Extending the basic ICA model

Part 1: reducing the number of souces

Version 0.1

Timothy Corbett-Clark, 12.3.1999
SP&NN Research Group, Oxford

Abstract

This internal report follows on from the introduction to Independent Component Analysis (ICA) given in [1]. An approach is described which enables the basic ICA model to be extended to include fewer sources than observed components, non-identical non-Gaussian source distributions, and an additative "noise" term. The approach is based on the idea of modelling each source using a fixed 1D mixture of Gaussians. The parameters of the model (mixing matrix and noise variances) are then optimsed using the Expectation Maximisation (EM) algorithm.

1 Introduction

Independent Component Analysis (ICA) is a solution to the problem of determining the matrix which will unmix the (unknown) linear combination of (unknown) independent sources. The only information available is a set of observations of the linearly combined signals.

The basic density estimation approach uses the following model:

$$\begin{aligned} \mathbf{x} &= \mathbf{A}\mathbf{s} \\ p(\mathbf{s}) &= \prod_{i=1}^{m} p(s_i), \end{aligned} \tag{1}$$

where the vector $\mathbf{s}$ represents m independent sources (or latent variables), the square mixing matrix $\mathbf{A}$ represents the linear mixing of the sources, and the vector $\mathbf{x}$ represents the m components of the observed signals. Note that this model makes several simplifying assumptions, such as equal number of observations and sources, time independent samples, and no additive measurement "noise" term.

Assuming Gaussian sources results in a maximum likelihood solution for $\mathbf{A}$ which is only unique up to an arbitrary rotation. This *may* be a reasonable density model, but is clearly useless for seperating source components.

Assuming non-Gaussian source distributions (such as the logistic) elliminates the arbitrary rotation problem, but also causes other problems to become dominant. In particular, assuming super(sub)-Gaussian[1] sources in the model enables super(sub)-Gaussian sourced data to be separated, but trying to fit a super(sub)-Gaussian sourced model on data from *sub(super)*-Gaussian sources produces an optimally poor solution.

[1] Recall that a super/sub-Gaussian distribution has heavier/lighter tails than a Gaussian distribution.

Insight into the density estimation approach to ICA can be gained from observing that a linear combination of random variables is more Gaussian than the original random variables. This is a crude interpretation of the central limit theorem [3]. Thus assuming super-Gaussian source distributions optimises the output from the "unmixing" matrix $\mathbf{A}^{-1}$ to be also super-Gaussian, which will coincide with the least mixed up combination of sources *so long as the actual sources are also super (and not sub) Gaussian*. It is therefore vital to assume source distributions of an appropriate form for the density model solution to solve the blind separation of sources problem. This is a tightening of the widely held belief that it is unnecessary to accurately model the density function. Of course this tightened version may also turn out to have exceptions; however we will accept it for the moment.

Section 2 begins by describing the problems arrising from assuming fewer source components than observation components, and ends by outlining the solution described in the rest of this report. Section 3 derives the complete algorithm, assuming that the model coefficients for each source distribution have alreay been obtained. Section 4 discusses methods for finding these fixed source distribution coefficients. Finally section 5 gives some initial results, and section 6 concludes.

2 Reducing the number of sources

Perhaps the most obvious extension to ICA is to have a different number of sources from components in each observation vector, *i.e.* a non-square mixing matrix. Assuming more sources than observations introduces unnecessary redundancy, and assuming fewer sources than observations gives rise to a density function which is only non-zero in a linear sub-space of observation space. Such a density model says that *no* patterns fall outside this "pancake" sub-space. This will always be false due to measurement noise, and so the likelihood of any real data under this model will always be zero. The obvious way to solve the problem and give this "pancake" some thickness is to include an additive noise term $\mathbf{v}$. As before, we shall assume zero mean data; clearly an extra constant term could be included to take non zero-mean data into account.

$$\begin{aligned} \mathbf{x} &= \mathbf{A}\mathbf{s} + \mathbf{v} \\ p(\mathbf{s}) &= \prod_{j=1}^{m} p(s_j), \\ p(\mathbf{v}) &= \prod_{i=1}^{d} p(v_i), \end{aligned} \tag{2}$$

where $\mathbf{A}$ now spans a sub-space ($m < d$), and the components of $\mathbf{v}$ are independent of one another and also independent of $\mathbf{s}$. This model is very similar to both the standard factor analysis model[2] [2] and the probabilistic principal component analysis (PPCA) model

[2]In the terminology of factor analysis, the sources are called *factors*, the columns of the mixing matrix are called *loadings*, and the noise term is called the vector of *specific* factors for each observation.

[5]. Both of these models assume Gaussian sources and Gaussian noise. However factor analysis assumes the noise term to have an unknown diagonal covariance matrix, whilst PPCA assumes the noise term to have a covarariance matrix equal to some unknown multiple of the identity. Note that the former makes it possible to scale individual variables and keep essentially the same density model, whilst the latter is a more general case of classical principal component analysis.

The key property of the above model is that the components of each observation are conditionally independent given the sources. Thus the sources are intended to model the dependencies between the observations while $\mathbf{v}$ represents the independent noise on each observation.

As far as solving the blind separation of sources problem is concerned, neither the factor analysis model nor the PPCA model are helpful. This is because both models assume Gaussian sources and hence have maximum likelihood solutions which are only unique up to an arbitrary rotation of source space.

Writing down the expression for $p(\mathbf{x})$ is more complicated than for the noiseless model because each observation could conceivably have been generated by any value of source vector $\mathbf{s}$, with an appropriate value of the error $\mathbf{v}$. Thus $p(\mathbf{x})$ involves the integral,

$$\begin{aligned} p(\mathbf{x}) &= \int_{\mathbf{s}} p(\mathbf{x}|\mathbf{s})p(\mathbf{s}) \\ p(\mathbf{x}|\mathbf{s}) &= p(\mathbf{v} = \mathbf{x} - \mathbf{A}\mathbf{s}). \end{aligned} \tag{3}$$

The basic problem is that this integral is only analytic for certain simple distributions. For example, it is clearly soluble if both the measurement error and the source distributions are Gaussian since the convolution of two Gaussian distributions is also Gaussian. But as described above, this fails to obtain a solution which unmixes the sources. Thus we need to model the sources using non-Gaussian distributions, only this unfortunately tends to prevent analytic solutions to the above integral.

The solution explored here is conceptually very simple. Approximate each desired source distribution using a 1D mixture of Gaussians. This does *not* give rise to a closed form solution to equation (3), but fortunately the Expectation Maximisation (EM) algorithm can be used to provide an iterative solution.

It is important to realise that the coefficients of each 1D Gaussian Mixture Model (GMM) are fixed in advance by the criteria that they model an assumed source distribution (*e.g.* a logisitic). Thus the parameters in the density model for $\mathbf{x}$ are still limited to the mixing matrix $\mathbf{A}$ and those parameters governing the distribution of $\mathbf{v}$. This has the advantage of not introducing extra parameters into the model, and thus avoids increasing the problems of overfitting. In any case, as explained in the previous section, it may be unnecessary to model the sources particularly accurately in order to achieve source separation.

3 The *constrained* mixture of Gaussians model

This section derives the iterative algorithm for finding the parameters of a model made from sources which are each a fixed mixture of Gaussians. Section 4 shows how to find suitable values for the mixture coefficients such that the GMM for each source approximates different assumed distributions.

3.1 The model

Let the j^{th} source component be written

$$p(s_j) = \sum_{k_j=1}^{K_j} p(s_j|k_j)P(k_j),$$
$$p(s_j|k_j) = \phi(s_j; \mu_{k_j}, \Sigma_{k_j}), \tag{4}$$

where $\phi(x; \text{mean}, \text{covariance})$ describes the probability density function (PDF) of a (multivariate) Gaussian. An alternative, equivalent, and advantageous notation is

$$p(\mathbf{s}) = \sum_{\mathbf{k=1}}^{K} p(\mathbf{s}|\mathbf{k})P(\mathbf{k}), \tag{5}$$
$$p(\mathbf{s}|\mathbf{k}) = \phi(\mathbf{s}; \mu_{\mathbf{k}}, \Sigma_{\mathbf{k}}), \tag{6}$$
$$P(\mathbf{k}) = \prod_{j=1}^{m} P(k_j), \tag{7}$$

where

$$\sum_{\mathbf{k}}^{K} \equiv \sum_{\mathbf{k}_1}^{K_1} \cdots \sum_{\mathbf{k}_m}^{K_m}$$

and $\Sigma_{\mathbf{k}}$ is a diagonal covariance, so

$$\phi(\mathbf{s}; \mu_{\mathbf{k}}, \Sigma_{\mathbf{k}}) = \prod_{j=1}^{m} \phi(s_j; \mu_{k_j}, \Sigma_{k_j}). \tag{8}$$

For completeness, we will write the noise distribution in the same way,

$$p(\mathbf{v}) = \phi(\mathbf{v}; 0, \Psi), \tag{9}$$

where Ψ is also a diagonal covariance matrix.

The following relationship will be much used throughout the following analysis:

$$p(\mathbf{s}, \mathbf{k}, \mathbf{x}) = p(\mathbf{x}|\mathbf{s}, \mathbf{k})p(\mathbf{s}|\mathbf{k})P(\mathbf{k}) \tag{10}$$
$$= \phi(\mathbf{x}; A\mathbf{s}, \Psi)\phi(\mathbf{s}; \mu_{\mathbf{k}}, \Sigma_{\mathbf{k}})P(\mathbf{k}). \tag{11}$$

3.2 The unconditional density

With the new notation, the unconditional density can be written

$$p(\mathbf{x}) = \sum_{\mathbf{k}=\mathbf{1}}^{K} p(\mathbf{x}|\mathbf{k})P(\mathbf{k}) \tag{12}$$

where $p(\mathbf{x}|\mathbf{k})$ is found by integrating out $\mathbf{s}$,

$$\begin{aligned} p(\mathbf{x}|\mathbf{k}) &= \int_{\mathbf{s}} p(\mathbf{x}|\mathbf{s},\mathbf{k})p(\mathbf{s}|\mathbf{k}) \\ &= \int_{\mathbf{s}} \phi(\mathbf{x}; A\mathbf{s}, \Psi)\phi(\mathbf{s}; \mu_{\mathbf{k}}, \Sigma_{\mathbf{k}}) \\ &= \phi(\mathbf{x}; A\mu_{\mathbf{k}}, \Psi + A\Sigma_{\mathbf{k}}A^T). \end{aligned} \tag{13}$$

Recalling that $P(\mathbf{k})$, $\mu_{\mathbf{k}}$ and $\Sigma_{\mathbf{k}}$ are fixed for all $\mathbf{k}$, and that the parameters of the model are A and Ψ, it can be seen that the density model is simply a highly constrained mixture of Gaussians.

3.3 Preparing for EM

The EM algorithm is a powerful technique for maximising likelihood in the presence of unknowns. For this problem, the vectors

$$\mathbf{X} = \{\mathbf{x}^n\}_{n=1}^N$$

are observed and hence known, and the vectors

$$\mathbf{S} = \{\mathbf{s}^n\}_{n=1}^N \qquad \mathbf{K} = \{\mathbf{k}^n\}_{n=1}^N$$

are unobserved and hence unknown.

Thus the EM update step is (see [4] for an excellent explanation),

$$\Theta^{t+1} \longleftarrow \arg\max_{\Theta} \int_{\mathbf{S}} \sum_{\mathbf{K}} p(\mathbf{S},\mathbf{K}|\mathbf{X};\Theta^t) \log p(\mathbf{X}|\mathbf{S},\mathbf{K};\Theta). \tag{14}$$

Note that the parameters of the model, $\Theta = \{A, \Psi\}$, are *not* random variables, hence the ';' notation. Note also that the summations have become even larger, for example:

$$\sum_{\mathbf{K}} \equiv \sum_{\mathbf{k}^1} \cdots \sum_{\mathbf{k}^N}.$$

We will (still !) not use the fact the the data is a time series, so

$$p(\mathbf{S},\mathbf{K}|\mathbf{X};\Theta^t) = \prod_{n=1}^{N} p(\mathbf{s}^n,\mathbf{k}^n|\mathbf{x}^n;\Theta^t), \tag{15}$$

$$\log p(\mathbf{X}|\mathbf{S},\mathbf{K};\Theta) = \sum_{n=1}^{N} \log p(\mathbf{x}^n|\mathbf{s}^n,\mathbf{k}^n;\Theta) \tag{16}$$

3.4 Simplifying the E-step

This subsection shows how to simplify the summations in the EM update step (equation 14). This essentially involves the following "trick",

$$\sum_n f(\mathbf{k}^n) = \sum_n \sum_{\mathbf{k}} \delta_{\mathbf{k},\mathbf{k}^n} f(\mathbf{k}), \tag{17}$$

followed by marginalisations using summations and integrations. Thus[3]

$$\begin{aligned}
&\int_{\mathbf{S}} \sum_{\mathbf{K}} p(\mathbf{S},\mathbf{K}|\mathbf{X};\Theta^t) \log p(\mathbf{X}|\mathbf{S},\mathbf{K};\Theta) \\
&= \int_{\mathbf{S}} \sum_{\mathbf{k}^1} \cdots \sum_{\mathbf{k}^N} \prod_{n'} p(\mathbf{s}^{n'},\mathbf{k}^{n'}|\mathbf{x}^{n'};\Theta^t) \sum_n \log p(\mathbf{x}^n|\mathbf{s}^n,\mathbf{k}^n;\Theta) \\
&= \int_{\mathbf{S}} \sum_{\mathbf{k}^1} \cdots \sum_{\mathbf{k}^N} \prod_{n'} p(\mathbf{s}^{n'},\mathbf{k}^{n'}|\mathbf{x}^{n'};\Theta^t) \sum_n \sum_{\mathbf{k}} \delta_{\mathbf{k},\mathbf{k}^n} \log p(\mathbf{x}^n|\mathbf{s}^n,\mathbf{k};\Theta) \\
&= \int_{\mathbf{S}} \sum_n \sum_{\mathbf{k}} \sum_{\mathbf{k}^1} \cdots \sum_{\mathbf{k}^N} \delta_{\mathbf{k},\mathbf{k}^n} \prod_{n'} p(\mathbf{s}^{n'},\mathbf{k}^{n'}|\mathbf{x}^{n'};\Theta^t) \log p(\mathbf{x}^n|\mathbf{s}^n,\mathbf{k};\Theta) \\
&= \int_{\mathbf{s}^1} \cdots \int_{\mathbf{s}^N} \sum_n \sum_{\mathbf{k}} \frac{\prod_{n'} p(\mathbf{s}^{n'},\mathbf{k}^{n'}|\mathbf{x}^{n'};\Theta^t)}{p(\mathbf{s}^n|\mathbf{x}^n;\Theta^t)} p(\mathbf{s}^n,\mathbf{k}|\mathbf{x}^n;\Theta^t) \log p(\mathbf{x}^n|\mathbf{s}^n,\mathbf{k};\Theta) \\
&= \sum_n \sum_{\mathbf{k}} \int_{\mathbf{s}} p(\mathbf{s},\mathbf{k}|\mathbf{x}^n;\Theta^t) \log p(\mathbf{x}^n|\mathbf{s},\mathbf{k};\Theta) \\
&= \sum_n \int_{\mathbf{s}} \sum_{\mathbf{k}} p(\mathbf{s},\mathbf{k}|\mathbf{x}^n;\Theta^t) \log p(\mathbf{x}^n|\mathbf{s};\Theta) \\
&= \sum_n \int_{\mathbf{s}} p(\mathbf{s}|\mathbf{x}^n;\Theta^t) \log p(\mathbf{x}^n|\mathbf{s};\Theta).
\end{aligned} \tag{18}$$

This may be interpreted as the log likelihood of the observed vectors $\mathbf{x}^n$ given the unobserved vector $\mathbf{s}$, *averaged* with respect to our estimate of the unobserved distributions $p(\mathbf{s}|\mathbf{x}^n;\Theta^t)$ at this iteration step t. Note that the unobserved variables $\mathbf{k}^n$, which denote the source kernel for each observation, have been summed out.

[3] Don't panic.

3.5 Finding the E-step in terms of $<\mathbf{s}>_{n,t}$ and $<\mathbf{s}\mathbf{s}^T>_{n,t}$

Continuing on from the "expectation" equation, (18), we can simplify further:

$$
\begin{aligned}
&= \sum_n \int_{\mathbf{s}} p(\mathbf{s}|\mathbf{x}^n;\Theta^t)\ \log\phi(\mathbf{x}^n; A\mathbf{s}, \Psi) \\
&= \sum_n \int_{\mathbf{s}} p(\mathbf{s}|\mathbf{x}^n;\Theta^t) \left[\text{constant} + \frac{1}{2}\log|\Psi^{-1}| - \frac{1}{2}(\mathbf{x}^n - A\mathbf{s})^T\Psi^{-1}(\mathbf{x}^n - A\mathbf{s})\right] \\
&= \sum_n \text{constant} + \frac{1}{2}\log|\Psi^{-1}| - \frac{1}{2}\int_{\mathbf{s}} p(\mathbf{s}|\mathbf{x}^n;\Theta^t)(\mathbf{x}^n - A\mathbf{s})^T\Psi^{-1}(\mathbf{x}^n - A\mathbf{s}). \qquad (19)
\end{aligned}
$$

Defining

$$
<f>_{n,t} = \int_{\mathbf{s}} f\ p(\mathbf{s}|\mathbf{x}^n;\Theta^t) \qquad (20)
$$

allows equation (19) to be written

$$
\begin{aligned}
&= \sum_n \text{constant} + \frac{1}{2}\log|\Psi^{-1}| + \\
&\quad - \frac{1}{2}\left[(\mathbf{x}^n)^T\Psi^{-1}\mathbf{x}^n - 2(\mathbf{x}^n)^T\Psi^{-1}A <\mathbf{s}>_{n,t} + \mathrm{tr}(A^T\Psi^{-1}A <\mathbf{s}\mathbf{s}^T>_{n,t})\right]. \qquad (21)
\end{aligned}
$$

If we simplify further by assuming the variances on the measurement noise components are all equal[4], $\Psi = \sigma^2 I$, then equation (21) becomes

$$
\begin{aligned}
&= \sum_n \text{constant} - \frac{d}{2}\log\sigma^2 + \\
&\quad - \frac{1}{2\sigma^2}(\mathbf{x}^n)^T\mathbf{x}^n + \frac{1}{\sigma^2}(\mathbf{x}^n)^T A <\mathbf{s}>_{n,t} - \frac{1}{2\sigma^2}\mathrm{tr}(A^T A <\mathbf{s}\mathbf{s}^T>_{n,t}). \qquad (22)
\end{aligned}
$$

3.6 Finding the M-step in terms of $<\mathbf{s}>_{n,t}$ and $<\mathbf{s}\mathbf{s}^T>_{n,t}$

This simply involves finding the values of the parameters which maximise equation (22). Setting the derivatives with respect to A and σ^2 equal to zero gives

$$
\left(A^{t+1} =\right)\ A = \left(\sum_n \mathbf{x}^n <\mathbf{s}>_{n,t}^T\right)\left(\sum_n <\mathbf{s}\mathbf{s}^T>_{n,t}\right)^{-1} \qquad (23)
$$

$$
\left((\sigma^2)^{t+1} =\right)\ \sigma^2 = \frac{1}{Nd}\left[(\mathbf{x}^n)^T\mathbf{x}^n - 2(\mathbf{x})^T A <\mathbf{s}>_{n,t} + \mathrm{tr}(A^T A <\mathbf{s}\mathbf{s}^T>_{n,t})\right]. \qquad (24)
$$

This is a very satisfying result, since by dropping the bra-kets, <and >, and the reference to t, we obtain exactly the closed form solution for the case when the source vectors $\{\mathbf{s}^n\}_{n=1}^N$ are in fact known.

[4] A later version of this document will not make this assumption.

3.7 Calculating $p(\mathbf{s}|\mathbf{x}^n, \mathbf{k}; \Theta^t)$

It is likely from the definition of $<f>_{n,t}$ (equation 20) that an expression for $p(\mathbf{s}|\mathbf{x}^n, \mathbf{k}; \Theta^t)$ will be required. This is indeed the case. Noting that

$$\begin{aligned} p(\mathbf{s}|\mathbf{x}^n, \mathbf{k}; \Theta^T)p(\mathbf{x}^n|\mathbf{k}; \Theta^t) &= p(\mathbf{x}^n|\mathbf{s}, \mathbf{k}; \Theta^t)p(\mathbf{s}|\mathbf{k}; \Theta^t) \\ &= \phi(\mathbf{x}^n; A^t\mathbf{s}, \Psi^t)\phi(\mathbf{s}; \mu_\mathbf{k}, \Sigma_\mathbf{k}) \end{aligned}$$

is proportional to

$$\exp -\frac{1}{2}\left[(\mathbf{x}^n - A\mathbf{s})^T(\Psi^t)^{-1}(\mathbf{x}^n - A\mathbf{s}) + (\mathbf{s} - \mu_\mathbf{k})(\Sigma_\mathbf{k})^{-1}(\mathbf{s} - \mu_\mathbf{k})\right], \tag{25}$$

and by refactorising to give a "Gaussian in $\mathbf{s}$", one can deduce that

$$p(\mathbf{s}|\mathbf{x}^n, \mathbf{k}; \Theta^t) = \phi(\mathbf{s}; M^t_\mathbf{k} m^t_{n,\mathbf{k}}, M^t_\mathbf{k}) \tag{26}$$

where

$$(M^t_\mathbf{k})^{-1} = (A^t)^T(\Psi^t)^{-1}(A^t) + \Sigma^{-1}_\mathbf{k} \tag{27}$$

$$m^t_{n,\mathbf{k}} = \Sigma^{-1}_\mathbf{k}\mu_\mathbf{k} + (A^t)^T(\Psi^t)^{-1}\mathbf{x}^n. \tag{28}$$

3.8 Calculating $<f>_{n,t}$

Straightforward manipulation provides a convenient form for $<f>_{n,t}$. Starting from the definition, equation (20),

$$<f>_{n,t} = \int_\mathbf{s} f\ p(\mathbf{s}|\mathbf{x}^n; \Theta^t) \tag{29}$$

$$= \int_\mathbf{s} f\ \sum_\mathbf{k} \frac{p(\mathbf{s}|\mathbf{x}^n, \mathbf{k}; \Theta^t)p(\mathbf{x}^n|\mathbf{k}; \Theta^t)P(\mathbf{k})}{p(\mathbf{x}^n; \Theta^t)} \tag{30}$$

$$= \sum_\mathbf{k} \frac{p(\mathbf{x}^n|\mathbf{k}; \Theta^t)P(\mathbf{k})}{p(\mathbf{x}^n; \Theta^t)} \int_\mathbf{s} f\ p(\mathbf{s}|\mathbf{x}^n, \mathbf{k}; \Theta^t). \tag{31}$$

Finally, given the form for $p(\mathbf{s}|\mathbf{x}^n, \mathbf{k}; \Theta^t)$ shown in equation (26), we obtain the last of the required equations:

$$<\mathbf{s}>_{n,t} = \sum_\mathbf{k} \frac{p(\mathbf{x}^n|\mathbf{k}; \Theta^t)P(\mathbf{k})}{p(\mathbf{x}^n; \Theta^t)}\ M^t_\mathbf{k} m^t_{n,\mathbf{k}} \tag{32}$$

and

$$<\mathbf{s}\mathbf{s}^T>_{n,t} = \sum_\mathbf{k} \frac{p(\mathbf{x}^n|\mathbf{k}; \Theta^t)P(\mathbf{k})}{p(\mathbf{x}^n; \Theta^t)}\left[M^t_\mathbf{k} + (M^t_\mathbf{k} m^t_{n,\mathbf{k}})(M^t_\mathbf{k} m^t_{n,\mathbf{k}})^T\right]. \tag{33}$$

3.9 The algorithm

We are now in a position to summarise the algorithm. Three extra variables have been defined for algorithmic convenience:

$$R_a = \sum_n \mathbf{x}^n (\mathbf{x}^n)^T, \quad R_b = \sum_n <\mathbf{s}>_{n,t} (\mathbf{x}^n)^T, \quad R_c = \sum_n <\mathbf{s}\mathbf{s}^T>_{n,t} . \tag{34}$$

---

set $R_a = \sum_n (\mathbf{x}^n)(\mathbf{x}^n)^T$

loop over iterations, t

set $R_b = 0$
set $R_c = 0$

loop over patterns, n

set $p(\mathbf{x}^n; \Theta^t) = 0$
set $<\mathbf{s}>_{n,t} = 0$
set $<\mathbf{s}\mathbf{s}^T>_{n,t} = 0$

loop over all kernels, $\mathbf{k}$

compute $p(\mathbf{x}^n|\mathbf{k}; \Theta^t)$ using equation (13)
compute $M^t_{\mathbf{k}}$ using equation (27)
compute $m^t_{n,\mathbf{k}}$ using equation (28)

accumulate $p(\mathbf{x}^n; \Theta^t)$ with $p(\mathbf{x}^n|\mathbf{k}; \Theta^t)P(\mathbf{k})$
accumulate $<\mathbf{s}>_{n,t}$ as per equation (32), but without normalisation
accumulate $<\mathbf{s}\mathbf{s}^T>_{n,t}$ as per equation (33), but without normalisation

normalise $<\mathbf{s}>_{n,t}$ with $p(\mathbf{x}^n; \Theta^t)$
normalise $<\mathbf{s}\mathbf{s}^T>_{n,k}$ with $p(\mathbf{x}^n; \Theta^t)$

accumulate R_b with $<\mathbf{s}>_{n,t} (\mathbf{x}^n)^T$
accumulate R_c with $<\mathbf{s}\mathbf{s}^T>_{n,t}$

update parameters $A^{t+1} = R_b^T R_c^{-1}$
update parameters $(\sigma^2)^{t+1} = \frac{1}{Nd}\mathrm{tr}\left(R_a - 2A^{t+1}R_b + A^{t+1}R_c(A^{t+1})^T\right)$

3.10 The MATLAB code

The above code design maps directly into MATLAB. Note that this implementation is limited to exactly two sources.

```
function [A,var] = ica_with_noise(X, sources)

% Each pattern is in a column of 'X'
% 'sources' is a 1xE array of the structure 'source':
%   source.K is a number of kernels
%   source.p is a vector of priors
%   source.mu is a vector of centres
%   source.sigma is a vector of variances
% 'A' is the mixing matrix
% 'var' is measurement variance ie v = Gauss(v; 0,var*I)

% maximum number of iterations
ITS = 100;

% set #observed dimensions, #patterns, and #sources
[d,N] = size(X);
m = size(sources,2);

% randomly initialise A and var (?)
A = randn(d,m)
var = 0.1

% can find Ra now because it is fixed
Ra = X*X';

% loop over iterations
for its=1:ITS,

  % initialise Rb and Rc
  Rb = zeros(m,d);
  Rc = zeros(m,m);

  % loop over patterns
  for n=1:N,

    % pattern x^n
    x = X(:,n);

    % initialise <s>_{n,t}
    s = zeros(m,1);

    % initialise <s*s'>_{n,t}
    ss = zeros(m,m);
```

```
    % initialise p(x^n) for accumulation
    px = 0;

    % loop over all kernels
    for k1=1:sources(1).K,
      for k2=1:sources(2).K,

        % form k, p, mu, sigma, sigmainv, phi
        k = [k1;k2];
        p = sources(1).p(k1) * sources(2).p(k2);
        mu = [sources(1).mu(k1) ; sources(2).mu(k2)];
        sigma = diag([sources(1).sigma(k1) ; sources(2).sigma(k2)]);
        sigmainv = diag(1./[sources(1).sigma(k1) ; sources(2).sigma(k2)]);
        phi = var*eye(d);

        % compute p(x^n , k)
        pxk = gauss((A*mu)', phi+A*sigma*A', x') * p;
        % and also accumulate ready for normalisation latter
        px = px + pxk;

        % compute M and m, and M*m
        M = inv(A' * A / var + sigmainv);
        m = sigmainv*mu + A' * x / var;
        Mm = M*m;

        % accumulate <s>_{n,t} and <s*s'>_{n,t}
        s = s + pxk * Mm;
        ss = ss + pxk * (M + Mm*Mm');

      end
    end

    % normalise <s>_{n,t} and <s*s'>_{n,t}
    s = s / px;
    ss = ss / px;

    % accumulate Rb and Rc
    Rb = Rb + s * x';
    Rc = Rc + ss;

  end

  % update model paramters, A and var
  A = Rb' * inv(Rc);
  var = trace(Ra-2*A*Rb+A*Rc*A') / (N*d);
end
```

4 Modelling each source with a Gaussian mixture

The algorithm described so far has assumed that the coefficients governing each source distribution are already known. This section will discuss possible choices.

For the purposes of investigating the method as a solution to the blind separation of sources problem, we will initially have only two types of source: a super-Gaussian source approximating a logistic distribution, and a sub-Gaussian source approximating a uniform distribution. The problem is thus of finding suitable values for the coefficients K_j, $P(k_j)$, μ_{k_j} and Σ_{k_j} for source j such that the model given by equation (4) approximates either a logistic or a uniform. These equations are repeated below but without the cluttering 'j' sub-script.

$$\begin{aligned} p(s) &= \sum_{k=1}^{K} p(s|k)P(k), \\ p(s|k) &= \phi(s; \mu_k, \Sigma_k), \end{aligned} \tag{35}$$

This is therefore a 1-dimensional density estimation problem (s is a scalar) for which we actually know the desired target distribution. This contrasts with the data estimation scenario for which only a *sample* from the target distribution is known.

The obvious approach is to maximise the normalised log-likelihood on an infinite sample,

$$L = \int_s f(s) \log p(s), \tag{36}$$

where $f(s)$ is the target distribution[5]. Rather than attempting to find an analytic solution, we will approximate the integral with a large sample of N points from $f(\cdot)$ to obtain the normalised log-likelihood,

$$L \approx \frac{1}{N} \sum_{n=1}^{N} \log p(s^n) \tag{37}$$

This mixture model is then easily optimised with respect to the model coefficients using EM. Figure 1 shows that a logistic distribution is accurately modelled using only three Gaussians, whilst figure 2 shows that rather more Gaussians are required to approximate a uniform. However this does necessarily mean that we should use more Gaussians to approximate uniform distributions than to approximate logistic. Recall that the (excess) kurtosis measures the weight of the tails, where the kurtosis of a logistic is 1.2 and the kurtosis of a uniform is -1.2. Compare with the kurtosis obtained from the approximations using Gaussian mixtures[6] shown in table 1. Ok, so this requires more interpration.

[5]This expression is also minus the (unnormalised) Kullback-Leibler divergence

[6]These numbers are only approximate because they were estimated from samples. A later version of this document will (I hope !) give the analytic solution.

#kernels	logistic	uniform
3	0.5	-1.1
5	2	-1.2
7	4	-1.1

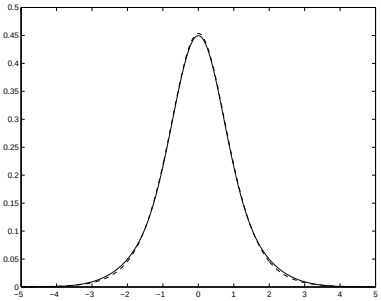
Table 1: Excess kurtosis for approximations of both logistic and uniform distributions using mixture of Gaussians with various number of kernels.

5 Results

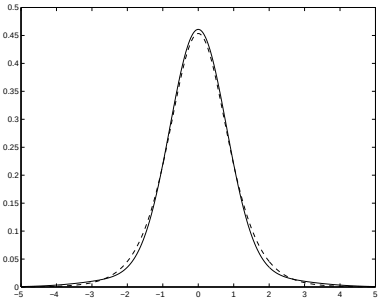
The algorithm works on data sets generated from correctly assumed distributions. Unfortunately, due to its currently un-vectored form, the MATLAB implementation is very slow. I have not yet tested it on more interesting examples.

6 Conclusions

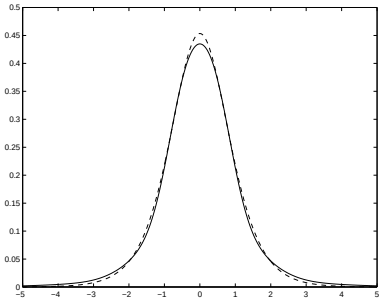
Inconclusive so far. However I think it is worth pursuing, not necessarily because the current model (if made faster) is particularly good, but because it is a promising approach to solving the blind deconvolution problem. A solution to this problem, which would finally take advantage of the time-series nature of the data, could lead to useful filter.



$p(k)$	μ_k	Σ_k
0.2633	-0.2477	1.4873
0.4734	0	0.4523
0.2633	-0.2477	1.4873

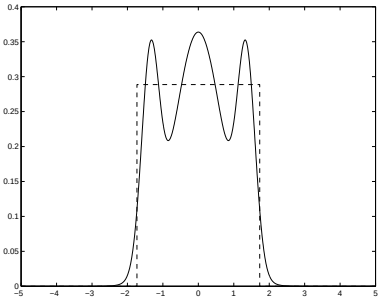


$p(k)$	μ_k	Σ_k
0.0669	-1.3801	2.0138
0.2786	-0.3750	0.5309
0.3089	0	0.5246
0.2786	0.3750	0.5309
0.0669	1.3801	2.0138

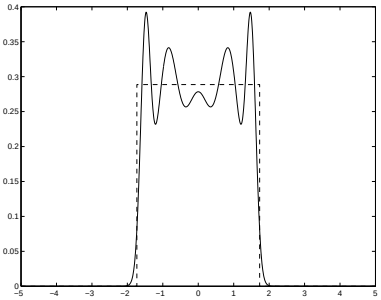


$p(k)$	μ_k	Σ_k
0.0580	-1.3165	1.7092
0.1591	-0.8372	0.3562
0.1856	-0.1188	0.3108
0.1946	0	0.5808
0.1856	0.1188	0.3108
0.1591	0.8327	0.3562
0.0580	1.3165	1.7092

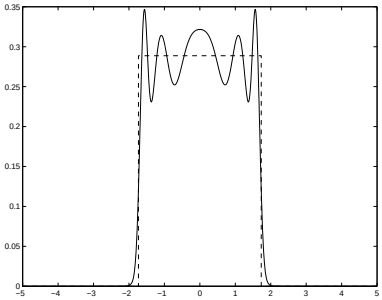
Figure 1: Using a mixture of Gaussians (shown solid) to approximate a logistic distribution (shown dashed).



$p(k)$	μ_k	Σ_k
0.1844	-1.3552	0.0616
0.6312	0	0.4791
0.1844	-1.3552	0.0616



$p(k)$	μ_k	Σ_k
0.1190	-1.4880	0.0194
0.2422	-0.8879	0.0964
0.2776	0	0.1708
0.2422	0.8879	0.0964
0.1190	1.4880	0.0194



$p(k)$	μ_k	Σ_k
0.0739	-1.5819	0.0112
0.1634	-1.1508	0.0632
0.1816	-0.4382	0.1592
0.1623	0	0.2772
0.1816	0.4382	0.1592
0.1634	1.1508	0.0632
0.0739	1.5819	0.0112

Figure 2: Using mixtures of Gaussians (shown solid) to approximate a uniform distribution (shown dashed).

References

[1] Timothy A Corbett-Clark. Introductory notes on independent component analysis and the blind separation of sources problem. Technical report, Signal Processing and Neural Network research group, Engineering Department, University of Oxford, 1999.

[2] D N Lawley and A E Maxwell. *Factor Analysis as a Statistical Method*. London: Butterworths, second edition edition, 1971.

[3] John A Rice. *Mathematical Statistics and Data Analysis*. Duxbury Press, 1995.

[4] Sam Roweis and Zoubin Ghahramani. A unifying review of linear gaussian models. *Neural Computation*, 11(2), 1999.

[5] Michael E Tipping and Christopher M Bishop. Probabilistic component analysis. Technical report, Neural Computing Research Group, Dept of Computer Science and Applied Mathematics, Aston University, Birmingham, `http://www.ncrg.aston.ac.uk`, September 1997. NCRG/97/010.