Independent Component Analysis

Max Welling

Gatsby Computational Neuroscience Unit
London WC1N 3AR
welling@gatsby.ucl.ac.uk

1 Square Noiseless ICA

We have seen in the previous section that PCA can be interpreted as finding the high variance directions in data space, or minimizing the reconstruction error of data when we project them down. There is yet another interpretation of PCA (or PPCA), which will lead us to ICA. Therefore we first notice that once we have estimated the matrix $\mathbf{A}$ (assuming the noise is zero), we can change to a basis in which the data are "decorrelated". This means that the sample covariance of the data vanishes. For that we first define the "pseudo-inverse" of a rectangular $d \times k$ matrix with $d > k$ to be,

$$\mathbf{A}^{+} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T \tag{1}$$

which has the property $\mathbf{A}^{+}\mathbf{A} = \mathbf{I}$. Remember that the covariance in the noiseless limit was given by,

$$\mathbf{E}[(\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^T] = \mathbf{A}\mathbf{E}[\mathbf{y}\mathbf{y}^T]\mathbf{A}^T = \mathbf{A}\mathbf{A}^T \tag{2}$$

It is now easy to verify that the variable,

$$\mathbf{v} = \mathbf{A}^{+}(\mathbf{x}-\boldsymbol{\mu}) \tag{3}$$

is uncorrelated and has unit variance.

$$\mathbf{E}[\mathbf{v}\mathbf{v}^T] = \mathbf{I} \tag{4}$$

A variable with this property is sometimes called "white". So PCA could also be interpreted as finding a set of basisvectors onto which the data are decorrelated. But, decorrelation does not necessarily imply that the data are completely independent. Only if the data are distributed according to a Gaussian is this actually true. Technically we say that only in second order statistics of the data are independent. ICA is developed to find the axes onto which the data are made independent in to all statistic orders if possible, and otherwise to find the axes onto which they are made as independent as possible.

Consider the following example. Suppose we have installed 10 microphones (sensors) in a room where we record sound. In the room we find a bird singing, a radio playing and two people chatting with each other. Because the microphones are scattered around in the room, they are at different distances from the sound sources. Therefore every microphone will pick up a different (approximately linear) combination of the sound sources. Consider every recorded sample as a new (approximately independent) datapoint and plot it in a 10 dimensional "sensor-space". Since, all data are different linear combinations of only 4 sources, the data will describe (approximately) a 4 dimensional plane in this 10 dimensional sensor-space. Our task will be to uncover the sound sources from the sensor data. The main ingredient will be that the we will assume these sources to be statistically independent. This above example is sometimes referred to as blind source separation.

First we assume zero mean data and a linear tranformation between the observed and the latent random variables,

$$\mathbf{x} = \mathbf{A}\mathbf{y} \tag{5}$$

For now we will also assume a noiseless setting. The latent variables $\mathbf{y}$ are supposed to represent the independent variables, and could also be called sources or factors. We will furthermore assume that there are as many sources as observed dimensions ($d = k$). Later, we will relax the above asssumptions.

The sources are assumed independent. This is mathematically expressed as,

$$p(\mathbf{y}) = p_1(y_1)\ p_2(y_2)...p_D(y_D). \tag{6}$$

We will now also assume that $\mathbf{A}$ is invertible and denote $\mathbf{A}^{-1} = \mathbf{W}$. Using the general transformation rule for random variables,

$$p(\mathbf{x}) = p(\mathbf{y})\ |\det(\mathbf{J})|_{\ \mathbf{y}=\mathbf{W}\mathbf{x}} \tag{7}$$

where $\mathbf{J}$ is the Jacobian of the transformation,

$$\mathbf{J}_{ij} = \frac{\partial \mathbf{y}_i}{\partial \mathbf{x}_j} \tag{8}$$

we can write,

$$p(\mathbf{x}) = p_1(\mathbf{w}_1^T\mathbf{x})\ p_2(\mathbf{w}_2^T\mathbf{x})...p_D(\mathbf{w}_D^T\mathbf{x})\ |\det(\mathbf{W})| \tag{9}$$

where $\mathbf{w}_i$ are the rows of the matrix $\mathbf{W}$. From here it is a simple step to write down the log-likelihood,

$$L(\mathbf{x}_{1:N}|\mathbf{W}) = N\log(\det[\mathbf{W}]) + \sum_{n=1}^{N}\sum_{i=1}^{D}\log[p_i(\mathbf{w}_i^T\mathbf{x}_n)] \tag{10}$$

We will first derive a gradient descent algorithm for ICA. Gradient descent is easy, since we only have to take derivatives of L with respect to $\mathbf{W}$. For that we use,

$$\frac{\partial\sum_j \mathbf{W}_{ij}\mathbf{x}_j}{\partial\mathbf{W}_{kl}} = \delta_{ik}\mathbf{x}_l \tag{11}$$

The result of this calculation is postmultiplied by $\mathbf{W}^T\mathbf{W}$ which will speed up convergence considerably for reasons we cannot explain here. The final result is then the following gradient descent algorithm for ICA,

$$\Delta\mathbf{W} = \eta\left(\mathbf{I} + \frac{1}{N}\sum_{n=1}^{N}\left[\frac{\partial\log[p(\mathbf{y}_n)]}{\partial\mathbf{y}_n}\right]\mathbf{y}_n^T\right)\mathbf{W} \tag{12}$$

Alternatively, we can drop the sum over all datapoints and update after considering smaller batches. We could even use just one sample per update. The name for this is stochastic gradient descent. We should also still choose the explicit form of the density $p(\mathbf{y})$. As mentioned before, we should not choose a Gaussian since they can only discriminate second order dependencies (correlations), and those are simply removed by the PCA transormation (3). The model for $p(\mathbf{y}) = \prod_{i=1}^{K} p_i(y_i)$ should model the source distributions as accurate as possible. They could even depend on some parameters which we could learn simulteneously. If we think the distribution has long tails, which is for instance true if we measure the loudness of sound (actually many natural phenomena have this property), then the following distribution is usefull,

$$p(y) = \frac{1}{4\cosh^2(\frac{1}{2}y)} \tag{13}$$

where

$$\cosh(x) = \frac{1}{2}(e^x + e^{-x}) \tag{14}$$

The derivative of the log-likelihood is given by,

$$\frac{\partial}{\partial y}\log[p(y)] = -\tanh(\frac{1}{2}y) \tag{15}$$

This algorithm was first derived in (Bell and Sejnowski, 1995), from the "infomax" principle. We don't need to use gradient descent techniques. Efficient ICA algorithm have been developed using Jacobi algorithms (Comon, 1994), fixed point algorithms (Hyvärinen, 1997) and EM algorithms (see below).

2 EM algorithm for ICA

The reason that we limited ourselves to square noiseless ICA with gradient descent optimization, instead of deriving an EM algorithm, is that it is typically very hard to compute the posterior distribution of the sources given the data analytically. The problem is caused by the fact that the prior is non-Gaussian. One way to circumvent this, is to model the source densities by 1 dimensional mixtures of Gaussians,

$$p_i(y_i) = \sum_{a=1}^{J}\pi_{i,a}\mathcal{G}_{y_i}[\boldsymbol{\mu}_{i,a}, \sigma_{i,a}^2] \tag{16}$$

i.e. a different mixture for every source. These d one dimensional mixtures of Gaussians are capable of approximating any probability density. For instance, long tailed distributions can be made by adding zero mean Gaussians with different variances. More uniformly distributed densities and multimodal ones can be modeled by moving the means apart. In general, we would like to learn the source parameters (i.e. the means and the variances) from the data. This is sometimes called adaptive ICA. We will now also relax the requirement that the number of sources needs to match the number of signals and allow Gaussian additive noise,

$$\mathbf{x} = \mathbf{A}\mathbf{y} + \boldsymbol{\nu} \tag{17}$$

where $\boldsymbol{\nu} \sim \mathcal{G}_{\boldsymbol{\nu}}[0, \boldsymbol{\Sigma}]$. Since we know how to integrate over Gaussians we can in principle compute $p(\mathbf{y}_n|\mathbf{x}_n)$ and derive update rules for $\{\mathbf{A}, \boldsymbol{\Sigma}, \pi, \boldsymbol{\mu}, \sigma^2\}$. The case where we have fewer sources than signals has been appropriately named "Independent Factor Analysis" (IFA) (Attias, 1999). Also the case where $d < k$ is interesting, because of its tendency to produce sparse codes (only a small fraction of the sources colaborate to explain the output) and is named "overcomplete ICA" (Lewicki and Sejnowski, 1998). However, the above method suffers from the problem that its complexity scales exponentially with the number of sources. This can be seen from the fact that the sources are multiplied together, and since every source consists of a sum of muliple Gaussians (say M), the total number of terms is M^k in general. In (Attias, 1999) a solution is found by using an approximate, factorized, distribution for the posterior densities (again a product of mixtures of Gaussians) and applying the machinery of the variational EM algorithm. Since the equations are quite involved, we will not spell them out here but refer to the literature. If one is willing to use square mixing matrices ($d = k$), and a simple isotropic noise model ($\boldsymbol{\Sigma} = \beta^2\mathbf{I}$), then one can write down an EM algorithm that scales linearly with the number of sources (in fact, the trick involves pre-sphering the data and optimizing over orthogonal mixing matrices) (Welling and Weber, 1999).

3 Variational Noisy ICA

In this section we will derive a simple ICA algorithm that can deal with general additive Gaussian noise and $d \neq k$. The idea is to use a factorized Gaussian density to model the posterior and to optimize its parameters using standard variational techniques. For computational reasons we will restrict ourselves to a (full) covariance matrix common to all data (i.e. independent of n),

$$Q(\mathbf{y}_n) = \mathcal{G}_{\mathbf{y}_n}[\boldsymbol{\mu}_n, \boldsymbol{\Sigma}] \tag{18}$$

Note that we allow dependencies between the sources given the data to incorporate explaining away phenomena. The generative model is as in section 1:

$$p(\mathbf{y}_n) = \prod_{i=1}^{k} \frac{1}{4\cosh^2(\frac{1}{2}y_i)} \tag{19}$$

$$p(\mathbf{x}_n|\mathbf{y}_n) = \mathcal{G}_{\mathbf{x}_n}[\mathbf{A}\mathbf{y}_n, \boldsymbol{\Gamma}] \tag{20}$$

where $\boldsymbol{\Gamma}$ is the noise covariance. The free energy is thus given by,

$$F = E - S \tag{21}$$

$$= \sum_{n=1}^{N} [\ \frac{1}{2} < (\mathbf{x}_n - \mathbf{A}\mathbf{y}_n)^T \boldsymbol{\Gamma}^{-1} (\mathbf{x}_n - \mathbf{A}\mathbf{y}_n) > + \frac{1}{2}\log\det\boldsymbol{\Gamma} \tag{22}$$

$$+ \sum_{i=1}^{k} < \log\cosh^2(\frac{1}{2}y_{in}) > - \frac{1}{2}\log\det\boldsymbol{\Sigma}\ \ ] \tag{23}$$

where we omitted irrelevant constants and $<>$ denotes averaging over Q. The thing to note here is that the averages over the log-prior terms are one dimensional and therefore easy to compute numerically. We will however use a well known bound for these averages, which implies that we are deteriorating our bound on the log-likelihood even further (without serious consequences fortunately). We will use the fact that,

$$< \log(2\cosh(\frac{1}{2}x)) >_{\mathcal{G}_x[\mu,\sigma^2]} \quad \leq \quad \frac{1}{8}\sigma^2(1+\alpha^2) + \log(2\cosh(\frac{1}{2}\mu - \frac{1}{4}\alpha\sigma^2)) \tag{24}$$

which is based on Jensen's inequality. The parameter α will be treated as an additional variational parameter. The value that minimizes the bound satisfies the fixed point equation,

$$\alpha = \tanh(\frac{1}{2}\mu - \frac{1}{4}\alpha\sigma^2) \tag{25}$$

Note, that there are $k \times N$ such equations since the parameters have the indices α_{in}. It can be shown that iterating these equations makes the bound tighter at each step. Inserting this bound in F and using the sufficient statistics,

$$< \mathbf{y}_n > \; = \; \boldsymbol{\mu}_n \tag{26}$$

$$< \mathbf{y}_n \mathbf{y}_n^T > \; = \; \boldsymbol{\mu}_n \boldsymbol{\mu}_n^T + \boldsymbol{\Sigma} \tag{27}$$

we arrive at,

$$F = \sum_{n=1}^{N} [\; \frac{1}{2}\mathbf{x}_n^T \boldsymbol{\Gamma}^{-1}\mathbf{x}_n - \mathbf{x}_n^T \boldsymbol{\Gamma}^{-1}\boldsymbol{\mu}_n + \frac{1}{2}\mathbf{tr}[\mathbf{A}^T \boldsymbol{\Gamma}^{-1}\mathbf{A}(\boldsymbol{\Sigma} + \boldsymbol{\mu}_n \boldsymbol{\mu}_n^T)] + \frac{1}{2}\log\det\boldsymbol{\Gamma} \tag{28}$$

$$+ \sum_{i=1}^{k} \left(\frac{1}{4}\boldsymbol{\Sigma}_{ii}(1+\alpha_{in}^2) + 2\log(2\cosh(\frac{1}{2}\mu_{in} - \frac{1}{4}\boldsymbol{\Sigma}_{ii}\alpha_{in})) \right) - \frac{1}{2}\log\det\boldsymbol{\Sigma} \;\;] \tag{29}$$

This is just a function of the parameters $\boldsymbol{\mu}_n, \boldsymbol{\Sigma}, \boldsymbol{\alpha}_n, \mathbf{A}, \boldsymbol{\Gamma}$. Taking derivatives and equating to zero gives the following update rules or fixed point equations,

$$\mathbf{A} = (\sum_{n=1}^{N} \mathbf{x}_n \boldsymbol{\mu}_n^T)(\sum_{n=1}^{N}(\boldsymbol{\Sigma} + \boldsymbol{\mu}_n\boldsymbol{\mu}_n^T))^{-1} \tag{30}$$

$$\boldsymbol{\Gamma} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \mathbf{A}\boldsymbol{\mu}_n)\mathbf{x}_n^T \tag{31}$$

Note that these are identical to the EM-updates in Factor Analysis model. The fixed point equation for the variational parameters (or mean field equations) are given by,

$$\boldsymbol{\alpha}_n = \tanh(\frac{1}{2}\boldsymbol{\mu}_n - \frac{1}{4}\mathbf{diag}(\boldsymbol{\Sigma})\boldsymbol{\alpha}_\mathbf{n}) \tag{32}$$

$$\boldsymbol{\mu}_n = (\mathbf{A}^T\boldsymbol{\Gamma}^{-1}\mathbf{A})^{-1}[\mathbf{A}^T\boldsymbol{\Gamma}^{-1}\mathbf{x}_n - \tanh(\frac{1}{2}\boldsymbol{\mu}_n - \frac{1}{4}\mathbf{diag}(\boldsymbol{\Sigma})\boldsymbol{\alpha}_\mathbf{n})] \tag{33}$$

$$\boldsymbol{\Sigma} = (\mathbf{A}^T\boldsymbol{\Gamma}^{-1}\mathbf{A} + \frac{1}{N}\sum_{n=1}^{N}\mathbf{D}_n)^{-1} \tag{34}$$

$$\tag{35}$$

where we have defined

$$\mathbf{D}_n = \frac{1}{2}(1+\boldsymbol{\alpha}_n^2) - \tanh(\frac{1}{2}\boldsymbol{\mu}_n - \frac{1}{4}\mathbf{diag}(\boldsymbol{\Sigma})\boldsymbol{\alpha}_\mathbf{n}).*\boldsymbol{\alpha}_\mathbf{n} \tag{36}$$

and **diag** denotes putting all off-diagonal elements to zero (producing a diagonal matrix). Note that $\mathbf{D}_n$ is therefore a diagonal matrix (tanh is defined componentwise).

One now simply cycles through the update equations in random order (properly initializing) to minimize F. Note that the fixed point equations are not guaranteed to improve F (except the one for $\boldsymbol{\alpha}$), so one has to be careful about convergence. In practice, the mean field equation for $\boldsymbol{\mu}_n$ easily oscillates and one can counteract this with a damping term,

$$\boldsymbol{\mu}^* = \gamma\boldsymbol{\mu} + (1-\gamma)\mathbf{f}(\boldsymbol{\mu}) \tag{37}$$

where the new $\boldsymbol{\mu}$ is encouraged to stay close to the old one. This does not influence the location of the fixed point. One could also use gradient steps to optimize for $\boldsymbol{\mu}_n, \boldsymbol{\Sigma}$.

References

Attias, H. (1999). Independent factor analysis. *Neural Computation*, 11:803–851.

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

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

Hyvärinen, A. (1997). Independent component analysis by minimization of mutual information. Technical report, Helsinki University of Technology, Laboratory of Computer and Information Science.

Lewicki, M. and Sejnowski, T. (1998). Learning overcomplete representations. Technical report, Howard Hughes Medical Institute, Computational Neurobiology Lab, The Salk Institute. Accepted for publication in Neural Computation.

Welling, M. and Weber, M. (1999). A constrained e.m. algorithm for independent component analysis. In *Proceedings of the 6th Joint Symposium on Neural Computation*.