An Unsupervised Ensemble Learning Method for Nonlinear Dynamic State-Space Models

Harri Valpola and Juha Karhunen
Helsinki University of Technology, Neural Networks Research Centre
P.O.Box 5400, FIN-02015 HUT, Espoo, FINLAND
email: `Harri.Valpola@hut.fi, Juha.Karhunen@hut.fi`
URL: `http://www.cis.hut.fi/projects/ica/bayes/`

Neural Computation 14(11), pp. 2647–2692, (2002), MIT Press.

Abstract

A Bayesian ensemble learning method is introduced for unsupervised extraction of dynamic processes from noisy data. The data are assumed to be generated by an unknown nonlinear mapping from unknown factors. The dynamics of the factors are modeled using a nonlinear state-space model. The nonlinear mappings in the model are represented using multilayer perceptron networks. The proposed method is computationally demanding, but it allows the use of higher dimensional nonlinear latent variable models than other existing approaches. Experiments with chaotic data show that the new method is able to blindly estimate the factors and the dynamic process which have generated the data. It clearly outperforms currently available nonlinear prediction techniques in this very difficult test problem.

Keywords: Ensemble learning, dynamic process analysis, blind signal separation, state-space models, time series prediction, neural networks, Bayesian modeling.

1 Introduction

In unsupervised learning, the goal is to build a model which captures the statistical regularities in the observed data and provides a compact and meaningful representation for the data. From such a representation it is often much easier to understand the basic characteristics of the data than directly from the raw data.

In our case, we wish to find a good nonlinear model for the dynamic process and the underlying factors or sources that have generated the observed data. The proposed method can be viewed as a nonlinear dynamic extension to standard linear principal component analysis (PCA) or factor analysis (FA). It is also related to nonlinear blind source separation (BSS) and independent component analysis (ICA). For these problems we have recently developed several methods described in more detail in (Hyvärinen et al., 2001), Chapter 17, and in (Lappalainen and Honkela, 2000; Valpola et al., 2000; Valpola, 2000b).

In particular, a method called nonlinear factor analysis (NFA) (Lappalainen and Honkela, 2000) serves as the starting point of the new method introduced in this paper. In nonlinear factor analysis, it is assumed that the noisy data vectors $\mathbf{x}(t)$ have been generated by the factors or source signals $\mathbf{s}(t)$ through a nonlinear mapping $\mathbf{f}$. Hence the data vectors are modeled by

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

where $\mathbf{n}(t)$ denotes additive gaussian noise. More specifically, in the NFA data model (1) it is assumed that the factors or sources $s_i(t)$ (components of the vector $\mathbf{s}(t)$) are gaussian, and the nonlinear mapping $\mathbf{f}$ is modeled by the standard multilayer perceptron (MLP) network. The MLP network (Bishop, 1995; Haykin, 1998) is used here because of its universal approximation property and ability to model well both mildly and strongly nonlinear mappings $\mathbf{f}$. However, the cost function used in Bayesian ensemble learning is quite different from the standard mean-square error used in the back-propagation algorithm and its variants (Bishop, 1995; Haykin, 1998). Another important difference is that the learning procedure is completely unsupervised or blind, because in NFA only the outputs $\mathbf{x}(t)$ of the MLP network are known.

We have extended NFA somewhat to so-called nonlinear independent factor analysis (NIFA) (Hyvärinen et al., 2001; Lappalainen and Honkela, 2000; Valpola et al., 2000; Valpola, 2000b). There the same model form (1) is still employed, but now the sources $s_i(t)$ are assumed to be statistically independent, and they are modeled as mixtures-of-gaussians instead of plain gaussians. NIFA can be understood as a nonlinear blind source separation method, and it seems to be applicable to much larger scale problems than most previously introduced nonlinear BSS methods, discussed in (Hyvärinen et al., 2001), Chapter 17. We have successfully applied the NFA and NIFA methods to finding useful compact representations for both artificial and real-world data sets in (Lappalainen and Honkela, 2000; Valpola et al., 2000; Honkela and Karhunen, 2001).

However, the NFA and NIFA methods have still the drawback that they do not take into account the possible dependences of subsequent data vectors. This means that like in standard principal or independent component analysis, the data vectors $\mathbf{x}(t)$ can be randomly shuffled and the found factors or sources remain the same. While this may be acceptable in certain situations, temporally (or spatially) adjacent data vectors are often in practice more or less statistically dependent. This dependence should then be modeled appropriately, too, and utilized in learning for improving the results.

In this paper, we extend the NFA method so that temporal dependences are taken into account by using a nonlinear state-space model for the factors $\mathbf{s}(t)$. An important advantage of the proposed new method is its ability to learn a high-dimensional latent (source) space. We have also reasonably solved computational and over-fitting problems which have been major obstacles in developing this kind of unsupervised methods thus far. Potential applications for the present method include prediction and process monitoring, control and identification.

The rest of the paper is organized as follows. In the next section, we briefly overview Bayesian methods, in particular ensemble learning, giving references to related work. The structure of the dynamic model is represented in more detail in Section 3. After this, we consider the evaluation of the cost function. The overall description and more detailed implementation of the learning method are presented in Section 5, followed by experimental results in Section 6. Future directions and potential problems are treated in Section 7, and the conclusions of the study are given in the last section. Some detailed derivations have been moved to appendices for better readability.

2 Bayesian methods and ensemble learning

2.1 Bayesian inference

In Bayesian data analysis and estimation methods (see for example (Bishop, 1995; Jordan, 1999; Gelman et al., 1995; Neal, 1996)) for continuous variables, all the uncertain quantities are modeled in terms of their joint probability density function (pdf). The key principle is to construct the joint posterior pdf for all the unknown quantities in a model, given the data sample. This posterior density contain all the relevant information on the parameters to be estimated in parametric models, or the predictions in non-parametric prediction or classification tasks.

Denote by $\mathcal{H}$ the particular model under consideration, and by $\boldsymbol{\theta}$ the set of model parameters that we wish to infer from a given data set X. The posterior probability density $p(\boldsymbol{\theta}|X, \mathcal{H})$ of the parameters given the data X and the model $\mathcal{H}$ can be computed from the Bayes' rule

$$p(\boldsymbol{\theta}|X, \mathcal{H}) = \frac{p(X|\boldsymbol{\theta}, \mathcal{H})p(\boldsymbol{\theta}|\mathcal{H})}{p(X|\mathcal{H})} \tag{2}$$

Here $p(X|\boldsymbol{\theta}, \mathcal{H})$ is the likelihood of the parameters $\boldsymbol{\theta}$, $p(\boldsymbol{\theta}|\mathcal{H})$ is the prior pdf of the parameters, and $p(X|\mathcal{H})$ is a normalizing constant[1]. The term $\mathcal{H}$ denotes all the assumptions made in defining the model, such as choice of a MLP network, specific noise model, etc. We have explicitly included it in the Bayes' rule (2) to emphasize the dependence of the results on these modeling assumptions.

Using the above notation, the normalizing term

$$p(X|\mathcal{H}) = \int_{\boldsymbol{\theta}} p(X|\boldsymbol{\theta}, \mathcal{H})p(\boldsymbol{\theta}|\mathcal{H})d\boldsymbol{\theta} \tag{3}$$

can be directly understood as the marginal probability of the observed data X assuming the model $\mathcal{H}$ ((Bishop, 1995), Chapter 10). Therefore it is called the evidence of the model $\mathcal{H}$. By evaluating the evidences $p(X|\mathcal{H}_i)$ for different models, one can choose the model with the highest evidence. This is the model which describes the observed data with the highest probability among the compared models.

The parameters $\boldsymbol{\theta}$ of a particular model $\mathcal{H}_i$ are often estimated using the maximum likelihood (ML) method. This non-Bayesian method chooses the estimates $\hat{\boldsymbol{\theta}}$ which maximize the likelihood $p(X|\boldsymbol{\theta}, \mathcal{H})$ of the data, assuming that the parameters $\boldsymbol{\theta}$ are non-random constants and omitting the potential prior information on them. The maximum a posteriori (MAP) method is the simplest Bayesian method. It applies the same strategy to the posterior distribution $p(\boldsymbol{\theta}|X, \mathcal{H})$ by finding the parameter values that maximize this distribution. This can be done by maximizing the numerator in (2), because the normalizing denominator (3) does not depend on the parameters $\boldsymbol{\theta}$.

However, using point estimates like those provided by the ML or MAP methods is often problematic. In these methods, the model order estimation and over-fitting (choosing too complicated a model for the given data) are severe problems. In high dimensions, the width of a peak of the posterior density is far more important to the probability mass than its height. For complex models, it is usually computationally prohibitive to try to express the posterior pdf exactly. This prevents the use of classical Bayesian methods, for example the minimum mean-square error estimation in all but the simplest problems. Furthermore, such classical methods still provide a point estimate $\hat{\boldsymbol{\theta}}$ which is somewhat arbitrarily chosen from the possible values of $\boldsymbol{\theta}$ in the posterior density (Sorenson, 1980).

Instead of searching for some point estimate, the correct Bayesian procedure is to perform estimation by averaging over the posterior distribution $p(\boldsymbol{\theta}|X, \mathcal{H})$. This means that the estimates will be sensitive to regions where the probability mass is large instead of being sensitive to high values of the pdf (Lappalainen and Honkela, 2000; Miskin and MacKay, 2001). One can go even one step further if possible and reasonable (for example in supervised learning problems), and make use of the complete set of models. This means that predicted values are obtained by computing a weighted sum over the predictions of several models, where the weighting coefficients are proportional to the evidence of each model (Bishop, 1995). More probable models therefore contribute more to the predicted or estimated values. Such a procedure appropriately solves the

[1] We have dropped out the subscripts of all pdf's for simplifying the notation; they are assumed to be the same as the arguments of the pdf's in question unless otherwise stated.

issues related to the model complexity and choice of a specific model among several candidates. The evidences $p(X|\boldsymbol{\theta}, \mathcal{H})$ of too simple models are low, because they leave a lot of the data unexplained. On the other hand, too complex models occupy little probability mass, even though they often show a high but very narrow peak in their posterior pdf's corresponding to the over-fitted parameters.

The first practical Bayesian method for neural networks was introduced by MacKay (MacKay, 1992). He used for the posterior density of the parameters a gaussian approximation around their MAP estimate to evaluate the evidence of the model. However, such parametric approximation methods (Bishop, 1995) often do not perform well enough in practical real-world problems. Another group of Bayesian methods used in neural network models consists of Markov Chain Monte Carlo (MCMC) techniques for numerical integration (Neal, 1996; Robert and Gasella, 1999). They perform the necessary integrations needed in evaluating the evidence (3) and the posterior density (2) numerically by drawing samples from the true posterior distribution. MCMC techniques have been successfully applied to practical supervised learning problems, for example in (Lampinen and Vehtari, 2001) by using a MLP network structure. However, MCMC methods cannot be used in large scale unsupervised learning problems, because the number of parameters grows far too large for estimating them in any reasonable time.

Other fully Bayesian approaches include various variational methods (Jordan et al., 1999) for approximating the integration by a tractable problem, and mean field approach (Winther, 1998), where the problem is simplified by neglecting certain dependences between the random variables. In this paper, we use a particularly powerful variational approximation method, called ensemble learning. It is applicable to unsupervised learning problems, too, and is discussed in more detail in the following.

2.2 Ensemble learning

Ensemble learning (Barber and Bishop, 1998; Lappalainen and Miskin, 2000), also called variational Bayes, is a method developed recently (Hinton and van Camp, 1993; MacKay, 1995) for approximating the posterior density (2). It can be used both for parametric and variational approximation. In the former, some parameters characterizing the posterior pdf are optimized while in the latter, a whole function is optimized.

More specifically, ensemble learning employs the Kullback-Leibler (KL) divergence (also called KL distance or information) between two probability distributions $q(v)$ and $p(v)$. This is defined by the cost function

$$\mathcal{J}_{\mathrm{KL}}(q \parallel p) = \mathrm{E}_q \left\{ \ln \frac{q}{p} \right\} = \int_v q(v) \ln \frac{q(v)}{p(v)} dv \tag{4}$$

The KL divergence $\mathcal{J}_{\mathrm{KL}}$ measures the difference in the probability mass between the densities $q(v)$ and $p(v)$. Its minimum value 0 is achieved when the two densities are the same.

A key idea in ensemble learning is to minimize the misfit between the actual posterior pdf and its parametric approximation. The exact posterior pdf (2) is written in the form $p(S, \boldsymbol{\theta}|X, \mathcal{H})$, where we have now separated the source signals or latent variables S corresponding to the data X from the other parameters $\boldsymbol{\theta}$ for clarity and later use. Denoting respectively the approximation of this posterior pdf by $q(S, \boldsymbol{\theta})$ yields for the KL divergence of these two distibutions

$$\mathcal{J}_{\mathrm{KL}}(q \parallel p) = \int_S \int_{\boldsymbol{\theta}} q(S, \boldsymbol{\theta}) \ln \frac{q(S, \boldsymbol{\theta})}{p(S, \boldsymbol{\theta}|X, \mathcal{H})} \boldsymbol{\theta} dS \tag{5}$$

The cost function used in ensemble learning is defined as follows (Hinton and van Camp, 1993; MacKay, 1995):

$$\mathcal{C}_{\mathrm{KL}} = \mathcal{J}_{\mathrm{KL}}(q \parallel p) - \ln p(X|\mathcal{H}) \tag{6}$$

The main reason for using (6) instead of (5) is that the posterior pdf $p(S, \boldsymbol{\theta}|X, \mathcal{H})$ includes the evidence term $p(X|\mathcal{H})$. This requires an integration (3) which is usually intractable. Inserting (5) into (6) yields after slight manipulation

$$\mathcal{C}_{\mathrm{KL}} = \int_S \int_{\boldsymbol{\theta}} q(S, \boldsymbol{\theta}) \ln \frac{q(S, \boldsymbol{\theta})}{p(X, S, \boldsymbol{\theta}|\mathcal{H})} d\boldsymbol{\theta} dS \tag{7}$$

which does not need $p(X|\mathcal{H})$. In fact, the cost function provides a bound for the evidence $p(X|\mathcal{H})$. Recalling that the KL divergence $\mathcal{J}_{\mathrm{KL}}(q \parallel p)$ is always nonnegative, it follows directly from (6) that the cost function $\mathcal{C}_{\mathrm{KL}}$ satisfies the inequality

$$\mathcal{C}_{\mathrm{KL}} \geq - \ln p(X|\mathcal{H}) \tag{8}$$

This shows that minimizing the cost function $\mathcal{C}_{\mathrm{KL}}$ used in ensemble learning is equivalent to maximizing the bound $\ln p(X|\mathcal{H})$ for the evidence $p(X|\mathcal{H})$ of the model $\mathcal{H}$.

For computational reasons, the approximation $q(S, \boldsymbol{\theta})$ of the posterior pdf must be simple enough. This is achieved by neglecting some or even all the dependences between the variables. Our choice for $q(S, \boldsymbol{\theta})$ is a multivariate gaussian density with an almost diagonal covariance matrix. A gaussian distribution with mutually uncorrelated and hence also statistically independent components is often used as the approximation of the posterior pdf in ensemble learning. Even this crude approximation is adequate for finding the region where the mass of the actual posterior density is concentrated. The mean value of each component of the gaussian approximation can be used as a reasonably good point estimate of the corresponding parameter or variable if necessary, and the respective variance provides a useful simple measure of the reliability of this estimate.

The main motivation in resorting to ensemble learning is that it avoids over-fitting which would be difficult to avoid in our problem if ML or MAP estimates were used. Ensemble learning allows one to select a model having appropriate complexity, making often possible to infer the correct number of sources or factors in our case. The Bayes' rule yields for the posterior probability of the model $\mathcal{H}_i$

$$p(\mathcal{H}_i|X) = \frac{p(X|\mathcal{H}_i)p(\mathcal{H}_i)}{p(X)} \tag{9}$$

Assume now that all the models $\mathcal{H}_i$ have the same constant prior probability $p(\mathcal{H}_i)$. Then the model with the highest posterior probability $p(\mathcal{H}_i|X)$ is the model with the highest evidence $p(X|\mathcal{H}_i)$. The inequality (8) shows that minimization of the cost function $\mathcal{C}_{\mathrm{KL}}$ provides a bound on the logarithm of the evidence of the model. If this bound is close to the correct value, the model $\mathcal{H}_i$ that maximizes the posterior probability (9) is also the one that minimizes the cost function $\mathcal{C}_{\mathrm{KL}}$ used in ensemble learning (Miskin and MacKay, 2001).

The cost function (7) has also a simple but insightful information theoretic interpretation (Hinton and van Camp, 1993; Valpola, 2000a). It represents the combined cost of coding the model and the data using that model. The optimal solution uses the minimum total amount of bits for encoding. If the model $\mathcal{H}$ is made more complicated, the amount of bits needed for coding the data usually decreases because the model is able to represent the data more accurately. On the other hand, the amount of bits needed for coding the model then increases. Hence ensemble learning tries to find the optimum balance between the representation power and the complexity of the model. It behaves in this respect qualitatively quite similarly to the well-known minimum description length method (see for example (Bishop, 1995)) of model selection.

Until recently, ensemble learning has been applied mainly to supervised learning problems (for example (Barber and Bishop, 1998)), but it can be used for unsupervised learning as well. The first author employed it for standard linear independent component analysis in (Lappalainen, 1999) using a fixed form approximation, with application to real-world speech data. Recently, several authors have studied ensemble learning in similar problems using slightly different assumptions and methods (Choudrey et al., 2000; Miskin and MacKay, 2000; Miskin and MacKay, 2001; Roberts and Everson, 2001). The work by Attias (Attias, 1999a; Attias, 1999b; Attias, 2000), summarized in (Attias, 2001), is also closely related to ours, even though point estimates are partly used in his early work. In (Attias, 2000), this type of methods have been extended to take into account temporal dependences, and in (Miskin and MacKay, 2000; Miskin and MacKay, 2001) for spatial dependences in image deconvolution applications. However, in all these previous works unsupervised ensemble learning or related Bayesian methods have been applied to linear generative data models only. The learning process then becomes clearly simpler, but on the other hand the representation power of such linear models is clearly weaker than in this work.

3 The dynamic model

3.1 Model structure

In this section, we first introduce our dynamic model and its neural network realization using a graphical Bayesian model, and define the distributions of parameters needed in the Bayesian approach. In the next subsection, we justify our choices and discuss the properties of the selected model. Finally, we deal briefly with relevant related work.

The generative model for the observations $\mathbf{x}(t)$ is as follows:

$$\mathbf{x}(t) = \mathbf{f}(\mathbf{s}(t)) + \mathbf{n}(t) \tag{10}$$

$$\mathbf{s}(t) = \mathbf{g}(\mathbf{s}(t-1)) + \mathbf{m}(t) \tag{11}$$

Equation (10) is the same as in the nonlinear factor analysis model (1). Hence we assume that the factors or sources $s_i(t)$ (components of $\mathbf{s}(t)$) are gaussian. The gaussian additive noise $\mathbf{n}(t)$ models the imperfections and errors in the nonlinear mapping $\mathbf{f}$.

The model (11) for the dynamics of the factors $\mathbf{s}(t)$ is fairly similar. The current factors $\mathbf{s}(t)$ are generated through another nonlinear tranformation $\mathbf{g}$ from the factors $\mathbf{s}(t-1)$ at the previous time instant. This means that the factors can be interpreted as the states of a dynamical system. The noise or error term $\mathbf{m}(t)$ is again gaussian. In the dynamic model (11) it is often called process noise or innovation process.

The nonlinear mappings $\mathbf{f}$ and $\mathbf{g}$ are modeled by MLP networks. The functional form of observation or data mapping $\mathbf{f}$ in (10) is

$$\mathbf{f}(\mathbf{s}(t)) = \mathbf{B}\tanh[\mathbf{A}\mathbf{s}(t) + \mathbf{a}] + \mathbf{b} \tag{12}$$

where $\mathbf{A}$ and $\mathbf{B}$ are the weight matrices of the hidden and output layers of the MLP network, respectively, and $\mathbf{a}$ and $\mathbf{b}$ the corresponding bias vectors. The hyperbolic tangent nonlinearity tanh is applied separately to each component of its argument vector. The dimensions of the vectors and matrices are chosen so that they match each other and the number of neurons in each layer.

The dynamic mapping $\mathbf{g}$ in (11) is modeled using another MLP network as follows:

$$\mathbf{g}(\mathbf{s}(t-1)) = \mathbf{s}(t-1) + \mathbf{D}\tanh[\mathbf{C}\mathbf{s}(t-1) + \mathbf{c}] + \mathbf{d} \tag{13}$$

The matrices $\mathbf{C}$ and $\mathbf{D}$ contain the weights of the hidden and output layers, respectively, and the vectors $\mathbf{c}$ and $\mathbf{d}$ their biases. Because the factors $\mathbf{s}(t)$ do usually not change much from their previous values $\mathbf{s}(t-1)$ during a single time step, we have used the MLP network for modeling only the change in factors in (13).

In order to apply the Bayesian approach, each unknown variable in the network is assigned a probability density function of its own. We apply the usual hierarchic definition of priors

(Bishop, 1995; Gelman et al., 1995; Neal, 1996). For many parameters, for instance the biases, it is difficult to assign a prior distribution. We can then utilize the fact that each individual bias has a similar role in the MLP network by assuming that the distribution of each element of a bias vector has the same, albeit unknown distribution which is then further modeled by a parametric distribution. These new parameters need to be assigned a prior, too, but there are far fewer of them.

Consider first the observation model (10). The noise vector $\mathbf{n}(t)$ in (10) is assumed to be i.i.d. at different time instants and gaussian with a zero mean. However, the variances of different components of the noise vector $\mathbf{n}(t)$ can be different. Given the source vector $\mathbf{s}(t)$, the variance of the data vector $\mathbf{x}(t)$ in (10) is due to the noise. Therefore $\mathbf{x}(t)$ has otherwise the same distribution as the noise but its mean is now $\mathbf{f}(\mathbf{s}(t))$. Denote generally the multivariate gaussian pdf of a random vector $\mathbf{x}$ having the mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ by $N[\mathbf{x}; \boldsymbol{\mu}; \boldsymbol{\Sigma}]$. The pdf of the vector $\mathbf{x}(t)$ in (10) is then

$$p(\mathbf{x}(t)|\mathbf{s}(t), \boldsymbol{\theta}) = N[\mathbf{x}(t); \mathbf{f}(\mathbf{s}(t)); \mathrm{diag}[\exp(2\mathbf{v}_n)]] \tag{14}$$

where diag means a diagonal matrix. The components of the vector $\exp(2\mathbf{v}_n)$ specify the diagonal covariance matrix of $\mathbf{x}(t)$ and therefore also the variances of the respective components of $\mathbf{x}(t)$. Hence the components of $\mathbf{x}(t)$ are assumed to be uncorrelated (which now implies independence due to the gaussianity assumption) in this prior model for $\mathbf{x}(t)$. The exponential parametrization of the variance corresponds to a log-normal distribution when the variance parameter has a gaussian prior. This choice is motivated later in Section 3.2.

Assuming again that the noise term $\mathbf{m}(t)$ is i.i.d. and gaussian with a zero mean and diagonal covariance matrix, we get from (11) for the distribution of the source vector $\mathbf{s}(t)$ in a similar way

$$p(\mathbf{s}(t)|\mathbf{s}(t-1), \boldsymbol{\theta}) = N[\mathbf{s}(t); \mathbf{g}(\mathbf{s}(t-1)); \mathrm{diag}[\exp(2\mathbf{v}_m)]] \tag{15}$$

where the vector $\mathbf{v}_m$ is used to define the variances of the components of $\mathbf{s}(t)$.

Denote by A_{ij} the element (i, j) of the weight matrix $\mathbf{A}$, and by a_i the ith component of the bias vector $\mathbf{a}$. The elements of the weight matrices $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{D}$ are denoted respectively by B_{ij}, C_{ij}, and D_{ij}, and b_i, c_i, and d_i denote the components of the bias vectors $\mathbf{b}$, $\mathbf{c}$, and $\mathbf{d}$. The distributions of these parameters defining the MLP network mappings (12) and (13) are again taken to be gaussian. They are specified as

$$\begin{aligned}
p(A_{ij}) &= N[A_{ij}; 0; 1] && (16)\\
p(B_{ij}) &= N[B_{ij}; 0; \exp(2v_{B_j})] && (17)\\
p(a_i) &= N[a_i; m_a; \exp(2v_a)] && (18)\\
p(b_i) &= N[b_i; m_b; \exp(2v_b)] && (19)\\
p(C_{ij}) &= N[C_{ij}; 0; \exp(2v_{C_j})] && (20)\\
p(D_{ij}) &= N[D_{ij}; 0; \exp(2v_{D_j})] && (21)\\
p(c_i) &= N[c_i; m_c; \exp(2v_c)] && (22)\\
p(d_i) &= N[d_i; m_d; \exp(2v_d)] && (23)
\end{aligned}$$

The prior distributions of the parameters m_a, m_b, m_c, m_d and v_a, v_b, v_c, v_d defining the distributions of the bias vectors are assumed to be gaussian with zero mean and standard deviation 100. Hence these distributions are assumed to be very flat[2]. The distributions (17)–(23) are also conditional, because all the other parameters in $\boldsymbol{\theta}$ are assumed to be nonrandom in defining them. This has not been shown explicitly to keep the notation simpler.

[2] Each observed component is pre-normalized to have unit variance.

For specifying the model completely, we still need to define the pdf's of the hyperparameters controlling the noise variances $\exp(2\mathbf{v}_n)$ and $\exp(2\mathbf{v}_m)$ in the prior distributions (14) and (15) of $\mathbf{x}(t)$ and $\mathbf{s}(t)$. For doing this, we denote the ith components of the vectors $\mathbf{v}_n$ and $\mathbf{v}_m$ respectively by v_{n_i} and v_{m_i}. Furthermore, we specify the distributions of the hyperparameters m_{v_B}, v_{v_B}, m_{v_C}, v_{v_C}, m_{v_D}, and v_{v_B}, which in turn define the variances of the elements of the matrices $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{D}$. These distributions are given by

$$
\begin{aligned}
p(v_{n_i}) &= N[v_{n_i}; m_{v_n}; \exp(2v_{v_n})] && (24)\\
p(v_{m_i}) &= N[v_{m_i}; m_{v_m}; \exp(2v_{v_m})] && (25)\\
p(v_{B_j}) &= N[v_{B_j}; m_{v_B} : \exp(2v_{v_B})] && (26)\\
p(v_{C_j}) &= N[v_{C_j}; m_{v_C} : \exp(2v_{v_C})] && (27)\\
p(v_{D_j}) &= N[v_{D_j}; m_{v_D} : \exp(2v_{v_D})] && (28)
\end{aligned}
$$

The prior distributions of the ten hyperparameters $m_{v_n}, \ldots, v_{v_D}$ defining the distributions (24)–(28) are similarly as before chosen gaussian with a zero mean and standard deviation 100.

3.2 Properties of the model

We have parameterized all the distributions so that the resulting parameters have a roughly gaussian posterior distribution when possible. This is because the posterior will be approximated by a gaussian distribution. For example, the variances of the gaussian distributions have been parameterized on a logarithmic scale for this reason. In ensemble learning, the conjugate gamma distributions are often used (for example in (Miskin and MacKay, 2001)) since they yield simple update equations. However, our choice of log-normal model for the variances makes it much easier to build a hierarchical model for the variances (Valpola et al., 2001). It should be noted that assuming all the parameters gaussian is not too restrictive, because the tanh nonlinearities in the MLP networks are able to transform the gaussian distributions to virtually any output distribution. On the other hand, the use of gaussian distributions makes the evaluation of the KL cost function and the learning process mathematically tractable.

The gaussian posterior approximation can be inaccurate particularly for the sources if the mappings $\mathbf{f}$ and $\mathbf{g}$ are strongly nonlinear. In this case the misfit between the approximation and the true posterior is large. The learning procedure aims at minimizing the misfit and consequently it favors smooth mappings which result in roughly gaussian posterior distribution for the sources.

Model indeterminacies are handled by restricting some of the distributions. For instance, there is the well-known scaling indeterminacy (Hyvärinen et al., 2001) between the matrix $\mathbf{A}$ and the source vector $\mathbf{s}(t)$ in the term $\mathbf{As}(t) + \mathbf{a}$ in (12). This is taken care of by setting the variance of $\mathbf{A}$ to unity instead of parameterizing and estimating it. The weight matrix $\mathbf{B}$ of the output layer does not suffer from such an indeterminacy, and so the variance of each column of this 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 $\mathbf{B}$ can be given small values.

In principle at least, one could use any sufficiently general neural network or graphical model structure for modeling the nonlinear mappings $\mathbf{f}$ and $\mathbf{g}$. We have chosen MLP networks, because they suit well for both mildly and strongly nonlinear mappings. Their another, even more important benefit here is that the number of parameters grows only quadratically with the dimension of the latent space. In the networks (12) and (13), the hidden layer is linear but the output layer is nonlinear. This is because using a nonlinear mapping in the hidden layer would only complicate the learning process while not bringing any notable improvement in the generality of the modeled dynamic system. It should also be noted that the MLP network (13) modeling the dynamics can learn to overrule the term $\mathbf{s}(t-1)$ if it turns out that it does not bear information about the current value $\mathbf{s}(t)$ of the source vector.

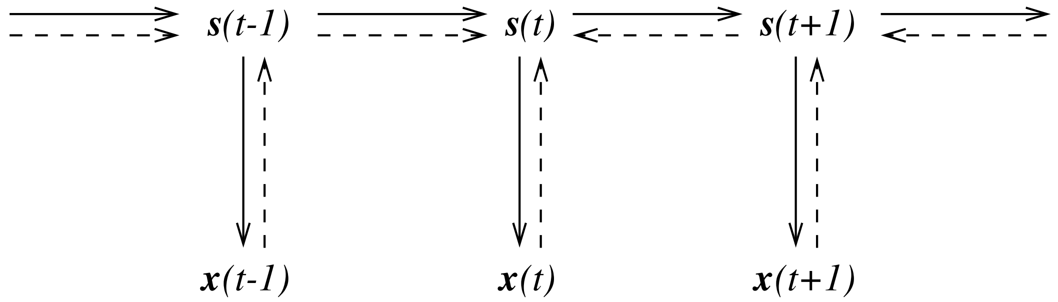

Figure 1: The causal relations assumed by the model are depicted by solid arrows. Observations $\mathbf{x}(i)$ give information about the values of the factors. The flow of information to the factor $\mathbf{s}(t)$ is represented by dashed arrows. Bayes' rule is used for reversing the solid arrows when necessary.

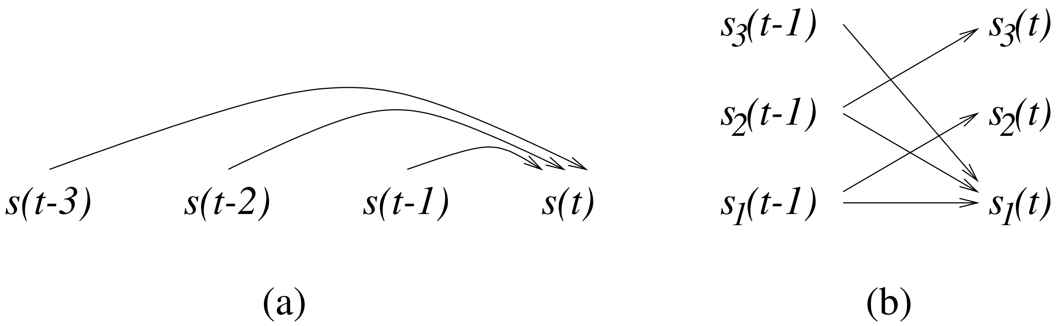

Figure 2: (a) The factor $s(t)$ depends on the three previous values $s(t-3)$, $s(t-2)$ and $s(t-1)$. (b) This dynamics can be transformed into an equivalent state representation where $s_1(t)$ corresponds to $s(t)$ while $s_2(t)$ and $s_3(t)$ store the values $s(t-1)$ and $s(t-2)$, respectively.

The model defined by (10) and (11) aims at finding factors that facilitate prediction. It is often easier to predict future factors from past factors than directly future observations from past observation. Learning of the factors $\mathbf{s}(t)$ at the time instant t takes into account three sources of information: 1) the factors should be able to represent the observations $\mathbf{x}(t)$, 2) the factors should be able to predict the factors $\mathbf{s}(t+1)$ at the next time step and 3) the factors should be well predicted by the factors $\mathbf{s}(t-1)$ at the previous time step. This is depicted schematically in Figure 1.

The model (11) assumes that the factors $\mathbf{s}(t)$ can be predicted from the immediately preceding factors $\mathbf{s}(t-1)$ without knowing the previous factors. This is not a restriction because any dynamic model depending on older factors $\mathbf{s}(t-2), \mathbf{s}(t-3), \ldots$ can be converted into an equivalent model of the type (11) in the way shown in Figure 2.

It is well known (Haykin and Principe, 1998) that the nonlinear model (10)–(11) is not uniquely identifiable. This is because any smooth transformation of the latent space can be absorbed in the mappings $\mathbf{f}$ and $\mathbf{g}$. If several different parameter values result in equivalent external behaviour of the model, no amount of data can distinguish between the alternatives. On the other hand, if external behaviour only is of interest, the models agree and the choice between the models becomes irrelevant.

Ensemble learning can utilize the indeterminacies in the prior model by rendering the posterior model closer to the allowed functional forms of the posterior approximation. A manifestation of this behavior is seen later in the experiments.

3.3 Related work

Before proceeding, we give relevant references on nonlinear dynamic modeling. A review article on nonlinear predictive dynamic models containing comparisons of various techniques for chaotic time series is (Casdagli, 1989). A tutorial paper on using neural networks, in particular the well-known radial-basis function (RBF) networks (Bishop, 1995; Haykin, 1998) to dynamically model nonlinear chaotic time series is (Haykin and Principe, 1998). Dynamic modeling using various neural network structures such as recurrent or time-delayed neural networks is discussed on an introductory level in (Principe et al., 2000), and on somewhat more advanced level in (Haykin, 1998).

Quite recently, Bayesian techniques have been introduced for the problem of learning the mappings $\mathbf{f}$ and $\mathbf{g}$ in Eqs. (10) and (11) in (Briegel and Tresp, 1999; Ghahramani and Roweis, 1999; Roweis and Ghahramani, 2001). In (Ghahramani and Roweis, 1999; Roweis and Ghahramani, 2001), the nonlinear mappings are modeled by RBF networks, and only the linear output layers of the RBF networks are adapted. This yields a very efficient learning algorithm, but in practice restricts the dimension of the dynamic system which can be learned. This is because the required number of radial basis functions increases exponentially with the dimension of the internal state of the process. Briegel and Tresp (Briegel and Tresp, 1999) model the nonlinearities using MLP networks with sampling.

The most important difference between these earlier methods and the one described in this paper is that the parameters of the mappings $\mathbf{f}$ and $\mathbf{g}$ have been so far summarized using point estimates while here full distributions are used.

Bayesian techniques for hidden Markov models are reviewed in (Ghahramani, 2001). These models comprise another popular technique of dynamic modeling, which in practice often provides fairly similar results as state-space models. See for instance (Ghahramani and Hinton, 2000) where variational methods resembling ensemble learning are applied to a state-space model where a hidden Markov model switches between different linear dynamical models.

Cichocki et al. (Cichocki et al., 1999) have considered a nonlinear dynamic extension of the standard linear model used in ICA and BSS by applying state-space models and hyper RBF networks. However, their learning algorithms are not Bayesian and they are only partly blind or unsupervised, requiring some amount of known training data in the beginning of learning.

An interesting line of research stems from the traditional Kalman filtering where the distribution of the unknown state is inferred given the observations and the known mappings $\mathbf{f}$ and $\mathbf{g}$. It has also been modified for joint estimation of the model parameters and the state. A review of these methods can be found in (Wan and Nelson, 2001) and Kalman filtering is also breafly discussed in Section 5.1. An interesting modification called the unscented Kalman filter is presented in (Wan and Merwe, 2001).

In (Wan and Nelson, 2001; Wan and Merwe, 2001), the unknown state and dynamics are estimated using extended Kalman filtering while the observation mapping $\mathbf{f}$ is assumed to be the identity mapping. This means that the observations are assumed to be noisy versions of the underlying states. It is unclear whether the approach is suitable for our problem where the unknown observation mapping is a general nonlinear function and the dimensionality of the state-space is high.

More research is needed to investigate the relative merits of the approaches, but advantages of ensemble learning include its ability to handle state-spaces with large dimensionality, to learn the structure of the model and to find a meaningful solution among a family of infinite possible state representations.

4 The cost function

Let us denote the two parts of the cost function (7) arising from the numerator and the denominator of the logarithm respectively by

$$\mathcal{C}_q = \mathrm{E}_q\{\ln q\} = \int_S \int_{\boldsymbol{\theta}} q(S, \boldsymbol{\theta}) \ln q(S, \boldsymbol{\theta}) d\boldsymbol{\theta} dS \tag{29}$$

$$\mathcal{C}_p = \mathrm{E}_q\{-\ln p\} = -\int_S \int_{\boldsymbol{\theta}} q(S, \boldsymbol{\theta}) \ln p(X, S, \boldsymbol{\theta}) d\boldsymbol{\theta} dS \tag{30}$$

Hence the total cost (7) is

$$\mathcal{C}_{\mathrm{KL}} = \mathcal{C}_q + \mathcal{C}_p \tag{31}$$

In (30) and later on, we do not show the explicit dependence of the pdf $p(X, S, \boldsymbol{\theta})$ on the model $\mathcal{H}$ any more for making the notation simpler. However, it should be remembered that such a dependence still exists.

For approximating and then minimizing the total cost $\mathcal{C}_{\mathrm{KL}}$, we need two things: the exact formulation of the joint density $p(X, S, \boldsymbol{\theta})$ of the data X, sources S, and other parameters $\boldsymbol{\theta}$, and the approximation of the posterior pdf $q(S, \boldsymbol{\theta})$. Assume that there are available a total of T data vectors $\mathbf{x}(t)$. The complete data set is then $X = \{\mathbf{x}(1), \ldots, \mathbf{x}(T)\}$, and the respective set of source vectors is $S = \{\mathbf{s}(1), \ldots, \mathbf{s}(T)\}$.

The joint density can be expressed as the product

$$p(X, S, \boldsymbol{\theta}) = p(X|S, \boldsymbol{\theta}) p(S|\boldsymbol{\theta}) p(\boldsymbol{\theta}) \tag{32}$$

The probability density $p(X|S, \boldsymbol{\theta})$ of the data X given the sources S and the parameters $\boldsymbol{\theta}$ can be evaluated from the equations (14) and (12). Clearly, the ith component of the mapping vector $\mathbf{f}(\mathbf{s}(t))$ in (12) is

$$f_i(\mathbf{s}(t)) = \mathbf{b}_i^T \tanh[\mathbf{A}\mathbf{s}(t) + \mathbf{a}] + b_i \tag{33}$$

where $\mathbf{b}_i^T$ denotes the ith row vector of the weight matrix $\mathbf{B}$, and b_i is the ith component of the bias vector $\mathbf{b}$. From (14), the distribution of the ith component $x_i(t)$ of the data vector $\mathbf{x}(t)$ is thus gaussian

$$p(x_i(t)|\mathbf{s}(t), \boldsymbol{\theta}) = N[x_i(t); f_i(\mathbf{s}(t)); \exp(2v_{n_i})] \tag{34}$$

Assuming that the data vectors $\mathbf{x}(t)$ are independent at different time instants and taking into account the independence of the components of the noise vector we get

$$p(X|S, \boldsymbol{\theta}) = \prod_{t=1}^{T} \prod_{i=1}^{n} p(x_i(t)|\mathbf{s}(t), \boldsymbol{\theta}) \tag{35}$$

where n is the common dimension of the vectors $\mathbf{x}(t)$, $\mathbf{f}(\mathbf{s}(t))$, and $\mathbf{n}(t)$ in (10).

The distribution $p(S|\boldsymbol{\theta})$ can be derived similarly as $p(X|S, \boldsymbol{\theta})$ above, but taking now into account that the subsequent values $\mathbf{s}(t)$ and $\mathbf{s}(t-1)$ of the source vector depend on each other due to the model (11), (13). Thus we get

$$p(S|\boldsymbol{\theta}) = p(\mathbf{s}(1)|\boldsymbol{\theta}) \prod_{t=1}^{T} p(\mathbf{s}(t)|\mathbf{s}(t-1), \boldsymbol{\theta}) \tag{36}$$

where $p(\mathbf{s}(t)|\mathbf{s}(t-1), \boldsymbol{\theta})$ can be evaluated quite similarly as above from Eqs. (15) and (13), yielding

$$p(\mathbf{s}(t)|\mathbf{s}(t-1), \boldsymbol{\theta}) = \prod_{i=1}^{m} p(s_i(t)|\mathbf{s}(t-1), \boldsymbol{\theta}) = \prod_{i=1}^{m} N[s_i(t); g_i(\mathbf{s}(t-1)); \exp(2v_{m_i})] \tag{37}$$

Here m is the number of components of the vectors $\mathbf{s}(t)$, $\mathbf{g}(\mathbf{s}(t-1))$, and $\mathbf{m}(t)$ in (11), and $g_i(\mathbf{s}(t-1))$ is the ith component of the vector $\mathbf{g}(\mathbf{s}(t-1))$ obtained directly from (13).

The prior distribution of the parameters $p(\boldsymbol{\theta})$ can be evaluated as the product of the gaussian distributions of the parameters specified in Eqs. (16)–(23), but here one must take into account their dependences on the distributions of the hyperparameters given in Eqs. (24)–(28). Finally, inserting the expressions obtained in this way for $p(X|S,\boldsymbol{\theta})$, $p(S|\boldsymbol{\theta})$, and $p(\boldsymbol{\theta})$ into (32) provides the desired joint density $p(X,S,\boldsymbol{\theta})$. We shall not write it explicitly here due to its complicated form.

Another distribution that is needed in evaluating the cost functions $\mathcal{C}_q$ and $\mathcal{C}_p$ is the approximation $q(S,\boldsymbol{\theta})$ of the posterior density of $\boldsymbol{\theta}$ and $S = \{\mathbf{s}(1),\ldots,\mathbf{s}(T)\}$. It must be simple enough for mathematical tractability and computational efficiency. First, we assume that the source vectors are independent of the other parameters $\boldsymbol{\theta}$, so that $q(S,\boldsymbol{\theta})$ decouples into

$$q(S,\boldsymbol{\theta}) = q(S)q(\boldsymbol{\theta}) \tag{38}$$

For the parameters $\boldsymbol{\theta}$, a gaussian density $q(\boldsymbol{\theta})$ with a diagonal covariance matrix is used. This implies that the approximation $q(\boldsymbol{\theta})$ is a product of independent gaussian distributions $q_j(\theta_j)$ of each individual parameter θ_j:

$$q(\boldsymbol{\theta}) = \prod_j q_j(\theta_j) = \prod_j N[\theta_j;\overline{\theta}_i;\widetilde{\theta}_i] \tag{39}$$

The mean $\overline{\theta}_i$ and variance $\widetilde{\theta}_i$ define each component pdf $q_j(\theta_j) = N[\theta_j;\overline{\theta}_i;\widetilde{\theta}_i]$.

Quite similarly as in modeling the pdf $p(S|\boldsymbol{\theta})$, the components of each source vector $\mathbf{s}(t)$ are assumed to be statistically independent. The subsequent source vectors are also otherwise independent, but their corresponding components $s_j(t)$ and $s_j(t-1)$, $j = 1,\ldots,m$, are pairwise correlated. This is a realistic minimum assumption for modeling the dependence of the vectors $\mathbf{s}(t)$ and $\mathbf{s}(t-1)$ which follows from the chosen dynamic model (11), and it does not increase the computational load significantly.

These considerations lead to the approximation

$$q(S) = \prod_{t=1}^{T}\prod_{i=1}^{m} q(s_i(t)|s_i(t-1)) \tag{40}$$

where the conditional pdf $q(s_i(t)|s_i(t-1))$ describing the dependence is again modeled as a gaussian distribution. Conditioned on $s_i(t-1)$, its variance is $\mathring{s}_i(t)$, and its mean

$$\mu_{si}(t) = \overline{s}_i(t) + \breve{s}_i(t,t-1)[s_i(t-1) - \overline{s}_i(t-1)] \tag{41}$$

depends linearly on the previous source value $s_i(t-1)$. The form of (41) is chosen such that $\overline{s}_i(t)$ is the mean of $q(s_i(t))$ when $s_i(t-1)$ is marginalized out. Hence

$$q(s_i(t)|s_i(t-1)) = N[s_i(t);\mu_{si}(t);\mathring{s}_i(t)] \tag{42}$$

The approximate density $q(s_i(t)|s_i(t-1))$ is thus parameterized by the mean $\overline{s}_i(t)$, linear dependence $\breve{s}_i(t,t-1)$ and variance $\mathring{s}_i(t)$ as defined in Eqs. (42) and (41). This concludes the specification of the approximation of the pdf $q(S)$ of the sources S in (40).

Inserting Eqs. (40) and (39) into (38) finally provides the desired approximating posterior density $q(S,\boldsymbol{\theta})$ of the sources S and the parameters $\boldsymbol{\theta}$, completing the specification of the cost functions (29) and (30).

5 Learning method

The overall learning procedure resembles somewhat the batch version of the standard back-propagation algorithm (see (Haykin, 1998)) in that it consists of forward and backward phases for each iteration, In the forward phase, the value of the cost function (31) is computed using the current values of the source signals S and other parameters $\boldsymbol{\theta}$. Essentially, in this phase the posterior means and variances of all the sources are propagated through the MLP networks (12) and (13), and the value of the cost function is then calculated.

In the backward phase, parameters defining the distributions of the sources $s_i(t)$ and the parameters $\boldsymbol{\theta}$ of the MLP networks are updated using EM-type algorithms. This in turn takes place in certain order, which will be described later on. The special properties of the problem are utilized for developing efficient updating algorithms. The updating procedure is simpler for the hyperparameters. The forward and backward phases are then repeated over sufficiently many sweeps until the method has converged with the desired accuracy. The learning is batch type, so that all the data vectors are used for updating all the parameters and sources during each sweep.

In the following subsections, we describe in more detail the forward and backward phases and the learning algorithms used in them, as well as initialization of the learning process.

5.1 Relation to Kalman filtering and smoothing

The standard technique for estimating the state of linear gaussian state-space models is Kalman filtering or smoothing. In smoothing, the posterior probability of the states is computed in a two-phase forward and backward pass which collects the information from past and future observations (filtering consists only of the forward pass). The posterior probability of the state can be solved in this way because the effect of the value of the observation $\mathbf{x}(t)$ on the posterior mean of the states is linear and does not depend on the values of the other observations. The posterior variance only depends on the mappings and noise levels, not on the observations.

Neither of these properties hold for nonlinear models because the values of the states affect the way in which the information passes through the dynamics. It is possible to extend linear Kalman filtering to nonlinear dynamics for instance by linearizing the dynamics for the current estimate of the sources. Several iterations are required because the linearized model and the estimated operating point depend on each other. It is also possible that the iterations become unstable.

For linear gaussian models it is possible to derive algorithms similar to Kalman smoothing using ensemble learning as was done in (Ghahramani and Beal, 2001). Our algorithm is designed for learning nonlinear models, and only propagates information one step forward and backward in time in the forward and backward phase. This makes learning stable but does not significantly slow down convergence because the slowest part is in any case the adaptation of the MLP networks. Each iteration is also more efficient than the Kalman smoothing step since no inversions of matrices are required. This is due to the factorial form of the posterior approximation $q(S)$ of the sources.

5.2 Forward phase: computation of the cost function

Consider now in more detail how the current value of the KL cost function $\mathcal{C}_{\mathrm{KL}}$, defined in Eqs. (29)–(31), is calculated for our model in practice.

Both the joint density $p(S, \boldsymbol{\theta}, X)$ and the approximation of the posterior density $q(S, \boldsymbol{\theta})$ are products of many simple terms. This property facilitates the calculation of the cost functions (29) and (30) greatly, because they split into expectations of many simple terms. This can be seen as follows. Assume that $h(\boldsymbol{\Gamma})$ and $f(\boldsymbol{\Gamma})$ are some joint pdf's depending on a set of parameters $\boldsymbol{\Gamma}$ $= \{\gamma_1, \ldots, \gamma_l\}$. Assuming further that these parameters are statistically independent, the joint densities $h(\mathbf{\Gamma})$ and $f(\mathbf{\Gamma})$ decouple into products of densities $h_i(\gamma_i)$ and $f_j(\gamma_j)$ of independent parameters:

$$h(\mathbf{\Gamma}) = \prod_{i=1}^{l} h_i(\gamma_i) \,, \qquad f(\mathbf{\Gamma}) = \prod_{j=1}^{l} f_j(\gamma_j) \tag{43}$$

and the integral

$$\begin{aligned}\int_{\mathbf{\Gamma}} h(\mathbf{\Gamma}) \ln f(\mathbf{\Gamma}) d\mathbf{\Gamma} &= \int_{\gamma_1} \cdots \int_{\gamma_l} \prod_{i=1}^{l} h_i(\gamma_i) \sum_{j=1}^{l} \ln f_j(\gamma_j) d\gamma_1 \cdots d\gamma_l \\ &= \sum_{j=1}^{l} \int_{\gamma_j} h_j(\gamma_j) \ln f_j(\gamma_j) d\gamma_j \prod_{i \neq j}^{l} \int_{\gamma_i} h_i(\gamma_i) d\gamma_i \\ &= \sum_{j=1}^{l} \int_{\gamma_j} h_j(\gamma_j) \ln f_j(\gamma_j) d\gamma_j \end{aligned} \tag{44}$$

Thus one must compute integrals of densities of each independent scalar parameter γ_i only (or more generally integrals which are functions of mutually dependent parameters only). Typically terms in the prior probability of the model are conditional but the same principle can be utilized efficiently in evaluating the costs (29) and (30): if the term is of the form $f_j(\gamma_j|\gamma_k, \gamma_l)$, for instance, then the integral needs to be computed over $h_j(\gamma_j)$, $h_k(\gamma_k)$ and $h_l(\gamma_l)$ only.

The computations in the forward phase start with the parameters of the posterior approximation $q(S, \boldsymbol{\theta})$ of the unknown variables of the model. The posterior mean of each parameter θ_j belonging to the parameter set $\boldsymbol{\theta}$ is denoted by $\overline{\theta}_j$, and $\widetilde{\theta}_j$ denotes the respective posterior variance. For the components $s_i(t)$ of the source vectors $\mathbf{s}(t)$ in the set S, the parameters of their posterior approximation are the posterior mean $\overline{s}_i(t)$, the posterior variance $\mathring{s}_i(t)$, and the dependence $\breve{s}_i(t, t-1)$.

The first stage of the forward computations is the iteration

$$\widetilde{s}_i(t) = \mathring{s}_i(t) + \breve{s}_i^2(t, t-1)\widetilde{s}_i(t-1) \tag{45}$$

to obtain the marginalized posterior variances $\widetilde{s}_i(t)$ of the sources. It is easy to see by induction that if the past values $s_i(t-1)$ of the sources are marginalized out, the posterior mean of $s_i(t)$ is indeed $\overline{s}_i(t)$ and its marginalized variance is given by (45). The derivation is presented in Appendix A. The marginalized variances $\widetilde{s}_i(t)$ are computed recursively in a sweep forward in time t.

Thereafter the posterior means and variances are propagated through the MLP networks. The final stage is the computation of the KL cost function $\mathcal{C}_{\mathrm{KL}}$ in (31).

The first part (29) of the KL cost function (31) depends only on the approximation $q(S, \boldsymbol{\theta})$ of the posterior density. It is straightforward to calculate, because in the previous section it was shown that $q(S, \boldsymbol{\theta})$ consists of products of independent gaussian pdf's of the form $q_j(\theta_j)$ or $q(s_i(t)|s_i(t-1))$. Applying (44), we see that the cost (29) becomes the sum of the negative entropies of gaussian random variables θ_j and $s_i(t)$ given $s_i(t-1)$. Recalling that the entropy $H(y)$ of a gaussian random variable y is

$$H(y) = -\int_{-\infty}^{+\infty} p(y) \ln p(y) dy = \frac{1}{2}[1 + \ln(2\pi\sigma_y^2)] = \frac{1}{2}\ln(2\pi e \sigma_y^2) \tag{46}$$

where σ_y^2 is the variance of y, we get

$$\mathcal{C}_q = -\frac{1}{2}\sum_j \ln(2\pi e \widetilde{\theta}_j) - \frac{1}{2}\sum_i \sum_t \ln(2\pi e \mathring{s}_i(t)) \tag{47}$$

where $\widetilde{\theta}_j$ is the posterior variance of the parameter θ_j, and $\mathring{s}_i(t)$ the posterior variance of $s_i(t)$ given $s_i(t-1)$.

Evaluation of the data part (30) of the KL cost function $\mathcal{C}_{\mathrm{KL}}$ is more involved. For notational simplicity, we denote by $\mathrm{E}_q\{y\}$ the expectation of the random variable y over the distribution $q(S, \boldsymbol{\theta})$. This notation has already been used in Eqs. (29) and (30).

Consider now evaluation of the expectations of the form $\mathrm{E}_q\{-\ln p(\gamma|\cdots)\}$, where $p(\gamma|\cdots)$ is the pdf of a random variable γ conditioned on some quantities. Assume further that the prior distribution of γ is gaussian, having the general form $N[\gamma; m_\gamma; \exp(2v_\gamma)]$. Using the fact that γ, m_γ, and v_γ are independent, one can show (see Appendix B) that

$$\mathrm{E}_q\{-\ln p(\gamma|m_\gamma, v_\gamma)\} = \frac{1}{2}[(\overline{\gamma} - \overline{m}_\gamma)^2 + \widetilde{\gamma} + \widetilde{m}_\gamma]\exp(2\widetilde{v}_\gamma - 2\overline{v}_\gamma) + \overline{v}_\gamma + \frac{1}{2}\ln(2\pi) \tag{48}$$

Here $\overline{\gamma}$, $\overline{m}_\gamma$, and $\overline{v}_\gamma$ denote the posterior means of γ, m_γ, and v_γ, respectively, and $\widetilde{\gamma}$, $\widetilde{m}_\gamma$, and $\widetilde{v}_\gamma$ their posterior variances.

The formula (48) can be directly applied to the distributions (16)–(28) of all the parameters and hyperparameters $\boldsymbol{\theta}$. Moreover, it can be applied to the posterior distributions (34) of the components $x_i(t)$ of the data vectors $\mathbf{x}(t)$ as well, yielding now

$$\begin{aligned}\mathrm{E}_q\{-\ln p(x_i(t)|\mathbf{s}(t), \boldsymbol{\theta})\} &= \frac{1}{2}\left\{\left[x_i(t) - \overline{f}_i(\mathbf{s}(t))\right]^2 + \widetilde{f}_i(\mathbf{s}(t))\right\}\exp\left(2\widetilde{v}_{n_i} - 2\overline{v}_{n_i}\right) \\ &+ \overline{v}_{n_i} + \frac{1}{2}\ln(2\pi)\end{aligned} \tag{49}$$

In this case, $\gamma = x_i(t)$ is a nonrandom quantity, and so its mean is simply $\overline{\gamma} = x_i(t)$ and variance $\widetilde{\gamma} = 0$. For computing the expectations (49), one must evaluate the posterior means $\overline{f}_i$ and variances $\widetilde{f}_i$ of the components f_i of the nonlinear mapping $\mathbf{f}$. These, in turn, depend on the posterior means and variances of the weights and sources. The posterior variances of the sources are obtained from (45).

Approximate formulas for $\overline{f}_i$ and $\widetilde{f}_i$ can be derived by expanding $\mathbf{f}$ into its Taylor series (Lappalainen and Honkela, 2000). These formulas are given and discussed in Appendix C.

The calculation of $\mathrm{E}_q\{-\ln p(\mathbf{s}(t)|\mathbf{s}(t-1), \boldsymbol{\theta})\}$ is otherwise similar to (49) but the interaction of the conditional distributions (37) and (40) of the sources $\mathbf{s}(t)$ give rise to extra terms. More specifically, the posterior dependence of both the source signal $s_i(t)$ and the nonlinear mapping $g_i(\mathbf{s}(t-1))$ on the previous value $s_i(t-1)$ of that source results in a cross-term which can be approximated by making use of the Taylor series expansion of the nonlinearity $\mathbf{g}$; see Appendix D. The terms due to the posterior variances of the outputs of the nonlinear mappings $\mathbf{f}$ and $\mathbf{g}$ have the highest computational load in evaluating the cost function, because they require evaluation of the Jacobian matrix for these nonlinearities.

Collecting and summing up the terms contributing to the cost function (30), the cost $\mathcal{C}_p$ due to the data can be computed in a similar manner as is done in (47) for the cost $\mathcal{C}_q$. Probably the easiest way to think the data part $\mathcal{C}_p$ of the KL cost function $\mathcal{C}_{\mathrm{KL}}$ is the following. Each single parameter θ_j contributes to the cost $\mathcal{C}_p$ one term, which depends on the posterior mean $\overline{\theta}_j$ and variance $\widetilde{\theta}_j$ of that parameter θ_j. Quite similarly, each source signal $s_i(t)$ gives rise to a term depending on the posterior mean $\overline{s}_i(t)$ and variance $\widetilde{s}_i(t)$ of that particular source. Furthermore, each component $x_i(t)$ of the data vectors also contributes one term to the cost $\mathcal{C}_p$.

The cost $\mathcal{C}_q$ due to the approximation, given in (47), has also one term for each parameter θ_j and source signal $s_i(t)$. Because these terms are entropies of gaussian pdf's, they depend solely on the posterior variances $\widetilde{\theta}_j$ and $\mathring{s}_i(t)$. The full form of the cost function $\mathcal{C}_{\mathrm{KL}}$ is given in Appendix E.

5.3 Backward phase: updating the parameters and the sources

In our ensemble learning method, a posterior pdf of its own is assigned to each variable appearing in the computations. This posterior pdf is described in terms of a small number of parameters characterizing it, typically by its mean and variance. Such a representation is called posterior summary. If the posterior pdf in question is gaussian, the summary defines it completely, otherwise it gives at least a rough idea of the pdf.

In the backward phase, the posterior summary of each variable is updated by assuming that the summaries of the other variables remain constant. This learning stage takes place in the following order:

1. The sources and weights are updated.
2. The parameters of the noise and weight distributions are updated.
3. The hyperparameters are updated.

The posterior summaries are updated by solving the zero of the gradients of the cost function $\mathcal{C}_{\mathrm{KL}}$. The gradients are evaluated for the mean and variance of the posterior summary of each needed quantity. This takes place in terms of a back-propagation type algorithm by reversing the steps in the forward computations and propagating the gradients backwards. The update formulas are given and discussed in the next subsection and in Appendix F.

Computation of the new posterior summaries is approximate for the values before the nonlinearities, that is, for the sources and for the weight matrices $\mathbf{A}$, $\mathbf{C}$ and bias vectors $\mathbf{a}$, $\mathbf{b}$ of the first layer. In practice, it is necessary to update the weights and sources simultaneously for keeping the computation time tolerable. However, the learning process must then be modified to take into account the couplings between these quantities. Subsection 5.3.3 presents how this can be done.

Updating the hyperparameters is simpler, because the zero of the gradient can be solved exactly. These updates are discussed in the subection 5.3.2.Finally, we discuss initialization of the learning process briefly in the last subsection.

5.3.1 Updating the sources and weights

All the weights of the MLP networks (12) and (13) are assumed to have gaussian pdf's detailed in Eqs. (16)–(23). Denoting generally any of these weights by w_i, the mean $\overline{w}_i$ and variance $\widetilde{w}_i$ of w_i specify its gaussian posterior pdf $N[w_i; \overline{w}_i; \widetilde{w}_i]$ completely.

The variances $\widetilde{w}_i$ are obtained by differentiating the KL cost function $\mathcal{C}_{\mathrm{KL}}$ with respect to $\widetilde{w}_i$ (Lappalainen and Honkela, 2000; Valpola, 2000b):

$$\frac{\partial \mathcal{C}_{\mathrm{KL}}}{\partial \widetilde{w}_i} = \frac{\partial \mathcal{C}_p}{\partial \widetilde{w}_i} + \frac{\partial \mathcal{C}_q}{\partial \widetilde{w}_i} = \frac{\partial \mathcal{C}_p}{\partial \widetilde{w}_i} - \frac{1}{2\widetilde{w}_i} \tag{50}$$

Equating this to zero yields a fixed-point iteration for updating the variances:

$$\widetilde{w}_i = \left[2\frac{\partial \mathcal{C}_p}{\partial \widetilde{w}_i} \right]^{-1} \tag{51}$$

For a linear gaussian model the cost $\mathcal{C}_p$ would be quadratic with respect to the posterior means $\overline{w}_i$ and a Newton iteration would thus converge in one step. It would also hold that $\frac{\partial^2 \mathcal{C}_p}{\partial \overline{w}_i^2} = 2\frac{\partial \mathcal{C}_p}{\partial \widetilde{w}_i}$. In the nonlinear model we can assume that this holds approximately (Lappalainen and Honkela, 2000; Valpola, 2000b) and use the following update rule

$$\overline{w}_i \leftarrow \overline{w}_i - \frac{\partial \mathcal{C}_p}{\partial \overline{w}_i} \left[\frac{\partial^2 \mathcal{C}_{\mathrm{KL}}}{\partial \overline{w}_i^2} \right]^{-1} \approx \overline{w}_i - \frac{\partial \mathcal{C}_p}{\partial \overline{w}_i} \widetilde{w}_i \, . \tag{52}$$

The formulas (51) and (52) have a central role in learning. For linear gaussian models the formulas give the optimum in one step assuming that there is only one parameter which is changing.

In the static nonlinear factor analysis (NFA) method which does not include the dynamic model (11) for the sources $\mathbf{s}(t)$, the means $\overline{s}_i(t)$ and variances $\widetilde{s}_i(t)$ of the gaussian posterior distributions of the sources $s_i(t)$ are updated quite similarly as in Eqs. (51) and (52). However, the dynamic model (11) makes these updates somewhat more involved.

In the computation of the KL cost function $\mathcal{C}_{\mathrm{KL}}$, the marginalized posterior variances $\widetilde{s}_i(t)$ of the sources are used as intermediate variables. Hence the gradients of the parameters of the posterior pdf's of the sources must be computed also through these variables. These gradients can be evaluated by applying the chain rule to the forward phase recursion formula (45) for the variances $\widetilde{s}_i(t)$. The derivations and formulas are presented in Appendix F. We give here the resulting adaptation formula for the linear dependence parameter $\breve{s}_i(t, t-1)$ and the conditional posterior variance $\mathring{s}_i(t)$:

$$\breve{s}_i(t, t-1) = \frac{\partial g_i(t)}{\partial s_i(t-1)} \exp(2\widetilde{v}_i - 2\overline{v}_i) \left[2 \frac{\partial \mathcal{C}_{\mathrm{KL}}(\widetilde{s}_i(t))}{\partial \widetilde{s}_i(t)} \right]^{-1} \quad (53)$$

$$\mathring{s}_i(t) = \left[2 \frac{\partial \mathcal{C}_{\mathrm{KL}}(\widetilde{s}_i(t))}{\partial \widetilde{s}_i(t)} \right]^{-1} . \quad (54)$$

5.3.2 Updating the other parameters

Updating the parameters of the noise and weight distributions and the hyperparameters is simpler, because the forward and backward phases are not needed for them. For these parameters, the means of their gaussian posterior distributions are updated directly. This is done after every sweep, after all the sources and the weights have been updated. The respective posterior variances are adapted using Newton's iteration.

Consider first updating of the means, recalling that all the parameters considered in this subsection have gaussian pdf's. A well-known result from classical estimation theory states that the mean of a gaussian pdf has a gaussian posterior pdf when the other parameters are assumed to be fixed and the prior distributions are gaussian (Gelman et al., 1995; Sorenson, 1980). Updating is then simple.

In particular, let γ_i, $i = 1, 2, \ldots$, be parameters which are gaussian distributed with a common mean m_γ and variance $\exp(2v_\gamma)$:

$$p(\gamma_i | m_\gamma, v_\gamma) = N[\gamma_i; m_\gamma; \exp(2v_\gamma)] . \quad (55)$$

Then the contribution arising from the parameters γ_i, $i = 1, 2, \ldots$, to the cost function $\mathcal{C}_p$ is

$$\begin{aligned}\sum_i \mathcal{C}_p(\gamma_i) &= \sum_i \mathrm{E}_q\{-\ln p(\gamma_i | m_\gamma, v_\gamma)\} \\ &= \sum_i \frac{1}{2}\ln(2\pi) + \overline{v}_\gamma + \frac{1}{2}[(\overline{\gamma}_i - \overline{m}_\gamma)^2 + \widetilde{\gamma}_i + \widetilde{m}_\gamma] \exp(2\widetilde{v}_\gamma - 2\overline{v}_\gamma) , \quad (56)\end{aligned}$$

where we have used the result (48). Similar terms but without the summation are obtained for the hyperparameters m_γ and v_γ:

$$\begin{aligned}\mathcal{C}_p(m_\gamma) &= \frac{1}{2}\ln(2\pi) + \overline{v}_{m_\gamma} + \\ &\quad \frac{1}{2}[(\overline{m}_\gamma - \overline{m}_{m_\gamma})^2 + \widetilde{m}_\gamma + \widetilde{m}_{m_\gamma}] \exp(2\widetilde{v}_{m_\gamma} - 2\overline{v}_{m_\gamma}) \quad (57) \\ \mathcal{C}_p(v_\gamma) &= \frac{1}{2}\ln(2\pi) + \overline{v}_{v_\gamma} + \\ &\quad \frac{1}{2}[(\overline{v}_\gamma - \overline{m}_{v_\gamma})^2 + \widetilde{v}_\gamma + \widetilde{m}_{v_\gamma}] \exp(2\widetilde{v}_{v_\gamma} - 2\overline{v}_{v_\gamma}) \quad (58)\end{aligned}$$

The hyperparameters m_γ and v_γ are updated by differentiating the cost function $\mathcal{C}_{\mathrm{KL}}$ with respect to $\overline{m}_\gamma$, $\widetilde{m}_\gamma$, $\overline{v}_\gamma$ and $\widetilde{v}_\gamma$ and solving for zero. For $\overline{m}_\gamma$ and $\widetilde{m}_\gamma$ the equivalents of (51) and (52) give the optimum in one step but for $\overline{v}_\gamma$ and $\widetilde{v}_\gamma$ the resulting equations need to be solved numerically by iteration. For $\overline{v}_\gamma$, for instance, the problem is essentially to solve v from $a+bv+c\exp(-2v)=0$ where a, b and c are constants. This equation is solved by Newton iteration.

5.3.3 Correction of the step size

The formulas (51) and (52) can be used as such for updating the posterior pdf's of the sources and weights if each of these quantities is adapted sequentially independently of the other weights and sources. However, in practice it is necessary to update the sources and weights simultaneously. This reduces the computational load of the proposed method dramatically, because it is almost as easy to compute new posterior summaries for all these variables simultaneously as for a single variable. Even then, the computation time required by our method tends to be high, because the evaluation of the cost function in the forward step requires a lot of computation. The drawback of parallel updating of the weights and sources is that the assumption of independent updates is violated for them. The error caused by parallel updates is compensated as follows.

For the weights w_i, a learning parameter α_i is introduced for each posterior mean $\overline{w}_i$ and quite similarly β_i for each posterior variance $\widetilde{w}_i$ to stabilize the oscillations caused by parallel updating. This is the standard method used in context with fixed-point algorithms. The oscillations can be detected by monitoring the directions of updates of individual parameters. The learning parameter α_i is updated using the rule

$$\alpha_i \leftarrow \begin{cases} 0.8\alpha_i & \text{if the sign of the update changed} \\ \min(1, 1.05\alpha_i) & \text{if the sign of the update remains the same} \end{cases} \tag{59}$$

The parameter β_i is computed using the same rule.

This provides the following refined fixed-point update rules for the posterior means and variances of the weights:

$$\overline{w}_i \leftarrow \overline{w}_i - \alpha_i \frac{\partial \mathcal{C}_p}{\partial \overline{w}_i} \widetilde{w}_i \tag{60}$$

$$\widetilde{w}_i \leftarrow \left[2 \frac{\partial \mathcal{C}_p}{\partial \widetilde{w}_i} \right]^{-\beta_i} \widetilde{w}_i^{1-\beta_i} \tag{61}$$

In (61), a weighted geometric rather than arithmetic mean is applied to the posterior variances of the weights because variance is a scale parameter. This is because the relative effect of adding a constant to the variance varies with the magnitude of the variance whereas the relative effect of multiplying it by a constant is invariant to the magnitude.

For the sources, the Jacobian matrices computed during the updates are used to infer the corrections needed in parallel updating. First the effect of all the tentative updates of the posterior means of the sources on the vectors $\mathbf{x}(t)$ and $\mathbf{s}(t)$ is computed, and then this effect is projected back to the sources. This is compared with the result which would be obtained by sequential updating, that is, by updating only one posterior source mean at a time. The amount of the update is then corrected by the ratio of the assumed and actual effect of the proposed updates. The corrections made to the sources are explained in detail in (Lappalainen and Honkela, 2000) for the static NFA method. For the dynamic model of this paper, the situation is slightly more complex. This follows from the fact that the source values $\mathbf{s}(t-1)$ at the previous time step affect also the "prior" $p(\mathbf{s}(t)|\mathbf{s}(t-1))$ of the sources $\mathbf{s}(t)$. However, essentially the same correction procedure can be used.

5.4 Initialization with embedding

The learning procedure starts by initialization of the sources to meaningful values. They are not adapted until the estimates of the nonlinear mappings $\mathbf{f}$ and $\mathbf{g}$ have been improved from their rather random initial guesses. In order to find out sources $\mathbf{s}(t)$ which describe not only the static features but also the dynamics of the data $\mathbf{x}(t)$, the data is augmented by using embedding in the initialization stage.

Embedding is a standard technique in the analysis of nonlinear dynamic systems (Haykin and Principe, 1998). In embedding, time-shifted versions $\mathbf{x}(t-k)$ of the data vectors are used in addition to the original ones. This is based on the theoretical result which states that under suitable conditions, a sequence $[\mathbf{x}(t)\,\mathbf{x}(t-1)\,\ldots\,\mathbf{x}(t-D)]$ of the data vectors contains all the information needed to reconstruct the original state if the number D of the delays is large enough (Takens, 1981).

The solution used here is to initialize the sources to the principal components of concatenated $2d+1$ subsequent data vectors

$$\mathbf{z}^T(t) = [\mathbf{x}^T(t+d)\ \mathbf{x}^T(t+d-1)\ \ldots\ \mathbf{x}^T(t-d+1)\ \mathbf{x}^T(t-d)]\,. \tag{62}$$

The dimension of $\mathbf{z}(t)$ is $2d+1$ times larger than the original data vectors.

In general, the state of a dynamic system is nonlinearly embedded in $\mathbf{z}(t)$. However, already the principal components of the concatenated vectors $\mathbf{z}$ contain useful information about the temporal behavior, giving good initial values for the sources. During the first sweeps, the data vectors $\mathbf{x}(t)$ are replaced by the concatenated vectors $\mathbf{z}(t)$. This promotes the model to find sources which describe the dynamics even when the dynamic mapping does not make use of these sources.

After some iterations the dynamic mapping starts using them to predict the future values of the sources and the augmented part of the data. The corresponding parts of the observation network (matrix $\mathbf{B}$ and vector $\mathbf{b}$) can then be removed. After that, the learning process continues using the original data $\mathbf{x}(t)$.

6 Experiments

6.1 Data

The working hypothesis tested was that our method should find such a representation which can model more easily the dynamics of the process that has generated the observed data than the data directly.

The dynamic process used to test our method was a combination of three independent dynamic processes with a total dimension of 8. One of the processes was a harmonic oscillator with angular velocity 1/3. The harmonic oscillator has a two-dimensional state representation and linear dynamics. The two other dynamic processes were chosen to be independent Lorenz processes (Lorenz, 1963). The time series of the 8 states of the combined process are shown in the topmost subfigure in Fig. 3. There the three uppermost signals correspond to the first Lorenz process, the next three to the second Lorenz process, and the last two to the harmonic oscillator.

The Lorenz system has a three-dimensional state space whose dynamics is governed by the following set of differential equations:

$$\frac{dz_1}{dt} = \sigma(z_2 - z_1) \tag{63}$$

$$\frac{dz_2}{dt} = \rho z_1 - z_2 - z_1 z_3 \tag{64}$$

$$\frac{dz_3}{dt} = z_1 z_2 - \beta z_3 \tag{65}$$

The parameter vectors $[\sigma \;\; \rho \;\; \beta]$ of the Lorenz processes were $[3 \;\; 26.5 \;\; 1]$ and $[4 \;\; 30 \;\; 1]$. The Lorenz system has three-dimensional chaotic nonlinear dynamics. The time evolution of the state of the sampled first Lorenz process is shown in Figure 4.

For making the learning problem more challenging, the dimensionality of the original state space was first reduced to five using a linear projection. These 5 projections are depicted in the middle subfigure of Fig. 3. Now the dynamics of the observations are needed in order to reconstruct the original state space since only five out of the eight dimensions are visible. Finally, the 10-dimensional data vectors $\mathbf{x}(t)$ shown in the lowermost subfigure of Fig. 3 were generated by nonlinearly mixing the five linear projections and then adding gaussian noise. The standard deviations of the signal and noise were 1 and 0.1, respectively. The nonlinear mixing was carried out using an MLP network which had randomly chosen weights and used the $\sinh^{-1}$ nonlinearity.

The processes used in the experiments are deterministic, but (11) includes a noise term. The noise term is needed even in the case of deterministic underlying process because there is always some degree of modeling error. The method can also learn noisy processes but it would be much more difficult to evaluate the results. We use prediction of the future data to measure the performance of the learned dynamical process, but there would be no single true continuation of the data if the process were noisy.

6.2 Results

The proposed method was used for learning a dynamic model of the observations[3]. Several different random initializations of the MLP networks and structures of the model were tested. For the first 500 sweeps, the concatenated vectors $\mathbf{z}(t)$ defined in (62) were used instead of the original data vectors $\mathbf{x}(t)$ as explained in Section 5.4. The suitable number of concatenated data vectors depends on the problem at hand. In these experiments, the dimension of $\mathbf{z}(t)$ was 50, that is five consecutive original data vectors form $\mathbf{z}(t)$. After 500 iterations, the time-lagged parts of $\mathbf{z}(t)$ were removed, leaving only $\mathbf{x}(t)$, and the observation MLP (12) was reduced accordingly.

In all the simulations both the MLP network (12) describing the observations $\mathbf{x}(t)$ and the MLP network (13) describing the dynamics of the sources $\mathbf{s}(t)$ had one hidden layer of 30 neurons but the size of the number of sources was varied between 8 and 11. It turned out that in the beginning of the learning, a model with 8 sources attained the lowest values of the cost function, but as the learning continued, models with 9 sources had achieved the lowest values. Figure 5 shows the best values of the cost function and the average of eight different random initializations as a function of the number of sources.

After 7500 sweeps the method had learned a dynamic process which was able to represent the observation vectors $\mathbf{x}(t)$. The standard deviation of the observations was estimated to be 0.106 on the average which is in reasonably good agreement with the actual value of 0.1. The quality of the dynamic model learned by our method was tested by predicting 1000 new source vectors using the estimated mapping $\mathbf{g}$ from the recursion $\mathbf{s}(t) = \mathbf{g}(\mathbf{s}(t-1))$. The estimated and predicted posterior means of the sources are shown in the upper subfigure of Figure 6.

Our earlier experiments (Honkela and Karhunen, 2001; Hyvärinen et al., 2001; Lappalainen and Honkela, 2000; Valpola et al., 2000) with the static nonlinear factor analysis (NFA) and nonlinear independent factor analysis (NIFA) methods indicated that 7500 sweeps were sufficient for learning the factors or sources and the nonlinear mapping $\mathbf{f}$. Clearly more sweeps are needed for learning completely the underlying dynamic process and mapping $\mathbf{g}$. Most of the learning process has ended after 100,000 sweeps, but some progress was observed even after that. After 500,000 sweeps there was no significant improvement and the simulations were finished after 1,000,000 sweeps.

[3]Matlab source code for the simulations is available at `http://www.cis.hut.fi/projects/ica/bayes/`.

In any case, the experiments confirmed that the introduced ensemble learning method is robust against over-learning. Hence there is no need to control the complexity of the resulting mappings by early stopping of learning. The lower part of Figure 6 shows the posterior means of the estimated and predicted sources at the end of the learning after 500,000 sweeps.

Visual inspection of the time series in Figure 6 confirms that the developed method is able to capture the characteristics of the dynamics of the data generating process. It also shows that only eight out of the nine factors are actually used at the end. However, it is difficult to compare the estimated dynamics with the original one by looking only at the predicted sources $\mathbf{s}(t)$. This is because the model learned by the method uses a state representation which differs from the original one.

Two processes can be considered equivalent if their state representations differ only by an invertible nonlinear transformation. Because the original states of the studied process are known, it is possible to examine the dynamics in the original state space. An MLP network was used for finding the mapping from the learned 8- and 9-dimensional sources or factors to the original 8-dimensional states. The mapping was then used for visualizing the dynamics in the original state space.

Figure 7 shows the reconstruction of the original states made from the predicted states $\mathbf{s}(t)$. It is evident that the sources contain all the required information about the state of the underlying dynamic process because the reconstructions are quite good for $1 \leq t \leq 1000$ even after 7500 sweeps. Initially our method does not model the dynamics accurately enough for being able to simulate the long term behavior of the process, but after 500,000 sweeps, the dynamics of all the three underlying subprocesses are well captured.

In Figure 8, the estimated states corresponding to the Lorenz process shown in Figure 4 are plotted after 500,000 sweeps. Note that the MLP networks modeling $\mathbf{f}$ and $\mathbf{g}$ could have represented any rotation of the state space. Hence the separation of the independent processes is only due to the form of the posterior approximation.

The ability to separate independent processes is important from a practical point of view because it usually makes it much easier to interpret the state representation. This makes it possible for example to analyse the type of change in process change detection problem as was proposed in (Iline et al., 2001).

The quality of the estimate of the underlying process was tested by studying the prediction accuracy for new samples. It should be noted that since the Lorenz processes are chaotic, the best that any method can do is to capture their general long-term behavior—exact numerical prediction is possible only in short term. Figure 7 confirms that our method achieves this goal.

The proposed approach was compared with a nonlinear autoregressive (NAR) model

$$\mathbf{x}(t) = \mathbf{h}(\mathbf{x}(t-1), \ldots, \mathbf{x}(t-d)) + \mathbf{n}(t) \tag{66}$$

which makes its predictions directly in the observation space. The nonlinear mapping $\mathbf{h}(\cdot)$ was again modeled by an MLP network, but with NAR standard back-propagation algorithm was applied in learning. Several strategies were tested, and the best performance was provided by an MLP network with 20 inputs and one hidden layer of 30 neurons. The number of delays was $d = 10$, and the dimension of the inputs to the MLP network $\mathbf{h}(\cdot)$ was compressed from 100 to 20 using standard PCA.

Figure 9 shows the performance of this NAR model. It is evident that the prediction of new observations is a challenging problem. Principal component analysis used in the model can extract features which are useful in predicting the future observations, but clearly the features are not adequate for modeling the long-term behaviour of the observations. The noise process was omitted like in the results shown in Figs. 6 and 7.

Both the NAR model and the model used in our method contain a noise process which can be taken into account in the prediction by Monte-Carlo simulation. Figure 10 shows the results averaged over 100 Monte-Carlo simulations. The average predicted observations are compared to the actual continuation of the process. The figure shows the average cumulative squared prediction error. Since the variance of the noise on the observations is 0.01, the results can be considered perfect if the average squared prediction error is 0.01. The signal variance is 1, which gives the practical upper bound of the prediction accuracy. The figure confirms that our method has been able to model the dynamics of the process, because the short-term average prediction error is close to the noise variance. Even the long-term prediction falls below the signal variance, which indicates that at least some part of the process can be predicted in the more distant future. The steady progress made during learning is also evident.

Already after 7500 sweeps our method is better than the NAR-based method in predicting the process $\mathbf{x}(t)$ few steps ahead. The performance improves greatly when learning is continued. The final predictions of our method after 500,000 sweeps are excellent up to the time $t = 1010$ and good up to $t = 1022$, while the NAR method is quite inaccurate already after $t \geq 1003$. Here the prediction started at time $t = 1000$. We have also made experiments with recurrent neural networks, which provide slightly better results than the NAR model but significantly worse than our method. Results for individual time series have been presented in (Valpola, 2000c).

An interesting finding was that the method always converged to almost the same representation of the state space. This cannot be completely due to similar initializations of the sources as the retrieved sources had different orderings and signs. Even though the state representations were similar, the prediction performance varied significantly. This is probably due to local minima of the MLP network modeling the dynamics and the chaotic nature of the underlying process which emphasizes any differences in the quality of the estimated model. It seems that the value of the cost function for the learning data is a relatively reliable indicator of the quality of the model as the prediction performance of the model with the best value of the cost function was also among the best in terms of prediction accuracy.

7 Discussion

Our experiments have shown that in its current form the described method is able to learn up to about 15-dimensional latent spaces, or to blindly extract roughly 15 source processes. Unsupervised linear techniques such as PCA and ICA can handle clearly higher dimensional latent spaces. However, our method is nonlinear, and should therefore be compared with other unsupervised nonlinear techniques. In nonlinear kernel models such as the self-organizing map (SOM) (Haykin, 1998; Principe et al., 2000), the number of parameters scales exponentially with the dimension of the latent space. This restricts in practice the use of such methods to small dimensional latent spaces, the typical dimension being only two. This is also the case with almost all the methods proposed thus far for nonlinear blind source separation and ICA (Hyvärinen et al., 2001). In our method, latent (source) space dimensions of the order of tens is attainable by splitting the model into parts. This approach was applied to modeling magnetoencephalographic data in (Särelä et al., 2001).

The emphasis in our method is on finding a good model describing well the observed data. A sufficiently good posterior approximation is important, but in large problems at least, resources are better invested in improving the model structure and prior information rather than the quality of the posterior approximation. The model is nonlinear and consequently the actual posterior cannot be described using any simple analytic form accurately. We have used gaussian approximation mainly due to its computational tractability. Posterior densities are asymptotically gaussian, but that result is not applicable to the sources as the ratio of observations to unknown variables in sources is constant. Gamma distributions could have been used as prior distributions for the variance parameters, but that modification would be of little practical significance.

The observations induce dependences between variables which were independent a priori. A key idea in ensemble learning is to neglect many of these dependences and instead try to find the best possible summary of the posterior probability having the given restricted form. In this work, almost all the dependences are neglected. Only the posterior dependences of consecutive values of the sources are taken into account. This results in automatic pruning: some sources or parts of the network can be shut down. This can be a useful property but in the beginning of the learning it can be harmful. If the estimates of the mappings are nearly random, sources seem to be useless, resulting in pruning them away. The learning algorithm is unable to escape this local minimum later on. Premature pruning is prevented by the initialization procedure described in Section 5.4.

Future work is needed for decreasing the computational load of the method. Currently learning requires a lot of computer time, taking easily days or even more. The most time consuming part is the computation of the Kullback-Leibler cost in the forward phase, because every source value, network weight, and parameter gives rise to a term of its own in the cost function. Especially the evaluation of the Jacobian matrices needed in the Taylor series approximation increases the computation time. At the moment, the computational load increases quadratically with the dimension of the latent space in practice. It is possible to reach linear complexity in this respect as proposed in (Valpola et al., 2001), although it may be that the complexity of the mapping which can be learned with the method is restricted in practice.

Another, related issue is the need to simplify the method itself. Now it is quite complicated, and therefore learning can sometimes fail. We are currently working towards that goal (Valpola et al., 2001). As usual, our method does not suit well to all types of data. For achieving good results, the model (10)–(11) should describe reasonably well the observed data.

8 Conclusions

We have introduced a Bayesian ensemble learning method for unsupervised extraction of dynamic processes from noisy data. The data are assumed to be generated by an unknown nonlinear mapping from unknown sources or factors. The dynamics of the sources are modeled using a nonlinear state-space model. The nonlinear mappings in the model are represented using MLP networks which can desribe well both mild and strong nonlinearities.

Usually MLP networks learn the nonlinear input-output mapping in a supervised manner using known input-output pairs, for which the mean-square error (MSE) of the mapping is minimized using the well-known back-propagation algorithm. Our learning procedure resembles standard back-propagation in that in the forward phase, the cost function is computed by propagating the input values through the network while keeping its weights frozen. In the backward phase, the parameters of the network are then updated.

However, our learning method differs significantly from back-propagation in several aspects. In our case, learning is unsupervised because only the outputs of the MLP networks are known. The MLP networks are used as generative models for the observations and sources rather than nonlinear signal transformation tools. The cost function employed is the Kullback-Leibler information instead of the MSE error. Probably the most significant difference lies in the Bayesian philosophy, where each variable is described using its own posterior distribution instead of a single point estimate.

The proposed method is computationally demanding, but it allows the use of higher dimensional nonlinear latent variable models than other existing approaches. The new method is able to learn a dynamic process responsible for generating the observations. Experiments with difficult chaotic data show excellent estimation and prediction performance compared with currently available nonlinear prediction techniques.

Acknowledgments

The authors thank Antti Honkela and J.-P. Barnard for useful comments and Alexander Ilin for performing some additional experiments. This research has been funded by the European Commission project BLISS and the Finnish Center of Excellence Programme (2000–2005) under the project New Information Processing Principles.

References

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

Attias, H. (1999b). Learning a hierarchical belief network of independent factor analyzers. In *Advances in Neural Information Processing Systems*, volume 11, pages 361–367. MIT Press.

Attias, H. (2000). Independent factor analysis with temporally structured sources. In Solla, S., Leen, T., and Müller, K.-R., editors, *Advances in Neural Information Processing Systems*, volume 12, pages 386–392. MIT Press.

Attias, H. (2001). ICA, graphical models and variational methods. In Roberts, S. and Everson, R., editors, *Independent Component Analysis: Principles and Practice*, pages 95–112. Cambridge University Press.

Barber, D. and Bishop, C. (1998). Ensemble learning in Bayesian neural networks. In Jordan, M., Kearns, M., and Solla, S., editors, *Neural Networks and Machine Learning*, pages 215–237. Springer, Berlin.

Bishop, C. (1995). *Neural Networks for Pattern Recognition.* Clarendon Press.

Briegel, T. and Tresp, V. (1999). Fisher scoring and a mixture of modes approach for approximate inference and learning in nonlinear state space models. In Kearns, M., Solla, S., and Cohn, D., editors, *Advances in Neural Information Processing Systems 11*, pages 403–409, Cambridge, MA, USA. The MIT Press.

Casdagli, M. (1989). Nonlinear prediction of chaotic time series. *Physica D*, 35:335–356.

Choudrey, R., Penny, W., and Roberts, S. (2000). An ensemble learning approach to independent component analysis. In *Proc. of the IEEE Workshop on Neural Networks for Signal Processing, Sydney, Australia, December 2000.* IEEE Press.

Cichocki, A., Zhang, L., Choi, S., and Amari, S.-I. (1999). Nonlinear dynamic independent component analysis using state-space and neural network models. In *Proc. of the 1st Int. Workshop on Independent Component Analysis and Signal Separation (ICA'99)*, pages 99–104, Aussois, France, January 11-15.

Gelman, A., Carlin, J., Stern, H., and Rubin, D. (1995). *Bayesian Data Analysis.* Chapman & Hall/CRC Press, Boca Raton, Florida.

Ghahramani, Z. (2001). An introduction to hidden Markov models and Bayesian networks. *Int. J. of Pattern Recognition and Artificial Intelligence*, 15(1):9–42.

Ghahramani, Z. and Beal, M. (2001). Propagation algorithms for variational Bayesian learning. In Leen, T., Dietterich, T., and Tresp, V., editors, *Advances in Neural Information Processing Systems 13*, pages 507–513, Cambridge, MA, USA. The MIT Press.

Ghahramani, Z. and Hinton, G. (2000). Variational learning for switching state-space models. *Neural Computation*, 12(4):963–996.

Ghahramani, Z. and Roweis, S. (1999). Learning nonlinear dynamical systems using an EM algorithm. In Kearns, M., Solla, S., and Cohn, D., editors, *Advances in Neural Information Processing Systems 11*, pages 599–605, Cambridge, MA, USA. The MIT Press.

Haykin, S. (1998). *Neural Networks – A Comprehensive Foundation, 2nd ed.* Prentice-Hall.

Haykin, S. and Principe, J. (May 1998). Making sense of a complex world. *IEEE Signal Processing Magazine*, 15(3):66–81.

Hinton, G. and van Camp, D. (1993). Keeping neural networks simple by minimizing the description length of the weights. In *Proc. of the 6th Ann. ACM Conf. on Computational Learning Theory*, pages 5–13, Santa Cruz, CA, USA.

Honkela, A. and Karhunen, J. (2001). An ensemble learning approach to nonlinear independent component analysis. In *Proc. European Conf. on Circuit Theory and Design (ECCTD'01)*, pages I–41/I–44, Espoo, Finland.

Hyvärinen, A., Karhunen, J., and Oja, E. (2001). *Independent Component Analysis.* J. Wiley.

Iline, A., Valpola, H., and Oja, E. (2001). Detecting process state changes by nonlinear blind source separation. In *Proc. of the 3rd Int. Workshop on Independent Component Analysis and Signal Separation (ICA2001)*, pages 704–709, San Diego, California.

Jordan, M., editor (1999). *Learning in Graphical Models.* The MIT Press, Cambridge, MA, USA.

Jordan, M., Ghahramani, Z., Jaakkola, T., and Saul, L. (1999). An introduction to variational methods for graphical models. In Jordan, M., editor, *Learning in Graphical Models*, pages 105–161. The MIT Press, Cambridge, MA, USA.

Lampinen, J. and Vehtari, A. (2001). Bayesian approach for neural networks - review and case studies. *Neural Networks*, 14(3):7–24.

Lappalainen, H. (1999). Ensemble learning for independent component analysis. In *Proc. Int. Workshop on Independent Component Analysis and Signal Separation (ICA'99)*, pages 7–12, Aussois, France.

Lappalainen, H. and Honkela, A. (2000). Bayesian nonlinear independent component analysis by multi-layer perceptrons. In Girolami, M., editor, *Advances in Independent Component Analysis*, pages 93–121. Springer-Verlag.

Lappalainen, H. and Miskin, J. (2000). Ensemble learning. In Girolami, M., editor, *Advances in Independent Component Analysis*, pages 75–92. Springer, Berlin.

Lorenz, E. (1963). Deterministic nonperiodic flow. *Journal of Atmospheric Sciences*, 20:130–141.

MacKay, D. (1992). A practical Bayesian framework for backpropagation networks. *Neural Computation*, 4(3):448–472.

MacKay, D. (1995). Developments in probabilistic modelling with neural networks - ensemble learning. In *Neural Networks: Artificial Intelligence and Industrial Applications. Proc. of the 3rd Annual Symposium on Neural Networks*, pages 191–198.

Miskin, J. and MacKay, D. (2000). Ensemble learning for blind image separation and deconvolution. In Girolami, M., editor, *Advances in Independent Component Analysis*, pages 123–141. Springer-Verlag.

Miskin, J. and MacKay, D. (2001). Ensemble Learning for blind source separation. In Roberts, S. and Everson, R., editors, *Independent Component Analysis: Principles and Practice*, pages 209–233. Cambridge University Press.

Neal, R. (1996). *Bayesian Learning for Neural Networks, Lecture Notes in Statistics No. 118.* Springer-Verlag.

Principe, J., Euliano, N., and Lefebvre, C. (2000). *Neural and Adaptive Systems - Fundamentals Through Simulations.* Wiley.

Robert, C. and Gasella, G. (1999). *Monte Carlo Statistical Methods.* Springer-Verlag.

Roberts, S. and Everson, R. (2001). Introduction. In Roberts, S. and Everson, R., editors, *Independent Component Analysis: Principles and Practice*, pages 1–70. Cambridge University Press.

Roweis, S. and Ghahramani, Z. (2001). Chapter 6: An EM algorithm for identification of nonlinear dynamical systems. In Haykin, S., editor, *Kalman Filtering and Neural Networks*, pages 175–220. Wiley, New York.

Särelä, J., Valpola, H., Vigário, R., and Oja, E. (2001). Dynamical factor analysis of rhythmic magnetoecephalographic activity. In *Proc. of the 3rd Int. Workshop on Independent Component Analysis and Signal Separation (ICA2001)*, pages 451–456, San Diego, California.

Sorenson, H. (1980). *Parameter Estimation - Principles and Problems.* Marcel Dekker.

Takens, F. (1981). Detecting strange attractors in turbulence. In Rand, D. and Young, L.-S., editors, *Dynamical Systems and Turbulence*, pages 366–381. Springer-Verlag, Berlin.

Valpola, H. (2000a). *Bayesian Ensemble Learning for Nonlinear Factor Analysis.* PhD thesis, Helsinki University of Technology, Espoo, Finland. Published in Acta Polytechnica Scandinavica, Mathematics and Computing Series No. 108.

Valpola, H. (2000b). Nonlinear independent component analysis using ensemble learning: theory. In *Proc. Int. Workshop on Independent Component Analysis and Blind Signal Separation (ICA2000)*, pages 251–256, Espoo, Finland.

Valpola, H. (2000c). Unsupervised learning of nonlinear dynamic state-space models. Technical Report A59, Lab of Computer and Information Science, Helsinki University of Technology, Finland.

Valpola, H., Giannakopoulos, X., Honkela, A., and Karhunen, J. (2000). Nonlinear independent component analysis using ensemble learning: experiments and discussion. In *Proc. of the 2nd Int. Workshop on Independent Component Analysis and Blind Signal Separation (ICA2000)*, pages 351–356, Espoo, Finland.

Valpola, H., Raiko, T., and Karhunen, J. (2001). Building blocks for hierarchical latent variable models. In *Proc. 3rd Int. Workshop on Independent Component Analysis and Signal Separation (ICA2001)*, pages 710–715, San Diego, California. Available at http://www.cis.hut.fi/harri/.

Wan, E. and Merwe, R. (2001). Chapter 7: The unscented Kalman filter. In Haykin, S., editor, *Kalman Filtering and Neural Networks*, pages 221–280. Wiley, New York.

Wan, E. and Nelson, A. (2001). Chapter 5: Dual extended Kalman filter methods. In Haykin, S., editor, *Kalman Filtering and Neural Networks*, pages 123–173. Wiley, New York.

Winther, O. (1998). *Bayesian Mean Field Algorithms for Neural Networks and Gaussian Processes*. PhD thesis, University of Copenhagen, Denmark.

Appendix: Derivations

A. Derivation of equation (45)

The result in (45) follows from (40)–(42). Note that according to (42), $s_i(t) - \mu_{si}(t)$ is gaussian with zero mean and variance $\mathring{s}_i(t)$ and independent on $\mu_{si}(t)$. Therefore

$$\begin{aligned}\widetilde{s}_i(t) = \mathrm{Var}\{s_i(t) - \mu_{si}(t) + \mu_{si}(t)\} = \mathrm{Var}\{s_i(t) - \mu_{si}(t)\} + \mathrm{Var}\{\mu_{si}(t)\} = \\ \mathring{s}_i(t) + \mathrm{Var}\{\overline{s}_i(t) + \breve{s}_i(t,t-1)[s_i(t-1) - \overline{s}_i(t-1)]\} = \\ \mathring{s}_i(t) + \mathrm{Var}\{\breve{s}_i(t,t-1)s_i(t-1)\} = \mathring{s}_i(t) + \breve{s}_i^2(t,t-1)\widetilde{s}_i(t-1)\,, \quad (67)\end{aligned}$$

where $\mathrm{Var}\{\cdot\}$ denotes the variance over the distribution $q(S, \boldsymbol{\theta})$.

B. Derivation of equation (48)

Assume that the prior distribution of γ is gaussian, having the general form $N[\gamma; m_\gamma; \exp(2v_\gamma)]$. Then

$$-\ln p(\gamma|m_\gamma, v_\gamma) = \frac{1}{2}(\gamma - m_\gamma)^2 \exp(-2v_\gamma) + v_\gamma + \frac{1}{2}\ln(2\pi)\,. \quad (68)$$

Using the fact that γ, m_γ, and v_γ are independent, expectation of (68) yields

$$\mathrm{E}_q\{-\ln p(\gamma|m_\gamma, v_\gamma)\} = \frac{1}{2}\mathrm{E}_q\{(\gamma - m_\gamma)^2\}\mathrm{E}_q\{\exp(-2v_\gamma)\} + \mathrm{E}_q\{v_\gamma\} + \frac{1}{2}\ln(2\pi)\,. \quad (69)$$

From the identity $\widetilde{\theta} = \mathrm{Var}\{\theta\} = \mathrm{E}_q\{\theta^2\} - \mathrm{E}_q\{\theta\}^2$ it follows

$$\begin{aligned}\mathrm{E}_q\{(\gamma - m_\gamma)^2\} = \mathrm{E}_q\{\gamma^2\} - 2\mathrm{E}_q\{\gamma m_\gamma\} + \mathrm{E}_q\{m_\gamma^2\} = \\ \overline{\gamma}^2 + \widetilde{\gamma} - 2\overline{\gamma}\,\overline{m}_\gamma + \overline{m}_\gamma^2 + \widetilde{m}_\gamma = (\overline{\gamma} - \overline{m}_\gamma)^2 + \widetilde{\gamma} + \widetilde{m}_\gamma\,. \quad (70)\end{aligned}$$

The result $\mathrm{E}_q\{\exp(-2v_\gamma)\} = \exp(2\widetilde{v}_\gamma - 2\overline{v}_\gamma)$ can be derived starting from the definition

$$\mathrm{E}_q\{\exp(-2v_\gamma)\} = \frac{1}{\sqrt{2\pi\widetilde{v}_\gamma}}\int \exp\left(-\frac{(v_\gamma - \overline{v}_\gamma)^2}{2\widetilde{v}_\gamma}\right)\exp\left(-2v_\gamma\right)dv_\gamma \quad (71)$$

and manipulating the integrand as follows:

$$\begin{aligned}-\frac{(v_\gamma - \overline{v}_\gamma)^2}{2\widetilde{v}_\gamma} - 2v_\gamma = -\frac{v_\gamma^2 - 2v_\gamma\overline{v}_\gamma + \overline{v}_\gamma^2 + 4v_\gamma\widetilde{v}_\gamma}{2\widetilde{v}_\gamma} = \\ -\frac{(v_\gamma - \overline{v}_\gamma + 2\widetilde{v}_\gamma)^2 + 4\overline{v}_\gamma\widetilde{v}_\gamma - 4\widetilde{v}_\gamma^2}{2\widetilde{v}_\gamma} = -\frac{(v_\gamma - \overline{v}_\gamma + 2\widetilde{v}_\gamma)^2}{2\widetilde{v}_\gamma} + 2\widetilde{v}_\gamma - 2\overline{v}_\gamma\,. \quad (72)\end{aligned}$$

This can be recognized as the expectation of the term $\exp(2\widetilde{v}_\gamma - 2\overline{v}_\gamma)$ over v_γ where v_γ has a gaussian distribution with mean $\overline{v}_\gamma - 2\widetilde{v}_\gamma$ and variance $\widetilde{v}_\gamma$. As the term is constant w.r.t. v_γ, it follows that $\mathrm{E}_q\{\exp(-2v_\gamma)\} = \exp(2\widetilde{v}_\gamma - 2\overline{v}_\gamma)$. Inserting this, (70) and $\mathrm{E}_q\{v_\gamma\} = \overline{v}_\gamma$ into (68) then yields (48).

C. Posterior mean and variance after nonlinear mapping

According to the prior model (14) for the observations $\mathbf{x}(t)$, $x_i(t)$ is gaussian with mean $f_i(\mathbf{s}(t))$ and variance v_{n_i} (here f_i denotes the ith component of the vector $\mathbf{f}$ and v_{n_i} similarly for $\mathbf{v}_n$). Similar reasoning as in Appendix B yields (49) which requires $\overline{f}_i(\mathbf{s}(t))$ and $\widetilde{f}_i(\mathbf{s}(t))$, the posterior mean and variance of the nonlinear mapping $\mathbf{f}(\mathbf{s}(t))$. A detailed derivation of these terms is given in (Lappalainen and Honkela, 2000) and here we discuss only the main results.

The derivation is based on the following approximations for the mean and variance of the output of a nonlinear scalar function $\phi(\xi)$:

$$\overline{\phi}(\xi) \approx \phi(\overline{\xi}) + \frac{1}{2}\phi''(\overline{\xi})\widetilde{\xi} \tag{73}$$

$$\widetilde{\phi}(\xi) \approx [\phi'(\overline{\xi})]^2\widetilde{\xi} \tag{74}$$

A second order Taylor series expansion about $\overline{\xi}$ yields (73) while (74) is derived from a first order expansion.

The computations with the mapping $\mathbf{f}(\mathbf{s}(t))$ proceed layer by layer much like in an ordinary MLP network except that means and variances are propagated through the network instead of simple scalar values. The nonlinearities in the hidden layer are handled by the approximations (73) and (74). In the first linear mapping from the sources to the hidden layer, the computation of the means and variances of the inputs of the hidden layer neurons are relatively simple as all the inputs and weights are mutually independent according to the posterior assumption. This is not the case in the second layer as the outputs of the hidden neurons are dependent due to the correlations induced by common inputs. The variance of the output is computed using the formula

$$\widetilde{f}_i(\mathbf{s}(t)) = \widetilde{f}_i^*(\mathbf{s}(t)) + \sum_j \widetilde{s}_j(t)\left[\frac{\partial f_i(\mathbf{s}(\mathbf{t}))}{\partial s_i(t)}\right]^2 , \tag{75}$$

where $\widetilde{f}_i^*(\mathbf{s}(t))$ denotes the variance resulting from the weights. In an MLP network with one hidden layer, the value of a weight cannot propagate through multiple paths and therefore $\widetilde{f}_i^*(\mathbf{s}(t))$ can be computed as in the first layer. Equation (75) is computationally the most expensive part of the algorithm as it requires the Jacobian matrix of $\mathbf{f}(\mathbf{s}(t))$, that is the terms $\partial f_i(\mathbf{s})/\partial s_i(t)$. This essentially requires multiplying two matrices $\mathbf{A}$ and $\mathbf{B}$ and the nonlinearity makes it impossible to make the computations in advance.

As in the standard back-propagation algorithm, the computation of the gradients involves layer-wise computations in reverse order starting from terms such as $\partial\mathcal{C}_p/\partial\widetilde{f}_i(\mathbf{s}(t)$ and yielding terms such as $\partial\mathcal{C}_p/\partial\overline{s}_i(t)$ in the end. The compuational complexity of these backward-phase computations is of the same order as that of the forward phase.

D. Computation of the terms $\mathbf{E}_q\{-\ln p(\mathbf{s}(t)|\mathbf{s}(t-1),\boldsymbol{\theta})\}$

The computation of the terms $\mathrm{E}_q\{-\ln p(\mathbf{s}(t)|\mathbf{s}(t-1),\boldsymbol{\theta})\} = \sum_i \mathrm{E}_q\{-\ln p(s_i(t)|\mathbf{s}(t-1),\boldsymbol{\theta})\}$ in the cost function $\mathcal{C}_p$ is very similar to (49) and the posterior mean $\overline{g}_i(t)$ and variance $\widetilde{g}_i(t)$ of the mapping are computed analogously to Appendix C. There are two differences: first, the sources $s_i(t)$ are random variables with variance $\widetilde{s}_i(t)$. The term thus includes $\widetilde{s}_i(t)$ in addition to $\widetilde{g}_i(\mathbf{s}(t-1))$. More importantly, the source $s_i(t)$ is assumed to have a posterior dependence with $s_i(t-1)$ according to (40)–(42). This results in an extra term.

We can approximate the dependence of $g_i(\mathbf{s}(t-1))$ on $s_i(t-1)$ by making a linear approximation $g_i(\mathbf{s}(t-1)) = \overline{g}_i(\mathbf{s}(t-1)) + \partial g_i(\mathbf{s}(t-1))/\partial s_i(t-1)[s_i(t-1) - \overline{s}_i(t-1)]$. This means that the counter-part of (70) produces a cross term of the form

$$
\begin{aligned}
& -2\mathrm{E}_q\{s_i(t)g_i(\mathbf{s}(t-1))\} = \\
& \qquad -2\mathrm{E}_q\Big\{[\overline{s}_i(t) + \breve{s}_i(t,t-1)(s_i(t-1) - \overline{s}_i(t-1))] \\
& \qquad [\overline{g}_i(\mathbf{s}(t-1) + \frac{\partial g_i(\mathbf{s}(t-1))}{\partial s_i(t-1)}(s_i(t-1) - \overline{s}_i(t-1))]\Big\} = \\
& \qquad -2\overline{s}_i(t)\overline{g}_i(\mathbf{s}(t-1)) - 2\breve{s}_i(t,t-1)\frac{\partial g_i(\mathbf{s}(t-1))}{\partial s_i(t-1)}\widetilde{s}_i(t-1)\,. \qquad (76)
\end{aligned}
$$

This differs from (70) by the second term and the final result is thus

$$
\frac{1}{2}\alpha_i(t)\exp(2\widetilde{v}_i - 2\overline{v}_i) + \overline{v}_i + \frac{1}{2}\ln 2\pi \qquad (77)
$$

$$
\alpha_i(t) = [\overline{s}_i(t) - \overline{g}_i(t)]^2 + \widetilde{s}_i(t) + \widetilde{g}_i(t) - 2\breve{s}_i(t,t-1)\frac{\partial g_i(t)}{\partial s_i(t-1)}\widetilde{s}_i(t-1)\,, \qquad (78)
$$

where the ith component of the vector $\mathbf{g}(\mathbf{s}(t-1))$ is denoted by $g_i(t)$, and the variance parameter of the ith factor by v_i.

E. The full cost function

The joint pdf $p(X, S, \boldsymbol{\theta})$ of the variables of the model is a product of many simple terms defined by (14)–(28). There is one term for each variable of the model. The approximation $q(S, \boldsymbol{\theta})$ of the posterior pdf of the unknown variables has a similar factorial form and hence the cost function splits into a sum of many simple terms due to the logarithm in (7).

All the distributions in the model are assumed gaussian and thus the terms have very similar forms. Each unknown variable generates terms for the numerator and denominator of (7) while the observed variables X only appear in the denominator. The terms originating from the sources differ from the ones originating from other unknown variables since some of the posterior correlations of the sources are modeled. Therefore there are three kinds of terms ($\mathcal{C}_\theta$, $\mathcal{C}_s$ and $\mathcal{C}_x$) corresponding to general parameters, sources and data.

In general, an unknown variable θ with mean m and standard deviation $\exp(v)$ generates the term

$$
\begin{aligned}
\mathcal{C}_\theta(\theta, m, v) &= \frac{1}{2}\left[(\overline{\theta} - \overline{m})^2 + \widetilde{\theta} + \widetilde{m}\right]\exp(2\widetilde{v} - 2\overline{v}) + \overline{v} + \frac{1}{2}\ln 2\pi - \frac{1}{2}\ln 2\pi e\widetilde{\theta} \\
&= \frac{1}{2}\left[(\overline{\theta} - \overline{m})^2 + \widetilde{\theta} + \widetilde{m}\right]\exp(2\widetilde{v} - 2\overline{v}) + \overline{v} - \frac{1}{2}\ln e\widetilde{\theta} \qquad (79)
\end{aligned}
$$

which is a combination of (48) and the appropriate term from (47). Note that $\mathcal{C}_\theta(\cdot)$ is a function of the posterior means and variances $\overline{\theta}$, $\overline{m}$, $\overline{v}$, $\widetilde{\theta}$, $\widetilde{m}$ and $\widetilde{v}$ but for short the arguments are θ, m and v.

The terms for the sources are given by (77), (78) and the appropriate term from (47):

$$
\mathcal{C}_s(s(t), g(t), v) = \frac{1}{2}\alpha(t)\exp(2\widetilde{v} - 2\overline{v}) + \overline{v} - \frac{1}{2}\ln e\mathring{s}(t) \qquad (80)
$$

$$
\alpha(t) = (\overline{s}(t) - \overline{g}(t))^2 + \widetilde{s}(t) + \widetilde{g}(t) - 2\breve{s}(t,t-1)\frac{\partial g(t)}{\partial s(t-1)}\widetilde{s}(t-1)\,. \qquad (81)
$$

The term for an observed variable is similar to (79) except that observed variables do not have posterior variance and there are no terms originating from (47):

$$
\mathcal{C}_x(x, m, v) = \frac{1}{2}\left[(x - \overline{m})^2 + \widetilde{m}\right]\exp(2\widetilde{v} - 2\overline{v}) + \overline{v} + \frac{1}{2}\ln 2\pi\,. \qquad (82)
$$

Some of the unknown parameters have hand-set prior means or variances. For instance all hyperparameters in the model are gaussian with zero mean and standard deviation 100. We shall use a short-hand notation where $\overline{m} = m$ and $\widetilde{m} = 0$ if m is a given constant. Thus the terms for the hyperparameters can be given as

$$\mathcal{C}_{\text{hyper}}(\theta) = \mathcal{C}_\theta(\theta, 0, \ln 100^2) = \frac{1}{2}\left(\overline{\theta}^2 + \widetilde{\theta}\right)/100^2 + \ln 100 - \frac{1}{2}\ln e\widetilde{\theta}\,. \tag{83}$$

We can now write down the full cost function used in this paper. The terms of the cost function can be grouped into parts related to the data, sources and the mappings $\mathbf{f}$ and $\mathbf{g}$. Let us denote them by $\mathcal{C}_{\text{data}}$, $\mathcal{C}_{\text{sources}}$, $\mathcal{C}_{\text{f}}$ and $\mathcal{C}_{\text{g}}$, respectively. The full cost function is a sum of these terms:

$$\mathcal{C}_{\text{KL}} = \mathcal{C}_{\text{data}} + \mathcal{C}_{\text{sources}} + \mathcal{C}_{\text{f}} + \mathcal{C}_{\text{g}}\,. \tag{84}$$

The individual terms are:

$$\begin{aligned}\mathcal{C}_{\text{data}} &= \sum_t \sum_i \mathcal{C}_x(x_i(t), f_i(t), v_{n_i}) + \\ &\quad \sum_i \mathcal{C}_\theta(v_{n_i}, m_{v_n}, v_{v_n}) + C_{\text{hyper}}(m_{v_n}) + C_{\text{hyper}}(v_{v_n})\end{aligned} \tag{85}$$

$$\begin{aligned}\mathcal{C}_{\text{sources}} &= \sum_t \sum_i \mathcal{C}_s(s_i(t), g_i(t), v_{m_i}) + \\ &\quad \sum_i \mathcal{C}_\theta(v_{m_i}, m_{v_m}, v_{v_m}) + C_{\text{hyper}}(m_{v_m}) + C_{\text{hyper}}(v_{v_m})\end{aligned} \tag{86}$$

$$\begin{aligned}\mathcal{C}_{\text{f}} &= \sum_i \sum_j \mathcal{C}_\theta(A_{ij}, 0, \ln 1^2) + \\ &\quad \sum_i \mathcal{C}_\theta(a_i, m_a, v_a) + C_{\text{hyper}}(m_a) + C_{\text{hyper}}(v_a) + \\ &\quad \sum_i \sum_j \mathcal{C}_\theta(B_{ij}, 0, v_{B_j}) + \\ &\quad \sum_j \mathcal{C}_\theta(v_{B_j}, m_{v_B}, v_{v_B}) + C_{\text{hyper}}(m_{v_B}) + C_{\text{hyper}}(v_{v_B}) + \\ &\quad \sum_i \mathcal{C}_\theta(b_i, m_b, v_b) + C_{\text{hyper}}(m_b) + C_{\text{hyper}}(v_b)\,.\end{aligned} \tag{87}$$

The term $\mathcal{C}_{\text{g}}$ is like $\mathcal{C}_{\text{f}}$ except A, a, B and b are replaced by C, c, D and d respectively.

F. Gradients of the cost function $\mathcal{C}_{\text{KL}}$ with respect to the source parameters

Let us use the notation $\mathcal{C}_{\text{KL}}(\widetilde{s}_i(t))$ to mean that the cost function $\mathcal{C}_{\text{KL}}$ is considered to be a function of the intermediate variables $\widetilde{s}_i(1), \ldots, \widetilde{s}_i(t)$ in addition to the parameters of the posterior approximation. The gradient computations resulting from (45) by the chain rule are then as follows:

$$\begin{aligned}\frac{\partial \mathcal{C}_{\text{KL}}}{\partial \mathring{s}_i(t)} &= \frac{\partial \mathcal{C}_{\text{KL}}(\widetilde{s}_i(t))}{\partial \mathring{s}_i(t)} + \frac{\partial \mathcal{C}_{\text{KL}}(\widetilde{s}_i(t))}{\partial \widetilde{s}_i(t)} \frac{\partial \widetilde{s}_i(t)}{\partial \mathring{s}_i(t)} \\ &= \frac{\partial \mathcal{C}_{\text{KL}}(\widetilde{s}_i(t))}{\partial \mathring{s}_i(t)} + \frac{\partial \mathcal{C}_{\text{KL}}(\widetilde{s}_i(t))}{\partial \widetilde{s}_i(t)}\end{aligned} \tag{88}$$

$$\begin{aligned}\frac{\partial \mathcal{C}_{\mathrm{KL}}}{\partial \breve{s}_i(t,t-1)} &= \frac{\partial \mathcal{C}_{\mathrm{KL}}(\widetilde{s}_i(t))}{\partial \breve{s}_i(t,t-1)} + \frac{\partial \mathcal{C}_{\mathrm{KL}}(\widetilde{s}_i(t))}{\partial \widetilde{s}_i(t)} \frac{\partial \widetilde{s}_i(t)}{\partial \breve{s}_i(t,t-1)} \\ &= \frac{\partial \mathcal{C}_{\mathrm{KL}}(\widetilde{s}_i(t))}{\partial \breve{s}_i(t,t-1)} + 2\frac{\partial \mathcal{C}_{\mathrm{KL}}(\widetilde{s}_i(t))}{\partial \widetilde{s}_i(t)} \breve{s}_i(t,t-1)\widetilde{s}_i(t-1) \end{aligned} \tag{89}$$

$$\begin{aligned}\frac{\partial \mathcal{C}_{\mathrm{KL}}(\widetilde{s}_i(t))}{\partial \widetilde{s}_i(t)} &= \frac{\partial \mathcal{C}_{\mathrm{KL}}(\widetilde{s}_i(t+1))}{\partial \widetilde{s}_i(t)} + \frac{\partial \mathcal{C}_{\mathrm{KL}}(\widetilde{s}_i(t+1))}{\partial \widetilde{s}_i(t+1)} \frac{\partial \widetilde{s}_i(t+1)}{\partial \widetilde{s}_i(t)} \\ &= \frac{\partial \mathcal{C}_{\mathrm{KL}}(\widetilde{s}_i(t+1))}{\partial \widetilde{s}_i(t)} + \frac{\partial \mathcal{C}_{\mathrm{KL}}(\widetilde{s}_i(t+1))}{\partial \widetilde{s}_i(t+1)} \breve{s}_i^2(t+1,t) \end{aligned} \tag{90}$$

The gradient $\partial \mathcal{C}_{\mathrm{KL}}(\widetilde{s}_i(t))/\partial \mathring{s}_i(t)$ can be computed from (47). Equating $\partial \mathcal{C}_{\mathrm{KL}}/\partial \mathring{s}_i(t)$ to zero results in

$$\frac{\partial \mathcal{C}_{\mathrm{KL}}}{\partial \mathring{s}_i(t)} = 0 \;\Rightarrow -\frac{1}{2\mathring{s}_i(t)} + \frac{\partial \mathcal{C}_{\mathrm{KL}}(\widetilde{s}_i(t))}{\partial \widetilde{s}_i(t)} = 0 \;\Rightarrow\; \mathring{s}_i(t) = \left[2\frac{\partial \mathcal{C}_{\mathrm{KL}}(\widetilde{s}_i(t))}{\partial \widetilde{s}_i(t)}\right]^{-1}. \tag{91}$$

This fixed-point update rule resembles (50) and a similar learning parameter is applied for stability as in (61).

Solving the equation $\partial \mathcal{C}_{\mathrm{KL}}/\partial \breve{s}_i(t,t-1) = 0$ from (90) yields the update rule (53) for the linear dependence parameter $\breve{s}_i(t,t-1)$:

$$\begin{aligned}\frac{\partial \mathcal{C}_{\mathrm{KL}}}{\partial \breve{s}_i(t,t-1)} = 0 \;\Rightarrow\; & 2\frac{\partial \mathcal{C}_{\mathrm{KL}}(\widetilde{s}_i(t))}{\partial \widetilde{s}_i(t)} \breve{s}_i(t,t-1)\widetilde{s}_i(t-1) = -\frac{\partial \mathcal{C}_{\mathrm{KL}}(\widetilde{s}_i(t))}{\partial \breve{s}_i(t,t-1)} = \\ & \frac{\partial g_i(t)}{\partial s_i(t-1)} \widetilde{s}_i(t-1) \exp(2\widetilde{v}_i - 2\overline{v}_i) \;\Rightarrow \\ & \breve{s}_i(t,t-1) = \frac{\partial g_i(t)}{\partial s_i(t-1)} \widetilde{s}_i(t-1) \exp(2\widetilde{v}_i - 2\overline{v}_i) \left[2\frac{\partial \mathcal{C}_{\mathrm{KL}}(\widetilde{s}_i(t))}{\partial \widetilde{s}_i(t)}\right]^{-1}, \end{aligned} \tag{92}$$

where $\partial \mathcal{C}_{\mathrm{KL}}(\widetilde{s}_i(t))/\partial \breve{s}_i(t,t-1)$ is obtained from (77)–(78).

The gradient $\partial \mathcal{C}_{\mathrm{KL}}(\widetilde{s}_i(t+1))/\partial \widetilde{s}_i(t)$ includes a term $\exp(2\widetilde{v}_i - 2\overline{v}_i)$ from (77)–(78). There are also terms originating from the nonlinear mappings $\mathbf{f}$ and $\mathbf{g}$ because their forward computation starts with the posterior means $\overline{s}_i(t)$ and variances $\widetilde{s}_i(t)$. The evaluation of these terms is discussed in Appendix C.

The marginalized variances $\widetilde{s}_i(t)$ are computed recursively forward in time from (45) and this has a counter-part in the computation of the gradients. The term $\partial \mathcal{C}_{\mathrm{KL}}(\widetilde{s}_i(t))/\partial \widetilde{s}_i(t)$ needed for the updates of $\mathring{s}_i(t)$ and $\breve{s}_i(t,t-1)$ depends on $\partial \mathcal{C}_{\mathrm{KL}}(\widetilde{s}_i(t+1))/\partial \widetilde{s}_i(t+1)$ according to (90). Similarly, equation (53) shows that $\breve{s}_i(t,t-1)$ depends on $\partial \mathcal{C}_{\mathrm{KL}}(\widetilde{s}_i(t))/\partial \widetilde{s}_i(t)$ which in turn depends on $\breve{s}_i(t+1,t)$ according to (90). Hence the dependences $\breve{s}_i(t,t-1)$ and the gradients with respect to the marginalized variances $\widetilde{s}_i(t)$ are computed recursively backward in time.

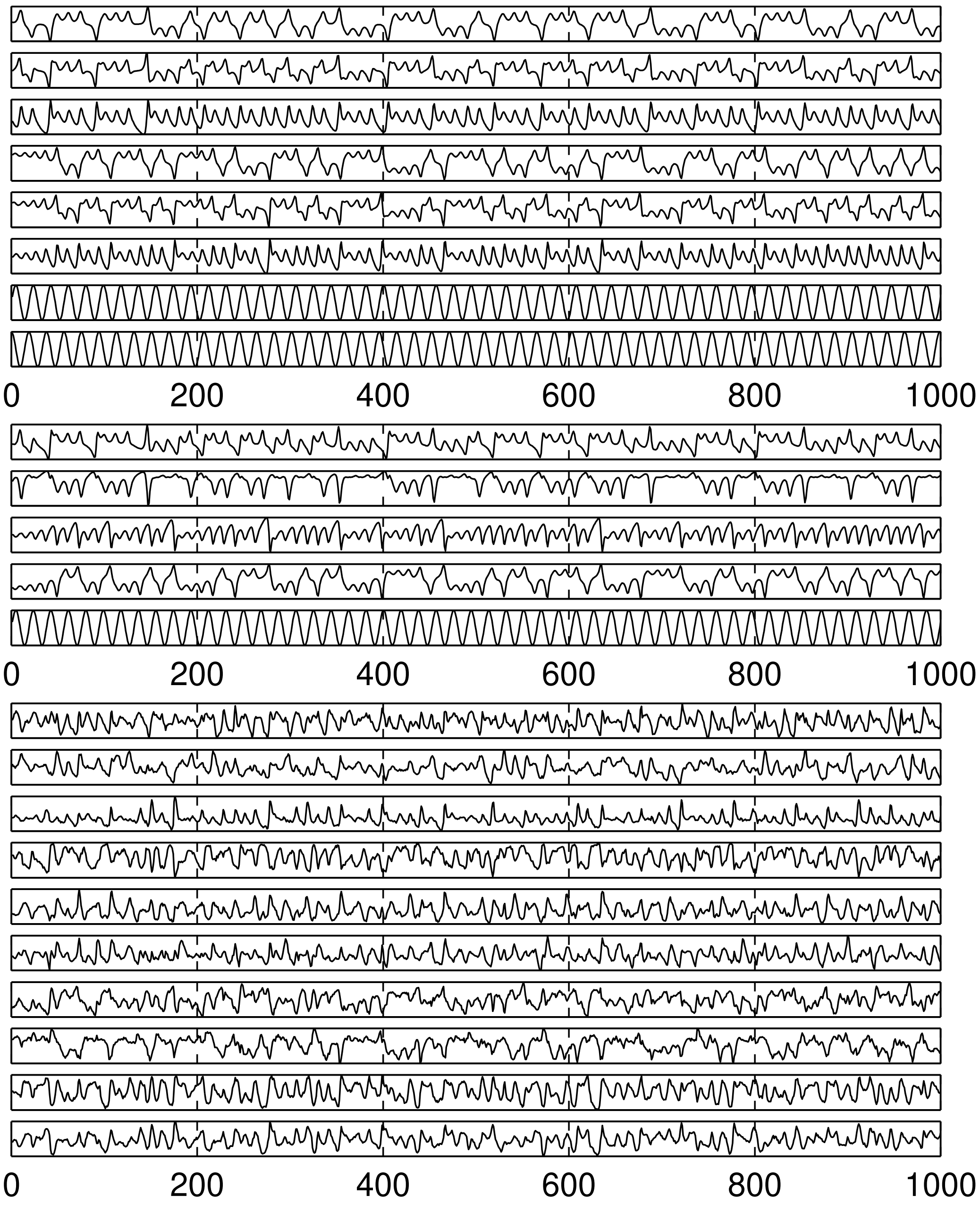

Figure 3: The 8 plots in the top subfigure show the 8 states defining the dynamics of the underlying process. The process is composed of two Lorenz processes, each having three states, and a harmonic oscillator which has two states. The middle subfigure shows the five projections used for generating the observations. The 10 plots in the lowermost subfigure represent the noisy observations obtained through nonlinear mapping from the five projections.

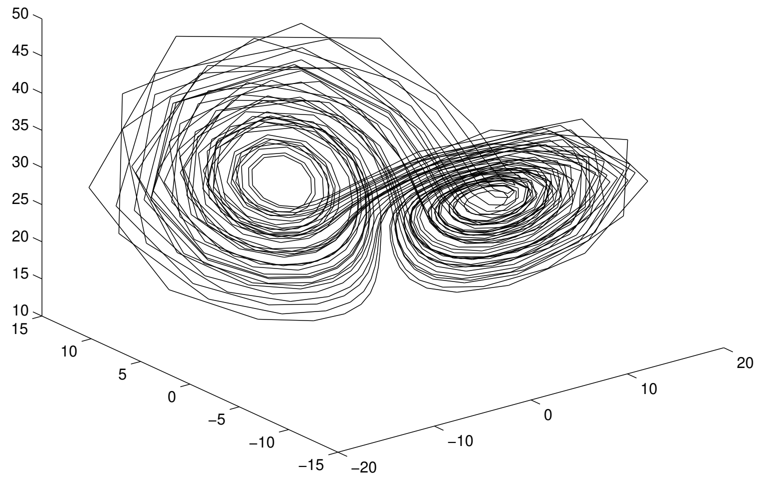

Figure 4: The original sampled Lorenz process.

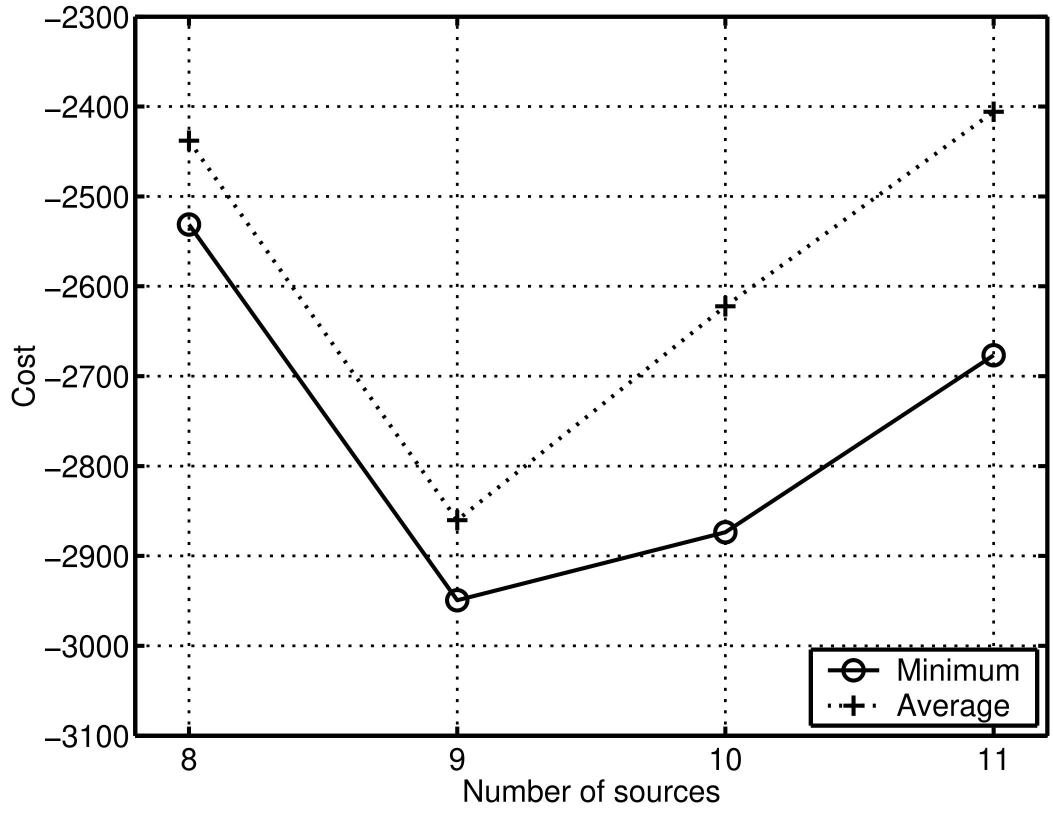

Figure 5: The best values of cost function and averages of eight simulations with different initializations are shown for different numbers of sources after 1,000,000 iterations.

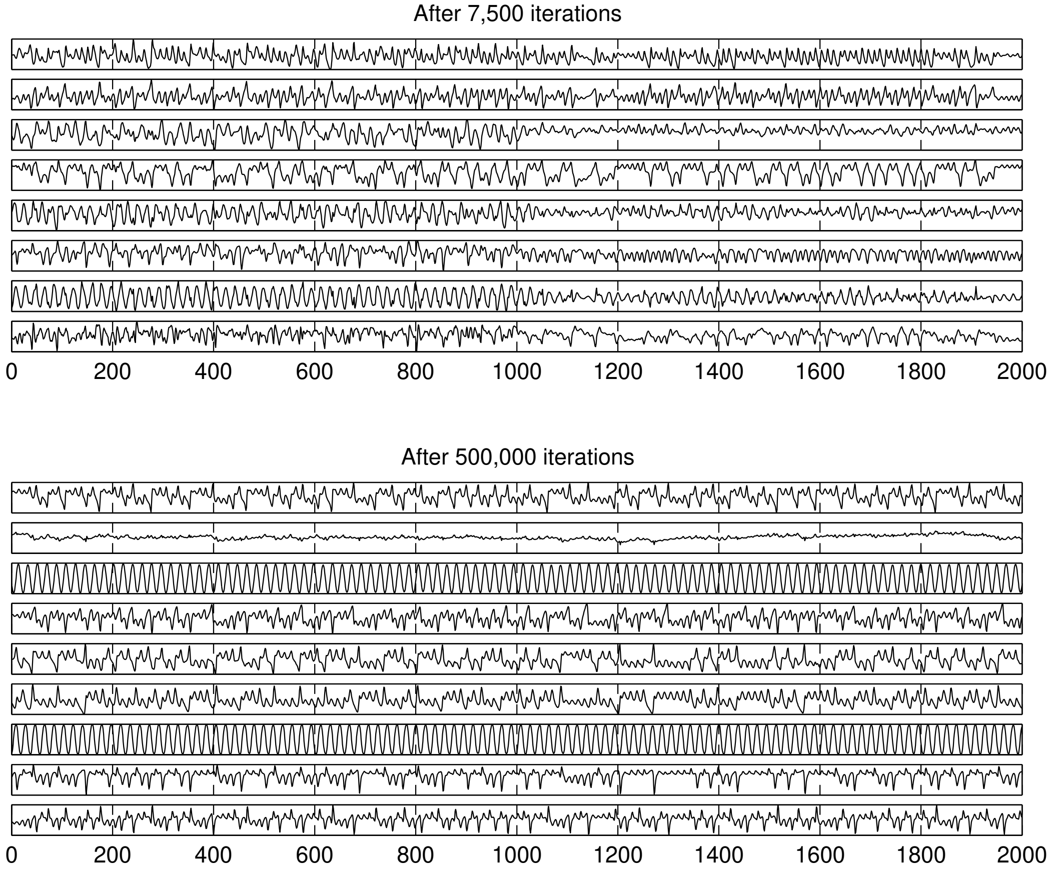

Figure 6: The 8 time series in the upper subfigure show the sources $\mathbf{s}(t)$ of the best model after 7500 iterations, and the 9 time series on the lower subfigure depict the sources of the best model after 500,000 iterations. The first 1000 values have been estimated based on the observations and the following 1000 values have been predicted using $\mathbf{s}(t) = \mathbf{g}(\mathbf{s}(t-1))$, that is, without the innovation process.

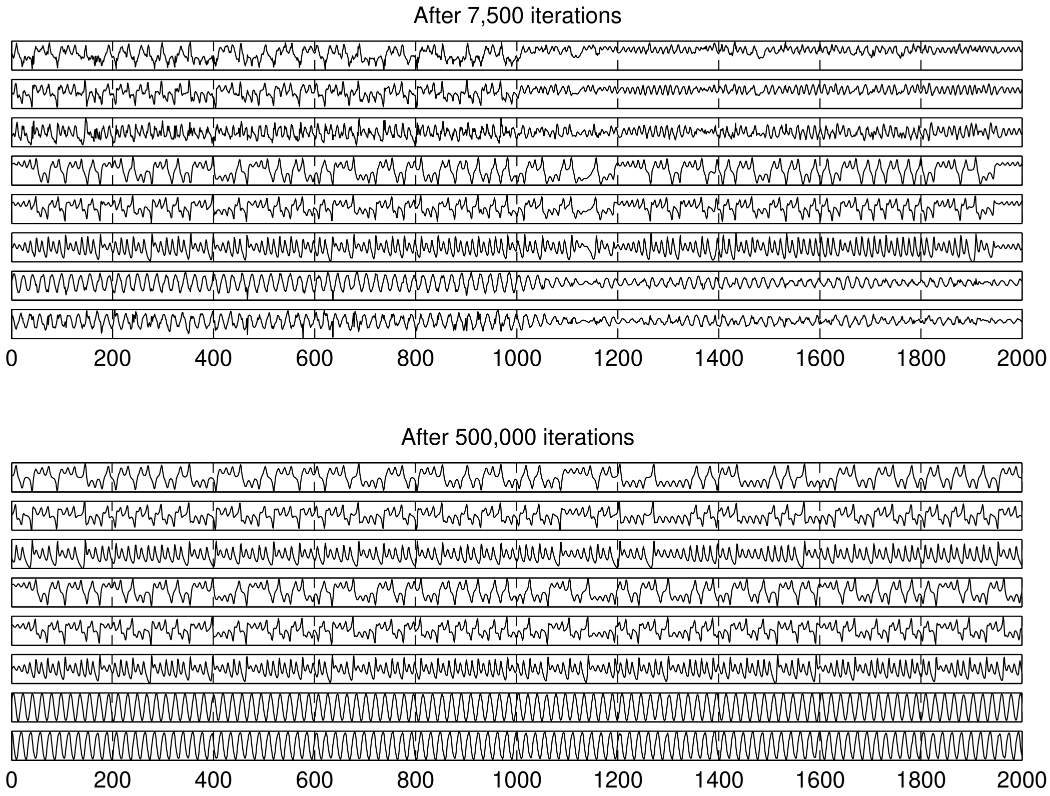

Figure 7: The estimated original 8-dimensional state of the dynamic process used for generating the data. This has been reconstructed from the predicted sources $\mathbf{s}(t)$ shown in Figure 6. After 7500 sweeps (upper subfigure), the estimated model is able to follow the dynamics of the process for a while. After 500,000 sweeps (lower subfigure), the essential characteristics of the dynamics are captured with high fidelity.

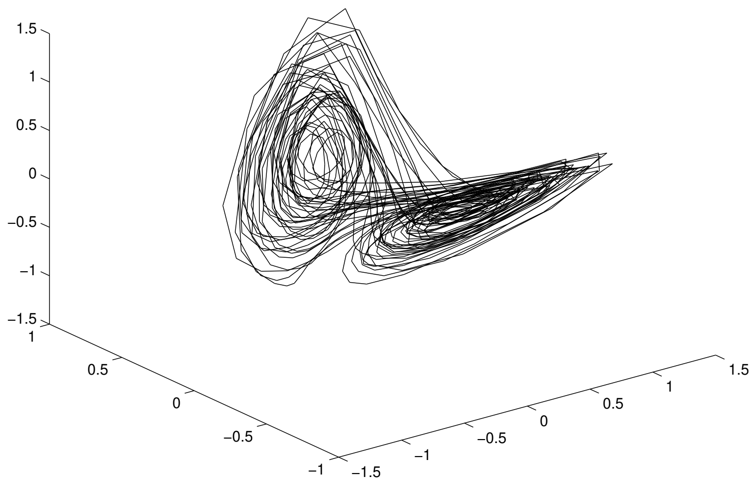

Figure 8: Three sources of the estimated process which correspond to the origial process shown in Figure 4.

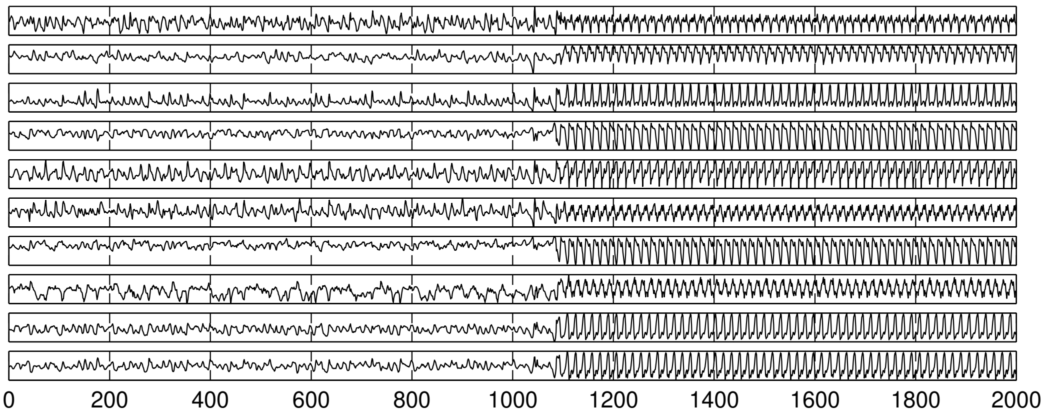

Figure 9: The ten plots show the original observations $\mathbf{x}(t)$ $(1 \leq t \leq 1000)$ and the prediction results $(t \geq 1001)$ given by a nonlinear autoregressive (NAR) model trained using standard back-prpagation algorithm.

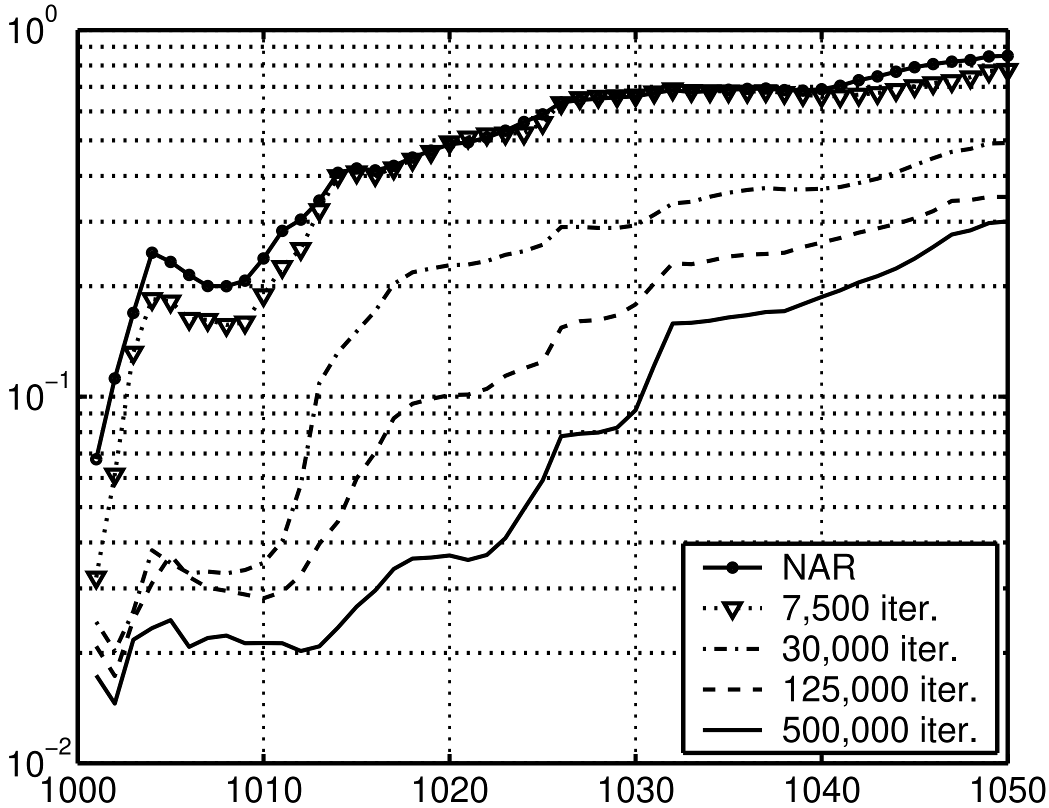

Figure 10: The average cumulative squared prediction error for the nonlinear autoregressive (NAR) model (solid with dots) and for our dynamic algorithm with different iteration numbers. Each result is for the model having the lowest value of the cost function at that stage.