

For learning the ICA expansion (1), an $m \times m$ inverse or separating matrix $\mathbf{B}(t)$ is updated so that the m-vector

$$\mathbf{y}(t) = \mathbf{B}(t)\mathbf{x}(t) \tag{2}$$

becomes an estimate $\mathbf{y}(t) = \hat{\mathbf{s}}(t)$ of the independent components. The estimate $\hat{s}_i(t)$ of the i:th independent component may appear in any component $y_j(t)$ of $\mathbf{y}(t)$. The amplitudes $y_j(t)$ are scaled to have a unit variance.

2 Neural ICA or BSS algorithms

In several neural ICA or BSS algorithms, the data vectors $\mathbf{x}(t)$ are preprocessed by whitening (sphering) them: $\mathbf{v}(t) = \mathbf{V}(t)\mathbf{x}(t)$. Here $\mathbf{v}(t)$ denotes the t:th whitened vector satisfying $\mathrm{E}[\mathbf{v}(t)\mathbf{v}(t)^T] = \mathbf{I}$, where $\mathbf{I}$ is the unit matrix, and $\mathbf{V}(t)$ is an $m \times m$ whitening matrix. Whitening can be done in many ways [7]. After prewhitening the subsequent separating matrix $\mathbf{W}(t)$ can be taken orthogonal, which often improves the convergence. Thus in whitening approaches the total separating matrix is $\mathbf{B}(t) = \mathbf{W}(t)\mathbf{V}(t)$.

Because of limited space, we describe the algorithms included in our study only briefly. For more details, see the references and [9].

Fixed-point (FP) algorithms. One iteration of the generalized fixed-point algorithm for finding a row vector $\mathbf{w}_i^T$ of $\mathbf{W}$ is [3, 11]

$$\begin{aligned} \mathbf{w}_i^* &= \mathrm{E}\{\mathbf{v}g(\mathbf{w}_i^T\mathbf{v})\} - \mathrm{E}\{g'(\mathbf{w}_i^T\mathbf{v})\}\mathbf{w}_i \\ \mathbf{w}_i &= \mathbf{w}_i^*/\|\mathbf{w}_i^*\|. \end{aligned} \tag{3}$$

Here $g(t)$ is a suitable nonlinearity, typically $g(t) = t^3$ or $g(t) = \tanh(t)$, and $g'(t)$ is its derivative. The expectations are in practice replaced by their sample means. Hence the fixed-point algorithm is not a truly neural adaptive algorithm. The algorithm requires prewhitening of the data. The vectors $\mathbf{w}_i$ must be orthogonalized against each other; this can be done either sequentially or symmetrically [11, 9]. Usually the algorithm (3) converges after 5-20 iterations.

Natural gradient algorithm (ACY). Originally proposed in [6] on heuristic grounds, this popular and simple neural gradient algorithm was later on derived from information-theoretic criteria [4]. The algorithm does not require prewhitening. The update rule for the separating matrix $\mathbf{B}$ is

$$\Delta\mathbf{B} = \mu_k[\mathbf{I} - \mathbf{g}(\mathbf{y})\mathbf{y}^T]\mathbf{B}. \tag{4}$$

The notation $\mathbf{g}(\mathbf{y})$ means that the nonlinearity $g(t)$ is applied to each component of the vector $\mathbf{y} = \mathbf{B}\mathbf{x}$. The learning parameter μ_k is usually a small constant. In practice, $\mathbf{B}$ is often updated using small batches of data [5].

Extended Bell-Sejnowski algorithm (ExtBS). The update rule is otherwise the same as in (4) but whitening is used to improve the convergence properties:

$$\Delta\mathbf{W} = \mu_k[\mathbf{I} - \mathbf{g}(\mathbf{y})\mathbf{y}^T]\mathbf{W} \tag{5}$$

where now $\mathbf{y} = \mathbf{W}\mathbf{v}$. In the extended form, kurtosis is estimated on-line for handling both super-Gaussian and sub-Gaussian sources [12], and the learning parameter μ_k is optimized using a momentum term and simulated annealing [12, 13].

EASI algorithm. Introduced as an adaptive signal processing algorithm in [10], EASI can be applied as a neural learning algorithm as well. The general update formula for the $\mathbf{B}$ contains two extra terms compared to (4):

$$\Delta\mathbf{B} = \mu_k \left[\frac{\mathbf{I} - \mathbf{y}\mathbf{y}^T}{1 + \mu_k\mathbf{y}^T\mathbf{y}} - \frac{\mathbf{g}(\mathbf{y})\mathbf{y}^T + \mathbf{y}\mathbf{g}(\mathbf{y}^T)}{1 + \mu_k|\mathbf{y}^T\mathbf{g}(\mathbf{y})|} \right] \mathbf{B} \tag{6}$$

In this comparison, we used the normalized version of EASI given above. In the unnormalized version, the terms in the denominator of (6) are left out, which may cause stability problems especially for nonlinearities growing faster than linearly [10].

RLS algorithm for a nonlinear PCA criterion (NPCA-RLS). The basic symmetric version, adapted for the BSS problem using prewhitened data vectors $\mathbf{v}(t)$, is [8]

$$\begin{aligned}
\mathbf{z}(t) &= \mathbf{g}(\mathbf{W}(t-1)\mathbf{v}(t)) = \mathbf{g}(\mathbf{y}(t)), \\
\mathbf{h}(t) &= \mathbf{P}(t-1)\mathbf{z}(t), \\
\mathbf{m}(t) &= \mathbf{h}(t)/(\beta + \mathbf{z}^T(t)\mathbf{h}(t)), \\
\mathbf{P}(t) &= \frac{1}{\beta}\mathrm{Tri}\left[\mathbf{P}(t-1) - \mathbf{m}(t)\mathbf{h}^T(t)\right], \\
\mathbf{e}(t) &= \mathbf{v}(t) - \mathbf{W}^T(t-1)\mathbf{z}(t), \\
\mathbf{W}(t) &= \mathbf{W}(t-1) + \mathbf{m}(t)\mathbf{e}(t)^T.
\end{aligned} \tag{7}$$

The forgetting constant $0 < \beta \leq 1$ should be close to unity. The notation Tri means that only the upper triangular part of the argument is computed and its transpose is copied to the lower triangular part. The NPCA-RLS algorithm (7) is a recursive least-squares version of the nonlinear PCA algorithm [7]. The learning parameter is determined so that it becomes roughly optimal [8].

3 Experimental results

3.1 Artificially generated data

We have thus far made simulations using mainly artificially generated data, because only then the accuracy and convergence speed of the algorithms can be measured reliably. For real-world data, the true independent components (or their best approximations) are unknown.

The experimental setup was the same for each algorithm in order to make the comparison fair; see [9]. We used the original MATLAB codes provided by the authors whenever possible, such as [14] for the fixed-point algorithms. For the experiments, both sub-Gaussian and super-Gaussian sources were generated [9], and the mixing matrix $\mathbf{A}$ consisted of uniformly distributed random numbers.

The accuracy was measured using two performance indexes. The first one, E_1, is defined by [4]

$$E_1 = \sum_{i=1}^{n}\Big(\sum_{j=1}^{n} \frac{|p_{ij}|}{max_k|p_{ik}|} - 1\Big) + \sum_{j=1}^{n}\Big(\sum_{i=1}^{n} \frac{|p_{ij}|}{max_k|p_{kj}|} - 1\Big) \tag{8}$$

where $\mathbf{P} = (p_{ij}) = \mathbf{BA}$ should be a permutation matrix if the sources have been separated perfectly. The second performance index E_2 is otherwise the same but the absolute values are replaced by squares in (8). For both indices, the greater the value, the poorer the performance. The minimum is zero.

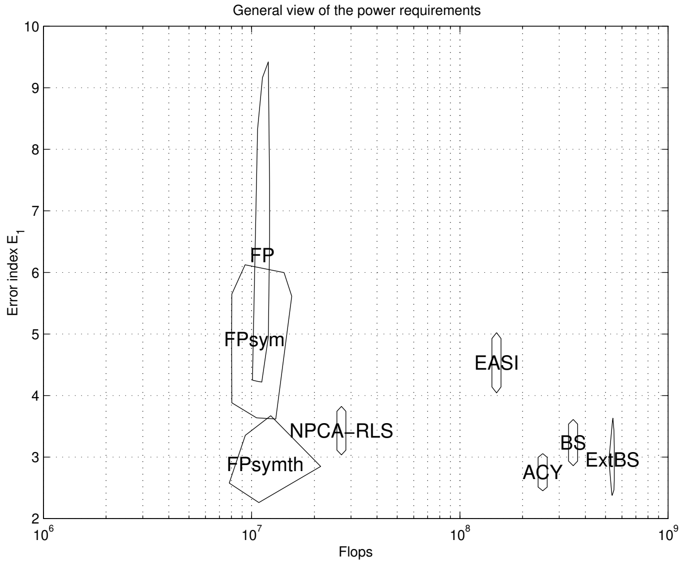

Figure 1: Power requirements in flops vs. error index E_1. The boxes typically contain 80% of the 100 trials.

Figure 1 is a schematic diagram of the results of basic experiments measuring both the accuracy and the computational load (in floating point operations) of the tested algorithms. The number of sources (independent components) was 10. Clearly, fixed-point algorithms require the smallest amount of computation. The symmetric version with a tanh nonlinearity (FPsymth) is the most accurate of them, while the accuracy of the basic FP algorithm using sequential orthogonalization is poorer than for the other algorithms. Of the adaptive algorithms, NPCA-RLS converges fastest. The natural gradient algorithm (ACY) and extended Bell-Sejnowski algorithm achieve a good final accuracy, but their computational load is much higher.

In figure 2, the error (square root of the index E_2) is plotted as a function of the number of super-Gaussian sources (for which all the algorithms worked). Generally, the natural gradient algorithm (ACY) and various modifications of it (BS, ExtBS, WACY) have the best accuracy, behaving very similarly as expected. Fixed-point algorithm (FP) has the poorest accuracy, but its error increases only slightly after 7 sources. For an unknown reason, the error of the EASI and NPCA-RLS algorithm has a peak around 5-6 sources. However, the error of all the algorithms is tolerable for most practical purposes.

In experiments where the number of sources was increased, it was necessary to replace the cubic nonlinearity $g(t) = t^3$ with the more stable $g(t) = \tanh(t)$ in the EASI algorithm and with $g(t) = \tanh(t) - t$ in the ACY algorithm to make them converge with more than 10 sources.

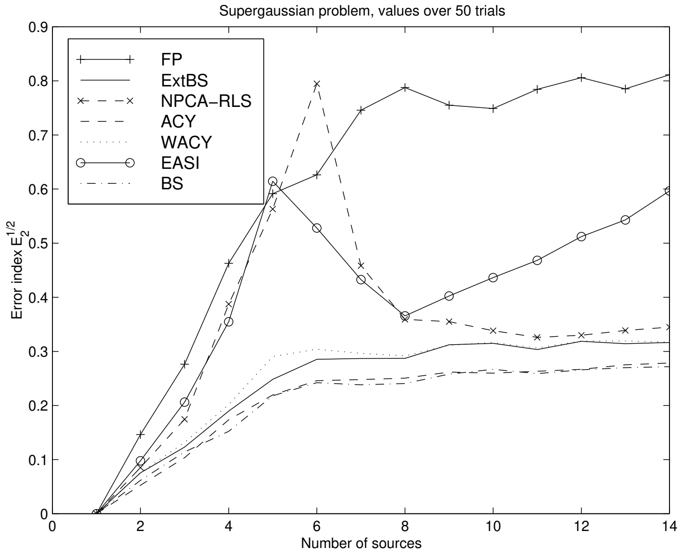

Figure 2: Error as a function of the number of sources.

When Gaussian noise was added to the data, the first conclusion was that degradation of the results is smooth at least until the noise power increases up to -20dB of the signal power. Another observation was that once there is even a little of noise present in the data, the error strongly depends on the condition number of the mixing matrix. This holds both for equivariant algorithms which compute the separating matrix $\mathbf{B}$ directly and for algorithms employing prewhitening, except for NPCA-RLS which behaves differently.

A general conclusion on the experiments with artificial data is that for designing an efficient ICA algorithm, one should split the problem into different parts. These include at least the following choices: the algorithm; the nonlinearity; and the control structure. Of course, the dependencies between these constituent parts must be taken into account. Such a design allows one to make a good practical compromise between efficiency, robustness, precision, and other relevant requirements of the problem at hand.

3.2 Real-world data

We have recently extended our comparisons to real-world data [9], trying to find good projection pursuit directions using ICA algorithms. In projection pursuit [15], the goal is to find for visualization purposes one-dimensional projections of multidimensional data containing as much "interesting" structural information as possible. ICA seems to be a good tool for projection pursuit, because it provides non-Gaussian projections containing meaningful structural information as suggested already in [7].

As an example, we considered 5-dimensional data set consisting of measurements made on two species of crabs. All the ICA algorithms except for (4) found two good ICA basis vectors separating the species and also to a certain extent males and females of both species; see [9]. The results were clearly better than those given by standard PCA.

4 Conclusions

The main conclusions of this experimental comparison of ICA algorithms are as follows. The semi-neural fixed-point algorithm converges fastest, and can deal with both sub-Gaussian and super-Gaussian sources without on-line estimation of the kurtosis required by the other algorithms. Of the adaptive neural algorithms, the recursive least-squares algorithm usually has the smallest computational load. The accuracy of the Bell-Sejnowski algorithm and its natural gradient extensions is very good. The main factor affecting the final accuracy of the algorithms is the choice of the nonlinearity.

References

[1] P. Comon, *Signal Processing*, vol. 36, pp. 287-314, 1994.
[2] C. Jutten and J. Herault, *Signal Processing*, vol. 24, no. 1, pp. 1-10, July 1991.
[3] E. Oja et al., in S.-I. Amari and N. Kasabov (Eds.), *Brain-Like Computing and Intelligent Information Systems*, Springer, Singapore, 1997, pp. 167-188.
[4] H. Yang and S.-I. Amari, *Neural Computation*, vol. 9, pp. 1457-1482, 1997.
[5] A. Bell and T. Sejnowski, *Neural Computation*, vol. 7, pp. 1129-1159, 1995.
[6] A. Cichocki and R. Unbehauen, *IEEE Trans. on Circuits and Systems-1*, vol. 43, pp. 894-906, Nov. 1996.
[7] J. Karhunen, E. Oja, L. Wang, R. Vigario, and J. Joutsensalo, *IEEE Trans. on Neural Networks*, vol. 8, pp. 486-504, May 1997.
[8] J. Karhunen and P. Pajunen, in *Proc. 1997 Int. Conf. on Neural Networks*, Houston, Texas, June 1997, pp. 2147-2152.
[9] X. Giannakopoulos, "Comparison of adaptive independent component analysis algorithms," Dipl.Eng thesis made for EPFL, Switzerland, at Helsinki Univ. of Technology, Finland, 58 p. Available at `http://www.cis.hut.fi/~xgiannak/`.
[10] J.-F. Cardoso and B. Hvam Laheld, *IEEE Trans. on Signal Processing*, vol. 44, pp. 3017-3030, Dec. 1996.
[11] A. Hyvärinen, in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, Munich, Germany, April 1997, pp. 3917-3920.
[12] M. Girolami and C. Fyfe, in *Proc. 1997 Int. Conf. on Neural Networks*, Houston, Texas, June 1997, pp. 1788-1791.
[13] M. McKeown et al., *Proc. Natl. Acad. Sci. USA*, vol. 95, pp. 803-810, 1998.
[14] *Fast ICA MATLAB package.* Available at `http://www.cis.hut.fi/projects/ica/fastica`.
[15] J. Friedman, *J. of Amer. Stat. Assoc.*, vol. 82, no. 397, pp. 249-266, March 1987.