# Maximum Likelihood and Covariant Algorithms for Independent Component Analysis

David J.C. MacKay
University of Cambridge
Cavendish Laboratory
Madingley Road
Cambridge CB3 0HE
mackay@mrao.cam.ac.uk

January 8th 1999 — Version 3.8 (including minor corrections added October 8, 2002)

**Abstract**

Bell and Sejnowski (1995) have derived a blind signal processing algorithm for a non-linear feedforward network from an information maximization viewpoint. This paper first shows that the same algorithm can be viewed as a maximum likelihood algorithm for the optimization of a linear generative model.

Second, a covariant version of the algorithm is derived. This algorithm is simpler and somewhat more biologically plausible, involving no matrix inversions; and it converges in a smaller number of iterations.

Third, this paper gives a partial proof of the 'folk-theorem' that any mixture of sources with high-kurtosis histograms is separable by the classic ICA algorithm.

Fourth, a collection of formulae are given that may be useful for the adaptation of the non-linearity in the ICA algorithm.

## 1 Blind separation

Algorithms for blind separation (Jutten and Herault 1991; Comon *et al.* 1991; Bell and Sejnowski 1995; Hendin *et al.* 1994) attempt to recover source signals $\mathbf{s}$ from observations $\mathbf{x}$ which are linear mixtures (with unknown coefficients $\mathbf{V}$) of the source signals

$$\mathbf{x} = \mathbf{V}\mathbf{s}. \tag{1}$$

The algorithms attempt to create the inverse of $\mathbf{V}$ (within a post-multiplicative factor) given only a set of examples $\{\mathbf{x}\}$.

Bell and Sejnowski (1995) have derived a blind separation algorithm from an information maximization viewpoint. The algorithm may be summarised as a linear mapping:

$$\mathbf{a} = \mathbf{W}\mathbf{x} \tag{2}$$

followed by a non-linear map:

$$z_i = \phi_i(a_i), \tag{3}$$

where, for example, $\phi = -\tanh(a_i)$, with a learning rule:

$$\Delta\mathbf{W} \propto [\mathbf{W}^{\mathsf{T}}]^{-1} + \mathbf{z}\mathbf{x}^{\mathsf{T}}. \tag{4}$$

Another non-linear function of $a_i$, $y_i = g(a_i)$, is also mentioned by Bell and Sejnowski, but it will not be needed here.

This paper has four parts. First it is shown that Bell and Sejnowski's (1995) algorithm may be derived as a maximum likelihood algorithm. This has been independently pointed out by Pearlmutter and Parra (1996) who also give an exciting generalization of the ICA algorithm.

Second, it is pointed out that the algorithm (4) is not *covariant*, and a covariant algorithm is described which is simpler, faster, and somewhat more biologically plausible. This covariant algorithm has been independently suggested by Amari *et al.* (1996) and is used by Pearlmutter and Parra (1996).

Third, this paper gives a partial proof of the 'folk-theorem' that any mixture of sources with high-kurtosis histograms is separable by the classic ICA algorithm.

Fourth, a collection of formulae are given that may be useful for the adaptation of the non-linearity in the ICA algorithm.

# 2 Maximum likelihood derivation of ICA

## 2.1 Latent variable models

Many statistical models are generative models that make use of latent variables to describe a probability distribution over observables (Everitt 1984).

Examples of latent variable models include mixture models, which model the observables as coming from a superposed mixture of simple probability distributions (Hanson *et al.* 1991) (the latent variables are the unknown class labels of the examples); hidden Markov models (Rabiner and Juang 1986); factor analysis; Helmholtz machines (Hinton *et al.* 1995; Dayan *et al.* 1995); and density networks (MacKay 1995; MacKay 1996).

Note that it is usual for the latent variables to have a simple distribution, often a separable distribution. Thus when we learn a latent variable model, we are finding a description of the data in terms of independent components. One thus might expect that an 'independent component analysis' algorithm should have a description in terms of a generative latent variable model. And this is indeed the case. Independent component analysis is latent variable modelling.

## 2.2 The generative model

Let us model the observable vector $\mathbf{x} = \{x_j\}_{j=1}^J$ as being generated from latent variables $\mathbf{s} = \{s_i\}_{i=1}^I$ via a linear mapping $\mathbf{V}$. The simplest derivation results if we assume $I = J$, *i.e.*, the number of sources is equal to the number of observations. The data we obtain are a set of $N$ observations $D = \{\mathbf{x}^{(n)}\}_{n=1}^N$. We assume that the latent variables are independently distributed, with marginal distributions $P(s_i|\mathcal{H}) \equiv p_i(s_i)$. Here $\mathcal{H}$ denotes the assumed form of this model and the assumed probability distributions $p_i$ of the latent variables.

The probability of the observables and the hidden variables, given $\mathbf{V}$ and $\mathcal{H}$, is:

$$\begin{aligned}
P(\{\mathbf{x}^{(n)}\}_{n=1}^N, \{\mathbf{s}^{(n)}\}_{n=1}^N|\mathbf{V},\mathcal{H}) &= \prod_{n=1}^N \left[P(\mathbf{x}^{(n)}|\mathbf{s}^{(n)},\mathbf{V},\mathcal{H})P(\mathbf{s}^{(n)}|\mathcal{H})\right] \qquad (5) \\
&= \prod_{n=1}^N \left[\left(\prod_j \delta\left(x_j^{(n)} - \textstyle\sum_i V_{ji} s_i^{(n)}\right)\right)\left(\prod_i p_i(s_i^{(n)})\right)\right]. \qquad (6)
\end{aligned}$$

Here it has been assumed that the vector $\mathbf{x}$ is generated without noise, because this is the assumption which leads to the Bell-Sejnowksi algorithm. It is straightforward to give another derivation in which the term $\delta\left(x_j^{(n)} - \sum_i V_{ji} s_i^{(n)}\right)$ is replaced by a probability distribution over $x_j^{(n)}$ with mean $\sum_i V_{ji} s_i^{(n)}$. If this noise distribution has sufficiently small standard deviation then the identical algorithm results.

### 2.3 The likelihood function

For learning about $\mathbf{V}$ from the data $D$, the relevant quantity is the likelihood function

$$P(D|\mathbf{V},\mathcal{H}) = \prod_n P(\mathbf{x}^{(n)}|\mathbf{V},\mathcal{H}) \tag{7}$$

which is a product of factors each of which is obtained by marginalizing over the latent variables. We adopt summation convention at this point, such that, for example, $V_{ji} s_i^{(n)} \equiv \sum_i V_{ji} s_i^{(n)}$. A single factor in the likelihood is given by[1]

$$\begin{aligned}
P(\mathbf{x}^{(n)}|\mathbf{V},\mathcal{H}) &= \int d^I \mathbf{s}^{(n)}\, P(\mathbf{x}^{(n)}|\mathbf{s}^{(n)},\mathbf{V}) P(\mathbf{s}^{(n)}) & (8)\\
&= \int d^I \mathbf{s}^{(n)} \prod_j \delta\left(x_j^{(n)} - V_{ji} s_i^{(n)}\right) \prod_i p_i(s_i^{(n)}) & (9)\\
&= \frac{1}{|\det \mathbf{V}|} \prod_i p_i(V_{ij}^{-1} x_j) & (10)\\
\Rightarrow \log P(\mathbf{x}^{(n)}|\mathbf{V},\mathcal{H}) &= -\log|\det \mathbf{V}| + \sum_i \log p_i(V_{ij}^{-1} x_j). & (11)
\end{aligned}$$

To obtain a maximum likelihood algorithm we find the gradient of the log likelihood. If we introduce $\mathbf{W} \equiv \mathbf{V}^{-1}$, the log likelihood for a single example may be written:

$$\log P(\mathbf{x}^{(n)}|\mathbf{V},\mathcal{H}) = \log\det \mathbf{W} + \sum_i \log p_i(W_{ij} x_j). \tag{12}$$

We will need the following identities:

$$\begin{aligned}
\frac{\partial}{\partial V_{ji}} \log\det \mathbf{V} &= V_{ij}^{-1} = W_{ij} & (13)\\
\frac{\partial}{\partial V_{ji}} V_{lm}^{-1} &= -V_{lj}^{-1} V_{im}^{-1} = -W_{lj} W_{im} & (14)\\
\frac{\partial}{\partial W_{ij}} f &= -V_{jm} \left(\frac{\partial}{\partial V_{lm}} f\right) V_{li}. & (15)
\end{aligned}$$

Let us define $a_i \equiv W_{ij} x_j$, $\phi_i(a_i) \equiv d\log p_i(a_i)/da_i$ and $z_i = \phi_i(a_i)$, which indicates in which direction $a_i$ needs to change to make the probability of the data greater. We may then obtain the gradient with respect to $V_{ji}$ using equations (13) and (14):

$$\frac{\partial}{\partial V_{ji}} \log P(\mathbf{x}^{(n)}|\mathbf{V},\mathcal{H}) = -W_{ij} - a_i z_{i'} W_{i'j}. \tag{16}$$

Or alternatively, the derivative with respect to $W_{ij}$:

$$\frac{\partial}{\partial W_{ij}} \log P(\mathbf{x}^{(n)}|\mathbf{V},\mathcal{H}) = V_{ji} + x_j z_i. \tag{17}$$

If we choose to change $\mathbf{W}$ so as to ascend this gradient, we obtain precisely the learning algorithm in Bell and Sejnowski (1995) (equation 4).

[1] Recall that for scalars, $\int ds\, \delta(x - vs) f(s) = \frac{1}{|v|} f(x/v)$.

### 2.4 Examples

To help explain the generative modelling viewpoint, figures 1a-c illustrate typical distributions generated by the independent components model when the components have $1/\cosh$ and Cauchy distributions. Figure 1d shows some samples from the Cauchy model. The Cauchy distribution, being the more heavy–tailed, gives the clearest picture of how the predictive distribution depends on the assumed generative parameters $\mathbf{V}$.

The two best known special cases are:

1. No nonlinearity. If $\phi_i(a_i) = -\kappa a_i$, then implicitly we are assuming a Gaussian distribution on the latent variables. It is well known that the H-J algorithm only works because of its non-linearities; if the algorithm has linear output $z$ then all that is obtained is second-order decorrelation. Equivalently, the Gaussian distribution on the latent variables is invariant under rotation of the latent variables, so there can be no evidence favouring any particular alignment of the latent variable space.

2. A tanh nonlinearity. If $\phi_i(a_i) = -\tanh(a_i)$ then implicitly we are assuming $p_i(s_i) \propto 1/\cosh(s_i) \propto \frac{1}{e^{s_i}+e^{-s_i}}$. This is a heavier-tailed distribution for the latent variables than the Gaussian distribution. But heavier tails still would be a possibility.

3. We could also use a tanh nonlinearity with gain $\beta$, that is, $\phi_i(a_i) = -\tanh(\beta a_i)$. As $\beta$ varies the implied probabilistic model changes, and is given by $p_i(s_i) \propto 1/[\cosh(\beta s_i)]^{1/\beta}$. In the limit of large $\beta$, the non-linearity becomes a step function and the probability distribution $p_i(s_i)$ becomes a biexponential distribution, $p_i(s_i) \propto \exp(-|s|)$. In the limit $\beta \to 0$ $p_i(s_i)$ approaches a Gaussian with mean zero and variance $1/\beta$.

## 3 A covariant, simpler and faster learning algorithm

We have thus derived a learning algorithm which performs steepest descents on the likelihood function.

Some designers of learning algorithms advocate the principle of covariance, which says, colloquially, that a consistent algorithm should give the same results independent of the units in which quantities are measured (Knuth 1968).

A prime example of a *non*-covariant algorithm is the popular steepest descents rule. A dimensionless objective function $L(\mathbf{w})$ is defined, its derivative with respect to some parameters $\mathbf{w}$ is computed, and then $\mathbf{w}$ is changed by the rule

$$\Delta w_i = \eta \frac{\partial L}{\partial w_i}. \tag{18}$$

This popular equation is dimensionally inconsistent: the left hand side of this equation has dimensions of $[w_i]$ and the right hand side has dimensions $1/[w_i]$. The behaviour of the learning algorithm (18) is not covariant with respect to linear rescaling of the vector $\mathbf{w}$. Dimensional inconsistency is not the end of the world, as the success of the backpropagation algorithm has demonstrated, and indeed if $\eta$ decreases with $n$ (during on-line learning) as $1/n$ then the Munro-Robbins theorem (Bishop 1992:p.41) shows that the parameters will asymptotically converge to the maximum likelihood parameters. But the non-covariant algorithm may take a very large number of iterations to achieve this convergence; indeed many former users of steepest descents algorithms prefer to use algorithms such as conjugate gradients that adaptively figure out the curvature of the objective function. The defense of equation (18) that points out $\eta$ could be a dimensional constant is untenable if not all the parameters $w_i$ have the same dimensions.

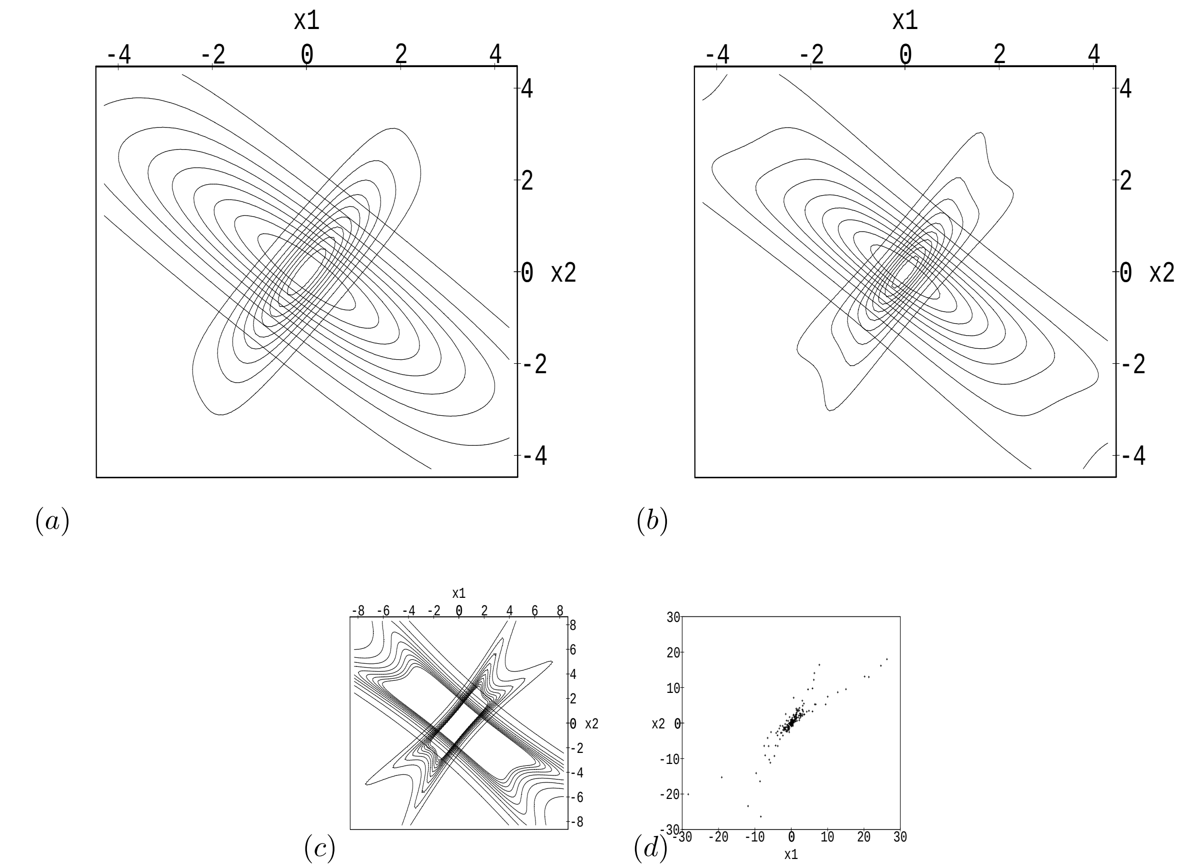


Figure 1: Illustration of the generative models implicit in the learning algorithm. (a) Distributions over two observables generated by $1/\cosh$ distributions on the latent variables, for $\mathbf{V} = \begin{bmatrix} 3/4 & 1/2 \\ 1/2 & 1 \end{bmatrix}$ (compact distribution) and $\mathbf{V} = \begin{bmatrix} 2 & -1 \\ -1 & 3/2 \end{bmatrix}$ (broader distribution). (b) Contours of the generative distributions when the latent variables have Cauchy distributions. The learning algorithm fits this amoeboid object to the empirical data in such a way as to maximize the likelihood. The contour plot in (b) does not adequately represent this heavy-tailed distribution. (c) Part of the tails of the Cauchy distribution, giving the contours $0.01 \ldots 0.1$ times the density at the origin. (d) Some data from one of the generative distributions illustrated in (b) and (c). Can you tell which? 200 samples were created, of which 196 fell in the plotted region.

The algorithm would be covariant if it had the form

$$\Delta w_i = \eta \sum_{i'} M_{ii'} \frac{\partial L}{\partial w_i}, \tag{19}$$

where $\mathbf{M}$ is a matrix whose $i, i'$ element has dimensions $[w_i w_{i'}]$. From where can we obtain such a matrix? Two sources of such matrices are *metrics* and *curvatures*.

### 3.1 Metrics and curvatures

If there is a natural metric that defines *distances* in our parameter space $\mathbf{w}$, then a matrix $\mathbf{M}$ can be obtained from the metric. There is often a natural choice. In the special case where there is a known quadratic metric defining the length of a vector $\mathbf{w}$, then the matrix can be obtained from the quadratic form. For example if the length is $\mathbf{w}^2$ then the natural matrix is $\mathbf{M} = \mathbf{I}$, and steepest descents is appropriate.

Another way of finding a metric is to look at the curvature of the objective function, defining $\mathbf{A} \equiv -\nabla\nabla L$ (where $\nabla \equiv \partial/\partial\mathbf{w}$). Then the matrix $\mathbf{M} = \mathbf{A}^{-1}$ will give a covariant algorithm; what is more, this algorithm is the Newton algorithm, so we recognize that it will alleviate one of the principle difficulties with steepest descents, namely its slow convergence to a minimum when the objective function is at all ill-conditioned. The Newton algorithm converges to the minimum in a single step if $L$ is quadratic.

In some problems it may be that the curvature $\mathbf{A}$ consists of both data-dependent terms and data-independent terms; in this case, one might choose to define the metric using the data-independent terms only (Gull 1989). The resulting algorithm will still be covariant but it will not implement an exact Newton step. Obviously there are many covariant algorithms; there is no unique choice. But covariant algorithms are a small subset of the set of *all* algorithms!

For the present maximum likelihood problem we have evaluated the gradient with respect to $\mathbf{V}$ and the gradient with respect to $\mathbf{W} = \mathbf{V}^{-1}$. Bell and Sejnowski (1995) chose to perform steepest ascents in $\mathbf{W}$, a procedure which is not covariant. Let us construct an alternative algorithm that is covariant with the help of the curvature of the log likelihood. Taking the second derivative of the log likelihood with respect to $\mathbf{W}$ we obtain two terms, the first of which is data-independent:

$$\frac{\partial V_{ji}}{\partial W_{kl}} = -V_{jk} V_{li}, \tag{20}$$

and the second of which is data-dependent:

$$\frac{\partial (z_i x_j)}{\partial W_{kl}} = x_j x_l \delta_{ik} z'_i, \text{(no sum over } i) \tag{21}$$

where $z'$ is the derivative of $z$. It is tempting to drop the data-dependent term and define the matrix $\mathbf{M}$ by $[M^{-1}]_{(ij)(kl)} = [V_{jk} V_{li}]$. However, this matrix is not positive definite (it has at least one non-positive eigenvalue), so it is a poor approximation to the curvature of the log likelihood, which must be positive definite in the neighbourhood of a maximum likelihood solution. We must therefore consult the data-dependent term for inspiration. The aim is to find a convenient approximation to the curvature and to obtain a covariant algorithm, not necessarily to implement an exact Newton step. What is the average value of $x_j x_l \delta_{ik} z'_i$? If the true value of $\mathbf{V}$ is $\mathbf{V}^*$, then

$$\langle x_j x_l \delta_{ik} z'_i \rangle = \left\langle V^*_{jm} s_m s_n V^*_{ln} \delta_{ik} z'_i \right\rangle. \tag{22}$$

We now make several severe approximations: we replace $\mathbf{V}^*$ by the present value of $\mathbf{V}$, and replace the correlated average $\langle s_m s_n z'_i \rangle$ by $\langle s_m s_n \rangle \langle z'_i \rangle \equiv \Sigma_{mn} D_i$. Here $\Sigma$ is the variance–covariance matrix of the latent variables (which is assumed to exist), and $D_i$ is the typical value of the curvature $d^2 \log p_i(a)/da^2$. Given that the sources are assumed to be independent, $\Sigma$ and $D$ are both diagonal matrices. These approximations motivate the matrix $\mathbf{M}$ given by:

$$[M^{-1}]_{(ij)(kl)} = V_{jm}\Sigma_{mn}V_{ln}\delta_{ik}D_i, \tag{23}$$

that is,

$$M_{(ij)(kl)} = W_{mj}\Sigma^{-1}_{mn}W_{nl}\delta_{ik}D_i^{-1} \tag{24}$$

For simplicity, we further assume that the sources are similar to each other so that $\Sigma$ and $D$ are both homogeneous and that $\Sigma D = 1$. This will ead us to an algorithm that is covariant with respect to linear rescaling of the data $x$, but not with respect to linear rescaling of the latent variables. For problems where these assumptions do not hold, it will be straightforward to retain inhomogeneous $\Sigma$ and $D$. We thus use:

$$M_{(ij)(kl)} = W_{mj}W_{ml}\delta_{ik} \tag{25}$$

Multiplying this matrix by the gradient in equation (17) we obtain the following covariant learning algorithm:

$$\Delta W_{ij} = \eta \left(W_{ij} + W_{i'j}a_{i'}z_i\right) \tag{26}$$

Notice that this expression does not require any inversion of the matrix $\mathbf{W}$. The only additional computation once $\mathbf{z}$ has been computed is a single backward pass through the weights to compute the quantity

$$x'_j = W_{i'j}a_{i'} \tag{27}$$

in terms of which the covariant algorithm reads:

$$\Delta W_{ij} = \eta \left(W_{ij} + x'_j z_i\right). \tag{28}$$

### 3.2 Control of the step size

Finally, a comment should be made regarding the value of the dimensionless quantity $\eta$, assuming that we implement the covariant algorithm 'on–line'. If $\eta$ is held to a constant value then one is implicitly solving a weighted maximum likelihood problem with an exponential weighting of the data. The parameters will not converge to a limiting value but will diffuse around the vicinity of the maximum likelihood parameters as old data points are forgotten. If one wants all data points to receive equal weight so that the parameters to converge to a limiting value then $\eta$ should go as $1/n$ asymptotically, where $n$ is the number of the current data point. If the objective function is quadratic and the algorithm implements the Newton step exactly then $\eta$ going as $1/n$ causes the parameters to be exactly the maximum likelihood parameters for all $n$. If the objective function is not quadratic and we didn't choose the matrix so that the Newton step is performed then at the $n$th iteration the parameters are not necessarily equal to the maximum likelihood parameters, but asymptotically the parameters are guaranteed to converge to the maximum likelihood parameters, by the Munro-Robbins theorem (Bishop 1992:p.41).

### 3.3 Comments

This covariant algorithm is simpler than the Bell and Sejnowski algorithm: it has no dependence on $\mathbf{V}$, so it requires no matrix inversion at all. This algorithm is thus not only dimensionally consistent but it is also somewhat closer to biological plausibility.

Key points:

1. The Bell-Sejnowski algorithm is non-local, because it involves a matrix inversion. The covariant algorithm is local, involving only a single extra back-propagation through $\mathbf{W}$.

2. The Bell-Sejnowski algorithm is not robust, because if the parameter matrix $\mathbf{W}$ strays towards a value that is not invertible, then the gradient will diverge. In contrast, the covariant algorithm has no such singular behaviour.

3. To make their algorithm well-behaved, Bell and Sejnowski made two modifications: (a) they spherized the data first, transforming to a representation in which the data has homogenous second order properties; (b) they used 'batches' when training. Without these two modifications, the algorithm is very poorly conditioned. Both these steps introduce a lag into the learning process.

   The preprocessing step can be interpreted as a way of making their steepest descents algorithm covariant. But it is quicker not to spherize the data and to use the covariant algorithm.

4. Note that if you want to solve a real application where $\mathbf{W}$ might be continuously adapting in time, the preprocessing requirement is kind of cumbersome.

### 3.4 Demonstration

I have made a simple comparison of the time taken for the two algorithms to solve a $2 \times 2$ decorrelation problem (without spherizing the data).

Data was generated using a biexponential distribution for the sources $s_1$ and $s_2$ and with $\mathbf{V} = [[2,1],[3,1]]$ (that is, source 1 is twice as loud as source 2 at microphone 1, and it is three times as loud at microphone 2).

For practical purposes one might wish to use 'momentum' $\mu$, and / or batching of the data. For simplicity I used no momentum and used batches of size either 1 or 200. (200 was the batch size used by Bell and Sejnowski.) The parameter $\eta$ was either optimized by trial and error to a single value giving the fastest convergence, or else it was set to go as $1/n$.

The results are shown in figures 2 and 3.

Notice how much faster the covariant algorithm is, and note the noisiness of the Bell-Sejnowski algorithm. Note also that the scatterplots show that the B-S algorithm has not converged to a perfect solution. There is still easily detected mixing of the two sources.

## 4 Robustness to choice of nonlinear function

The literature on independent component analysis contains a folk theorem that the algorithm is robust to the details of the sources that are being separated, and that in particular the $(-\tanh)$ function is likely to give good results for any sources with large kurtosis — that is, any sources with sufficiently heavy tails.

I now give a *partial* proof of this folk theorem. The following theorem is less general than the folk theorem in that it is only proven for two sources, for a limited set of high kurtosis distributions, and with both sources having the same distribution. It is slightly more general than the folk theorem in that it shows that a tanh function of any gain $\beta$ (including a step function) has the same robustness property.

Without loss of generality we assume that the true weights are $\mathbf{W} = \mathbf{V} = \mathbf{I}$, with the question then being, 'could ICA find an alternative $\mathbf{W}$ which achieves greater likelihood?' The following theorem asserts that any other $\mathbf{W}$ (apart from trivial cases) achieves a lesser expected log likelihood.

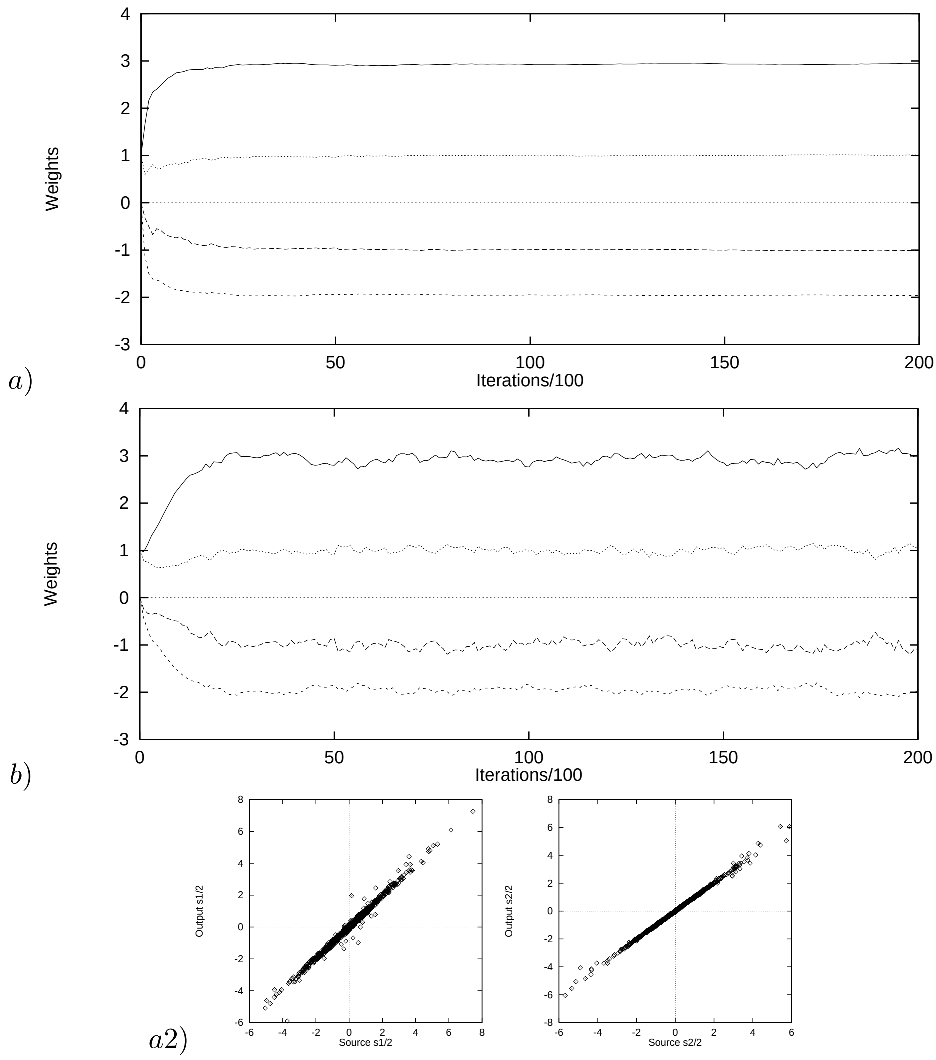


Figure 2: (a) Time course of weights using **covariant** algorithm with $\eta = 1/n$. (b) $\eta = 0.002$. (a2) Scatter plot of outputs versus true source signals, from example 20000 to 30000 using the weights in (a).

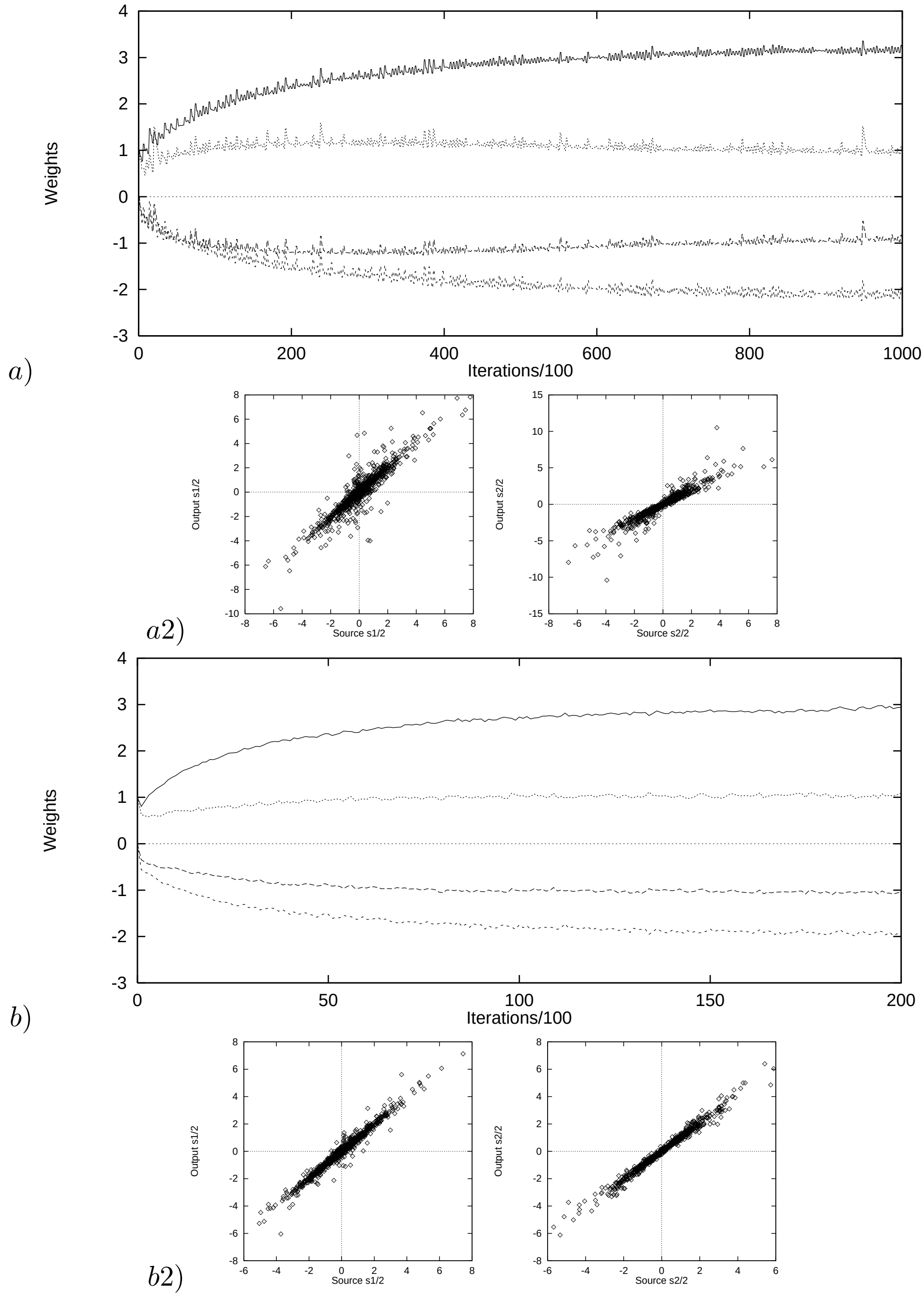


Figure 3: (a) Time course of weights using Bell-Sejnowski algorithm with $\eta = 0.0005$, batch size 200. (Faster learning rates $\eta$ proved unstable.) (a2) Scatter plot of outputs versus true source signals, using the weights in (a). Note, comparing with figure 2, that the horizontal time axis is 5 times longer. (b) $\eta = 0.002$, batch size 1. (Faster learning rates proved unstable.)

**Theorem 1** *Let data be generated from two sources both having symmetric density $p(s) = p(-s)$. Let $x_1 = s_1$ and $x_2 = s_2$, and define $P(\mathbf{x}) = p(x_1)p(x_2)$. Let the data be modelled with $Q(\mathbf{x}|\mathbf{W}) = \det \mathbf{W}\, Q_s(\mathbf{Wx})$ where $Q_s(\mathbf{s}) = \prod_i q(s_i)$, with $d \log q(s)/ds = -\tanh(\beta s)$. If the true source density $p(s)$ is a continuous density that satisfies the heavy-tail condition:*

$$\frac{d}{ds}\left(\frac{1}{s}\frac{d \log p}{ds}\right) > 0 \ \forall s > 0 \tag{29}$$

*and if $\mathbf{W}$ is a matrix with determinant 1 and $\mathbf{I}$ is the $2 \times 2$ identity matrix then*

$$\int d^2\mathbf{x}\, P(\mathbf{x}) \log Q(\mathbf{x}|\mathbf{I}) \geq \int d^2\mathbf{x}\, P(\mathbf{x}) \log Q(\mathbf{x}|\mathbf{W}) \tag{30}$$

*with equality holding only if $\mathbf{W} = \mathbf{I}$ and in the trivial cases where $\mathbf{W}$ is a rotation-reflection through $n\pi/2$.*

Comment: the constraint that $\mathbf{W}$ be a matrix with determinant 1 does not weaken the theorem. If someone claims to find a non-trivial solution which has $\det \mathbf{W} = \Delta \neq 1$, then we can stretch the axes of $\mathbf{s}$-space by a factor of $\sqrt{\Delta}$; then the theorem says that by replacing $\mathbf{W}$ by $\mathbf{I}$, the expected value of the likelihood function will be increased. Thus the likelihood is maximized by a $\mathbf{W}$ proportional to the identity matrix.

## 4.1 Proof

The proof proceeds in two steps, based on the following decomposition of $\mathbf{W}$. A $2 \times 2$ transformation $\mathbf{W}$ with determinant 1 has three degrees of freedom that can be represented in terms of pure shears $\mathbf{S}(\zeta) = \begin{bmatrix} e^{\zeta} & 0 \\ 0 & e^{-\zeta} \end{bmatrix}$ and pure rotations $\mathbf{R} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$ thus:

$$\mathbf{W} = \mathbf{E}^n \mathbf{R}(\theta_2)\mathbf{S}(\zeta)\mathbf{R}(\theta_1), \tag{31}$$

where $\mathbf{E} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$ and $n$ is zero or one. [We will ignore this optional inversion in the following.]

In the first step we will show that for any probability distribution $p(s)$ and for any rotations $(\theta_1, \theta_2)$, the biggest likelihood is achieved by using a shear-less transformation ($\zeta = 0$). In the second step we show that out of all rotations, rotation through $\theta = 0$ gives the greatest likelihood.

### 4.1.1 Zero shear is best

Let $\mathbf{W} = \mathbf{R}(\theta_2)\mathbf{S}(\zeta)\mathbf{R}(\theta_1)$, with $\theta_1$ and $\theta_2$ fixed. We will show that out of all values of $\zeta$, $\zeta = 0$ achieves the greatest likelihood.

Transforming to the basis $y_1, y_2$ in which the shear is diagonal $\mathbf{S}(\zeta) = \begin{bmatrix} e^{\zeta} & 0 \\ 0 & e^{-\zeta} \end{bmatrix}$, the density $P(\mathbf{x})$ transforms to $P(\mathbf{y}) = p\left[\cos(\theta)y_1 + \sin(\theta)y_2\right] p\left[\cos(\theta)y_2 - \sin(\theta)y_1\right]$ and $Q(\mathbf{x})$ transforms to

$$\begin{aligned} Q(\mathbf{y}) &= \frac{1}{Z} \Big/ \cosh\left[\cos(\theta_2)e^{\zeta}y_1 - \sin(\theta_2)e^{-\zeta}y_2\right] \cosh\left[\cos(\theta_2)e^{-\zeta}y_2 + \sin(\theta_2)e^{\zeta}y_1\right] && (32) \\ &\equiv 1 \Big/ \cosh\left(ce^{\zeta}y_1 - se^{-\zeta}y_2\right) \cosh\left(ce^{-\zeta}y_2 + se^{\zeta}y_1\right), && (33) \end{aligned}$$

where $c \equiv \cos\theta_2$ and $s \equiv \sin\theta_2$ and $Z$ is independent of $\zeta$, $\theta_1$ and $\theta_2$. To prove that $\zeta = 0$ is the maximum likelihood shear, the only property of $p$ that we will need is that the density $P(\mathbf{y})$ is invariant under a rotation through 90 degrees. That is $P(\mathbf{y}) = P(\mathbf{y}^{\perp})$, where $\mathbf{y}^{\perp} = (y_2, -y_1)$. So the integral of interest

$$L(\zeta) = \int P(\mathbf{y}) \log Q(\mathbf{y}) \tag{34}$$

can be rewritten

$$\begin{aligned} L(\zeta) &= \frac{1}{2} \int P(\mathbf{y}) \left[ \log Q(\mathbf{y}) + \log Q(\mathbf{y}^{\perp}) \right] & (35) \\ &= -\frac{1}{2} \int P(\mathbf{y}) \log \Big[ \cosh\left( c e^{\zeta} y_1 - s e^{-\zeta} y_2 \right) \cosh\left( c e^{-\zeta} y_2 + s e^{\zeta} y_1 \right) \\ &\qquad \cosh\left( c e^{\zeta} y_2 + s e^{-\zeta} y_1 \right) \cosh\left( -c e^{-\zeta} y_1 + s e^{\zeta} y_2 \right) \Big] & (36) \\ &= -\frac{1}{2} \int P(\mathbf{y}) \log \Big[ \Big( \cosh\left( (c y_1 + s y_2)(e^{\zeta} - e^{-\zeta}) \right) + \cosh\left( (c y_1 - s y_2)(e^{\zeta} + e^{-\zeta}) \right) \Big) \\ &\qquad \Big( \cosh\left( (c y_2 + s y_1)(e^{\zeta} + e^{-\zeta}) \right) + \cosh\left( (s y_1 - c y_2)(e^{\zeta} - e^{-\zeta}) \right) \Big) \Big] . & (37) \end{aligned}$$

Here, we have used $\cosh A \cosh B = \cosh(A+B) + \cosh(A-B)$. Examining the product inside the logarithm, we notice that $\cosh\left((cy_1 + sy_2)(e^{\zeta} - e^{-\zeta})\right)$ achieves its minimum value of 1 only when $\zeta = 0$ (unless $(cy_1 + sy_2) = 0$, in which case this particular term is independent of $\zeta$); $\cosh\left((cy_1 - sy_2)(e^{\zeta} + e^{-\zeta})\right)$ achieves its minimum value when $(e^{\zeta} + e^{-\zeta})$ is minimized, that is, when $\zeta = 0$ (unless $(cy_1 - sy_2) = 0$, in which case this particular term is independent of $\zeta$); and similarly the other two terms are minimized when $\zeta = 0$. So for any non-zero $\mathbf{y}$ the argument of the logarithm is minimized by $\zeta = 0$ (and only by this value of $\zeta$). So the maximum likelihood $\mathbf{W}$ is shearless.

### 4.1.2 Zero rotation is best

Having proved that the maximum likelihood $\mathbf{W}$ is shearless, we now need to prove that the maximum likelihood $\mathbf{W}$ of the form $\mathbf{W} = \mathbf{R}(\theta)$ has $\theta = 0$ (or a multiple of $\pi/2$). We restrict our attention to cases $\theta \in (0, \pi/4)$, since all other cases can be reduced to this interval. We parameterize $\mathbf{x}$ space by $\mathbf{x} = (r, \phi)$ such that $(x_1, x_2) = (r\cos\phi, r\sin\phi)$; we use the same notation for $\mathbf{y} = \mathbf{R}(\theta)\mathbf{x} = (r, \phi + \theta)$, and write $Q$ thus:

$$Q(r, \phi | \theta) = q(r\cos(\phi + \theta)) q(r \sin(\phi + \theta)), \tag{38}$$

where $q(s) = 1/(Z\cosh(s))$. The crucial property required here is that $P(r, \phi) = p(r\cos\phi)p(r\sin\phi)$ is a decreasing function of $\phi$ for $\phi \in (0, \pi/4)$, if $p$ satisfies the condition of the theorem.
**Proof:** If $(x_1, x_2) = (r\cos\phi, r\sin\phi)$ and $x_1 > x_2 > 0$ then

$$\begin{aligned} \frac{\partial}{\partial \phi} \log[p(r\cos\phi)p(r\sin\phi)] &= -\frac{d}{dx_1} \log p(x_1) r \sin\phi + \frac{d}{dx_2} \log p(x_2) r\cos\phi & (39) \\ &= x_1 x_2 \left\{ -\frac{1}{x_1} \frac{d}{dx_1} \log p(x_1) + \frac{1}{x_2} \frac{d}{dx_2} \log p(x_2) \right\} & (40) \\ &> 0 \text{ if } \frac{d}{ds} \left( \frac{1}{s} \frac{d \log p}{ds} \right) > 0 \ \forall s > 0. & (41) \end{aligned}$$

For example, since $1/\cosh(s)$ satisfies this heavy tail condition, $Q(r, \phi | 0)$ is a decreasing function of $\phi$ for $\phi \in (0, \pi/4)$. Notice that under the transformation $\mathbf{R}(\theta)$, for every point $\mathbf{x}_{\mathrm{A}} = (r, \phi)$ that is mapped to $\mathbf{y}_{\mathrm{A}} = (r, \phi + \theta)$, there is a point $\mathbf{x}_{\mathrm{B}} = (r, -(\phi + \theta))$ that is mapped to $\mathbf{y}_{\mathrm{B}} = (r, -\phi)$,

and these points satisfy $Q(r, \theta + \phi|\theta) = Q(r, -\theta - \phi|\theta) = Q(r, \phi|0)$.
**Proof**:

$$Q(r,-\theta-\phi|\theta) = q\left[r\cos(-\phi)\right] q\left[r\sin(-\phi)\right] = q\left[r\cos(\phi)\right] q\left[r\sin(\phi)\right] = Q(r, \phi|0). \tag{42}$$

Using these symmetries we can obtain

$$L(0)-L(\theta) \;=\; 4\int dr \int_{\phi=-\theta/2}^{\pi/4-\theta/2} d\phi \left[P(r, \phi) - P(r, \phi+\theta)\right] \left[\log Q(r, \phi|0) - \log Q(r, \phi+\theta|0)\right] \tag{43}$$

Now throughout the range of this integral, both the terms $[P(r, \phi) - P(r, \phi + \theta)]$ and $[\log Q(r, \phi|0) - \log Q(r, \phi + \theta|0)]$ are positive because $P$ and $Q$ are both monotonic as proved at equation (41). This completes the proof.

Perhaps a more general theorem could be proved which relies on some moment properties of $p$ rather than a heavy-tailedness defined in terms of smoothness properties.

# 5 Learning of the nonlinearity

Let us conclude by discussing how one might learn the density on the latent variables. (Bell and Sejnowski 1995) discuss the concept of learning the nonlinearity, but don't give an explicit algorithm for doing this. One can construct a family of parameterized distributions $p_i$ (which must be explicitly normalized), then differentiate the log likelihood.

## 5.1 Example of parameterization of non-linearity: Student distribution

$$P(x|\nu) = \frac{\Gamma[(\nu+1)/2]}{(\pi\nu)^{1/2}\Gamma[\nu/2]} \frac{1}{[1+(x^2/\nu)]^{(\nu+1)/2}} \tag{44}$$

The derivative $\phi$ is given by:

$$d\log P(x|\nu)/dx = -\frac{x\,(\nu+1)}{\nu+x^2} \tag{45}$$

Learning of $\nu$ is achieved using the following gradient:

$$\frac{d}{d\nu}\log P(x|\nu) = \frac{1}{2}\left[\Psi\left(\frac{\nu+1}{2}\right) - \Psi\left(\frac{\nu}{2}\right) - \ln\left(\frac{\nu+x^2}{\nu}\right) + \frac{x^2-1}{\nu+x^2}\right], \tag{46}$$

where the digamma function $\Psi(x) = \frac{d}{dx}\log\Gamma(x)$. The following approximation is accurate to within 8% for all $t$ (MacKay and Peto 1995):

$$\Psi(t+1/2) - \Psi(t) \simeq 1/t + \log(t/(t+1/2)). \tag{47}$$

## 5.2 Learning of a tanh non-linearity

Let the distribution be

$$p(x|\beta) = \frac{1/Z(\beta)}{(\cosh(\beta x))^{1/\beta}}. \tag{48}$$

As $\beta$ increases this distribution tends to the biexponential distribution. The quantity $z$ in the learning algorithm is given by

$$z = d\log p/dx = -d/dx\frac{1}{\beta}\log\cosh(\beta x) = -\tanh(\beta x). \tag{49}$$

The normalizing constant $Z(\beta)$ is given by the numerical approximation:

$$\log Z(\beta) = a \log\left(\frac{c}{\beta} + 1\right) + b \tag{50}$$

where $c = 1.397$, $a = 0.522$ and $b = 0.692$. Using this formula we get

$$d \log Z / d(\log \beta) \simeq -ac/(\beta + c) \tag{51}$$

To learn $\beta$ we need the derivative

$$d \log p(x|\beta)/d\beta \simeq -ac/(\beta + c) + \log(\cosh(\beta x))/\beta^2 - \frac{x}{\beta}\tanh(\beta x) \tag{52}$$

### 5.3 Non-equal dimensionalities

The **real** cocktail party problem, $I > J$ is a challenge not addressed in this paper.

Let us now work out a learning algorithm for the case $I < J$ (*i.e.*, fewer sources than measurements).

We define a generative model $\mathbf{V}$ with pseudoinverse $\mathbf{W} \equiv \left[\mathbf{V}^{\mathsf{T}}\mathbf{V}\right]^{-1}\mathbf{V}^{\mathsf{T}}$. We replace the $\delta$ function above by a narrow Gaussian distribution. We assume for simplicity that the noise on each component $j$ is indepependent with variance $\sigma_\nu^2 \equiv 1/\beta$.

The likelihood function is:

$$P(\{\mathbf{x}\}|\mathbf{V}) = \prod_n \int d^I\mathbf{s}\, P(\mathbf{x}^{(n)}|\mathbf{s}, \mathbf{V}) P(\mathbf{s}). \tag{53}$$

Let us assume that the noise level $\sigma_\nu^2$ is sufficiently small that the term $P(\mathbf{x}^{(n)}|\mathbf{s}, \mathbf{V})$ has a sharp peak that dominates each integral. Then each term in the product is a Gaussian integral and the log likelihood for a single term is (c.f. (MacKay 1992; Bretthorst 1988; Green and MacKay 1996)):

$$\log P(\mathbf{x}^{(n)}|\mathbf{V}) = \frac{J}{2}\log\frac{\beta}{2\pi} - \frac{\beta}{2}\left(\mathbf{x}^{(n)} - \mathbf{V}\mathbf{s}_{\mathrm{MP}}\right)^2 - \frac{1}{2}\log\det\left(\frac{\mathbf{V}^{\mathsf{T}}\mathbf{V}}{2\pi}\right) + \log P(\mathbf{s}_{\mathrm{MP}}), \tag{54}$$

where

$$\mathbf{s}_{\mathrm{MP}} = \mathbf{W}\mathbf{x}. \tag{55}$$

This expression may be differentiated to obtain a learning rule for $\mathbf{V}$. However, it may be more interesting to pursue an alternative algorithm in which we do not assume the ability to compute the pseudoinverse $\mathbf{W}$.

[Work in progress]


### Acknowledgements

I thank Robert MacKay and Graeme Mitchison for helpful discussions. This work was supported by the Gatsby charitable foundation.


## References


Amari, S., Cichocki, A., and Yang, H., (1996) A new learning algorithm for blind signal separation. To appear in NIPS96.

Bell, A. J., and Sejnowski, T. J. (1995) An information maximization approach to blind separation and blind deconvolution. *Neural Computation* **7** (6): 1129–1159.

Bishop, C. M. (1992) Exact calculation of the Hessian matrix for the multilayer perceptron. *Neural Computation* **4** (4): 494–501.

Bretthorst, G. (1988) *Bayesian spectrum analysis and parameter estimation.* Springer. Also available at `bayes.wustl.edu`.

Comon, P., Jutten, C., and Herault, J. (1991) Blind separation of sources .2. problems statement. *Signal Processing* **24** (1): 11–20.

Dayan, P., Hinton, G. E., Neal, R. M., and Zemel, R. S. (1995) The Helmholtz machine. *Neural Computation* **7** (5): 889–904.

Everitt, B. S. (1984) *An Introduction to Latent Variable Models*. London: Chapman and Hall.

Green, A. G., and MacKay, D. J. C. (1996) Bayesian analysis of linear phased-array radar. In *Maximum Entropy and Bayesian Methods, Santa Barbara 1993*, ed. by G. Heidbreder, pp. 309–318, Dordrecht. Kluwer.

Gull, S. F. (1989) Developments in maximum entropy data analysis. In *Maximum Entropy and Bayesian Methods, Cambridge 1988*, ed. by J. Skilling, pp. 53–71, Dordrecht. Kluwer.

Hanson, R., Stutz, J., and Cheeseman, P. (1991) Bayesian classification with correlation and inheritance. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence, Sydney, Australia*, volume 2, pp. 692–698. Morgan Kaufmann.

Hendin, O., Horn, D., and Hopfield, J. J. (1994) Decomposition of a mixture of signals in a model of the olfactory- bulb. *Proceedings of the National Academy of Sciences of the United States of America* **91** (13): 5942–5946.

Hinton, G. E., Dayan, P., Frey, B. J., and Neal, R. M. (1995) The wake-sleep algorithm for unsupervised neural networks. *Science* **268** (5214): 1158–1161.

Jutten, C., and Herault, J. (1991) Blind separation of sources .1. an adaptive algorithm based on neuromimetic architecture. *Signal Processing* **24** (1): 1–10.

Knuth, D. E. (1968) *The art of computer programming. Volume 1: fundamental algorithms.* Reading, Mass.: Addison Wesley.

MacKay, D. J. C. (1992) Bayesian interpolation. *Neural Computation* **4** (3): 415–447.

MacKay, D. J. C. (1995) Bayesian neural networks and density networks. *Nuclear Instruments and Methods in Physics Research, Section A* **354** (1): 73–80.

MacKay, D. J. C. (1996) Density networks and their application to protein modelling. In *Maximum Entropy and Bayesian Methods, Cambridge 1994*, ed. by J. Skilling and S. Sibisi, pp. 259–268, Dordrecht. Kluwer.

MacKay, D. J. C., and Peto, L. (1995) A hierarchical Dirichlet language model. *Natural Language Engineering* **1** (3): 1–19.

Pearlmutter, B. A., and Parra, L. C., (1996) A context-sensitive generalization of ica. To appear in ICONIP. Also available at `http://www.cnl.salk.edu/ ~bap/papers/iconip-96-cica.ps.gz`.

Rabiner, L. R., and Juang, B. H. (1986) An introduction to hidden Markov models. *IEEE ASSP Magazine* pp. 4–16.