ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Computer Science Department

Comparison of Adaptive Independent Component Analysis Algorithms

Xavier Giannakopoulos

Research work done at the

Helsinki University of Technology
Laboratory of Information and Computer Science
Rakentajanaukio 2 C
P.O. Box 2200 FIN-02015 HUT
Suomi-Finland

for the degree of

ingénieur informaticien diplômé EPFL

presented at the

Swiss Federal Institute of Technology
CH-1024 Ecublens
Switzerland

Espoo, Finland, March 20, 1998

Title: Comparison of Adaptive Independent Component Analysis Algorithms
Author: Xavier Giannakopoulos
Date: 13th March 1998
Unit at HUT: Laboratory of Computer and Information Science
Unit at EPFL: Center for Neuro-Mimetic Systems
Supervisors at HUT: Professor Erkki Oja and Doctor Juha Karhunen
Supervisor at EPFL: Professor Wulfram Gerstner
Keywords: independent component analysis, blind source separation, projection pursuit, algorithm performance, neural networks

Abstract

Independent Component Analysis (ICA) is a recent method for data analysis, based on statistical properties of multidimensional data. Many algorithms performing ICA have appeared in the last years, and there is a need to compare the different methods proposed.

In this work five algorithms performing ICA have been set into a common framework for comparing the results on the same inputs. In the first part, basic theoretical concepts are reviewed and ICA is exposed. Each algorithm is described with emphasis on the common points. In the first part of the experiments, comparisons are made on a simple, artificial blind source separation problem. Parameters of the problem that have a great influence on the process are given a range of different values, and the consequences on the ICA solution are examined. In the second part, ICA is used for obtaining meaningful projections of a multidimensional data set (Projection Pursuit). The quality of the projection is assessed in various ways.

This version contains some minor corrections made to the original version. It is therefore not the exact thesis in the strict sense.

Acknowledgments

I would like to thank the persons in the Laboratory of Computer and Information Science and the Neural Network Research Center for the rich environment provided on the scientific level as well as on the human level. Although sad events darkened this period of my life, joining the laboratory and the ICA team has been a great experience.

I am especially grateful to Professor Erkki Oja, who decided of my stay here and kept it on the right tracks, to Dr Juha Karhunen for the supervision and revision work, to Dr Aapo Hyvärinen and M.Sc. Petteri Pajunen for providing many theoretical answers and hints, and last but not least, to M.Sc. Ricardo Vigário for our numerous and vivid debates on each cornerstone of the work.

For my stay in Helsinki, I was accepted as a Socrates student. The Swiss Confederation also provided me with a mobility grant.

In Espoo, 13th March, 1998

Xavier Giannakopoulos

Contents

Abbreviations

ACY	Amari, Cichocki & Yang algorithm
BS	Bell & Sejnowski algorithm
BSS	Blind Source Separation
cdf	cumulative density function
EASI	Equivariant Adaptive Separation via Independence
ExtBS	Extended Bell & Sejnowski algorithm
FP	Fixed Point algorithm
ICA	Independent Component Analysis
NPCA-RLS	Nonlinear Principal Component Analysis by Recursive Least Squares algorithm
OO	Object-Oriented
PCA	Principal Component Analysis
pdf	probability density function
PP	Projection Pursuit
WACY	Amari, Cichocki & Yang algorithm, whitened version

1 Introduction

The idea of ICA can be tracked back to 1983, when Hérault and Jutten made a pioneering work on a real-time algorithm with a neuro-mimetic architecture dedicated to BSS. On the theoretical side, the framework of high-order moments allowed Cardoso [4, 5] to propose one of the first direct solutions to a BSS problem. In 1991, Hérault and Jutten [21] published a complete presentation of their adaptive algorithm. Their article outline the transition from PCA to ICA (INCA as they call it) very clearly.

More than 10 years later, a lot of work has been done, and more algorithms have emerged in different versions. Also several neural type algorithms have been proposed lately, but there has been no comprehensive comparison between them yet. The purpose of this study is to test different neural ICA algorithms on the same situation, and by analyzing the outcome extract the properties and abilities of each algorithm. The goal here is to backup the theoretical results as well as to provide some insight for persons interested about the properties of the algorithms for use in applications.

The algorithms included in this study are neural or semi-neural algorithms for ICA. I tried to include all the more prominent approaches, but the scheme is for sure not complete. Sometimes the algorithms have different qualities, and I tried to get them all inside a unifying framework.

Every algorithm must be put in the same situation for a fair comparison. Some algorithms come from the laboratory where I work, the others don't. It is clear that I can easier get an explanation or an improvement from the persons who are working here, and that sets me in a delicate position.

My information about fixed point and NPCA-RLS algorithm is more up to date than for the other algorithms. This shall be a call for corrections by persons that know a better version of the other algorithms or new, more powerful algorithms.

As a way of addressing this point, I will clearly define the limitations of my research. First, I operate on MATLAB code, for reasons of time and access to ready-made algorithms. Second, I will make use of the code provided on the Web as it is, as a first step. If there is no code available, I shall make a basic implementation of the missing algorithms that I want to include.

This study is organized as follow. In Section 2, the concept of independence is introduced and examined under different points of view. In Section 3, ICA is defined and some properties of this method are explained. Section 4 presents each algorithm included in this study in a way as uniform as possible. In Section 5 the details related to the actual implementation of the testbed are described. Section 6 reports the result of the experiments performed on artificial data. Finally Section 7 presents the results obtained on multidimensional data sets from the public domain.

Part I
Theory of ICA

2 Preliminaries

2.1 Independence

We are going to deal with statistical independence, as a way of characterizing signals. Independence is a statistical property. For two events A and B, independence is the fulfillment of

$$\mathrm{P}(A, B) = \mathrm{P}(A)\mathrm{P}(B) \tag{1}$$

where $\mathrm{P}(A, B)$ is the joint probability of observing that the events A and B happen. There are numerous ways of interpreting this relation. One of them is that the event A bears no information about the event B. In this case it is better written

$$\mathrm{P}(A|B) = \mathrm{P}(A) \tag{2}$$

where $\mathrm{P}(A|B)$ is the probability of observing A on the condition that event B has occurred. This is a conditional probability, and in the case of independence, the event B sets no condition on A, so that the probability is the same, conditioned by B or not.

Returning back to Eq. 1, one can see that the probability of the event "A and B" on the left side has been factored into two expressions joined by a multiplication. This is very important because thanks to independence, the joint probability has been split into the product of two probabilities. This simplifies greatly the computation of the joint probability. In many statistical developments, independence is assumed simply to make the calculations tractable.

The discrete case is useful for illustration purposes, but it is not interesting for this work. As soon as possible description shall be based on the general case of multidimensional continuous random variables. For a continuous multidimensional random variable $\mathbf{x} \in \mathbb{R}^N$ having a density function $f_{\mathbf{x}}(\mathbf{x})$, independence is defined by

$$f_{\mathbf{x}}(\mathbf{x}) = \prod_{i=1}^{N} f_{x_i}(x_i). \tag{3}$$

In this form the factorization clearly appears as a way of splitting a multivariate function into univariate functions which can be analyzed and processed separately. This property of being factorizable is very useful in the context of projection pursuit, where the factorized dimensions may be associated with variables governing the process that are usually hidden by the vast amount of data.

To understand better some of the advantages and drawbacks of the independence, let us make a graph representation of the structure of a multivariate density function. Let G be a graph constructed from $f_{\mathbf{x}}(\mathbf{x})$ with one vertex for each x_i, and one edge connecting each independent pair.

$$G = (V, E) \tag{4}$$
$$V = \{x_i\} \quad \forall i \in 1, ..., N \tag{5}$$
$$(x_i, x_j) \in E \Leftrightarrow f_{x_i,x_j}(x_i, x_j) = f_{x_i}(x_i) f_{x_j}(x_j) \quad \forall i, j \in 1, ..., N \tag{6}$$

The complementary graph $\overline{G}$ is then a dependency graph. A dependency graph expresses the structure of the whole random process and edges in this graph represent possible flux of information. Due to the transitivity of the dependency relationship, expressed by

$$\text{if } f_{x_i,x_j}(x_i, x_j) \neq f_{x_i}(x_i) f_{x_j}(x_j) \text{ and } f_{x_j,x_k}(x_j, x_k) \neq f_{x_j}(x_j) f_{x_k}(x_k) \text{ then } f_{x_i,x_k}(x_i, x_k) \neq f_{x_i}(x_i) f_{x_k}(x_k), \tag{7}$$

each subset of linked nodes form a fully connected subgraph, and two disconnected nodes pertain to different fully connected subgraphs. In other terms, the dependency graph is always the result of a transitive closure. The advantage is that independence allows to partition the graph and to split the problem into separate subproblems. One drawback of independence that comes from the same property is that there is no room for other structures than transitive closures. For instance chain dependency cannot be seen by inspecting the independence relationship.

2.2 Information theory

One way of dealing with independence is to rely on information theory. This means deriving the criterion from statistical properties of the data. Entropy is such a criterion, based on the amount of information contained in some occurrences of a random variable. I shall recall some basic facts about entropy and related criteria. This exposition and all the proofs can be found in [12] for the general part and in [10] applied to ICA.

The definition of entropy for a discrete variable x with K values $\{x_i\}, i = 1, ..., K$ is

$$H(x) = -\sum_{i=1}^{K} p_i \log(p_i) \tag{8}$$

where $p_i = P\{x = x_i\}$ is the probability of x taking the value x_i.

One can view this formula as the expectation of the $-\log(p_i)$ factor that relates to the average amount of information gained by observing one occurrence of x. Its properties are that

- $H(x) = 0$ iff the probability has a form $p_i = 1$ for some i, and 0 for the others. This is certainty, the minimum of entropy.
- $H(x) = \log(K)$ iff $p_i = 1/K, \forall i$, that is x is a uniform variable. This is uncertainty, the maximum of entropy.
- Entropy is invariant to invertible transformations.

Now let's move to the continuous case, which is the only one that is relevant in this study. Entropy does not exist for continuous random variables, but differential entropy does. Its definition for a continuous random variable x is

$$h(x) = -\int_{-\infty}^{\infty} f(u) \log f(u) du \tag{9}$$

where $f(x)$ is the probability density function of the variable x.

In case of a multidimensional variable $\mathbf{x}$, with distribution $f_{\mathbf{x}}(\mathbf{x})$, the joint entropy obviously becomes

$$h(\mathbf{x}) = -\int f_{\mathbf{x}}(\mathbf{u}) \log f_{\mathbf{x}}(\mathbf{u}) d\mathbf{u}. \tag{10}$$

Differential entropy is invariant only to orthogonal transformations, and it is also upper bounded. Given a variance σ^2, the distribution that has the maximum differential entropy is the Gaussian distribution of variance σ^2, and the value of Eq. 9 is then

$$h(x) = \frac{1}{2}(1 + \log(2\pi\sigma^2)). \tag{11}$$

One useful measure related to the entropy is the average mutual information that is, in a two variables case, the amount of information conveyed by an occurrence of one variable about the other. The average mutual information is formally defined based on the differential entropy as

$$I(x, y) = h(x) + h(y) - h(x, y) \tag{12}$$

where $h(x, y)$ is the joint entropy of the variables. The mutual information is symmetric and always nonnegative.

As seen before, the invariance property of entropy is lost when going to the continuous case. To overcome this drawback, it is useful to introduce negentropy, the difference between the entropy of a random variable x and a Gaussian variable x_G with the same variance as x.

$$J(x) = h(x_G) - h(x) \tag{13}$$

In the case of a multidimensional random variable $\mathbf{x}$, to which is associated a Gaussian variable $\mathbf{x}_G$ with the same covariance matrix as $\mathbf{x}$, negentropy is

$$J(\mathbf{x}) = h(\mathbf{x}_G) - h(\mathbf{x}) = \int f_{\mathbf{x}}(\mathbf{u}) \log \frac{f_{\mathbf{x}}(\mathbf{u})}{f_{\mathbf{x}_G}(\mathbf{u})} d\mathbf{u}. \tag{14}$$

The first form indicates that negentropy is a kind of distance from normality. The second form, written like a Kullback divergence, tells which distance is measured.

Negentropy is always nonnegative, and is invariant to linear invertible transformations. It will be seen that negentropy as a distance from the subspace of Gaussian variables is a good candidate for optimizing independence.

Now using these definitions, the independence condition can be expressed in terms of information criteria. In case of a random vector, mutual information becomes

$$I(\mathbf{x}) = \int f_{\mathbf{x}}(\mathbf{x}) \log \frac{f_{\mathbf{x}}(\mathbf{x})}{\prod f_{x_i}(x_i)} d\mathbf{x}, \tag{15}$$

which has the form of a Kullback divergence between both terms of Eq. 3. This means that mutual information is a measure of independence. It is zero only when the components of $\mathbf{x}$ are independent, and is strictly positive otherwise.

Comparing Eq. 14 and Eq. 15, it comes out that if a Gaussian multivariate is a reasonable approximation to the product of the marginal densities and if Eq. 14 is easier to compute (which is likely to be the case), then negentropy is an interesting method for estimating independence.

Negentropy, as well as Kullback divergence, is difficult to calculate and requires an estimate of the probability density function (pdf).

2.3 Moments and cumulants

Instead of defining the properties on the distributions, which is a generative way of describing independence, we could use some descriptive statistics. One class of such statistics is the moments. The rth moment about the origin of the random variable x, written μ'_r is defined as

$$\mu'_r(x) = \mathrm{E}\{x^r\}, \tag{16}$$

whereas the rth centered moment μ_r is defined as

$$\mu_r(x) = \mathrm{E}\{[x - \mathrm{E}\{x\}]^r\}. \tag{17}$$

To provide a more general framework, another way of calculating the moments is

$$\mu'_r(x) = \left.\frac{\partial^r M_x(t)}{\partial t^r}\right|_{t=0} \tag{18}$$

$$M_x(t) = \mathrm{E}\{e^{tx}\} \tag{19}$$

where $M_x(t)$ is called the moment generating function. Moments are used in classical statistics. Especially the moments up to the second order form part of the basic tools, since expectation is the first moment about the origin, and variance the second centered moment.

Recently, it has been noted that moments are not the best possible tools when dealing with higher-order statistics. Another class of descriptive statistics has emerged, called cumulants [26]. Cumulants are defined in the following way, first for a multidimensional variable $\mathbf{x} = [x_1, x_2, ..., x_n]^T$, then for a scalar variable x

$$\mathrm{cum}(x_1, x_2, ..., x_n) = \left.\frac{1}{k!}\frac{\partial}{\partial v_1}\cdots\frac{\partial}{\partial v_n} K_{\mathbf{x}}(v_1, ..., v_n)\right|_{v_1=0,\dots,v_n=0}, \tag{20}$$

$$K_{\mathbf{x}}(v_1, ..., v_n) = \ln\left(\mathrm{E}\left\{e^{\sum_i v_i x_i}\right\}\right), \tag{21}$$

$$\kappa_r(x) = \mathrm{cum}(\overbrace{x, ..., x}^{r \text{ times}}). \tag{22}$$

where $K_{\mathbf{x}}(\mathbf{v})$ is called the cumulant generating function. Note that cumulants are the coefficients of the Taylor series expansion of $K_{\mathbf{x}}(\mathbf{v})$.

An important property of cumulants is that they are linear in each entry, meaning

$$\begin{aligned} \mathrm{cum}(x_1,...,x_i+y,...,x_n) &= \mathrm{cum}(x_1,...,x_i,...,x_n) \\ &\quad +\mathrm{cum}(x_1,...,y,...,x_n) \qquad (23) \\ \mathrm{cum}(x_1,...,\alpha x_i,...,x_n) &= \alpha\cdot\mathrm{cum}(x_1,...,x_i,...,x_n), \qquad (24) \end{aligned}$$

and that cumulants of order more than two are zero for Gaussian distributed variables. Approximation of negentropy have been derived through the cumulants.

Now inside this frame independence can be described with the moments or with the cumulants. Let x and y be independent random variables. We have

$$\begin{aligned} \mu_{r,q}(x,y) &= \mu_r(x)\mu_q(y) \qquad (25) \\ \mathrm{cum}(x,y) &= 0 \qquad (26) \\ \kappa_r(x+y) &= \kappa_r(x)+\kappa_r(y) \qquad (27) \end{aligned}$$

Comparing Eq. 19 and Eq. 21 it becomes clearer why the properties of the cumulants are additive and not multiplicative, which makes them more suitable as operators.

Note that for characterizing completely statistical independence using cumulants or moments, one should know all of them (infinitely many).

3 The ICA problem

3.1 Description

Now begins the description of the special field of this thesis. ICA is by nature a mathematical tool. Its origins are probably an extension of the idea of PCA. ICA research started mainly from the Blind Source Separation problem in digital signal processing and array processing, but nowadays also people working with Projection Pursuit, or even Data Mining are using ICA.

The problem of BSS is a perfect illustration for ICA and shall be used to expose the idea. In this situation, the data are collected by many sensors simultaneously. It is then known that the collected data are a transformation of original data that are unreachable (because of inaccessibility, perturbation by communication, lack of knowledge). With little a priori information about the transformation and the original data, it is possible to approximate both the transformation and the original data. This little a priori information consist mainly in general assumptions about the model, but not in a parameterization.

Among all its different versions, the ICA problem can be described by two main properties.

- There must be some *separated* variables, called sources, components, or latent variables, in the problem (let them be the *sources* hereafter). These sources are not visible at first sight.

- The meaning of *separated* in this case is statistical independence, as defined above.

ICA in its full generality amounts almost to blind model identification, which is a very hard problem, and the conditions for resolution are still not exactly known. As most of the research today, this work is restricted to linear ICA. This means that the transformation from the sources to the observed data is assumed to be linear. As a consequence, the inverse transformation for estimating the sources from the data is also linear. Nonlinear ICA is much more difficult to handle, and only some methods capable of dealing with small size problems have been proposed.

Mathematically, I will use the definition from [10] where the theoretical foundations of ICA are well introduced. Defining $\mathbf{x}, \mathbf{n} \in \mathbb{R}^N, \quad \mathbf{s} \in \mathbb{R}^M$, three random vectors with zero mean and finite covariance, with the components of $\mathbf{s}$ being independent, and $\mathbf{A}$ a rectangular constant $N \times M$ matrix with at least as many rows as columns ($M \leq N$), the assumption of the linear model becomes

$$\mathbf{x} = \mathbf{A}\mathbf{s} + \mathbf{n} \tag{28}$$

where $\mathbf{s}, \mathbf{x}, \mathbf{n}$ and $\mathbf{A}$ are respectively the sources, the data, the noise (unknown) and the mixing matrix.

Then the ICA problem is the following. Given T realizations of $\mathbf{x}$, it is desired to estimate both the matrix $\mathbf{A}$ and the corresponding realizations of $\mathbf{s}$. There are several limitations to solving this problem. Without any assumption about the noise, it is impossible to introduce it into ICA. Thus generally the solution will look like

$$\begin{aligned} \hat{\mathbf{s}} &= \mathbf{B}\mathbf{x} \quad \text{or} & (29)\\ \mathbf{x} &= \hat{\mathbf{A}}\hat{\mathbf{s}}, & (30)\\ \hat{\mathbf{A}} &= \mathbf{B}^{-1}, & (31) \end{aligned}$$

where the noise has been embedded in the estimates of the sources.

Another limitation concerns the sizes of the different vectors. The size of $\mathbf{s}$, usually unknown, should not be greater than the size of the data $\mathbf{x}$. Otherwise the problem is under-determined, although a solution can be found in some special cases, for instance in the binary case [30]. If there are more sensors than sources, the problem is over-determined and the extra sources can be used for reducing noise [22].

There is a basic indeterminacy in Eq. 30 because it is possible to construct another solution $\hat{\mathbf{A}}'\hat{\mathbf{s}}'$ satisfying the same condition

$$\mathbf{x} = \hat{\mathbf{A}}\hat{\mathbf{s}} = \hat{\mathbf{A}}\mathbf{P}\mathbf{\Lambda}\mathbf{\Lambda}^{-1}\mathbf{P}^T\hat{\mathbf{s}} = \hat{\mathbf{A}}'\hat{\mathbf{s}}' \tag{32}$$

with any permutation matrix $\mathbf{P}$ and any diagonal scaling matrix $\mathbf{\Lambda}$. This appears quite natural, because the sources are unknown, and their ordering may only be arbitrary. The scale of the signal also can be modified by the mixing.

To conclude with the indeterminacy, the solution to an ICA problem is always defined up to permutation and scaling of each component.

3.2 ICA and PCA

The comparison between both methods is very enlightening because the difference is not in the form of the process but in what is looked for.

PCA as a signal analysis technique provides an optimal representation of the data in the sense that data represented in the PCA coordinate system can be truncated with the least reconstruction error in the Euclidean space. The same data model (Eq. 28) as in ICA is used in PCA, but the assumptions are somewhat different. In PCA the columns of $\hat{\mathbf{A}}$ are forced to be mutually orthogonal, and the variances of the components of $\hat{\mathbf{s}}$ are maximized under this constraint. The book [12] provides a good survey of the PCA technique, and [29] presents neural implementations of both PCA and ICA.

PCA relies on the covariance matrix, or on the second-order statistics, which are easy to estimate from the data. ICA relies on independence, which is not as such a computable measure. Even using the mutual information (Eq. 15) does not make it really more tractable. There is space for more than one computational definition of independence, and that accounts for the present study at last.

What is optimized in PCA is the variance of the projected data along the PCA coordinates, that is

$$\max_{\mathbf{w}} \mathbf{w}^T \mathbf{C}_{xx} \mathbf{w} \tag{33}$$

where $\mathbf{w}$ is a PCA basis vector having unit length and $\mathbf{C}_{xx}$ the covariance matrix of the data. Clearly PCA describes the data only by covariance, thus as if they were Gaussian variables. In terms of moments, only the moments up to order two are taken into consideration. In terms of information theory, all the distributions are considered unstructured, since it was stated that with given variance, the Gaussian variable has the least structured distribution, as measured by differential entropy (Eq. 10). With ICA, also the high-order moments are taken into account, and with it the structure of the distribution. It should be recalled that a Gaussian distribution is defined completely by its first and second-order statistics (mean and covariance matrix), while other distributions may contain a lot of extra information in their higher-order statistics. What will be optimized in ICA is a kind of non-gaussianity, and each algorithm will have a slightly different version of the optimized function making a different approximation of independence.

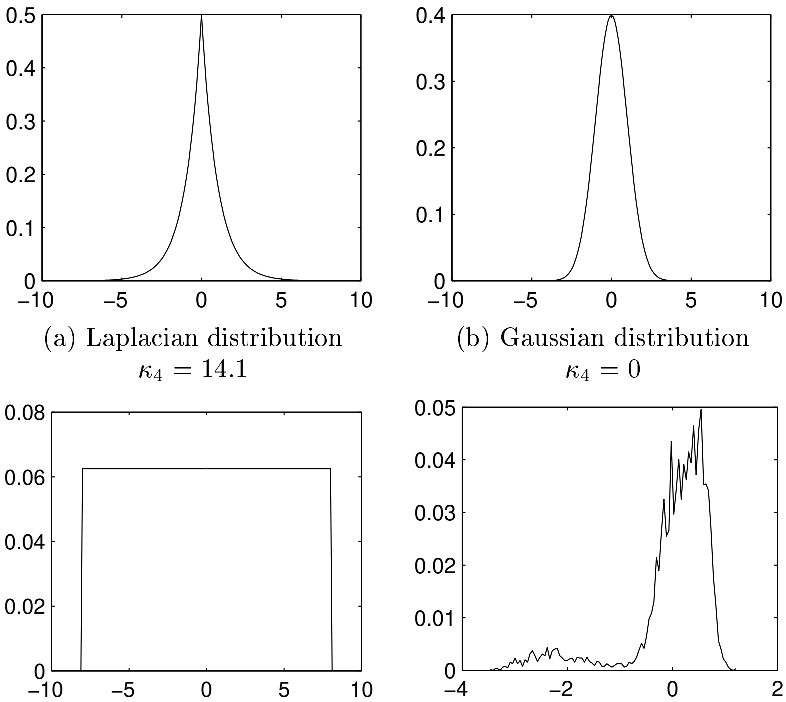
It is possible to view both PCA and ICA as data representation tools. Still, from PCA to ICA, the algorithm searches deeper for structure in the data and uses more information from the distribution, as can be seen from the number of moments involved in the computation of independence tests. This is indicated also by the fact that PCA algorithms are linear with respect to learning and linear to data processing, whereas ICA algorithms, also linear with respect to data processing, must be nonlinear in the learning phase.

The simplest way of looking closer into the structure of the distribution than PCA does is to consider statistics of order higher than two. For this purpose cumulants have been introduced before. The third and fourth order cumulants with zero lag, $\kappa_3(x)$ and $\kappa_4(x)$, are called *skewness* and *kurtosis*. They are defined following Eq. 22 for the random variable x as

$$\kappa_3(x) = \mathrm{E}\{x^3\} \tag{34}$$

$$\kappa_4(x) = \mathrm{E}\{x^4\} - 3\left(\mathrm{E}\{x^2\}\right)^2 . \tag{35}$$

Skewness is different from zero only when the distribution is asymmetric. This covers only a subset of the distributions, and it is not much used in practice.



(a) Laplacian distribution $\kappa_4 = 14.1$

(b) Gaussian distribution $\kappa_4 = 0$

(c) Uniform distribution $\kappa_4 = -1.2$

(d) Experimental distribution $\kappa_4 = 4.2$ $\kappa_3 = -2.1$

Figure 1: Partition of the distributions with normalized kurtosis

Kurtosis is the simplest measure of departure from Gaussian distribution. As such, it is used to split the distributions in two classes, supergaussian distributions whose kurtosis is positive, and subgaussian distributions whose kurtosis is negative. This distinction will be used throughout this work.

If normalized kurtosis is used, that is $\kappa_4(x)/\left(\mathrm{E}\{x^2\}\right)^2$, in addition to the usual properties of the cumulants, normalized kurtosis can take values only into $[-2, \infty)$. Thus the range of values for subgaussian sources is $[-2, 0)$, while it is $(0, \infty)$ for supergaussian sources, therefore it is difficult to directly compare the nongaussianity of supergaussian and subgaussian variables with each other.

When looking at the distribution of a random variable, high kurtosis is usually a mark of a peaked distribution, for instance Laplacian, whereas low kurtosis corresponds to flat or bimodal distributions, for instance uniform. Examples are given in Figure 1.

ICA can be approximately computed using cumulants. A general reference is Comon [10].

3.3 Prewhitening

As ICA deals with high-order statistics, it is meaningful to preprocess the data so that all the first and second-order moments are normalized. This means that the solution is split in two parts, the first one dealing with dependencies in the first two moments, and the second one dealing with other dependencies. This procedure is called whitening. The algorithms below need to receive the data more or less preprocessed, and I shall now describe each stage of the preprocessing.

I shall give here the data version of the equations. Let $\mathbf{x}$ be a raw data column vector. We have the following three stages:

$$\begin{aligned} \mathbf{x}' &= \mathbf{x} - \mathrm{E}\{\mathbf{x}\} & (36)\\ \mathbf{v} &= \mathrm{E}\{\mathbf{x}'\mathbf{x}'^T\}^{-1/2}\mathbf{x}' & (37)\\ \mathbf{v}' &= \mathbf{R}\mathbf{v} & (38) \end{aligned}$$

The first equation sets the first order moment to zero. The second equation sets the covariance matrix to a unit matrix, and the third equation performs additional PCA, with $\mathbf{R}$ being the PCA orthogonal rotation matrix. Usually one rather gets $\mathbf{v}'$ with

$$\begin{aligned} \mathbf{v}' &= \mathbf{D}^{-1/2}\mathbf{E}^T\mathbf{x}' \\ \mathbf{EDE}^T &= \mathrm{E}\{\mathbf{x}'\mathbf{x}'^T\} & (39) \end{aligned}$$

where $\mathbf{EDE}^T$ is the eigenvector decomposition of the covariance matrix of $\mathbf{x}$, implying that $\mathbf{D}$ is a diagonal matrix and $\mathbf{E}$ an orthogonal matrix.

There are neural ways of computing the PCA matrices as on-line estimates [23].

Every algorithm must be given the data in the right form, as will be explained in each section. When prewhitening is performed, the separating matrix is split into two components

$$\mathbf{B} = \mathbf{WQ} \tag{40}$$

where $\mathbf{Q}$ is the whitening matrix and $\mathbf{W}$ the separating matrix in the whitened space. Matrix $\mathbf{W}$ is then orthogonal as a consequence of

$$\hat{\mathbf{s}} = \mathbf{Bx} = \mathbf{WQx} = \mathbf{Wv}. \tag{41}$$

If $\mathbf{v}$ and $\hat{\mathbf{s}}$ are uncorrelated and the v_i and $\hat{s}_i$ have unit variances, then $\mathbf{W}$ is orthogonal. In case of finite data sample, this is not always exact.

4 Algorithms for ICA

In this section all the algorithms included in this study are described. I tried to unify the notations and the presentation. Each time that the update rule is given as a $\Delta\mathbf{W}$, it is meant that the update equation is

$$\mathbf{W} = \mathbf{W} + \Delta\mathbf{W} \tag{42}$$

and the same for $\mathbf{B}$.

Most of the algorithms are rather family of algorithms parameterized by the nonlinear function actually used. At the core of each algorithm, like a nut in a shell, there is an update rule and its associated minimized criterion. This part is described here, whereas description for the 'shell', namely the control loops, that take full benefit of the update rules, is deferred to the next section.

4.1 Fixed Point (FP) algorithm

General description

The fixed point algorithm for performing ICA is initially described in [19]. It derives from the entropy optimization methods. Thanks to a new approach to minimization/maximization, the algorithm converges faster than any other known algorithm. Convergence is at least quadratic.

For a complete description of the algorithm, see [16].

Fixed point algorithm is based on the mutual information. We have seen above (Eq. 15) that this measure is a kind of distance of independence, and minimizing mutual information leads to ICA solution. For the purpose of this algorithm, mutual information is rewritten as

$$I(\mathbf{y}) = J(\mathbf{y}) - \sum_i J(y_i) + \frac{1}{2}\log\frac{\prod c_{ii}}{\det\ \mathbf{C}_{yy}} \tag{43}$$

where $\mathbf{y} = \mathbf{W}\mathbf{x}$, $\mathbf{C}_{yy}$ is the correlation matrix, and c_{ii} is the i^{th} diagonal element of the correlation matrix. The last term is zero because the y_i are supposed to be uncorrelated. The first term is constant for a given problem, because of the invariance of negentropy. The problem is now reduced to separately maximizing the negentropy of each component.

Estimation of negentropy is a delicate problem. The papers [10], [17] and [18] have addressed this problem. For the general version of FP algorithm, the approximation was based on a maximum entropy principle.

The algorithm works with whitened data (in fact with PCA represented data in my version), although a version with non-whitened data exists.

Criterion

The maximization is performed over the following index

$$J_G(\mathbf{w}) = \left[\mathrm{E}\left\{G(\mathbf{w}^T\mathbf{v})\right\} - \mathrm{E}\left\{G(\nu)\right\}\right]^2 \tag{44}$$

to find one independent component, with ν, a standard Gaussian variable, and G, the one-unit contrast function.

Update rule

Update rule for the generic algorithm is

$$\begin{aligned} \mathbf{w}^* &= \mathrm{E}\{\mathbf{v}g(\mathbf{w}^T\mathbf{v})\} - \mathrm{E}\{g'(\mathbf{w}^T\mathbf{v})\}\mathbf{w} \\ \mathbf{w} &= \mathbf{w}^*/\|\mathbf{w}^*\| \end{aligned} \tag{45}$$

to extract one component.

There is a symmetric version of the FP algorithm, whose update rule is

$$\begin{aligned} \mathbf{W}^* &= \mathrm{E}\left\{\mathbf{g}(\mathbf{W}\mathbf{v})\mathbf{v}^T\right\} - \mathrm{Diag}\left(\mathrm{E}\{\mathbf{g}'(\mathbf{W}\mathbf{v})\}\right)\mathbf{W} \\ \mathbf{W} &= (\mathbf{W}^*\mathbf{W}^{*T})^{-1/2}\mathbf{W}^* \end{aligned} \tag{46}$$

where $\mathrm{Diag}(\mathbf{v})$ is a diagonal matrix with $\mathrm{Diag}_{ii}(\mathbf{v}) = v_i$.

Parameters

The nonlinearity I used was

$$g(y) = \begin{cases} y^3 \\ tanh(y). \end{cases} \tag{47}$$

The choice is free except that the symmetric algorithm with *tanh* nonlinearity does not separate supergaussian signals. Otherwise the choice can be devoted to other criteria, for instance the cubic nonlinearity is faster, whereas *tanh* nonlinearity is more stable. These questions are addressed in [16].

Other characteristics

This algorithm has no learning rate in its basic form, and the parameters allow a large range of values. The initial condition can be an orthogonalized random matrix.

In deflation version (Eq 45) the algorithm needs orthogonalization. For a complete description see Algorithm 5, but the idea is to find a component, then to project the data on the subspace orthogonal to the found component. That is if the components found so far are put in a matrix $\mathbf{W}$

$$\mathbf{W} = \begin{pmatrix} \mathbf{w}_1 \\ \vdots \\ \mathbf{w}_r \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{pmatrix} \tag{48}$$

after having found the rth component, then before starting to search for a new ICA component the data is projected with

$$\mathbf{v} = \mathbf{v} - \mathbf{W}^T \mathbf{W} \mathbf{v}. \tag{49}$$

The fixed point algorithm differs also from the others by being able to extract one component at a time, which should be useful especially in projection pursuit.

I will abbreviate this algorithm with **FP**. I used 4 versions of the algorithm, that is with deflation or symmetrical (**FPsym**) approach, and with cubic or tanh non-linearity (**FPtanh**). The algorithm separates every mixture of sub- and supergaussian sources, except for the deflation approach with tanh non-linearity, which does not separate supergaussian components without modification.

The deflation algorithm ends up with accumulation errors, because every error that is made in the extraction of the first components cannot be corrected in the following components.

4.2 NPCA-RLS algorithm

General description

The Nonlinear Principal Component Analysis by Recursive Least Squares algorithm for ICA was introduced in [24]. It uses an approximate recursive least-squares approach. The NPCA-RLS algorithm belongs to the adaptive family of algorithms, but it is faster than most other gradient descent algorithms.

The RLS approach is a classical approach in signal processing, belonging to the adaptive filtering techniques [13, ch. 13]. Yang [32] has applied the RLS technique to subspace tracking.

Criterion

The minimized criterion function for the NPCA algorithm [28] is

$$J_1(\mathbf{W}) = \mathrm{E}\{\|\mathbf{v} - \mathbf{W}\mathbf{g}(\mathbf{W}^T\mathbf{v})\|^2\} \tag{50}$$

which becomes in the NPCA-RLS

$$J_2(\mathbf{W}(t)) = \sum_{i=1}^{t} \beta^{t-i}\|\mathbf{v}(i) - \mathbf{W}(i)\mathbf{g}(\mathbf{W}^T(i-1)\mathbf{v}(i))\|^2 \tag{51}$$

if the β is different from 1.

In the case where data are whitened, one can rewrite J_1 as [25]

$$J_1(\mathbf{W}) = \sum_{i=1}^{m} \mathrm{E}\{[y_i - g(y_i)]^2\} \tag{52}$$

where $\mathbf{y} = \mathbf{W}^T\mathbf{v}$, because $\mathbf{W}$ can be assumed to be orthogonal.

Depending on the nonlinearity, the $[y_i - g(y_i)]$ term gives different estimates of output entropy, which accounts for the algorithm finding some interesting projection, while the orthogonality of $\mathbf{W}$, due to the whitening process, ensures that every output is different (although $\mathbf{W}$ may not be exactly orthogonal especially if the sources are correlated).

These criteria are closely related to the minimization of mutual information except that the sign is reversed. As a consequence, the nonlinearities to be used are not the same as for some other algorithms. This matter is discussed more in detail in [25].

Update rule

The algorithm estimates automatically an optimal learning rate, and the only parameter to be supplied, β, defines the rate at which old data influence over $\mathbf{W}$ fades out (called the forgetting parameter).

The update rule is [24]

$$\begin{aligned}
\mathbf{z}(t) &= \mathbf{g}(\mathbf{W}^T(t-1)\mathbf{v}(t)) = \mathbf{g}(\mathbf{y}(t)), \\
\mathbf{h}(t) &= \mathbf{P}(t-1)\mathbf{z}(t), \\
\mathbf{m}(t) &= \mathbf{h}(t)/(\beta + \mathbf{z}^T(t)\mathbf{h}(t)), \\
\mathbf{P}(t) &= \frac{1}{\beta}\mathrm{Tri}\left[\mathbf{P}(t-1) - \mathbf{m}(t)\mathbf{h}^T(t)\right], \\
\mathbf{e}(t) &= \mathbf{v}(t) - \mathbf{W}(t-1)\mathbf{z}(t), \\
\mathbf{W}(t) &= \mathbf{W}(t-1) + \mathbf{e}(t)\mathbf{m}^T(t)
\end{aligned} \tag{53}$$

where $\mathbf{v}(t)$ is the data vector in the PCA coordinates.

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, making thus the matrix $\mathbf{P}(t)$ symmetric.

Parameters

There is only one parameter in the NPCA-RLS algorithm, namely β. I set it to 0.999 (slow forgetting), since the goal was to get a more accurate solution. A β of 1 would have retained also the bad estimates of the first steps of convergence, resulting in a slower convergence.

The nonlinearity, here the **g** function, depends on the type of the sources. I used for all the experiments

$$g(y) = \begin{cases} y - tanh(y), & \text{supergaussian components} \\ tanh(y), & \text{subgaussian components.} \end{cases} \tag{54}$$

The $y - tanh(y)$ form is a stabilized version of the plain $-tanh(y)$, required by the fact that the algorithm is not normalized.

Other characteristics

The algorithm starts with $\mathbf{P}(0) = \mathbf{W}(0) = \mathbf{I}$.

The algorithm in this version requires whitened data, but there exists also a version which deals with non whitened data [25]. Due to some approximations, the NPCA-RLS algorithm is somewhat less precise than other gradient descent algorithms.

The algorithm in the form I have it can separate either sub- or supergaussian sources, but not both. This drawback can be overcome by an on-line estimate of kurtosis, the like is done in ExtBS. The version here is the symmetric version of NPCA-RLS. A sequential (deflatory) version exists, too [24]. It is simpler, requiring less computation per iteration, but is less accurate and converges slower.

4.3 Bell & Sejnowski (BS) algorithm

General description

The BS algorithm, proposed by Bell and Sejnowski [2], is also a gradient algorithm, based on maximization of information. In perfect cases, it does provide the best estimate to ICA components. The article presenting this algorithm appeared in 1995 [2]. The strength of this algorithm comes from its direct relationship to information theory, although after recent theoretical work it seems that most ICA algorithms can be related to the same principles (see [25] and [7]).

The algorithm is derived through an information maximization principle, applied here between the inputs and the nonlinear outputs. Given the form of the joint entropy

$$H(y_1, y_2) = H(y_1) + H(y_2) - I(y_1, y_2), \tag{55}$$

here for two variables $\mathbf{y} = \mathbf{g}(\mathbf{Bx})$, it is clear that maximizing the joint entropy of the outputs amounts to minimizing mutual information $I(y_1, y_2)$, unless it is more interesting to maximize the individual entropies than to reduce the mutual information. This is the point where the choice of the nonlinear function plays its critical role.

The basic idea of the information maximization is to match the slope of the nonlinear transfer function with the input probability density function. That is

$$y = g(x, \theta) \simeq \int_{-\infty}^{x} f_x(t)dt. \tag{56}$$

In case of a perfect matching, the resulting $f_y(y)$ looks like an uniform variable, whose entropy is large. If this is not possible because the shapes are different, the best solution found in some case is to mix the input distributions so that the resulting mix matches the slope of the transfer function better than a single input distribution. In this case the algorithm does not converge, and separation is not achieved.

Criterion

The algorithm is a stochastic gradient ascent that maximizes the joint entropy (Eq. 10).

Update rule

In its original form, the update rule is

$$\begin{aligned} \Delta\mathbf{B} &= \lambda\left[\left[\mathbf{B}^T\right]^{-1} + (\mathbf{1} - 2\mathbf{g}(\mathbf{Bx} + \mathbf{b}_0))\mathbf{x}^T\right] \\ \Delta\mathbf{b}_0 &= \lambda[\mathbf{1} - 2\mathbf{g}(\mathbf{Bx} + \mathbf{b}_0)]. \end{aligned} \tag{57}$$

The $\mathbf{b}_0$ coefficient is not interesting because the mean is removed from the samples, so I will not use it.

Nobody uses the original version of the algorithm now, but instead the version modified by Amari & Cichocki, see below. The algorithm processes the data in a whitened form given by Eq. 37, although it could process the raw data, but I did not use this form of the extended algorithm. The equation becomes

$$\Delta\mathbf{W} = \lambda\left[\mathbf{I} - \mathbf{g}(\mathbf{Wv})(\mathbf{Wv})^T\right]\mathbf{W} \tag{58}$$

Parameters

The nonlinearity used in the original article is

$$g(y) = \frac{1}{1 + e^{-y}} \tag{59}$$

and in the extended version, it is

$$g(y) = y \pm tanh(y) \tag{60}$$

where the sign is that of the estimated kurtosis of the signal.

It may seem that nonlinearity in Eq. 59 is different from the others, but it is just a sigmoidal function. Replacing $tanh(y) = 1 - \frac{2}{1+e^{2y}}$ in Eq. 58, one gets the same nonlinear part as Eq. 57, up to a coefficient.

Other characteristics

The algorithm can take as a starting point the identity matrix or a random matrix.

I have two programs for this algorithm, one with the basic algorithm provided on the Web and no optimization (**BS**), which is introduced in [2], and the other, optimized, developed by Makeig et alia for EEG analysis purposes (**ExtBS**).

The optimized version contains on-line kurtosis estimation to be able to separate subgaussian sources as well, and usual gradient-descent optimizations like momentum or simulated annealing.

The basic algorithm separates only supergaussian sources. The optimized version too, when the extended switch is off. With extended switch on, it separates everything, as mentioned above. Whenever there were subgaussian components in the mixtures, I turned on the extended feature, which is computationally demanding.

4.4 Natural gradient algorithm (ACY)

General description

Amari, Cichocki and Yang [1] proposed an improvement to the BS algorithm by using the natural gradient instead of the stochastic gradient. This method speeds up the convergence dramatically and is computationally simpler. Actually this algorithm was proposed earlier by Cichocki and his coauthors, on heuristic grounds. This approach is described in full detail in the later journal paper [9].

Update rule

The update rule in Eq. 57 becomes

$$\Delta\mathbf{B} = \lambda\left[\mathbf{I} - \mathbf{g}(\mathbf{Bx})(\mathbf{Bx})^T\right]\mathbf{B} \tag{61}$$

Parameters

In the original article [1], the authors proposed the theoretical contrast function

$$g(y) = \frac{3}{4}y^{11} + \frac{25}{4}y^9 - \frac{14}{3}y^7 - \frac{47}{4}y^5 + \frac{29}{4}y^3 \tag{62}$$

that does not work in practical problems because of too high powers of y, causing the nonlinearity in practice to depend only on few large values.

Two functions were used instead, proposed in [22], depending on the nature of the sources,

$$g(y) = \begin{cases} tanh(y), & \text{supergaussian components} \\ y^3, & \text{subgaussian components.} \end{cases} \tag{63}$$

Other characteristics

In the form I have it, the algorithm can not separate both type of components at the same time. The initial weights are the identity matrix, or a random matrix.

To be exact, I should merely not speak about Bell's and Sejnowski's algorithm and Amari, Cichocki and Yang algorithm, but rather of two version of the BS algorithm modified by Amari, Cichocki and Yang, differing only by their contrast function.

Anyway, I will call this algorithm **ACY**, or **WACY** if the entries are whitened.

4.5 EASI algorithm

General description

The Equivariant Adaptive Source separation algorithm was introduced by Laheld and Cardoso in 1994 at a conference, and then fully described in [8].

The idea of equivariance comes from the group theory. The main principle is that " When a transformation on the data is equivalent to a transformation of the parameter, the notion of equivariance is of relevance." In the case of ICA, the transformation is an invertible linear transformation. Since both the mixing transformation and the separating transformation belong to the same group, the properties of the parameter can be seen through the whole process. In a sense, the equivariance property makes the mixing transparent.

Criterion

The minimized criterion consists of two parts, one part for the whitening matrix estimate, based on the Kullback divergence between the output correlation matrix and the identity matrix,

$$K(\mathbf{C}_{yy}) = \frac{1}{2}\left(\mathrm{Trace}(\mathbf{C}_{yy}) - \log\det(\mathbf{C}_{yy}) - n\right) \tag{64}$$

where $\mathbf{y} = \mathbf{B}\mathbf{x}$.

Computing the criterion with respect to the whitening yields

$$J_4(\mathbf{Q}) = K(\mathbf{Q}\mathbf{C}_{xx}\mathbf{Q}^T) \tag{65}$$

which results in the $\mathbf{y}\mathbf{y}^T - \mathbf{I}$ in the update rule.

The second part performs ICA by minimizing the following function.

$$J_5(\mathbf{W}) = \mathrm{E}\{G(\mathbf{W}\mathbf{x})\} \tag{66}$$

with the constraint that $\mathbf{W}$ should be orthogonal. This constraint is enforced by projecting the gradient obtained, for instance, in Eq. 61 into a skew-symmetric matrix with formula $(M - M^T)/2$, giving the second part of the update rule.

Update rule

Finally, the update rule is

$$\Delta\mathbf{B} = -\lambda[\mathbf{y}\mathbf{y}^T - \mathbf{I} + \mathbf{g}(\mathbf{y})\mathbf{y}^T - \mathbf{y}\mathbf{g}(\mathbf{y})^T]\mathbf{B} \tag{67}$$

in its basic form and

$$\Delta\mathbf{B} = -\lambda\left[\frac{\mathbf{y}\mathbf{y}^T - \mathbf{I}}{1 + \lambda\mathbf{y}^T\mathbf{y}} + \frac{\mathbf{g}(\mathbf{y})\mathbf{y}^T - \mathbf{y}\mathbf{g}(\mathbf{y})^T}{1 + \lambda|\mathbf{y}^T\mathbf{g}(\mathbf{y})|}\right]\mathbf{B} \tag{68}$$

in the normalized form, where $\mathbf{y} = \mathbf{B}\mathbf{x}$.

Parameters

Nonlinearity is chosen according to the kurtosis of the sources,

$$g(y) = \begin{cases} tanh(y) - y, & \text{supergaussian components} \\ y^3, & \text{subgaussian components.} \end{cases} \tag{69}$$

I tried also $tanh(y)$ in the supergaussian case, that works the same. Note that with $tanh(y)$, Eq. 69 is the same as Eq. 63, whereas Eq. 54 is reversed.

Other characteristics

The algorithm starts with random weights. Simulated annealing was tried but not used, although it should give good results after tuning.

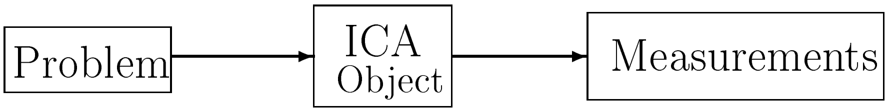


Figure 2: Coarse view of a testbed system

Part II
Experiments

5 Experimental setup

As mentioned before, the goal of this study is to gain some insight into the differences between algorithms. For achieving this, it is meaningful to study the algorithms in two types of situations.

- A typical BSS situation, to collect basic accuracy results and computational requirements. In this part ICA is performed for artificial, one-dimensional signals that are mixed through a matrix to provide the 'perceived' data. The requirements of ICA are almost exactly fulfilled, and the quality of the result can be directly assessed by index values. In this kind of perfect case, I will look at the precision of the solutions, the requirements of computing power, and the dependency on the parameters.
- A projection pursuit setting, where one tries to find interesting projections from classical data sets. In that part real data is fed into the algorithm, corresponding to various dimensions collected on a system. The idea is to look at the data through the independent components, evaluating whether the projections are meaningful with respect to some criterion that was not fed into the algorithm, for instance classification. In this case the presence of real independent components is speculative, and it is interesting to see what is happening if some of the hypotheses do not hold.

To perform these experiments, it was necessary to design a framework where the algorithms could be easily manipulated.

Here I describe every details regarding the implementation of the algorithms and the testbed used for the experiments. Two of the parts in Fig. 2 shall be described here, as well as the embedding of the algorithms in a general framework.

5.1 Programming the algorithms

Programming an algorithm is the act of bringing a theoretical algorithm into a form executable by the machine. Every choice deferred to the execution is a parameter. Every choice that is made inside the program is said to be 'hard coded'. One delicate point is to characterize the blurred frontier between parameters and programs, since one can always move the level at which a choice is made, allowing a caller program to set more or less parameters. In a strict sense, there are no parameters, since everything must be put into the machine,

Name	Type	Origin
FP	represent.	CIS laboratory
NPCA-RLS	original	CIS laboratory
BS	original	http://www.cnl.salk.edu/~tony/ica.html
ExtBS	represent.	http://www.cnl.salk.edu/ ~scott/ica-download-form.html
(W)ACY	original	personal version
EASI	original	http://sig.enst.fr/~cardoso/guidesepsou.html

Table 1: Code used for the tests

in a machine readable form, in order to be executed (that is the machine does not read your thoughts yet).

I propose to make a distinction between two types of codes, original and representative. For me an original code is programmed by its authors, or is a direct implementation of the original algorithm if the choices to be made are not too important. A representative code is a program that can be used for real problems, meaning that it addresses questions of capacity, efficiency and finer parameterization, and that it tries to get the best out of the algorithm. In this sense, BS is an original code, ExtBS is a representative code, and the FP code is a representative code. The ACY is the only one I programmed by myself.

5.1.1 Object oriented framework

The new object-oriented (OO) features of Matlab 5 were used to unify the code of the different algorithms. There each algorithm is a child object inheriting from a basic 'ICA algorithm' object, as can be seen from Figure 3.

This approach has the following advantages:

- The common properties of all algorithms are programmed only once. The parent class provides a common framework for the properties of the algorithms. The properties have the same name. It is especially useful to view an ICA algorithm as a black box, with some given inputs and outputs.
- The parameters that are specific to an algorithm (like learning rates, thresholds, etc.) are hidden inside the object. This reduces the number of global variables and increases the readability of the program.
- Each algorithm as an object is far more tractable for testing purposes.

On the other hand, OO approach suffers from the following disadvantages:

- Objects in Matlab require version 5.
- Some features of OO-programming in Matlab are rather tedious, especially assigning values to object slots inside a procedure. It reduces the readability and stability of the code.
- Generally speaking, the addition of OO-programming features into Matlab has not yet been finished.
- Bringing a new algorithm into the OO-framework still requires understanding of the structure of the parent object.

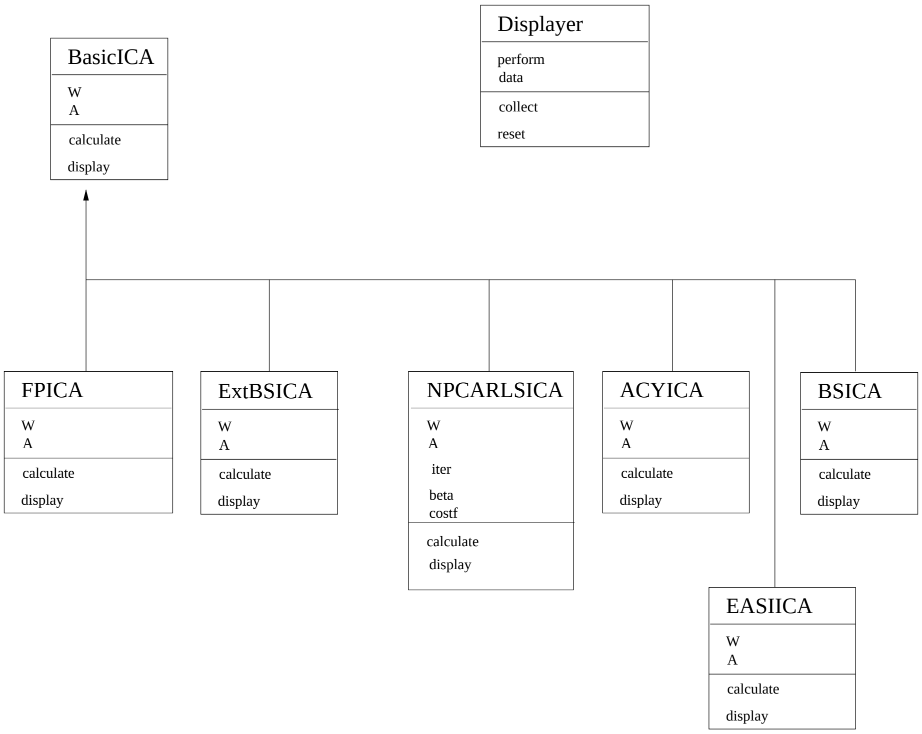


Figure 3: Inheritance diagram of the system. Not all parameters are reported on the scheme.

In the case of the test framework, the OO solution was a good one. The way it was implemented is far from serious software development, but it was not the central point in this kind of exploratory study.

5.1.2 Control loops

Around the basic parts of an algorithm, each program must set up the variables, provide the right number of iterations, test for terminating conditions and make some final calculations. Everything here is described in pieces, and each piece has to be embedded into the higher level piece.

The description here is not intended to be complete, but to provide the outlines of each algorithm. Some differences in implementation may bring great improvements to the final results, and the goal is to be able to locate where the improvement was made. As exposed before, the global framework for the experiments is an object oriented one. The pieces of code here represent rather methods of the ICA objects than real functions. Anyway the difference is small.

In this part all names are variables or functions and especially X is the matrix of the data with each data vector as a column of the matrix. The basic scheme of an ICA program is Alg. 1.

```
funct B = ica(X, ...)
  if prewhitening
     Q := whiten(X)
     X := QX
  fi
  W := properIca(X)
  if prewhitening
     B := WQ
  elsf
       B := W
  fi
.
```

Algorithm 1: Basic scheme for ICA calculation

In the case of gradient algorithms, the basic algorithm is Alg. 2, which uses one data vector at a time. Alg. 2 is to be used as *properIca* in Alg. 1. The NPCA-RLS and EASI algorithm are programmed following this scheme.

```
funct W = gradientIca(X, λ, nbSweeps)
  for i := 1 to nbSweeps do
      for j := 1 to nbSamples do
          ΔW := update(x_j, W)
          W := W + λΔW
      od
  od
.
```

Algorithm 2: Simple scheme for gradient algorithm

A method to stabilize the algorithm by computing the gradient in batches is

Parameter	NPCA-RLS	ACY	BS	EASI
λ		10^{-4}	10^{-4}	10^{-3}
β	0.99			
$batchSize$		30	30	
$nbSweeps$	1-30	200-400	200-400	1-10

Table 2: Values for the algorithms without exit test

proposed in [2]. This is the way BS and (W)ACY algorithms are implemented, like in Alg. 3.

```
funct W = gradBatchIca(X, λ, nbSweeps, batchSize)
  X = randomPermutation(X) /* scramble the input vectors to improve stationarity */
  /* not used in the ACY algorithm */
  for i := 1 to nbSweeps do
      for j := 1 to nbSamples step batchSize do
          ΔW := Σ_{k=0}^{batchSize−1} update(x_{j+k}, W)
          W := W + λΔW
      od
  od
.
```

Algorithm 3: The structure of batch gradient computation

It is tedious to take care of the exact number of iterations, especially because this changes with every problem and even with every run of the same program, and because it is not necessary since output conditions based on the changes of the separating matrix generally work well. All programs designed for real use should include some form of termination condition. The basic form of a controlled loop is exposed in Alg. 4, describing quite accurately the ExtBS version.

There is no obstacle in changing an update rule to another one, and the algorithms presented here are only shells which allows the update rules to work properly. Tuning is necessary in most cases, though.

The fixed point algorithm has a different structure in its deflation version because it extracts the components one by one. The algorithm is completely described in [14], and Alg. 5 is taken from there.

All the parameters that tune the algorithms are collected in Table 2 and Table 3. Some of them have ranges when modification was needed for each particular problem. Others were modified for special situations, as defined in the text.

5.2 Environment

The test environment of this study consisted mainly of the algorithm module. The two other parts are the problem given to the algorithm as input and the collection and analysis of the results. For each data set tested, there are particular measurements. First I describe the common features across all data sets, then I dedicate a small section for each case separately.

```
funct W = controlledIca(X, r_max, ε, ...)
  W := rand() /* Initialization */
  round := 0
  while r < r_max do
        X := randomPermute(X) /* Ensures stationarity */
        for j := 1 to nbSamples step batchSize do /* cf. gradBatchIca */
            ΔW := Σ_{k=0}^{batchSize-1} update(x_{j+k}, W, k)
            W := W + λΔW
            if W > max_W
              blowup := true
              break
            fi
            if extended
              k := estimateKurtosis(WX)
            fi
        od
        if blowup
          λ := λ · restartFactor
          restart
        elsf
            ΔW_old := W − W_old
            W_old := W
            r := r + 1
            d := Vec(ΔW_old)
            c := d^T d /* c contains the norm of the update */
            dα := acos( d^T d_old / sqrt(c c_old) ) /* is the angle for annealing */
            if dα > α_anneal
              λ := λ · annealStep
              d_old := d, c_old := c
            fi
            if c < ε
              r := r_max
            elsif c > c_blowup
                 λ := λ · restartFactor
            fi
        fi
  od
.
```

Algorithm 4: General structure of a controlled ICA program : ExtBS. *Vec* is the operator that transform a m x n matrix into an mn vector by putting each column one after the other.

funct $W = fp(X, r_{max}, \epsilon)$
  /* Returns the separating matrix W. */
  $W := \emptyset$ /* Start with an empty matrix. */
  $i := 1$ /* The extracted source. */
  while $i \leq n$ **do** /* Extract all sources. */
      $w := rand()$
      $w := w - WW^T w$ /* Gram-Schmidt orthogonalization */
      $w := w/\|w\|$
      $r := 0$ /* Current iteration round. */
      while $r < r_{max}$ **do** /* Maximum number of iterations. */
          $w_{old} := w$
          $w^* := \mathrm{E}\{g(w^T X)X\} - \mathrm{E}\{g'(w^T X)\}w$
          $w^\dagger := w^* - WW^T w^*$
          $w := w^\dagger/\|w^\dagger\|$
          if $\|w - w_{old}\| < \epsilon \vee \|w + w_{old}\| < \epsilon$
            $i := i + 1$
            $W := [W\ w]$ /* Augment W with the found basis vector. */
            break /* Break from the while-loop. */
          fi
          $r := r + 1$
      od
  od
  $W := W^T$
.

Algorithm 5: Fixed point algorithm with orthogonalization.

Parameter	ExtBS	FP
r_{max}	512	1000
ϵ	10^{-6}	10^{-4}
$batchSize$	$\sqrt{n}/3$	
λ_{init}	$0.015/\log m$	
max_W	10^8	
$annealStep$	0.9	
$restartFactor$	0.9	
α_{anneal}	$60°$	
c_{blowup}	10000	

Table 3: Values for the parameters of the controlled algorithms. n is the number of samples, m is the number of channels.

5.2.1 The problem

By the problem, I design the set of signals which are introduced as input to the ICA algorithm. The algorithm does not see any processing stage before its inputs.

The basic parameter of the problem is the *number of channels.* Unless in very special cases, the number of channels sets an upper limit to the number of independent components one can extract from the data. The number of channels is also important in terms of efficiency, because it represents the size of the problem.

The *number of sources* is not a parameter, but a result in the general case. There are methods for approximating the number of sources, exposed in [22], but this problem will not be addressed.

The *number of samples* is the second important parameter. The availability of the samples depends strongly on the problem. The samples can be unique, continuously incoming, generated artificially. All these cases must be treated by different versions of the algorithms. The main distinction to be made here is between off-line and on-line algorithms. On-line algorithms can cope with continuous data while keeping the speed at an interactive level (say one second), off-line algorithms need some time to perform the calculation on a finite set of data, and cannot handle the incoming of new data after the calculation has begun. This distinction depends from the computing power at hand and from the size of the problem, the rate of data income and all the timed parameters. I will make the comparisons in the frame of off-line calculation.

A minimal amount of samples should be collected, otherwise the calculation will certainly give meaningless results. This is due to the adaptive nature of the algorithms, as well as to some stages of preprocessing, like the covariance matrix, required for the whitening of the data.

The third point here is the *nature of the sources.* Statistical independence is a very strong condition. In most problems, it is not fulfilled completely. Given the classification into super- and subgaussian sources, there are algorithms that can cope with one, or both or these types of sources.

Obviously in a real problem, there are only assumptions about the nature of the sources (otherwise separation is not 'blind').

5.2.2 The outcome

The solution given by an algorithm consists in a separating matrix $\mathbf{B}$, with as many columns as the input vectors and as many rows as the number of extracted components. Its inverse $\hat{\mathbf{A}}$ has as its columns the basis vectors, that in image processing applications represent also synthesis filters for the data (see [15]). In some cases, the algorithm does not converge, and there is no result, except for the FP algorithm that can deliver partial results. It is also interesting to know how much calculation was needed to perform ICA.

There is not much to say here because the use of the outcome is completely dependent of the application.

6 Artificial data

6.1 Problem

In this part, the problem consists of separating artificial (pseudo-random generated or stored) signals mixed through a random matrix (usually uniformly distributed random matrix). The fact that the mixing matrix is known enables exact computation of the final accuracy, by using the indexes below.

The mixing matrix is square, and there is no under- or over-determination in the problem. The typical values used for the problem parameters were 10 channels, supergaussian, subgaussian or mixed type source signals, generated randomly or in a deterministic way (i.e. simulated spike train for supergaussian, sinusoid for the subgaussian sources), and 2000 samples, or in a range of 500 to 5000, depending on the number of sources.

6.1.1 Generating the sources

The sources are of two types, one subgaussian and the other supergaussian. The supergaussian sources are generated by the following pdf, for source number i

$$f_{s_i}(s) = \frac{|s|^{\frac{1-\beta}{\beta}}}{\sqrt{2\pi}e^{\alpha(i-1)}\beta} e^{\left(\frac{|s|^{1/\beta}}{e^{\alpha(i-1)}}\right)^2} \tag{70}$$

where α and β are two parameters. α controls the decreasing of variance when generating many sources of different powers. β controls the shape of the distribution. It is supergaussian if $\beta > 1$ and subgaussian if $\beta < 1$.

Two periodic signals were included in the supergaussian sources, namely a sine wave and a triangular signal

$$\begin{aligned} s_1(t) &= \sin(\gamma_1 t) \\ s_2(t) &= \begin{cases} -1 + \frac{2\gamma_2 t}{\pi}, & 0 \leq t - 2k\pi < \pi \\ 3 - \frac{2\gamma_2 t}{\pi}, & \pi \leq t - 2k\pi < 2\pi, \quad k \in \mathbb{Z}, \end{cases} \end{aligned} \tag{71}$$

and a uniform signal, with distribution

$$f_s(s) = \begin{cases} 1/(2\sqrt{3}), & s \in [-\sqrt{3}, \sqrt{3}], \\ 0, & \text{otherwise.} \end{cases} \tag{72}$$

The signals are then divided by their standard variance. There is a slight asymmetry in the experiments, since the successive supergaussian sources are weaker, whereas subgaussian source all have the same power.

The parameters are summarized into Table 4. γ_1 and γ_2 are different for the signals to be uncorrelated.

The typical realizations are shown in Figures 4 and 5.

A randomly generated matrix was used. The condition number does not have any great importance (the average correlation with results is around 0.06 when there is no noise).

Whenever not specified, each algorithm may run on a different realization of these data.

Parameter	super-	subgaussian
α	-0.05	
β	1.5	0.7
γ_1		$2\pi/50$
γ_2		$1.254 \cdot 2\pi/50$

Table 4: Parameters for generating the sources

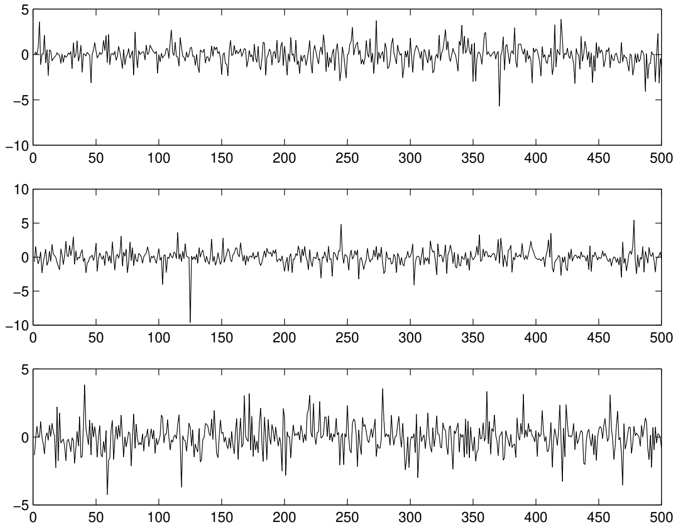


Figure 4: Samples from supergaussian distributions

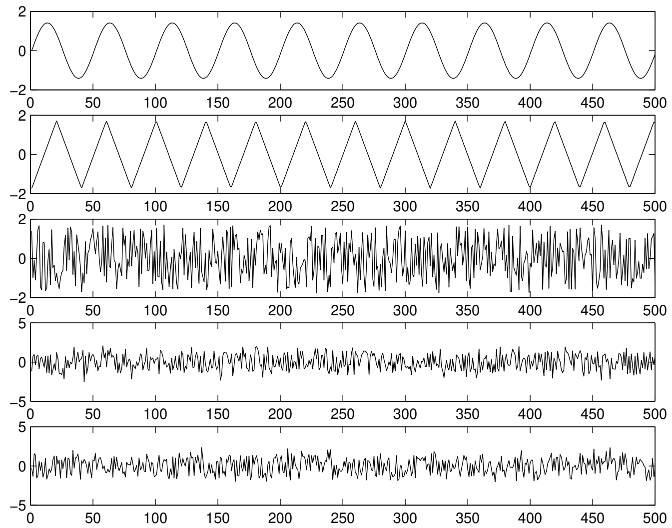


Figure 5: Samples from subgaussian distributions

6.2 Measurements

The accuracy was measured using two performance indexes, described in [1]. The first one, E_1, is defined by

$$E_1 = \sum_{i=1}^{n}(\sum_{j=1}^{n} \frac{|p_{ij}|}{max_k|p_{ik}|} - 1) + \sum_{j=1}^{n}(\sum_{i=1}^{n} \frac{|p_{ij}|}{max_k|p_{kj}|} - 1) \tag{73}$$

where $\mathbf{P} = (p_{ij}) = \mathbf{WA}$.

The second one is only a quadratic version of the same index, that is

$$E_2 = \sum_{i=1}^{n}(\sum_{j=1}^{n} \frac{p_{ij}^2}{max_k p_{ik}^2} - 1) + \sum_{j=1}^{n}(\sum_{i=1}^{n} \frac{p_{ij}^2}{max_k p_{kj}^2} - 1). \tag{74}$$

A value of zero indicates perfect match between the found solution and the sources. The greater the value, the poorer the performance.

It was observed that E_1 has better resolution, so I used it mainly to differentiate the algorithms in the same situation, when the problem does not change. A drawback of E_1 is that the value has little meaning and it is not normalized with respect to the number of sources.

I also used $\sqrt{E_2}$, which is strongly correlated with E_1. However $\sqrt{E_2}$ seems to be more normalized with respect to the size of the problem, see Fig. 15.

For measuring the speed of the algorithms, the actual time of execution, measured in seconds, seemed too arbitrary, first because the problem of the speed of an algorithm is rather a mathematical problem than an optimization problem (that is not handled here), and second because measuring speed on UN*X machines is delicate, because it is dependent of the current load, and is highly non reproducible.

Instead, Matlab provides a measure of flops, floating point operations that is independent of time, and directly related to the operations required by the algorithm. I will base my study of power consumption on this measure as provided by Matlab.

The idea is to reset a counter just before calling the algorithm and taking the value when the algorithm returns the result. This implies that some graphical job is inside, as well as the parts of code collecting various indices. Some profiler examination led to the conclusion that it is less than 1-2%, that is not significant.

The difficult question is that flops are sensitive to the size of the problem, to the number of sources as well as to the number of frames. One should take care to show on the same diagram only results with the same problem.

6.3 Test structure

What is here searched for is a bridge between the theoretical properties of the algorithms and the effective behavior in numerical runs. Because of that, the questions and the tests of this part are simple ones, avoiding to give a too difficult problem, and to get unreadable results.

The most important questions in these tests are:

- Is there a significant difference in the quality of the result obtained?
- Is the amount of computation used by different algorithms different and how much?

Test	Measure	Remarks
Accuracy	display with E_1, values from $\sqrt{E_2}$	importance of nonlinearity, ranking of the algorithms
Convergence	fine grain measure	computational load
Threshold	precision vs stop value	good stability
Comp. load	precision vs flops	FP alg. superior
Sources	$\sqrt{E_2}$ more stable	some alg. degrade quicker
Noise	display with E_1	regular increase, correlation

Table 5: General conclusions on the tests performed with artificial data

- Are the algorithms sensitive to other parameters, like the number of sources, or additive noise, and how much?

To answer these questions, 4 sets of tests were made:

1. Accuracy test with basic parameters, 100 trials.
2. (a) For algorithms where the number of iterations has to be set in advance: convergence test, 10 trials.
   (b) For algorithms that control the number of iterations: relation between the threshold parameter and the effective accuracy, 100 trials.
3. Noise test, 20 steps from 0% up to 20%, 50 trials each.
4. Test with 2 up to 20 sources when supported, 50 trials each.

6.4 Results

On Table 5 is a general view of the tests. The accuracy test is the only one where pure value distribution is searched for. All the other test compare some parameter with the accuracy as a general ranking value.

If one wants to draw a global picture of the insight in this work, then he should include the following points.

- All the algorithms approach a common lower bound. There is a lower bound due to orthogonalization, as exposed in [6]. Most algorithms are near to this lower bound.
- There seem to be three grades of speed in the algorithms. The gradient optimization are the slowest. The RLS technique gives a speedup of approximately 10. The fixed point method gives a speedup of approximately 50. With some compromises in precision and optimization, gradient algorithm can get close to the performance of RLS technique, or even do better in special cases (EASI in subgaussian case).
- In real world problems, the first emphasis should be put on a good estimate of the covariance matrix, if prewhitening is done as a first step. The covariance matrix is used for decorrelating the signals, and the lower bound on final accuracy depends on it. Then dimensionality should be reduced if the problem is ill-conditioned, because the exact source where confusion

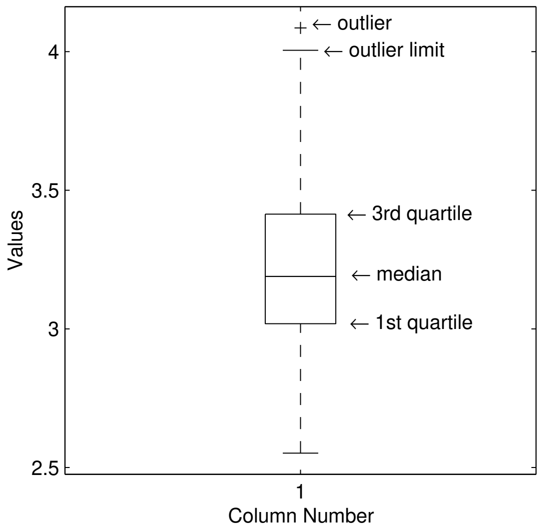


Figure 6: Example of a boxplot display

appears in the separating matrix is dependent of the algorithm, of the nonlinearity used, and on the mean projected variance of the source. If the ill-conditioned problem is given directly to ICA algorithm, the sources retrieved will depend on the initial condition.

Each algorithm has its own weak points, I cannot describe them all here, but they are mentioned along the presentation of the results.

6.4.1 Accuracy tests

Table 6 lists the median values of the error of each algorithm. A blank space indicates that the algorithm as provided cannot handle the problem.

Figure 7 graphically displays the supergaussian test measured by E_1, showing better what is the distribution of the results with boxplots[1].

As a first remark, one could say that most algorithms perform the same. The absolute values are near to each other, and especially correct separation is achieved by each algorithm. The tests with other sources perform quite the same, but with subgaussian sources there is even less difference between the results, except for the EASI algorithm.

The kurtosis deflation version of the fixed point algorithm performs the poorest. It is concluded that the deflation approach degrades the performance as well as the kurtosis, with their effect adding together. But in the symmetrical tanh version, FP algorithm gives as good results as the others.

This can be explained like as follows: first the $tanh$ nonlinearity is basically better than the cubic nonlinearity, and results with $tanh$ in the same situation

[1] A boxplot is a more refined way of viewing a probability distribution than a simple mean or variance display. On Figure 6 is an explanation of the signs. The only parameter is the relation between the outlier border and the interquartile range, here 1.5, who expresses how far a value can be without being an outlier (See [20, p.1-79]).

$\sqrt{E_2}$	superg.	subg.	mixed
FP	0.7869	0.4082	0.6005
FPth		0.3793	
FPsym	0.517	0.2874	0.4639
FPsymth	0.3062	0.27	0.2707
ExtBS	0.3052	0.2536	0.2953
NPCA-RLS	0.3237	0.2817	
ACY	0.2613	0.2657	
WACY	0.2568	0.2683	
BS	0.3076		
EASI	0.4325	0.575	

Table 6: Accuracy of the algorithms. Median value over 100 trials

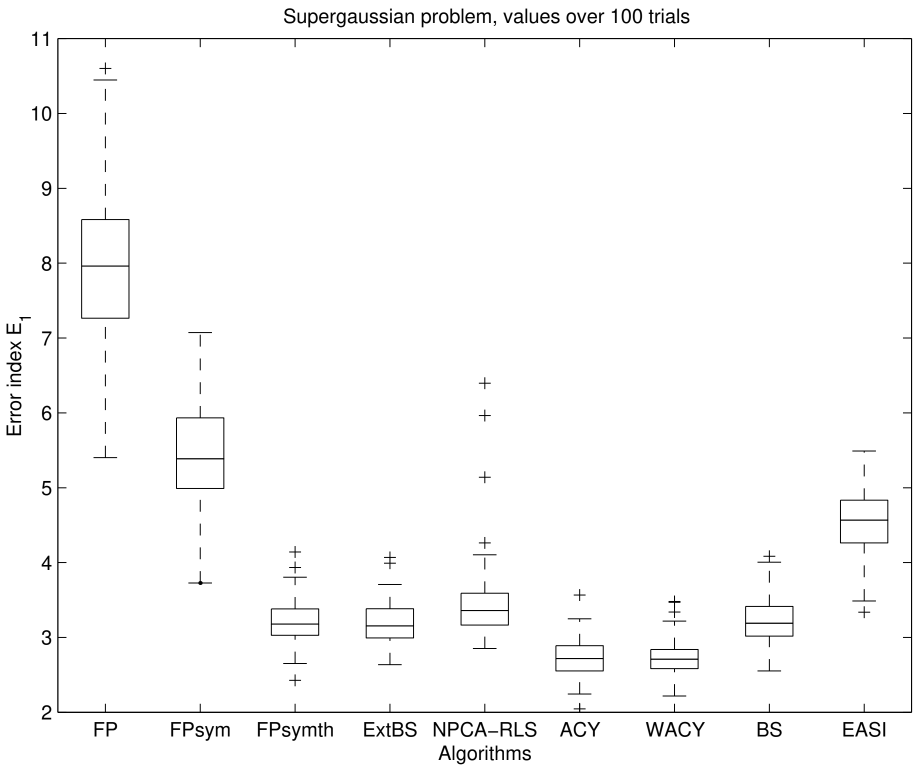


Figure 7: Accuracy of the algorithms over a supergaussian problem

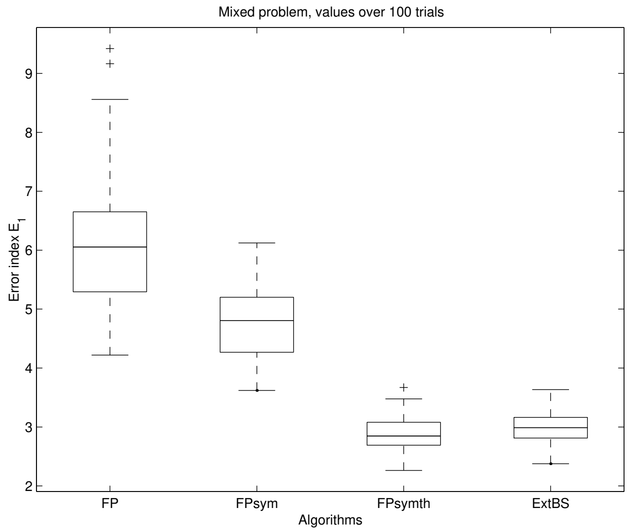


Figure 8: Accuracy of the algorithms over a problem with both components

are better. Second the deflation approach gives more variable results, as can be seen on Figure 8, because the error is set in the beginning, when the first components are extracted, and is more likely to depend on the initial condition. This difference between FP and FPsym algorithms is not clear in Figure 7 because cubic nonlinearity is anyway not optimal at all with supergaussian components.

The ExtBS algorithm is always among the best. This is in line with the theoretical results which say that in this problem, BS algorithm should give a nearly optimal result. On the supergaussian problem, there is little difference between the basic version and the optimized one.

There is a small difference between NPCA-RLS results and the other gradient descent algorithms. This is due to approximations in the algorithm.

The EASI algorithms yields the poorest results in a subgaussian problem. A possible explanation is that the 2 in $(M - M^T)/2$ was dropped in the update equation. Let **EASI2** be this algorithm with the 2. As seen on Figure 9, the difference is significant, and the error is in the same range as for the supergaussian problem, but the EASI algorithm remains the poorest.

Testing the algorithms on a perfect problem, one can approach the theoretical limitations. From Fig. 7 conclusion shall be

- that all the gradient algorithms get very similar results,
- that if one wants to fine tune the algorithm, the nonlinearity yields the biggest gain in precision.

6.4.2 Ensuring convergence

The user of the algorithm would like to be sure that the algorithm outputs a solution, especially when there is little information about the quality of this

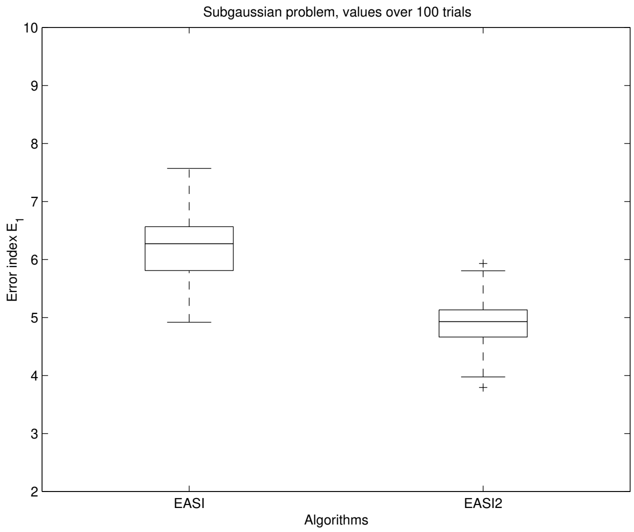


Figure 9: Test between two versions, differing by one coefficient

solution. This is the case in many problems where the mixing is difficult to model.

On the other hand, it is time-consuming and useless to let an algorithm run when it has already converged.

I will describe here some characteristics of convergence with two types of implementations:

- The first versions of the algorithms just perform a fixed number of iterations, large enough to ensure convergence. I will look at the evolution of the error through iterations.
- The sophisticated versions of the algorithms check what changes occur from one step to the other. It is then easy to exit when the change passes below some threshold. I will show how final accuracy depends from the threshold.

The programs of the first type are the NPCA-RLS, ACY, BS, WACY and EASI algorithms. On Fig. 11 is a plot of the error vs used flops. The flops were used because the notion of iteration is not the same for each algorithm. So in this case flops correspond to the amount of work executed as a function of the iterations.

There is a slight difference between WACY and BS algorithms. This is not however, the difference due to the natural gradient improvement, see Section 4.4.

The EASI presents a big variability in the results, and its convergence speed can range from double of the speed of NPCA-RLS to slightly worse than for the other gradient algorithms (See Fig. 10), on a supergaussian problem. On a subgaussian problem it can be faster than NPCA-RLS. One reason for that is that the learning rate for EASI is greater than for the other algorithms. An annealing scheme could easily overcome this problem, since EASI converges anyway with a large range of learning rates.

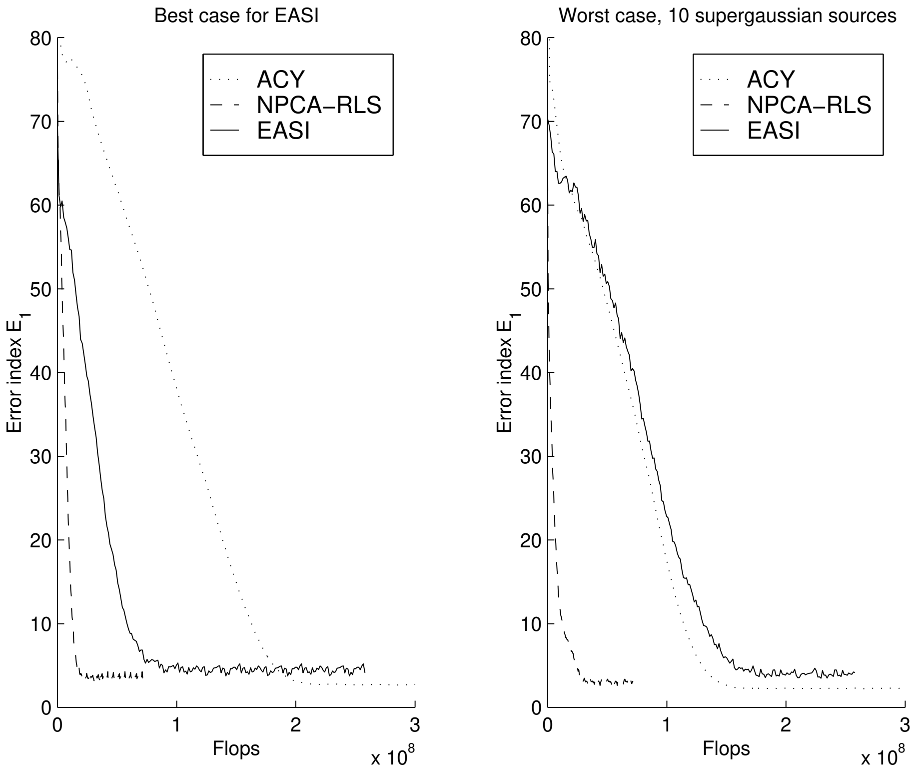


Figure 10: Extreme cases of convergence for EASI algorithm

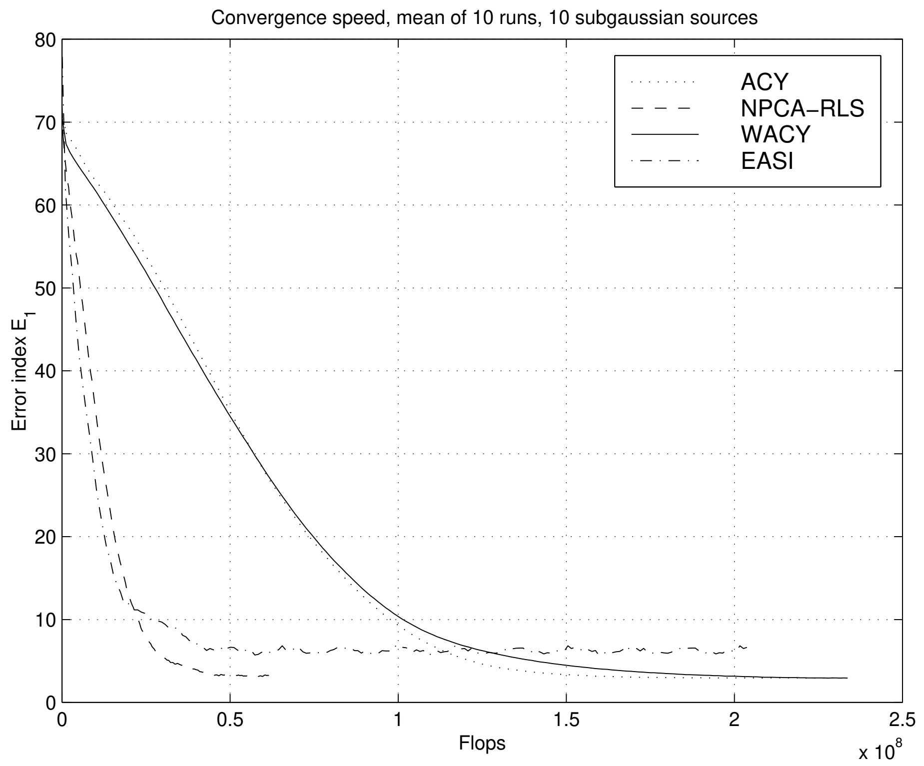


Figure 11: Convergence of the gradient descent algorithms as a function of flops needed.

The NPCA-RLS algorithm converges clearly faster, but its final error is somewhat greater than for the other algorithms. The difference between NPCA-RLS and the others in terms of speed could be reduced by using annealing or other common techniques for combinatorial optimization. The conclusion is, though, that NPCA-RLS is a good algorithm to get quickly a first solution, for instance in time-critical applications.

On the other hand, the programs of the second type are the ExtBS and the FP algorithm. Convergence was tested with a large range of thresholds (see Figure 12).

The ExtBS algorithm stabilizes with $\epsilon < 10^{-3}$, whereas the FP algorithm has few changes as soon as $\epsilon < 10^{-2}$.

The conclusion is that this parameter does not induce any important change. For the rest of the experiments, threshold will be set at 10^{-4} for FP algorithm and 10^{-7} for the ExtBS algorithm.

6.4.3 Convergence speed

As the ICA problem may require a lot of calculation, it is important to know what is the speed of each algorithm. The speed was measured through the flops, provided by Matlab as said before.

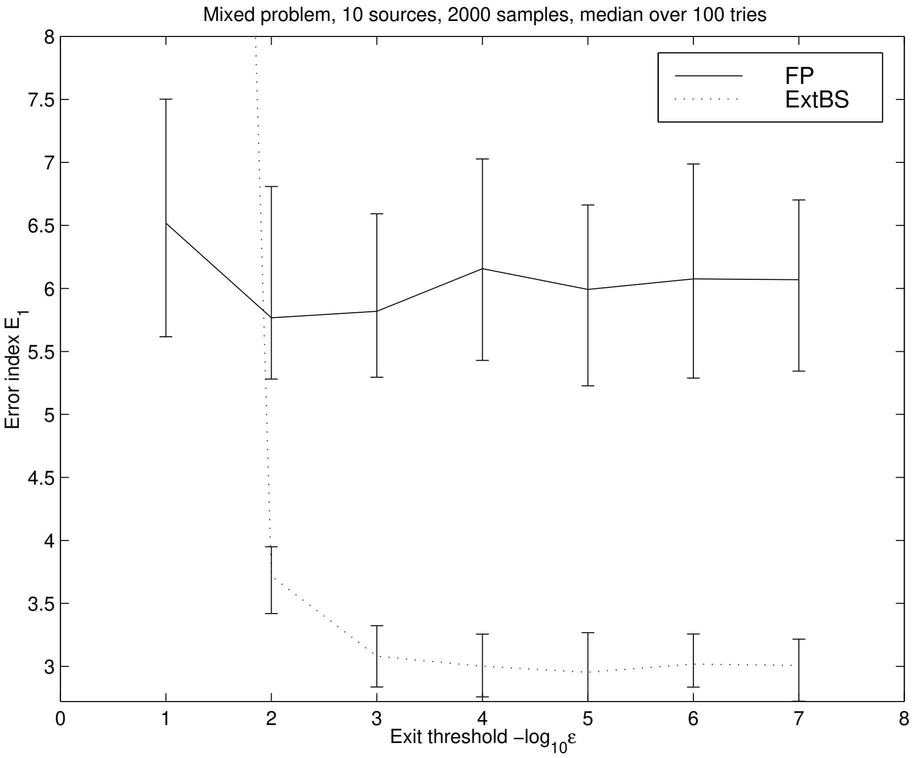


Figure 12: Error as a function of the exit threshold of the algorithms. Median over 100 values. The error bars indicate the 0.2 and 0.8 percentiles.

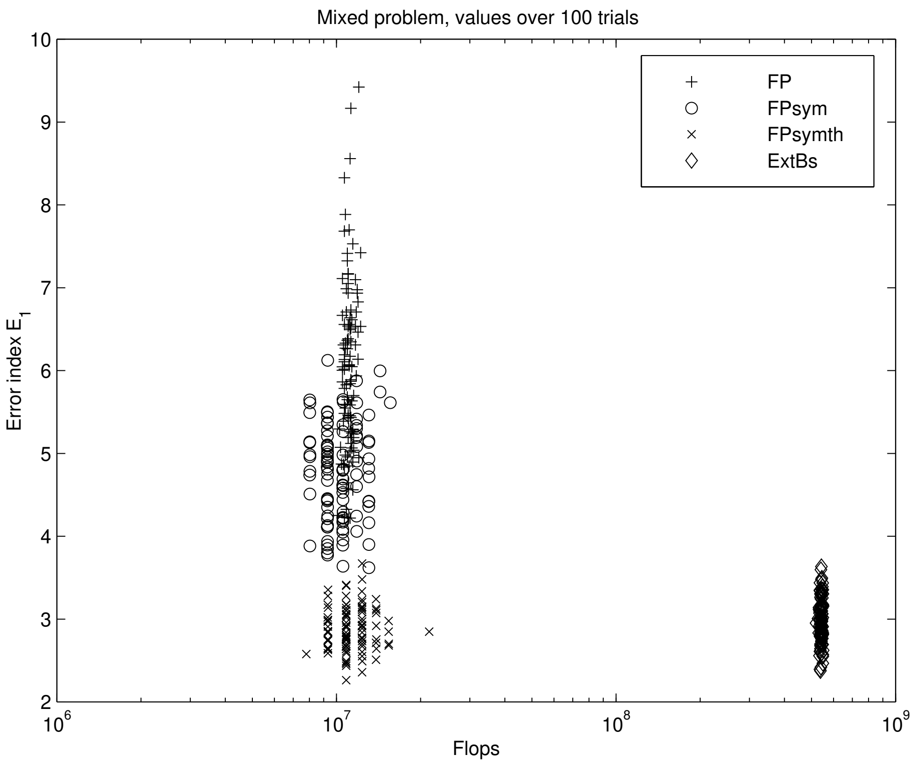


Figure 13: Power requirements of the thresholded algorithms. See Fig. 11 for plotting the accuracy vs the number of iterations.

The plot of accuracy vs flops reveals that the fixed point algorithm is the fastest by a 10 fold difference at least (See Fig. 13).

To match the consumption of algorithms that check for convergence and others that do not, one should combine Fig. 13 with Fig. 11. The result is shown on Figure 14.

The difference between BS and ExtBS amounts to the estimate of kurtosis and the other optimization like simulated annealing. This represents a twofold increase. The consumption for the EASI algorithm is shown here for supergaussian data. As said before, the EASI converges with a large range of speeds. On a subgaussian problem, it generally converges faster than the NPCA-RLS, but with reduced stability.

On this general view, fixed point algorithm remains the fastest. The ExtBS algorithm with the extended feature off is in fact near to EASI performances, due to some optimizations like simulated annealing (not shown on figure).

6.4.4 Variable number of sources

The number of sources does not affect greatly the quality of the result. One can see on Fig 15 that the error for each algorithm grows in a stable manner with the number of sources.

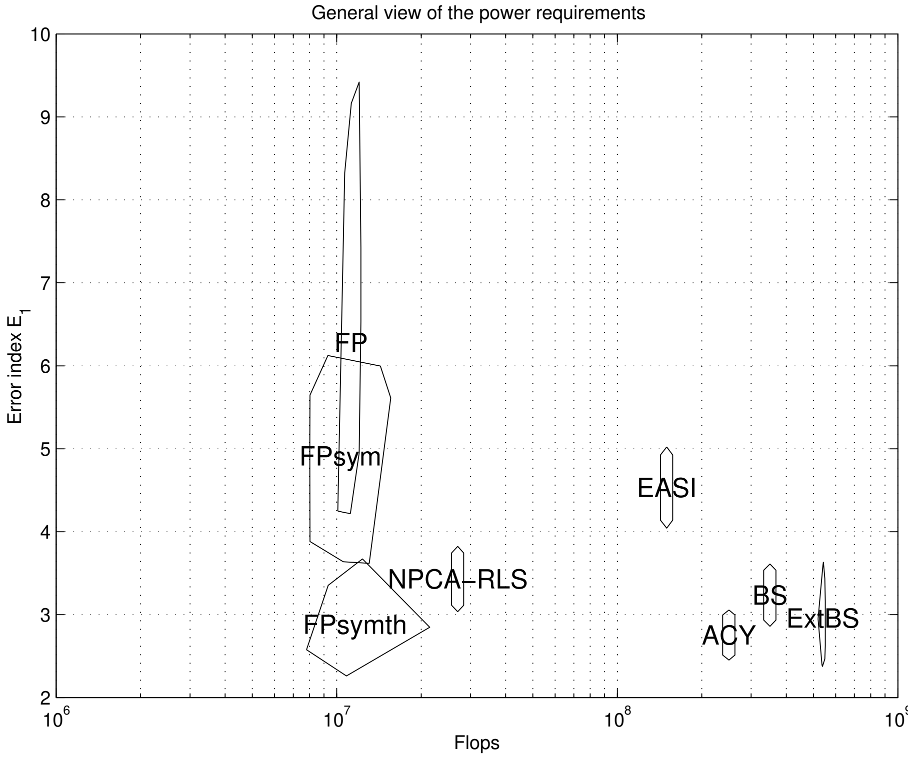


Figure 14: Schematic description of the power requirements in flops vs error index E_1. The thresholded algorithm are shown through the convex area of the same data as on Figure 13, the fixed iterations algorithms are represented by boxes situated at the region of convergence following Section 6.4.2, Figure 11 and the same for supergaussian sources. The height of the boxes correspond to the 10% and 90% quantiles on the basic accuracy experiment.

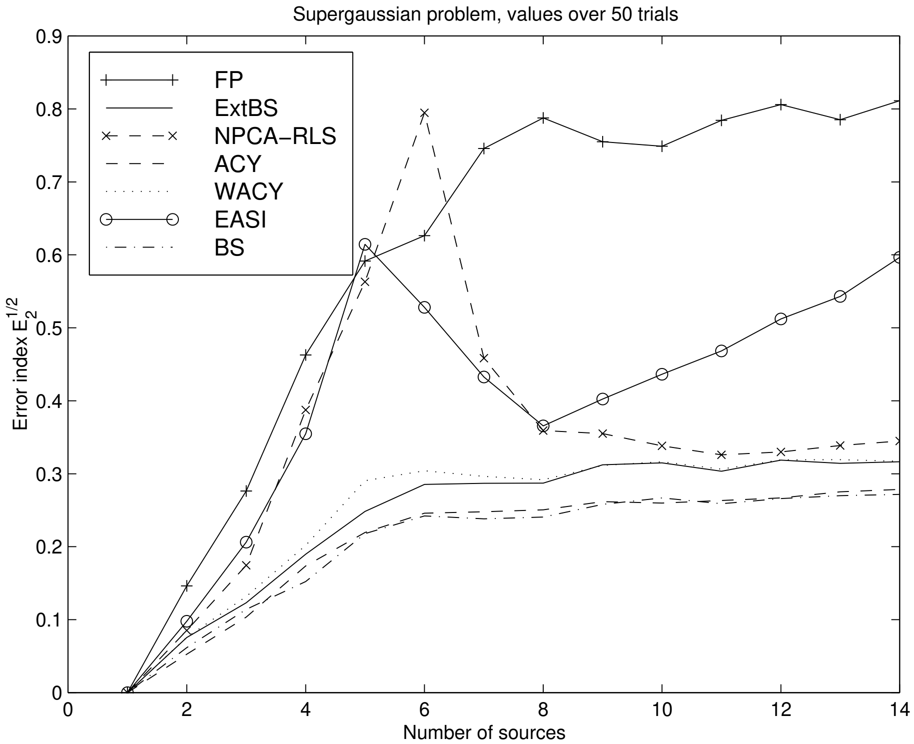


Figure 15: Error with various number of sources. The results of the NPCA-RLS algorithm around 6 sources is bad because the convergence was not completed. Normally it should be just above the WACY alg., like for 10 sources.

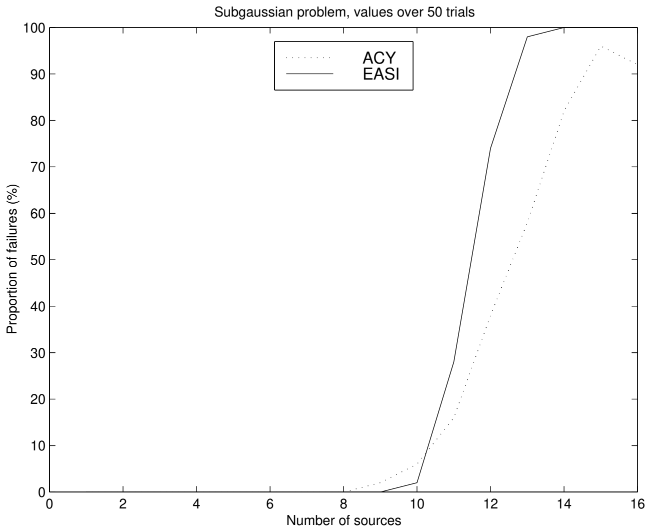


Figure 16: Change in the number of failed convergences. Both algorithms become unusable after 14 sources.

There is a couple of observations about Figure 15. One comment valid for every algorithm is that there is stabilization after 6 sources. About the NPCA-RLS and EASI algorithm, there is a peak around 6 sources. This is not an intrinsic feature of the algorithms, but pertains rather to the question of the number of samples required for proper convergence as a function of the number of sources.

On the other hand the degradation that occurs with EASI after 10 sources seems to be unavoidable. The algorithm is more sensitive than the others to the increase in the number of sources. In the case of subgaussian components, there is even a clear threshold around 10 sources for EASI and ACY, shown in Figure 16.

After examination, it appears that the default with growing number of subgaussian sources is due to the cubic nonlinearity, more sensible to high values. The ACY and EASI algorithms were the only ones to diverge because as pure stochastic gradient algorithms they are more sensible. By taking

$$g(y) = -tanh(y), \text{for EASI and} \tag{75}$$

$$g(y) = y - tanh(y), \text{stabilized version, for ACY} \tag{76}$$

because EASI has a normalized update rule, whereas ACY does not, the algorithms do not diverge any more.

6.4.5 Added Gaussian noise

By blurring the mixtures with Gaussian noise, the point is to obtain a more realistic picture of the error rates. I used the following model for generating the data

$$\mathbf{x} = \mathbf{A}\mathbf{s} + \alpha\mathbf{n} \tag{77}$$

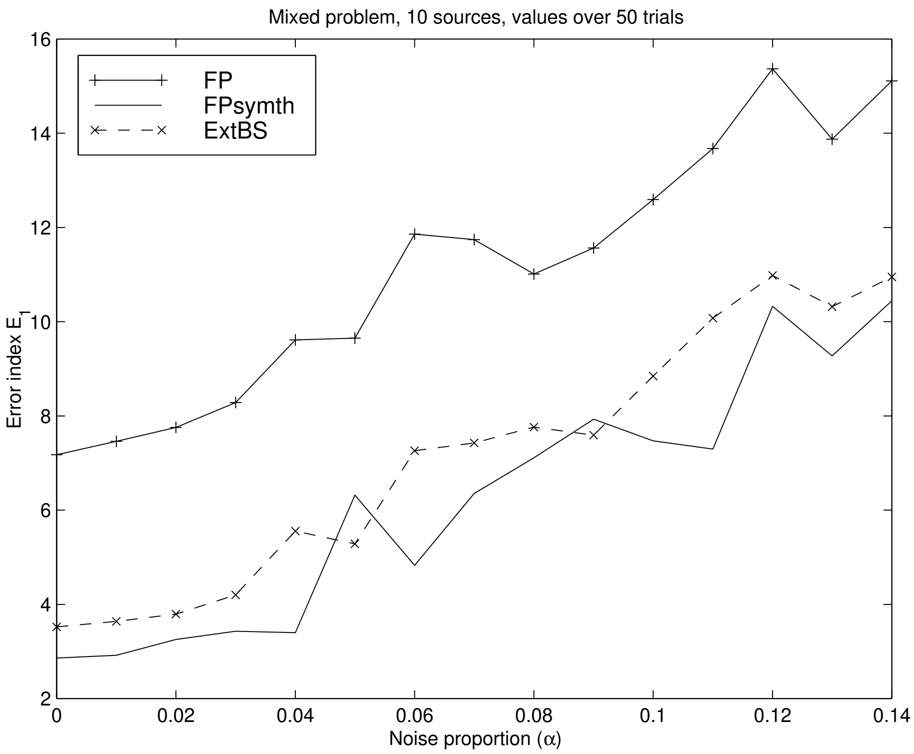


Figure 17: Error as a function of the added Gaussian noise.

where $\mathbf{n}$ is a Gaussian noise vector having the same variance as $\mathbf{s}$. The parameter α ranges from 0.01 to 0.2.

Figure 17 shows that the error slowly grows up to a point where the separation is no longer achieved. There is no clear discontinuity.

In fact there is a continuous degradation measured by the error index. Results are different if someone uses permutations to see if the final matrix $\mathbf{P} = (p_{ij}) = \mathbf{WA}$ contains some confusion. A small experiment was made with 7% of noise and the results were that in 3 cases out of 20 there was confusion, meaning that at least one source was lost.

Another observation is that with little noise, the error index becomes highly correlated with the condition number of the mixing matrix. The condition number of the matrix is defined as

$$\frac{\lambda_{max}}{\lambda_{min}} \tag{78}$$

where λ_{max} is the greatest eigenvalue of the eigenvalue decomposition of the matrix and λ_{min} the smallest. A matrix corresponding to an ill-conditioned linear system has high condition number.

It is natural that with an ill-conditioned matrix, the effect of the noise is stronger since some sources must be very weak in the mixtures. There has even been a mathematical analysis in the case of equivariant algorithms showing that the mixing matrix appears in the final solution only when noise is added [6].

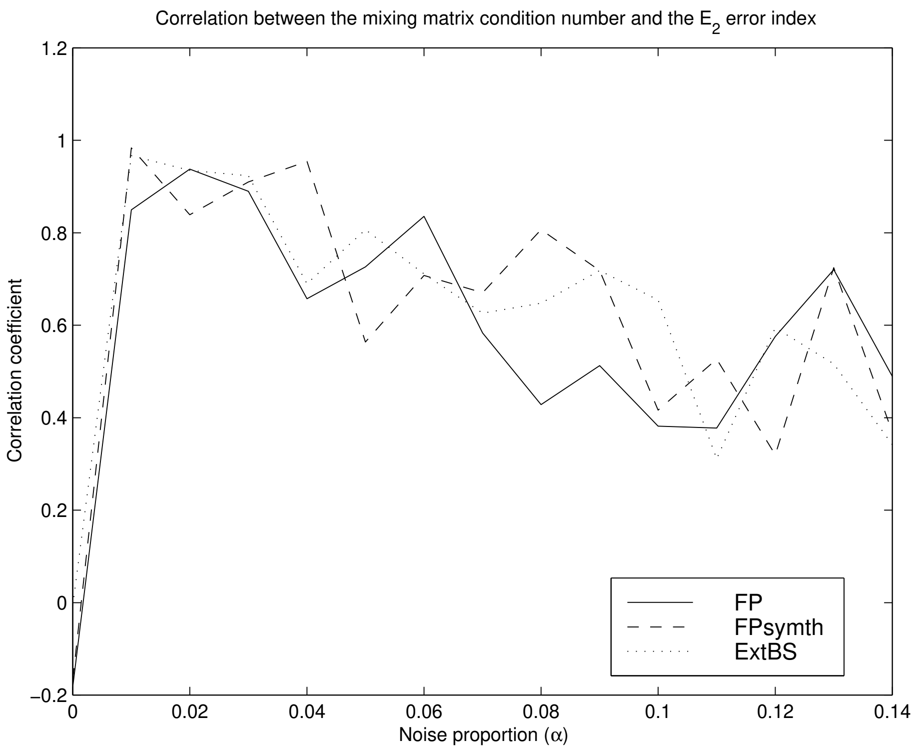


Figure 18: Plot of the rise of the correlation, showing dependency on the mixing matrix in the case of noise.

On Figure 18, the correlation between the condition number and the error is plotted as a function of the amount of noise. At zero noise, no correlation is present, when noise grows, correlation vanishes because the separation gets more serious problems like confusion for instance, which is not related to the condition number. Notice that none of the algorithms of the figure is an equivariant algorithm.

7 Projection Pursuit data

In this section comparison is applied with real world data. The objective of projection pursuit is to find interesting projections out of multidimensional data without prior knowledge of the data.

7.1 Problem

Most of the problems in projection pursuit are classification problems, that is each data point comes as a pair $(\mathbf{x}, c)$, $\mathbf{x}$ being the values of the features and c the class to which this point belongs. The class does not exist for new incoming data, but only for training and testing purpose. The ICA method excludes any supervised training, but the underlying assumption is that projections that are independent from each other are suitable for visualization. In a sense, ICA can perform in some case a blind classification (analog to blind source separation) by clustering the data following their independence properties.

7.1.1 Crab database

The crab database comes originally from [3], and is taken from [31]. Two species of *Leptograpsus variegatus* crabs were differentiated by color, blue and orange, but the specimens lose their color in the storage. The goal was to differentiate species by morphological characteristics. The data contains 50 specimens of each sex of each species, collected at Freemantle, Western Australia. Each specimen has measurements on the width of the frontal lip, the rear width, length along the midline and maximum width of the carapace and the body depth in mm. Thus the number of measurements made from each specimen is five. The database is classified, meaning that for each sample is given the species and the sex of the animal. The idea is to see if ICA finds clusters corresponding to the classes, performing a kind of self-organization.

The four classes span two dimensions, but it does not mean that the data contain nothing else. It may be so that the algorithm finds directions that would be meaningful if it was possible to attach any meaning to them. Regrettably, I am no specialist about crabs, and the provided classifications are the only ones taken into account.

7.1.2 Satellite image pixels

The satellite image pixel database (named like that because there is no complete image, but only a set of individual pixels) is taken from the UCI Machine Learning Repository [27], provided to UCI by Ross D. King at Univ. of Strathclyde from the European StatLog project. The original Landsat data were purchased from NASA by the Australian Centre for Remote Sensing (see Url [27] for more information).

This database was generated from Landsat Multi-Spectral Scanner image data. One frame of Landsat MSS imagery consists of four digital images of the same scene in different spectral bands. Two of these are in the visible region (corresponding approximately to green and red regions of the visible spectrum) and two are in the (near) infra-red region. Each pixel is a 8-bit binary word, with 0 corresponding to black and 255 to white. The spatial resolution corresponding to a pixel is about 80m x 80m. Each image contains 2340 x 3380 pixels.

The actual database is a tiny sub-area of a scene, consisting of 82 x 100 pixels. The classification for each pixel was performed on the basis of an actual site visit by Ms. Karen Hall, when working for Professor John A. Richards, at the Centre for Remote Sensing at the University of New South Wales, Australia.

Each row of data corresponds to a 3x3 square neighborhood of pixels completely contained within the 82x100 sub-area. Each row contains the pixel values in the four spectral bands (converted to ASCII) of each of the 9 pixels in the 3x3 neighborhood and a number indicating the classification label of the central pixel. The database contains thus 6435 patterns with 36 attributes (4 spectral bands x 9 pixels in neighborhood) plus the class label.

The classes are listed in Table 7, but there are no examples with class 6 in this dataset - they have all been removed because of doubts about the validity

Number	Class
1	red soil
2	cotton crop
3	grey soil
4	damp grey soil
5	soil with vegetation stubble
6	mixture class (all types present)
7	very damp grey soil

Table 7: Classes of the SatImage database.

of this class.

The data is given in random order and certain lines of the original image have been removed so it is impossible to reconstruct the image with this database.

6 classes are too much for linear separation. Thus only the most prominent class distinctions are sought. Only the central pixel values were used, the others were discarded. This means that the data has only 4 dimensions.

7.2 Measurements

Unlike the artificial data case, there is here no direct measure of the quality of the results. What can be done is

- compare the solutions found by different algorithms, hoping that there will be a pool of solutions centered around the optimal solution, but this method requires to be sure that there exist some optimal solution,
- use some information-theoretic measure to rate the quality of the result. This is a better founded solution, but most of the theoretical measures exposed in Section 2.2 are difficult to estimate from the data, and often computationally expensive.

When the problem has many dimensions, it is not trivial to see which ICA component contains information. To select the displayed information, the same rating index was used.

There are many possibilities to design a structural index. Some have been tried, as summarized below.

7.2.1 Estimation of negentropy

Negentropy should be a good estimate of the structure of a one-dimensional projection in the continuous case. As presented in Section 2.2, the Gaussian distribution can be considered the least structured among the distributions over infinite support that have a mean and a variance. It is theoretically justified to rely on negentropy as a distance from normality to provide a structural index.

Computation of negentropy is not easy. Many estimates have been proposed (see [18], and the references therein). An estimate of the pdf is needed, and many problems arise from that. Estimate of the pdf is a kind of numerical derivative, and it should be avoided because the irregularities in data are amplified by numerical derivation. Because of the log function, negentropy is also

Figure 19: Problematic case in histogram estimate

very sensitive to the small probabilities. Suppose a histogram is built over the data. Given a situation like on Figure 19, a shifting of the bins results in a dramatic change of the estimate. In case a), value is $-2\log 1/N$ and in case b) $-\log 2/N$, because the other bin is discarded for numerical reasons.

An estimate based on empirical histogram was first tried, but the results were poor. As soon as noise is included in a component, the estimate greatly depends on the number of bins.

Another method was, instead of dividing the data into fixed bins, to sum on variable bins containing always the same number of occurrences. But then zones with few occurrences will yield very small probabilities, with high influence on the result. Result is only slightly better than with fixed bin histograms, and especially the index in problematic cases prefers non structured projections.

Finally it seemed that this approach could not find the ICA directions which provided the best discrimination. Therefore, another class of indexes was examined.

7.2.2 Structural index

Friedman [11] described a projection pursuit index that differs from negentropy. Although he agrees that nongaussianity is a good criterion for finding interesting projection pursuit directions, he does not use a classical statistical measure like negentropy because negentropy is too sensitive to departure from gaussianity in the tails of the distribution, in a part which is not significant most of the time. What is to be searched anyway is a structure in the pdf, most probably in the central part, like separation into clusters.

The idea of this index is to sphere the data, then map it into the interval $[0, 1]$ with the Gaussian cumulative density function $\Phi(v)$. Given some data coming from a random variable x, we compute

$$v = \frac{x - \mathrm{E}\{x\}}{\sigma_x} \tag{79}$$

$$r = 2\Phi(v) - 1 \tag{80}$$

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-1/2t^2} dt \tag{81}$$

where σ_x is the standard deviation of x. With this transformation, if x is Gaussian, r is uniform. The subsequent step is to choose a distance from uniformity. In this case integral-squared distance is taken, yielding

$$I(x) = \int_{-1}^{1} f_r^2(r)dr - 1/2 \tag{82}$$

where $f_r(r)$ is the pdf of r and I is the index. In [11], a moment approximation of Eq. 82 is taken.

The main problem is that estimates of the pdf are difficult to obtain. On the contrary, an estimate of the cdf is very easy to obtain. Let the $\{x_i\}, i = 1, ..., N$ be N iid realizations of a random variable x with pdf $f_x(x)$ and cdf $F_x(x)$. Let $\{\sigma_i\}$ be the indexes of the ordered $\{x_i\}$, so that

$$x_{\sigma_i} \leq x_{\sigma_j} \Leftrightarrow i \leq j. \tag{83}$$

Then the $\{x_{\sigma_i}\}$ form an estimate of the inverse of the cdf, that is

$$x_{\sigma_i} = \hat{F}_x^{-1}(i/N), i = 1, ..., N. \tag{84}$$

Because there are i realizations not greater than x_{σ_i}, one can approximate

$$P\{x \leq x_{\sigma_i}\} = F_x(x_{\sigma_i}) \simeq i/N \tag{85}$$

on the points, and interpolate if necessary.

This approximation has the nice property of being monotonic because of the sorting.

With that approximation, it is easy to calculate another distance from normality as

$$I_2(x) = \int_{-1}^{1} \left(\hat{F}_x(u) - \frac{u+1}{2} \right)^2 du \tag{86}$$

or rather the inverse, covering the same area

$$I_3(x) = \int_{0}^{1} \left(\hat{F}_x^{-1}(p) - (2p-1) \right)^2 dp \tag{87}$$

which becomes in the data version

$$\hat{I}_3(x) = \sum_{i=1}^{N} \left(x_{\sigma_i} - \left(\frac{2i}{N} - 1 \right) \right)^2, \tag{88}$$

which is the index finally used in the experiments.

The advantage arising from the monotonic property is that the estimator is more robust to noise. With this method, it was possible to obtain estimates of the pdf through numerical derivation and polynomial regression on which clustering appeared very clearly. The next step would be to apply negentropy estimation on this pdf estimate. This was not done though.

Index was used in ranking the IC components according to the structure of their distribution. Small differences between algorithms are also displayed.

7.3 Results

7.3.1 Crab database

The experiment was made in two steps. First each algorithm was tuned so that it gets an acceptable solution. There is a brief comment on how it was made and how difficult it was. After that, the best results of each algorithm are compared. Visual comparison is one way of displaying the result, angle between the separating vectors is the other one.

Algorithm	Choice	Nonlinearity
ACY	WACY	subgaussian
ExtBS	extended, exactly 2 subgaussian sources set manually	*tanh*
EASI	15 sweeps through the data	subgaussian
NPCA-RLS	10 sweeps through the data	subgaussian
FP	FPsymth version	

Table 8: Choice among the versions of each algorithm for resolving the crab problem

ACY The non whitened version gave very poor results. The whitened version with nonlinearity for supergaussian sources did not show any convergence. I used WACY with subgaussian nonlinearity, that performs separation although results are poorer than with other algorithms.

ExtBS The crabs database is small (only 200 data points) and the kurtosis estimate did not work. It finds only one subgaussian component that converges to the species separation, but fails to find the other separating ICA component. By setting manually the number of subgaussian components, convergence is fine. The minimal value is 2 subgaussian components, 3 does not bring any improvement.

EASI The nonlinearity for supergaussian sources eventually managed to separate the species, but not the sex. The number of iterations is such that convergence is assured.

NPCA-RLS The nonlinearity corresponding to supergaussian sources converges towards something, but there is no way to know whether it corresponds to something meaningful, although the distribution looks clustered. The nonlinearity for subgaussian sources produces the same results as the other algorithms.

FP The original version of the fixed point algorithm with whitening, deflation and cubic nonlinearity fails to find more than two components. It seems that the problem is globally ill-conditioned, and that one of the components is near to a principal component. After whitening, the algorithm fails to retrieve this component and gets confused. The same algorithm without whitening works well.

Better results are obtained with the symmetric approach, using the *tanh* nonlinearity. The symmetric approach with cubic nonlinearity converges finally but with strong oscillating behavior ($\sim$ 300 steps to converge).

The final choice for each algorithm is summarized in Table 8. Globally, every algorithm is able to cluster the classes, after some tailoring.

In Figure 20, every algorithm shows almost exactly the same result except for the ACY algorithm which had more difficulties in converging to an acceptable solution.

The displayed dimensions are selected with the index presented in Section 7.2.2. The value of this index allows to measure how much the underlying

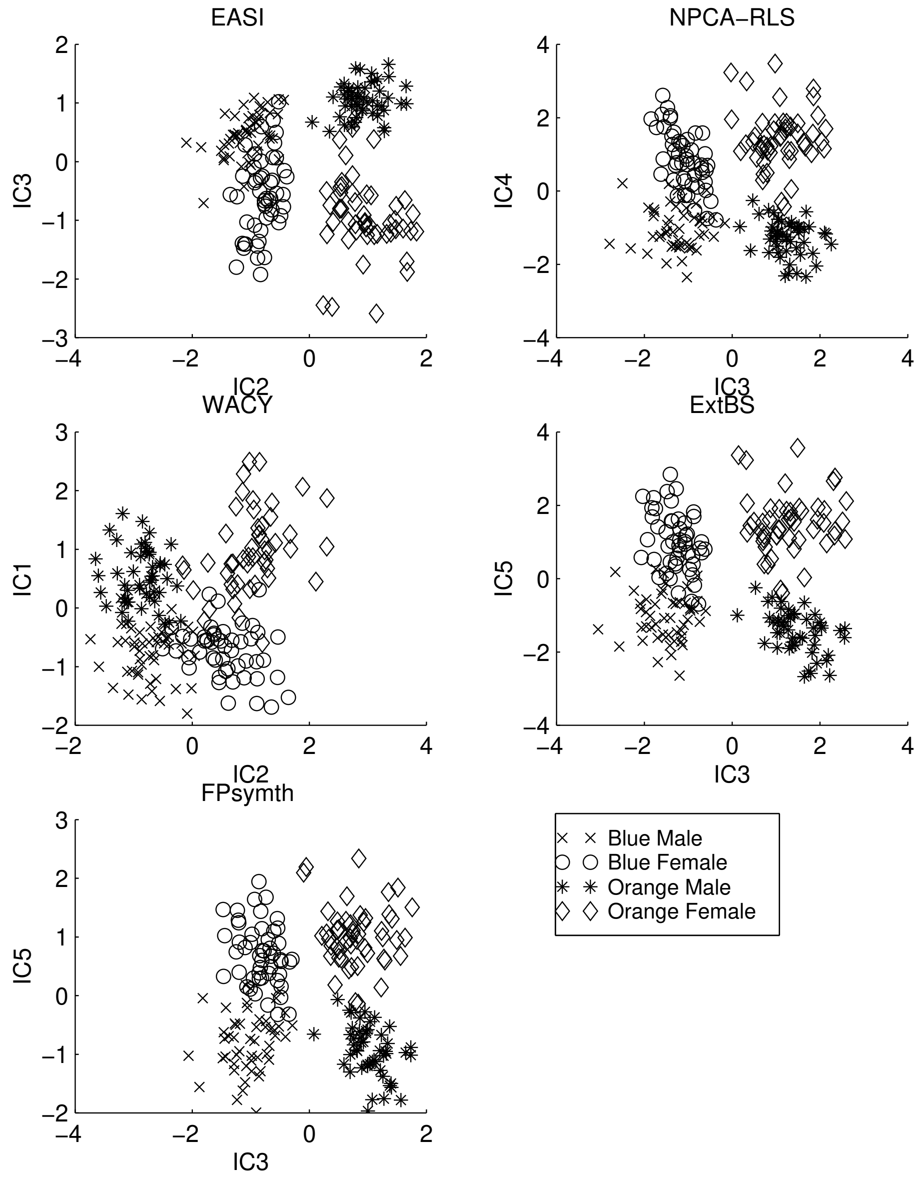


Figure 20: Display of the crab data along the two best dimensions

$\hat{I}_3$	1st	2nd
EASI	5.36	1.96
NPCA-RLS	5.31	2.68
WACY	2.35	1.21
ExtBS	5.97	2.32
FPsymth	5.19	1.86

Table 9: Values of the structural index for the two best components, mean of 10 trials.

angle(°)	EASI	NPCA-RLS	WACY	ExtBS
FPsymth	26.63	26.32	60.95	13.59
ExtBS	13.75	13.24	62.25	
WACY	19.06	21.41		
NPCA-RLS	2.906			

Table 10: Angles between the separating vectors of the best component, mean of 10 trials.

distribution is structured. The values are listed in Table 9. In agreement with Figure 20, WACY algorithm has lower values for both dimensions. The second worst is the FP algorithm, maybe for the same reasons that make the estimate of kurtosis useless. The size of the data is just sufficient for calculating the covariance matrix, and this results probably in errors at the whitening level that cannot be corrected by the proper ICA algorithm.

Table 10 shows the angle computed between the separating vectors corresponding to the most structured component. Only half of the symmetric matrix is given. The small angle between NPCA-RLS and EASI can be explained by the fact that both are realted to the same Nonlinear PCA criterion (see [25]). NPCA and EASI perform much the same, and NPCA-RLS only converges faster. The ACY has a bad projection for the species distinction, and thus the angle is greater.

As a conclusion, the results provide interesting clustering among the algorithms, but one should be very careful with the results, since the small size of the crab database makes numerical artifacts more probable.

7.3.2 Satellite image pixels

The procedure is more complicated in the case of the satellite pixels because there are more classes, and some classes (3, 4 and 7) are rather values of the same parameter than really distinct classes. Nevertheless, the same procedure was applied, with a separate ranking for each algorithm and comparison between the best of each algorithm.

ACY The algorithm without whitening did not converge with subgaussian nonlinearity, with whitening, the same nonlinearity provides a mixing solution: instead of being arranged on orthogonal dimensions, like in Figure 21, data point arrange on diagonal lines, because the algorithm maximizes the depen-

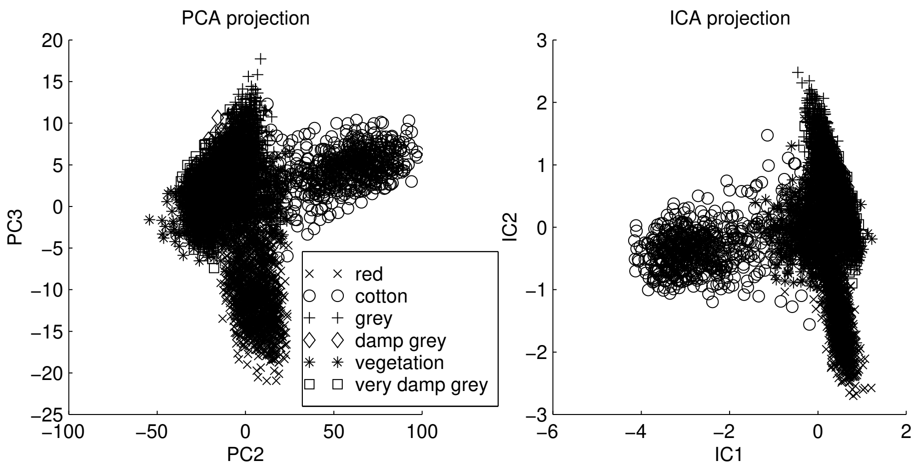


Figure 21: Display of the satellite pixel data along the best ICA dimension and the correspondent PCA dimensions. Important is the shape of the distributions, not the actual position of the classes, which is better seen on the histogram plots.

dence between the components. With supergaussian nonlinearity, the algorithm performs well with whitening, but see the remarks below.

EASI The same observation is valid for the EASI algorithm, and the supergaussian nonlinearity was chosen. The algorithm converges in one sweep.

NPCA-RLS This version also is not able to handle different types of components, and with the subgaussian nonlinearity it provides a mixing solution. With supergaussian nonlinearity convergence is achieved in one sweep.

ExtBS The estimate of kurtosis works well.

FP All the versions of the algorithm provide a solution, but the deflationary approach does not yield as orthogonal components as the symmetric approach. FPsymth version was chosen.

The data has only four dimensions and it is not obvious that ICA performs better than PCA for instance. There is a difference though. At visual inspection, both projections look the same, as in Figure 21, but by looking at the distributions of the first axis, one observes from Figures 22 and 23 that the ICA distribution is more clustered, as a result of using higher-order statistics. The projection is roughly the same, but the tuning correspond to different characteristics.

This can be viewed in another way as how many dimensions are required for describing the data. Let's say each projection retrieves information from the data until it is fully described. The proper method is called structure removal, but there has been no experiment of structure removal yet. Suppose the projections (the IC components) are ordered by the structural index $\hat{I}_3$. The first IC

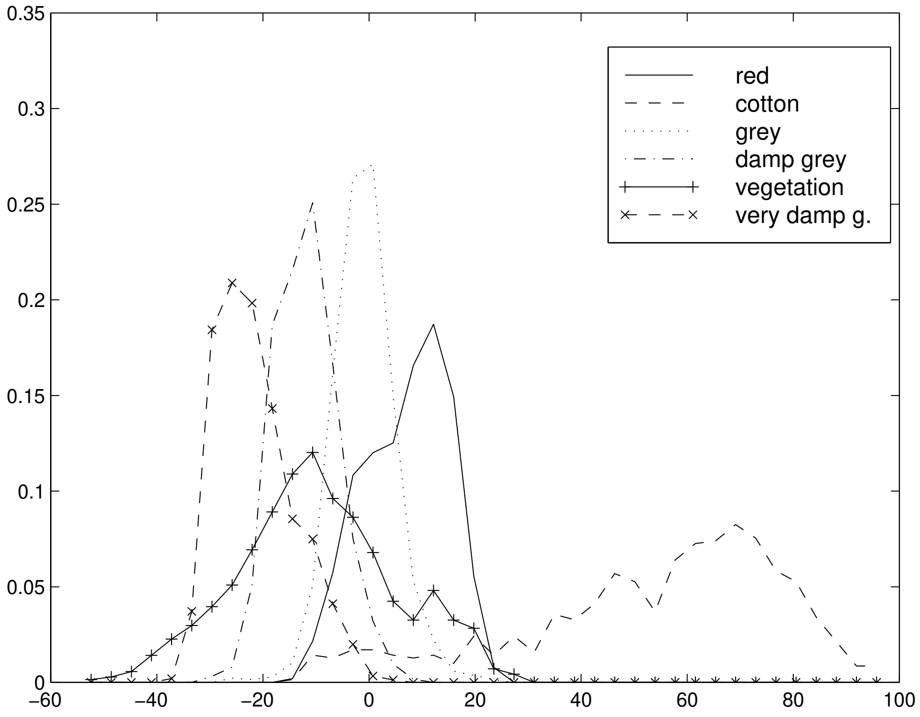


Figure 22: Distribution by class, along the second PC

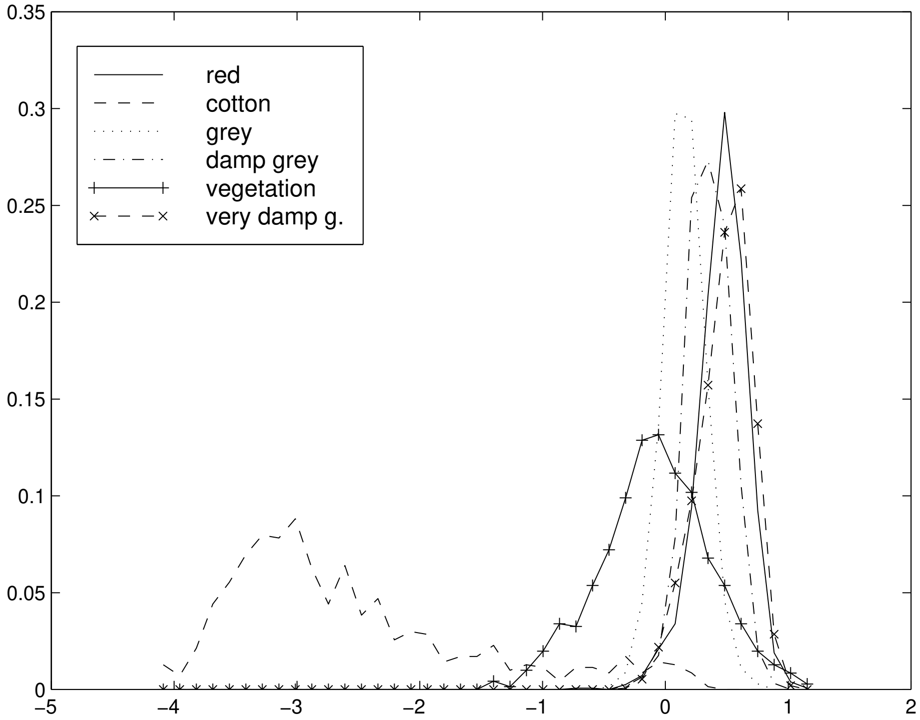


Figure 23: Distribution by class, along the first IC

$\hat{I}_3$	1st	2nd	3rd
FPsymth	694	34.4	26.1
NPCA-RLS	782	200	9.97
ExtBS	391	63.4	41.2
EASI	747	167	20.2
ACY	730	253	0.383

Table 11: Values of the structural index for the three interesting components, mean of 10 trials.

components converges to the "*cotton* vs the rest" distinction as on Figure 23, for all the algorithms.

The turning point is the projection that describe classes 3,4,7 (grey soil) and eventually class 5 (vegetation). These classes are not well clustered, but form a continuum. Consequently, this projection has to be subgaussian. Only the algorithms that can accommodate both types of components will output this component. There is here a difference with PCA about how information is shared among the components. What IC algorithms retrieve as a stand alone dimension (Figure 24, alg. FPsynth and ExtBS) is mixed by PCA with the first distinction (Figure 22), and discarded by the algorithm that do not adapt nonlinearity (Figure 24, alg. ACY and NPCA-RLS, EASI is between the two cases). The latter algorithms focus on clustering the *red soil* class.

The result in terms of distribution structure of the components is visible with the values of the structural index in Table 11. Whereas values for the FPsymth and ExtBS are similar for the 2nd and 3rd components, in the case of ACY the data are 'exhausted' after two components only.

This would certainly require more careful examination, which cannot be included in this work. Structure removal in the case of Projection Pursuit probably makes a great difference (see [11] for description of structure removal). It would be an elegant way of performing classification in terms of ICA projections.

8 Discussion and conclusion

In this work effort was made to design as objectively as possible a framework for comparing different algorithms for ICA. Purposedly, no ranking of the algorithms was provided, because each feature yields another classification. The most significant difference between the ICA algorithms is probably that FP algorithm is fast in comparison with the others, while there is no real difference in the precision.

Although some issues in this work can nowadays be deduced by theoretical ways, nothing replaces experience, and if the conclusions of this work are not new at least they back up the theoretical results. In addition, the theoretical effort like in [7] is a recent one, and it is probably not terminated. For instance, there are no theoretical results about the number of sources each algorithm can separate.

The two experiments designed are at both extremes of the range where ICA is used. It could be argued that the first one is too simple, and the other one too hazardous. But the first experiment provides numerical values and some basic

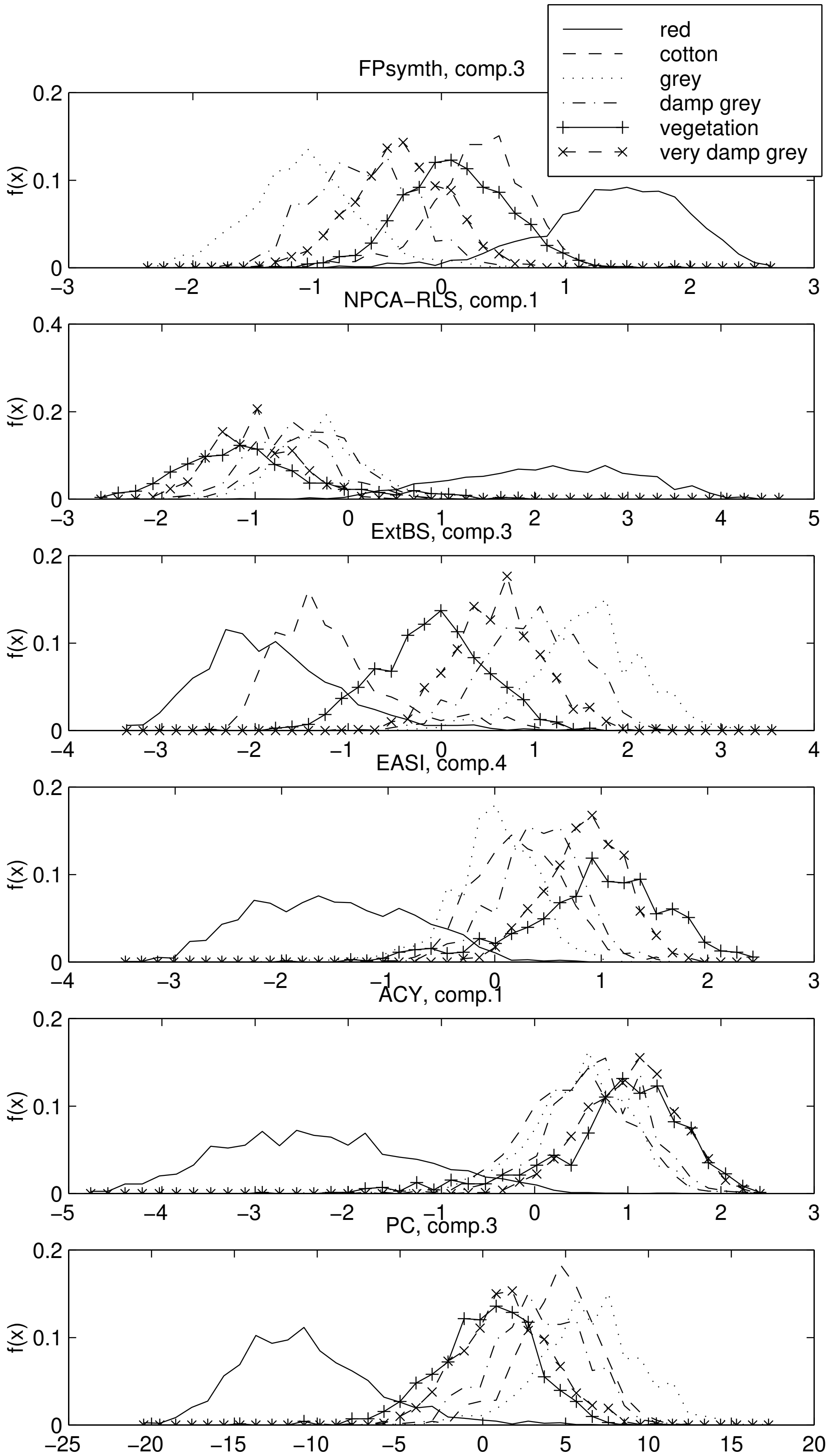


Figure 24: Distribution by class, along the second IC, ordered by structural index. The index of the component is in the separating matrix.

information about the use of the algorithms, which may serve as a tuning library for more delicate situations. The second experiment is designed to understand in which cases ICA does not work properly, in order to discern some limitations of the model.

As expected, every method for solving the ICA problem relies on a common theoretical background. It is not difficult, though, to obtain differences, sometimes small, sometimes more significant, in the functioning of these algorithms. It is hoped that this work helps to provide a picture for estimating what is the value of each algorithm, what is the adequate method for a particular problem and what are the limits and advantages of the ICA method.

References

[1] S. Amari, A. Cichocki, and H.H. Yang. A new learning algorithm for blind source separation. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing 8 (Proc. NIPS'95)*, pages 757–763. MIT Press, Cambridge, MA, 1996.

[2] A.J. Bell and T.J. Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7:1129–1159, 1995.

[3] N.A. Campbell and R.J. Mahon. A multivariate study of variation in two species of rock crab of genus *Leptograpsus*. *Australian Journal of Zoology*, 22:417–425, 1974.

[4] J.-F. Cardoso. Source separation using higher order moments. In *Proc. ICASSP'89*, pages 2109–2112, 1989.

[5] J.-F. Cardoso. Eigen-structure of the fourth-order cumulant tensor with application to the blind source separation problem. In *Proc. ICASSP'90*, pages 2655–2658, Albuquerque, NM, USA, 1990.

[6] J.-F. Cardoso. On the performance of orthogonal source separation algorithms. In *Proc. EUSIPCO*, pages 776–779, Edinburgh, September 1994.

[7] J.-F. Cardoso. *Entropic Contrasts for Source Separation*, chapter 2 of Adaptive unsupervised learning. S. Haykin ed., to appear. Available at http://sig.enst.fr/∼cardoso/stuff.html.

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

[9] A. Cichocki and R. Unbehauen. Robust neural networks with on-line learning for blind identification and blind separation of sources. *IEEE Trans. on Circuits and Systems*, 43(11):894–906, 1996.

[10] P. Comon. Independent component analysis – a new concept? *Signal Processing*, 36:287–314, 1994.

[11] J.H. Friedman. Exploratory projection pursuit. *J. of the American Statistical Association*, 82(397):249–266, 1987.

[12] S. Haykin. *Neural Networks. A Comprehensive Foundation.* Macmillan College Publishing, New York, 1994.

[13] S. Haykin. *Adaptive Filter Theory.* Prentice-Hall International, 3rd edition, 1996.

[14] J. Hurri. Independent component analysis of image data. Master's thesis, Helsinki University of Technology, 1997.

[15] J. Hurri, A. Hyvärinen, and E. Oja. Wavelets and natural image statistics. In *Proc. Scandinavian Conf. on Image Analysis '97*, Lappenranta, Finland, 1997.

[16] A. Hyvärinen. Independent component analysis by minimization of mutual information. Technical Report A46, Helsinki University of Technology, Laboratory of Computer and Information Science, 1997.

[17] A. Hyvärinen. New approximations of differential entropy for independent component analysis and projection pursuit. Technical Report A47, Helsinki University of Technology, Laboratory of Computer and Information Science, 1997.

[18] A. Hyvärinen. New approximations of differential entropy for independent component analysis and projection pursuit. In *Advances in Neural Information Processing 10 (Proc. NIPS*97).* MIT Press, 1998. To appear.

[19] A. Hyvärinen and E. Oja. A fast fixed-point algorithm for independent component analysis. *Neural Computation*, 9(7):1483–1492, 1997.

[20] B. Jones. *Statistics Toolbox.* The Math Works Inc., 24 Prime Park Way, Natick, Mass. 01760-1500, 1st edition, March 1996.

[21] C. Jutten and J. Herault. Blind separation of sources, part I: An adaptive algorithm based on neuromimetic architecture. *Signal Processing*, 24:1–10, 1991.

[22] J. Karhunen, A. Cichocki, W. Kasprzak, and P. Pajunen. On neural blind separation with noise suppression and redundancy reduction. *Int. Journal of Neural Systems*, 8(2):219–237, April 1997.

[23] J. Karhunen, E. Oja, L. Wang, R. Vigario, and J. Joutsensalo. A class of neural networks for independent component analysis. *IEEE Trans. on Neural Networks*, 8(3):486–504, 1997.

[24] J. Karhunen and P. Pajunen. Blind source separation and tracking using nonlinear PCA criterion: A least-squares approach. In *Proc. ICNN'97*, Houston, Texas, 1997.

[25] J. Karhunen and P. Pajunen. New results on nonlinear PCA criterion in blind source separation. In C. Fyfe, editor, *Proc. Independence and Artificial Neural Networks Workshop (I&ANN'98)*, Tenerife, Spain, February 1998.

[26] J.M. Mendel. Tutorial on higher-order statistics (spectra) in signal processing and system theory: Theoretical results and some applications. *Proceedings of the IEEE*, 79(3):278–305, 1991.

[27] C.J. Merz and P.M. Murphy. UCI repository of machine learning databases, 1998. http://www.ics.uci.edu/~mlearn/MLRepository.html.

[28] E. Oja. The nonlinear PCA learning rule in independent component analysis. *Neurocomputing*, 17(1):25–46, 1997.

[29] E. Oja, J. Karhunen, and A. Hyvärinen. From neural principal components to neural independent components. In *Proc. Int. Conf. on Artificial Neural Networks (ICANN'98)*, Lausanne, Switzerland, 1997.

[30] P. Pajunen. Blind separation of binary sources with less sensors than sources. In *Proc. ICNN'97*, Houston, Texas, 1997.

[31] Brian D. Ripley. *Pattern recognition and Neural networks.* Cambridge University Press, Cambridge, UK, 1996.

[32] B. Yang. Projection approximation subspace tracking. *IEEE Transactions on Signal Processing*, 43(1):95–107, January 1995.