NONLINEAR INDEPENDENT COMPONENT ANALYSIS USING ENSEMBLE LEARNING: THEORY

Harri Lappalainen

Helsinki University of Technology, Neural Networks Research Centre
P.O.Box 5400, FIN-02015 HUT, Espoo, Finland
E-mail: Harri.Lappalainen@hut.fi URL: http://www.cis.hut.fi/

ABSTRACT

A nonlinear version of independent component analysis is presented. The mapping from sources to observations is modelled by a multi-layer perceptron network and the distributions of sources are modelled by mixtures of Gaussians. The posterior probability of all the unknown parameters is estimated by ensemble learning. In this paper, we present the theory of the method, and in a companion paper experimental results.

1. INTRODUCTION

This paper presents a nonlinear independent component analysis (ICA) algorithm. In much of the ICA research the mapping from the sources $\mathbf{s}(t)$ to the observations $\mathbf{x}(t)$ has been assumed to be linear, but here we consider a more general model where the observations are assumed to be generated by a nonlinear mapping $\mathbf{f}$ from the sources as shown in (1).

$$\mathbf{x}(t) = \mathbf{f}(\mathbf{s}(t)) + \mathbf{n}(t) \tag{1}$$

The observations are assumed to be corrupted by Gaussian noise $\mathbf{n}(t)$. The sources are assumed to be generated by an i.i.d. process and the distribution of each of the sources is modelled by a mixture of Gaussians (MOG) and the nonlinear mapping $\mathbf{f}$ is modelled by a multi-layer perceptron (MLP) network.

It is well known that the problem of determining the nonlinearity $\mathbf{f}$ is indeterminate, that is, there exists an infinite number of different nonlinearities which can produce observations with the same distribution from some independent sources [2]. The Bayesian approach does not suffer from the indeterminacy because a posterior probability can be assigned to all the nonlinear models. Here we use ensemble learning which is a computationally efficient approximation to the full Bayesian treatment [5]. Although each model will have some probability of being responsible for generating the data, typically almost all probability is concentrated on a very small subset of models. It suffices to approximate this high probability region of the posterior probability of the models. In ensemble learning, a parametric approximation is fitted to the posterior probability.

The model structure is presented in Sect. 2. The cost function and details about how to efficiently evaluate it are discussed in Sect. 3. Finally, the learning algorithm is introduced in Sect. 4. This paper is accompanied by [3] which demonstrates the feasibility of the method in simulations with artificial and natural data sets and discusses other existing nonlinear ICA algorithms as well as the limitations and possible extensions of the algorithm described here.

2. MODEL STRUCTURE

The nonlinear mapping $\mathbf{f}$ is modelled by a multi-layer perceptron (MLP) network having two layers.

$$\mathbf{f}(\mathbf{s}(t)) = \mathrm{B}\mathbf{g}(\mathrm{A}\mathbf{s}(t) + \mathbf{a}) + \mathbf{b} \tag{2}$$

The activation function for each of the nonlinear hidden neurons is the hyperbolic tangent, that is, $g(y) = \tanh(y)$. In addition to the weight matrices A and B, both the hidden neurons and the linear output neurons have biases, denoted by $\mathbf{a}$ and $\mathbf{b}$, respectively.

In order to apply the Bayesian approach, each unknown variable in the network is assigned a probability density function (pdf). We apply the usual hierarchical definition of priors. For many parameters, for instance the biases $\mathbf{a}$, it is difficult to assign a prior distribution but we can utilise the fact that each bias occurs in a similar role in the network by assuming that the distribution for each element of vector $\mathbf{a}$ has the same, albeit unknown distribution which is then modelled by a parametric distribution. These new parameters need to be assigned a prior also, but there are far fewer of them.

The noise $\mathbf{n}(t)$ is assumed to be independent and Gaussian with a zero mean. The variance can be different on different channels, and hence the algorithm

can be more accurately be called nonlinear independent factor analysis. Given $\mathbf{s}(t)$, the variance of $\mathbf{x}(t)$ is due to the noise. Therefore $\mathbf{x}(t)$ has the same distribution as the noise except with the mean $\mathbf{f}(\mathbf{s}(t))$.

The distribution of each of the sources is modelled by a mixture of Gaussians. We can think that for each source $s_i(t)$ there is a discrete process which produces a sequence $M_i(t)$ of indices which tell from which Gaussian each $s_i(t)$ is originated. Each Gaussian has its own mean and variance and the probability of different indices is modelled by a soft-max distribution.

The model is defined by the following set of distributions:

$$\begin{aligned}
\mathbf{x}(t) &\sim N(\mathbf{f}(\mathbf{s}(t)), \exp(2\mathbf{v}_n)) && (3)\\
P(M_i(t)=l) &= \exp(c_{il})/\sum_{l'}\exp(c_{il'}) && (4)\\
s_i(t) &\sim N(m_{sil}, \exp(2v_{sil})) && (5)\\
\mathrm{A} &\sim N(0,1) && (6)\\
\mathrm{B} &\sim N(0, \exp(2\mathbf{v}_B)) && (7)\\
\mathbf{a} &\sim N(m_a, \exp(2v_a)) && (8)\\
\mathbf{b} &\sim N(m_b, \exp(2v_b)) && (9)\\
\mathbf{v}_n &\sim N(m_{v_n}, \exp(2v_{v_n})) && (10)\\
\mathbf{c} &\sim N(0, \exp(2v_c)) && (11)\\
\mathbf{m}_s &\sim N(0, \exp(2v_{m_s})) && (12)\\
\mathbf{v}_s &\sim N(m_{v_s}, \exp(2v_{v_s})) && (13)\\
\mathbf{v}_B &\sim N(m_{v_B}, \exp(2v_{v_B})) && (14)
\end{aligned}$$

The prior distributions of m_a, v_a, m_b, v_b and the eight hyperparameters $m_{v_n}, \ldots, v_{v_B}$ are assumed to be Gaussian with zero mean and standard deviation 100, that is, the priors are assumed to be very flat.

The parametrisation of all the distributions is chosen such that the resulting parameters have a roughly Gaussian posterior distribution. This is because the posterior will be modelled by a Gaussian distribution. For example, the variance of the Gaussian distributions is parametrised on a logarithmic scale.

Model indeterminacies are handled by restricting some of the distributions. There is a scaling indeterminacy between the matrix A and the sources, for instance. This is taken care of by setting the variance of A to unity instead of parametrising and estimating it. For the second layer matrix B there is no such indeterminacy. The variance of each column of the matrix is $\exp(2v_{Bj})$. The network can effectively prune out some of hidden neurons by setting the outgoing weights of the hidden neurons to zero, and this is easier if the variance of the corresponding columns of B can be given small values.

3. COST FUNCTION

The goal is to estimate the posterior pdf of all the unknown variables of the model. This is done by ensemble learning which amounts to fitting a simple, parametric approximation to the actual posterior pdf [5]. The cost function C is the misfit between the approximation and the actual posterior and is measured by the Kullback-Leibler information which is sensitive to the probability mass of densities. This is the most important advantage over maximum a posteriori (MAP) estimation which is computationally less expensive but is sensitive to probability density, not mass. This is why MAP estimation suffers from overfitting, which would be a serious problem since there are so many estimated variables, while ensemble learning is able avoid it.

For the time being, let us denote the set of all observations vectors $\mathbf{x}(t)$ by X and denote all the other parameters by a vector θ. The actual posterior pdf is thus $p(\boldsymbol{\theta}|X) = p(X,\boldsymbol{\theta})/p(X)$. The joint pdf $p(X,\boldsymbol{\theta})$ is obtained from the definition of the model in (3)–(14) and $p(X)$ is a normalising factor which does not depend on the unknown variables.

Let us denote the approximation of the posterior pdf by $q(\boldsymbol{\theta})$. In order for the cost function to be computable in practice, a simple factorial form needs to be chosen for the approximation $q(\boldsymbol{\theta})$. The maximally factorial form would be

$$q(\boldsymbol{\theta}) = \prod_i q(\theta_i)\,. \qquad (15)$$

Notice that we have used the usual notation with probability density functions where q with different arguments are taken to be different functions.

The assumption of factorial $q(\boldsymbol{\theta})$ is equivalent to assuming the unknown variables independent given the observations. This is not true, of course, but we have to make this approximation in order to obtain a practical algorithm. The only exception to this maximally factorial form is that the index $M_i(t)$ of the Gaussian and the corresponding source $s_i(t)$ are allowed to have posterior dependency, that is, the terms $q(M_i(t), s_i(t))$ are not further factorised.

The approximation $q(\theta_i)$ should be chosen so that it fits the actual posterior as closely as possible. This is accomplished by choosing $q(\theta_i)$ to be Gaussian for other variables than sources and for sources choosing $q(M_i(t), s_i(t)) = Q(M_i(t))q(s_i(t)|M_i(t))$, where $q(s_i(t)|M_i(t))$ is Gaussian.

Let us denote the mean and variance of $q(\theta_i)$ by $\bar{\theta}_i$ and $\tilde{\theta}_i$, respectively. The result of learning is then an estimate of $\bar{\boldsymbol{\theta}}$ and $\tilde{\boldsymbol{\theta}}$ which tell the posterior mean and variance of all the unknown variables.

The term $p(X)$ is constant with respect to the unknown parameters. Instead of the pure Kullback-Leibler information $K(q(\boldsymbol{\theta})||p(\boldsymbol{\theta}|X))$ it is therefore possible to use the following cost function:

$$\begin{aligned} C(\bar{\boldsymbol{\theta}}, \tilde{\boldsymbol{\theta}}) &= K(q(\boldsymbol{\theta})||p(\boldsymbol{\theta}|X)) - \ln p(X) = \\ &\int q(\boldsymbol{\theta}) \ln \frac{q(\boldsymbol{\theta})}{p(\boldsymbol{\theta}|X)} d\boldsymbol{\theta} - \ln p(X) = \\ &\int q(\boldsymbol{\theta}) \ln \frac{q(\boldsymbol{\theta})}{p(X, \boldsymbol{\theta})} d\boldsymbol{\theta} \,. \end{aligned} \tag{16}$$

Notice that the variables $M_i(t)$ are discrete and those terms are summed over, not integrated over, in the Kullback-Leibler information. For simplicity this is omitted from (16).

Due to simple factorial forms of $q(\boldsymbol{\theta})$ and $p(X, \boldsymbol{\theta})$ the cost function splits into simple terms which are easy to compute. Consequently, it is also easy to differentiate the cost function with respect to $\bar{\boldsymbol{\theta}}$ and $\tilde{\boldsymbol{\theta}}$ and use the derivatives for constructing the learning algorithm.

3.1. Terms of the Cost Function

Almost all the distributions appearing in our model are assumed to be Gaussians, and consequently almost all the terms appearing in the cost function are expectations of logarithms of Gaussian distributions. We shall use the second layer biases $\mathbf{a}$ as an example. For each element a_i, there is one term in $q(\boldsymbol{\theta})$ and $p(X, \boldsymbol{\theta})$, namely the terms $q(a_i)$ and $p(a_i|m_a, v_a)$. The cost function therefore includes terms $\int q(\boldsymbol{\theta}) \ln q(a_i) d\boldsymbol{\theta}$ and $-\int q(\boldsymbol{\theta}) \ln p(a_i|m_a, v_a) d\boldsymbol{\theta}$. In the first expectation the terms $q(a_i)$ only depends on a_i which means that we can integrate over the other variables and we have

$$\int q(\boldsymbol{\theta}) \ln q(a_i) d\boldsymbol{\theta} = \int q(a_i) \ln q(a_i) da_i \,. \tag{17}$$

The same happens for the other integral:

$$\begin{aligned} &- \int q(\boldsymbol{\theta}) \ln p(a_i|m_a, v_a) d\boldsymbol{\theta} = \\ &- \int q(a_i) q(m_a) q(v_a) \ln p(a_i|m_a, v_a) da_i dm_a dv_a \,. \end{aligned} \tag{18}$$

Recall that $q(a_i)$ is Gaussian with mean $\bar{a}_i$ and variance $\tilde{a}_i$. This means that the integral in (17) yields simply

$$\int q(a_i) \ln q(a_i) da_i = -\frac{1}{2} \ln 2\pi e \tilde{a}_i \,. \tag{19}$$

The integral in (18) also fairly easy and it can be shown that the result is

$$\begin{aligned} &- \int q(a_i) q(m_a) q(v_a) \ln p(a_i|m_a, v_a) da_i dm_a dv_a = \\ &\frac{1}{2}[(\bar{a}_i - \bar{m}_a)^2 + \tilde{a}_i + \tilde{m}_a] \exp(2\tilde{v}_a - 2\bar{v}_a) + \bar{v}_a + \frac{1}{2} \ln 2\pi \,. \end{aligned} \tag{20}$$

Again the result is based on the fact that $q(a_i)$, $q(m_a)$ and $q(v_a)$ are Gaussian with means $\bar{a}_i$, $\bar{m}_a$, $\bar{v}_a$ and variances $\tilde{a}_i$, $\tilde{m}_a$, $\tilde{v}_a$, respectively.

The following terms are the only ones whose expectations in (16) give different results than (19) or (20): $q(M_i(t), s_i(t))$, $p(M_i(t)|\mathbf{c}_i)$, $p(s_i(t)|M_i(t), \mathbf{m}_{si}, \mathbf{v}_{si})$ and $p(x_k(t)|\boldsymbol{\theta})$.

The index $M_i(t)$ is discrete and therefore we have a summation instead of integration in the cost function. Let us denote $\dot{s}_{il}(t) = Q(M_i(t) = l)$ and denote the mean and variance of the Gaussian $q(s_i(t)|M_i(t) = l)$ by $\bar{s}_{il}(t)$ and $\tilde{s}_{il}(t)$. Then the expectations of $\ln q(M_i(t), s_i(t))$ in (16) are given by

$$\begin{aligned} &\sum_l \int q(\boldsymbol{\theta}) \ln q(M_i(t) = l, s_i(t)) d\boldsymbol{\theta} = \\ &\sum_l Q(M_i(t) = l)[\ln Q(M_i(t) = l) + \\ &\int q(s_i(t)|M_i(t) = l) \ln q(s_i(t)|M_i(t) = l) ds_i(t)] = \\ &\sum_l \dot{s}_{il}(t)[\ln \dot{s}_{il}(t) - \frac{1}{2} \ln 2\pi e \tilde{s}_{il}(t)] \,. \end{aligned} \tag{21}$$

For the expectation of $-\ln p(M_i(t)|\mathbf{c}_i)$ we shall first evaluate the following integral:

$$\begin{aligned} &- \int q(\mathbf{c}_i) \ln p(M_i(t) = l|\mathbf{c}_i) d\mathbf{c}_i = \\ &- \int q(\mathbf{c}_i)[c_{il} - \ln \sum_{l'} \exp(c_{il'})] d\mathbf{c}_i = \\ &- \bar{c}_{il} + \int q(\mathbf{c}_i) \ln \sum_{l'} \exp(c_{il'}) d\mathbf{c}_i \end{aligned} \tag{22}$$

The resulting integral can be approximated by applying a second order Taylor's series expansion of $\ln \sum_{l'} \exp(c_{il'})$ with respect to $c_{il'}$ around the posterior mean $\bar{c}_{il'}$. This yields the following approximation for the integral:

$$\begin{aligned} &- \int q(\mathbf{c}_i) \ln p(M_i(t) = l|\mathbf{c}_i) d\mathbf{c}_i \approx \\ &- \bar{c}_{il} + \frac{1}{2} \sum_{l'} \phi_{il'}(1 - \phi_{il'}) \tilde{c}_{il'} \,, \end{aligned} \tag{23}$$

where $\phi_{il} = \exp(\bar{c}_{il}) / \sum_{l'} \exp(\bar{c}_{il'})$. Now we see that the expectation of $-\ln p(M_i(t)|\mathbf{c}_i)$ is

$$\begin{aligned} &- \sum_j \dot{s}_{il} \int q(\mathbf{c}_i) \ln p(M_i(t)|\mathbf{c}_i) d\mathbf{c}_i \approx \\ &- \sum_l \dot{s}_{il} \bar{c}_{il} + \frac{1}{2} \sum_{l'} \phi_{il'}(1 - \phi_{il'}) \tilde{c}_{il'} \,. \end{aligned} \tag{24}$$

Since both $q(s_i(t)|M_i(t))$ and $p(s_i(t)|M_i(t), \mathbf{m}_{si}, \mathbf{v}_{si})$ are Gaussian, the terms $-\ln p(s_i(t)|M_i(t), \mathbf{m}_{si}, \mathbf{v}_{si})$ have expectations which are similar to (20):

$$-\sum_l \dot{s}_{il} \int q(s_i(t)|M_i(t)=l)q(m_{sil})q(v_{sil}) \ln p(s_i(t)|M_i(t)=l, m_{sil}, v_{sil}) ds_i(t) dm_{sil} dv_{sil}\,, \quad (25)$$

which equals to the sum of terms

$$\frac{1}{2}[(\bar{s}_{il}(t) - \bar{m}_{sil})^2 + \tilde{s}_{il}(t) + \tilde{m}_{sil}] \exp(2\tilde{v}_{sil} - 2\bar{v}_{sil}) + \bar{v}_{sil} + \frac{1}{2}\ln 2\pi \quad (26)$$

weighted by $\dot{s}_{il}$.

The observations $x_k(t)$ are known — unless there are missing values — which means that there are no terms of the form $\ln q(x_k(t))$. The expectations of $-\ln p(x_k(t)|\boldsymbol{\theta})$ are the most difficult terms in the cost function. If the posterior mean and variance of the function $f_k(\mathbf{s}(t))$ are known — let us denote them by $\bar{f}_k(t)$ and $\tilde{f}_k(t)$ for short — then the expectation has a form similar to (20):

$$\frac{1}{2}[(x_k(t) - \bar{f}_k(t))^2 + \tilde{f}_k(t)] \exp(2\tilde{v}_{nk} - 2\bar{v}_{nk}) + \bar{v}_{nk} + \frac{1}{2}\ln 2\pi\,. \quad (27)$$

3.2. Posterior Mean and Variance of $\mathbf{f}(\mathbf{s}(t))$

This section describes how to compute the posterior mean and variance of the outputs $f_k(\mathbf{s}(t))$ of the MLP network. Ordinarily the inputs, weights and biases of an MLP network have fixed values. Here the inputs $\mathbf{s}(t)$, weights A, B and the biases $\mathbf{a}$, $\mathbf{b}$ have posterior distributions which means that we also have a posterior distribution of the outputs. One way to evaluate the posterior mean and variance is to propagate distributions instead of fixed values through the network. Whole distributions would be quite tricky to deal with, and therefore we are going to characterise the distributions by their mean and variance only.

The sources have mixture-of-Gaussians distributions for which it is easy to compute the mean and variance:

$$\bar{s}_i(t) = \sum_l \dot{s}_{il}(t)\bar{s}_{il}(t) \quad (28)$$

$$\tilde{s}_i(t) = \sum_l \dot{s}_{il}(t)[\tilde{s}_{il}(t) + (\bar{s}_{il}(t) - \bar{s}_i(t))^2]\,. \quad (29)$$

Then the sources are multiplied with the first layer weight matrix A and the bias $\mathbf{a}$ is added. Let us denote the result by $y_j(t) = a_j + \sum_i A_{ji}s_i(t)$. Since the sources, weights and biases are all mutually independent a posteriori, the following equations hold:

$$\bar{y}_j(t) = \bar{a}_j + \sum_i \bar{A}_{ji}\bar{s}_i(t) \quad (30)$$

$$\tilde{y}_j(t) = \tilde{a}_j + \sum_i \bar{A}^2_{ji}\tilde{s}_i(t) + \tilde{A}_{ji}[\bar{s}^2_i(t) + \tilde{s}_i(t)]\,. \quad (31)$$

Equation (31) follows from the identity

$$\mathrm{var}(\alpha) = \langle\alpha^2\rangle - \langle\alpha\rangle^2\,. \quad (32)$$

For computing the posterior mean of the output $g_j(y_j(t))$ of a hidden neuron, we shall utilise the second order Taylor's series expansion of g_j around the posterior mean $\bar{y}_j(t)$ of its input. This means that we approximate

$$g_j(y_j(t)) \approx g_j(\bar{y}_j(t)) + (y_j(t) - \bar{y}_j(t))g_i'(\bar{y}_j(t)) + \frac{1}{2}(y_j(t) - \bar{y}_j(t))^2 g_i''(\bar{y}_j(t))\,. \quad (33)$$

Since the posterior mean of $y_j(t)$ is by definition $\bar{y}_j(t)$, the second term vanishes when evaluating the posterior mean, while the posterior mean of $(y_j(t) - \bar{y}_j(t))^2$ is by definition the posterior variance $\tilde{y}_j(t)$. We thus have

$$\bar{g}_j(y_j(t)) \approx g_j(\bar{y}_j(t)) + \frac{1}{2}\tilde{y}_j(t)g_j''(\bar{y}_j(t))\,. \quad (34)$$

The second order expansion was chosen because those are the terms whose posterior mean can be expressed in terms of posterior mean and variance of the input. Higher order terms would have required higher order cumulants of the input, which would have increased the computational complexity with little extra benefit.

For the posterior variance of $g_j(y_j(t))$ the second order expansion would result in terms which need higher than second order knowledge about the inputs. Therefore we shall use the first order Taylor's series expansion which then yields the following approximation for the posterior variance of $g_j(y_j(t))$:

$$\tilde{g}_j(y_j(t)) \approx [g_j'(\bar{y}_j(t))]^2\tilde{y}_j(t)\,. \quad (35)$$

The next step is to compute the mean and variance of the output after the second layer mapping. The outputs are given by $f_k(t) = b_k + \sum_j B_{kj}g_j(t)$. The equation for the posterior mean $\bar{f}_k(t)$ is similar to (30):

$$\bar{f}_k(t) = \bar{b}_k + \sum_j \bar{B}_{kj}\bar{g}_j(t)\,. \quad (36)$$

The equation for the posterior variance $\tilde{f}_k(t)$ is more complicated than (31), however, since $s_i(t)$ are independent a posterior but $g_j(t)$ are not. This is because

each $s_i(t)$ affects several — potentially all — $g_j(t)$. In other words, each $s_i(t)$ affects each $f_k(t)$ through several paths which interfere. This interference needs to be taken into account when computing the posterior variance of $f_k(t)$.

We shall use a first order approximation of the mapping $\mathbf{f}(\mathbf{s}(t))$ for measuring the interference. This is consistent with the first order approximation of the nonlinearities g_j and yields the following equation for the posterior variance of $f_k(t)$:

$$\begin{aligned}\bar{f}_k(t) &\approx \sum_i \left(\frac{\partial f_k(t)}{\partial s_i(t)}\right)^2 \tilde{s}_i(t) + \tilde{b}_k + \\ &\sum_j \bar{B}_{kj}^2 \tilde{g}_j^*(t) + \tilde{B}_{jk}[\bar{g}_j^2(t) + \tilde{g}_j(t)]\,, \quad (37)\end{aligned}$$

where the posterior means of the partial derivatives are obtained by the chain rule

$$\begin{aligned}\frac{\partial f_k(t)}{\partial s_i(t)} &= \sum_j \frac{\partial f_k(t)}{\partial g_j(t)} \frac{\partial g_j(t)}{\partial y_j(t)} \frac{\partial y_j(t)}{\partial s_i(t)} = \\ &\sum_j \bar{B}_{kj} g_j'(\bar{y}_j(t)) \bar{A}_{ji} \quad (38)\end{aligned}$$

and $\tilde{g}_j^*(t)$ denotes the posterior variance of $g_j(t)$ without the contribution from the sources. It can be computed as follows:

$$\tilde{y}_j^*(t) = \tilde{a}_j + \sum_i \tilde{A}_{ji}[\bar{s}_i^2(t) + \tilde{s}_i(t)] \quad (39)$$

$$\tilde{g}_j^*(t) \approx [g_j'(\bar{y}_j(t))]^2 \tilde{y}_j^*(t)\,. \quad (40)$$

Notice that $\tilde{s}_i(t)$ appears in (39) and $\tilde{g}_j(t)$ appears in (37). These terms do not contribute to interference, however, because they are the parts which are randomised by multiplication with A_{ji} or B_{kj} and randomising the phase destroys the interference, to use an analogy from physics.

4. UPDATE RULES

In the previous section we derived all the equations needed for the computation of the cost function. Given the posterior means $\bar{\boldsymbol{\theta}}$ and variances $\tilde{\boldsymbol{\theta}}$ and discrete posterior probabilities $\dot{s}_{il}(t)$, we can compute the cost function which measures the quality of the approximation of the posterior pdf of the unknown variables. Any standard optimisation algorithm could be used for minimising the cost function, but it is sensible to utilise the particular form of the function. Due to lack of space, we shall only outline the update rules but a more detailed description can be found in [4].

Let us denote $C = C_q + C_p$, where C_q is the part originating from the expectation of $\ln q(\boldsymbol{\theta})$ and C_p is the part originating from expectation of $-\ln p(X, \boldsymbol{\theta})$. We shall see how it is possible to derive efficient fixed point algorithms for $\bar{\theta}$ and $\tilde{\theta}$ assuming that we have computed the gradients of C_p with respect to the current estimates of $\bar{\theta}$ and $\tilde{\theta}$.

Since C_q has a term $-1/2 \ln 2\pi e\tilde{\theta}$ for each $\tilde{\theta}$ whose posterior is approximated by Gaussian $q(\theta)$, solving for $\partial C/\partial\tilde{\theta} = 0$ yields an update rule for $\tilde{\theta}$:

$$0 = \frac{\partial C_p}{\partial\tilde{\theta}} + \frac{\partial C_q}{\partial\tilde{\theta}} = \frac{\partial C_p}{\partial\tilde{\theta}} - \frac{1}{2\tilde{\theta}} \;\Rightarrow\; \tilde{\theta} = \frac{1}{2\frac{\partial C_p}{\partial\tilde{\theta}}}\,. \quad (41)$$

Now suppose $\ln p(X, \boldsymbol{\theta})$ is roughly quadratic with respect to θ:

$$-\ln p(X, \boldsymbol{\theta}) \approx \alpha + (\theta - \theta_{\text{opt}})^2 \beta\,. \quad (42)$$

Then C_p would be

$$C_p \approx \alpha + [(\bar{\theta} - \theta_{\text{opt}})^2 + \tilde{\theta}]\beta \quad (43)$$

and hence the derivatives with respect to $\bar{\theta}$ and $\tilde{\theta}$ would be

$$\frac{\partial C_p}{\partial\bar{\theta}} = 2(\bar{\theta} - \theta_{\text{opt}})\beta \quad (44)$$

$$\frac{\partial C_p}{\partial\tilde{\theta}} = \beta\,. \quad (45)$$

As C_q does not depend on $\bar{\theta}$, the optimal value for $\bar{\theta}$ is evidently θ_{opt} and solving for that we obtain an update rule for $\bar{\theta}$:

$$\bar{\theta}_{\text{new}} = \theta_{\text{opt}} = \bar{\theta}_{\text{old}} - \frac{\frac{\partial C_p}{\partial\bar{\theta}}}{2\frac{\partial C_p}{\partial\tilde{\theta}}} = \bar{\theta}_{\text{old}} - \frac{\partial C_p}{\partial\bar{\theta}}\tilde{\theta}\,. \quad (46)$$

Since this update rule makes a quadratic approximation of the cost function C, it can be viewed as Newton iteration which assumes that $\bar{\theta}$ is the only variable which changes because the quadratic approximation does not take into account the cross terms $\partial^2 C/\partial\theta_i\partial\theta_j$. In practice all the weights A and B, for instance, are adapted simultaneously and each weight affects the optimal value of the other weights. In [4] it is explained how it is possible to compensate for the error which results in the invalid assumption of independent adaptations.

4.1. Update Rules for Posterior Source Distributions

The posterior distributions of the sources are effectively approximated by mixtures of Gaussians which means that the above update rules are not directly applicable

for them. The new values for discrete posterior probabilities of the source indices $\dot{s}_{il}(t)$ and the posterior means $\bar{s}_{il}(t)$ and variances $\tilde{s}_{il}(t)$ of the Gaussians corresponding to different source are most easily solved by making a quadratic approximation for C_f based on the derivatives $\partial C_f(t)/\partial \bar{s}_i(t)$ and $\partial C_f/\partial \tilde{s}_i(t)$, where C_f denotes the sum of terms of the form (27).

The usefulness of this approximation is based on the fact that the cost function tries to minimise the misfit between the approximation and the actual posterior. If we can solve the actual posterior and show that it can be described by our parametric approximation, we know that the cost function will be minimised by setting the parameters of the approximation to values corresponding to the actual posterior.

In this case, making a second order approximation for C_f is equivalent to approximating $p(X|s_i(t))$ by an unnormalised Gaussian distribution. Since the prior $p(s_i(t))$ is a mixture of Gaussians and the posterior will be given by $p(s_i(t)|X) = p(X|s_i(t))p(s_i(t))/p(X)$, we notice that the posterior is also a mixture of Gaussians since a Gaussian multiplied with an unnormalised Gaussian will produce another unnormalised Gaussian and the normalising factor $p(X)$ will then make sure that the posterior is a normalised mixture of Gaussians. From the resulting mixture of Gaussians one can then determine the values for $\dot{s}_{il}(t)$, $\bar{s}_{il}(t)$ and $\tilde{s}_{il}(t)$. In [1], a similar method without approximations was used for a linear model. Due to the linearity of the mapping, $\ln p(X|s_i(t))$ is quadratic and no approximations are needed.

4.2. Avoiding Problems Originating from Approximations

When constructing a learning algorithm which is based on approximations of the cost function, it is important to make sure that learning does not drive the network into areas of the parameter space where the approximations are no longer valid.

The approximations in (34) and (35) are based on a roughly quadratic or linear behaviour of the nonlinearities. This assumption is quite good if the posterior variance $\tilde{y}_j(t)$ of the inputs to the hidden neurons is not very large.

Since the approximations take into account only local behaviour of the nonlinearities g_j and MLP networks typically have multimodal posterior distributions, there must be areas of the parameter space where the second order derivative of the posterior probability with respect to one of the parameters θ is positive. This means that $\partial C_p/\partial \tilde{\theta}$ is negative which in turn means that it appears that the cost function can be made arbitrarily small by letting $\tilde{\theta}$ grow.

It is easy to see that the problem is due to the local estimate of g since the logarithm of the posterior eventually has to go to negative infinity. The derivative $\partial C_p/\partial \tilde{\theta}$ will thus be positive for large $\tilde{\theta}$, but the local estimate of g_j fails to account for this.

In order to discourage the network from adapting itself into areas of parameter space where the problems might occur and to deal with the problem if it nevertheless occurred, the terms in (34) which have negative contribution to $\partial C_p/\partial \tilde{\theta}$ will be neglected in the computation of the gradients. As this can only make the estimate of $\tilde{\theta}$ in (41) smaller, this leads, in general, to increasing the accuracy of the approximations in (34) and (35).

5. COMPUTATIONAL COMPLEXITY

Most of the computation in the forward phase is spent in (38). The computation of the gradients of (38) is also where most computation of the backward phase takes place. We have previously tried making the assumption that the outputs of the hidden neurons are independent a posteriori which then obviates the need of equation (38) because (37) can be replaced by an equation similar to (31). Simulations have shown that this assumption is too inaccurate. The computational complexity of this algorithm is proportional to $IJKT$, where I, J, K and T denote the source dimension, the number of hidden neurons, the number of outputs and the number of observation vectors. In a typical case $K > I$ which means that the computational complexity of the second layer dominates and the computational complexity is higher than in ordinary back-propagation by a factor I.

6. REFERENCES

[1] H. Attias. Independent factor analysis. *Neural Computation*, 11(4):803–851, 1999.

[2] A. Hyvärinen and P. Pajunen. Nonlinear independent component analysis: Existence and uniqueness results. *Neural Networks*, 12(2):209–219, February 1999.

[3] H. Lappalainen, X. Giannakopoulos, A. Honkela, and J. Karhunen. Nonlinear independent component analysis using ensemble learning: Experiments and discussion. In *Proc. ICA'2000*. Submitted.

[4] H. Lappalainen and A. Honkela. Bayesian nonlinear independent component analysis by multi-layer perceptrons. In M. Girolami, ed., *Advances in Independent Component Analysis*. Springer, Berlin, 2000. In Press.

[5] H. Lappalainen and J. W. Miskin. Ensemble learning. In M. Girolami, ed., *Advances in Independent Component Analysis*. Springer, Berlin, 2000. In Press.