NONLINEAR DYNAMICAL FACTOR ANALYSIS

XAVIER GIANNAKOPOULOS
IDSIA, Galleria 2, CH-6928 Manno, Switzerland‡

AND

HARRI VALPOLA
Neural Networks Research Centre, Helsinki University of Technology, P.O.Box 5400, 02015 HUT, Finland§

Abstract. A general method for state space analysis is presented where not only underlying factors generating the data are estimated, but also the dynamics behind time series in factor space are modelled. The mappings and the states are all unknown. The nonlinearity of the mappings makes the problem highly underdetermined and thus challenging. The Bayesian approach is able to find a set of mappings which has a high posterior probability. The model is very general: in principle any dynamical process can be modelled as a nonlinear state space model, and long-term dependencies can always be transformed into a model with more states and one-step dynamics. Potential applications are abundant. We present the results of experiments on real-world data.

Key words: Ensemble Learning, Nonlinear Factor Analysis, Dynamical Systems, Multi-Layer Perceptron Networks

1. Introduction

This work builds on [1] which introduces a nonlinear version of factor analysis capable of handling a moderately large number of factors. The key idea there is to represent the observation vectors $\mathbf{x}(t)$ as being generated by unknown factors (states, latent variables, sources) $\mathbf{s}(t)$ through an unknown nonlinear observation mapping $\mathbf{f}$ and additive measurement noise $\mathbf{n}(t)$:

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

As in [1], the nonlinear mapping is modelled by a multi-layer perceptron network. In this paper, we shall not consider known external inputs but their inclusion in the model would be straight forward.

‡Email: xavier@idsia.ch
§Email: Harri.Valpola@hut.fi, URL: http://www.cis.hut.fi/harri/

In many situations the measurements form a sequence in time and then it is reasonable to assume that the factors are the underlying states of a dynamical system. In this paper, we extend the nonlinear factor analysis by modelling the dynamics of the factors by another unknown nonlinear mapping $\mathbf{g}$ and additive process noise $\mathbf{m}(t)$:

$$\mathbf{s}(t) = \mathbf{g}(\mathbf{s}(t-1)) + \mathbf{m}(t)\,. \tag{2}$$

For any observed data there exist infinitely many different explanations in terms of (1) and (2), in other words, the problem of estimating both the unknown nonlinear functions $\mathbf{f}$ and $\mathbf{g}$ and the unknown states $\mathbf{s}(t)$ is ill-posed. The Bayesian approach does not suffer from this because all the explanations are considered, in principle at least. In practice the exact posterior probability of the unknown variables is approximated.

Section 2 introduces ensemble learning, the technique we have used for approximating the posterior probability. Section 3 then briefly outlines the approach used for nonlinear factor analysis in [1]. Equation (2) modelling the dynamics has a similar functional form as (1) and section 4 shows that it is straight forward to extend the nonlinear factor analysis into taking into account the dynamics of the factors. Section 5 discusses the case where the observation mapping $\mathbf{f}$ is linear. Then it is possible to simplify the model structure so that the resulting algorithm is more efficient than the fully nonlinear version.

2. Ensemble learning

In ensemble learning, a simple, computationally tractable factorial approximation is fitted to the true posterior probability by minimising their Kullback-Leiber information. The idea was first published in [2]. Introductory treatments of ensemble learning can be found in [3–5]. Before going into further detail, we shall briefly outline why we have not used some of the more traditional approaches.

MacKay and Gibbs have tried sampling for Bayesian learning of a nonlinear factor analysis model [6]. However, the resulting algorithm is very slow, and we therefore opt for less accurate but more efficient parametric approximation.

The standard Laplace's method for estimating the posterior density of the unknown variables fails in this problem because maximum a posteriori (MAP) estimator is sensitive to probability density, not probability mass. In the case where both the mapping $\mathbf{f}$ and the factors $\mathbf{s}(t)$ need to be estimated, MAP estimation leads to solutions where the posterior density is very high but the posterior peak is even more narrow.

The source of the problem is easily found by considering the case with linear observation mapping $\mathbf{f}(\mathbf{s}(t)) = \mathbf{A}\mathbf{s}(t)$. If the matrix $\mathbf{A}$ is multiplied by a constant and the factors $\mathbf{s}(t)$ are divided by the same constant, the model yields an identical density for the observations but a higher posterior density for the unknowns. This is because in a typical case there are far more unknown variables in the factors $\mathbf{s}(t)$ as there are elements in the matrix $\mathbf{A}$ (notice that $s_i(1)$ and $s_i(2)$ are considered two different unknown variables). The scaling increases the densities of $\mathbf{s}(t)$ and decreases the densities of the elements in the matrix $\mathbf{A}$. Since $\mathbf{s}(t)$ outnumber $\mathbf{A}$, the overall density increases.

The width of the posterior peak of the factors is inversely proportional to the determinant of the Jacobian matrix of the mapping $\mathbf{f}$. In the linear case this is simply $|\mathbf{A}|$ and it is possible to explicitly constrain $\mathbf{A}$ or $\mathbf{s}(t)$ such that the width of the posterior peak is always constant. Then a high posterior density implies high posterior mass. In the nonlinear case this would be very difficult and therefore MAP estimate is impractical.

The Kullback-Leibler information used in ensemble learning is sensitive to probability mass and therefore it avoids the problem from which MAP estimate suffers. It does not avoid the computation of the Jacobian, of course, but it results in a computationally tractable algorithm.

2.1. COST FUNCTION

The cost function in ensemble learning is (roughly) the Kullback-Leibler information between the true posterior and its factorial approximation. For the moment, let us denote the unknown variables (factors, parameters of the mappings, noise variance, etc.) by $\boldsymbol{\theta}$, the data by $\mathbf{X}$, the true posterior by $p(\boldsymbol{\theta}|\mathbf{X}) = p(\boldsymbol{\theta}, \mathbf{X})/p(\mathbf{X})$ and the factorial approximation by $q(\boldsymbol{\theta}|\mathbf{X})$. The Kullback-Leibler information between them is

$$I_{\mathrm{KL}}(q||p) = \int q(\boldsymbol{\theta}|\mathbf{X}) \ln \frac{q(\boldsymbol{\theta}|\mathbf{X})}{p(\boldsymbol{\theta}|\mathbf{X})} d\boldsymbol{\theta} = \int q(\boldsymbol{\theta}|\mathbf{X}) \ln \frac{q(\boldsymbol{\theta}|\mathbf{X})}{p(\boldsymbol{\theta}, \mathbf{X})} d\boldsymbol{\theta} + \ln p(\mathbf{X}) \,. \tag{3}$$

The term $p(\mathbf{X})$ acts as a normalising constant and it is not needed during learning because it does not depend on the unknown variables. Therefore the term $\ln p(\mathbf{X})$ can be omitted and the cost function is

$$C(\mathbf{X}; q) = I_{\mathrm{KL}}(q||p) - \ln p(\mathbf{X}) = \int q(\boldsymbol{\theta}|\mathbf{X}) \ln \frac{q(\boldsymbol{\theta}|\mathbf{X})}{p(\boldsymbol{\theta}, \mathbf{X})} d\boldsymbol{\theta} \,. \tag{4}$$

Since Kullback-Leibler information is always nonnegative and $\ln p(\mathbf{X}) = I_{\mathrm{KL}}(q||p) - C(\mathbf{X}; q)$, the cost function yields a lower bound for the log-probability of the data. This property can readily be used for model comparison.

2.2. CHOICE OF THE APPROXIMATION

For practical reasons, models are usually defined so that $p(\boldsymbol{\theta}, \mathbf{X})$ factorises into simple terms which means that $\ln p(\boldsymbol{\theta}, \mathbf{X})$ splits into a sum of simple terms. The computational efficiency of ensemble learning depends crucially on the factorial form of $q(\boldsymbol{\theta})$. It guarantees that $\ln q(\boldsymbol{\theta}|\mathbf{X})$ splits into a sum of simple terms but also that for each term, the integral weighted by $q(\boldsymbol{\theta}|\mathbf{X})$ is computationally tractable. This is because each logarithmic term depends only on a few unknown variables and all the rest of the unknown variables can be trivially integrated out if q has a factorial structure.

As an example, suppose there is a term $\ln p(\mathbf{X}|\theta_1)$ and $q(\boldsymbol{\theta}|\mathbf{X}) = q(\theta_1|\mathbf{X})q(\theta_2|\mathbf{X})$. Then the weighted integral over the term simplifies into

$$\int q(\boldsymbol{\theta}|\mathbf{X}) \ln p(\mathbf{X}|\theta_1) d\boldsymbol{\theta} = \int q(\theta_2|\mathbf{X}) \int q(\theta_1|\mathbf{X}) \ln p(\mathbf{X}|\theta_1) d\boldsymbol{\theta} = \int q(\theta_1|\mathbf{X}) \ln p(\mathbf{X}|\theta_1) d\theta_1 \, . \tag{5}$$

We see that θ_2, on which the term $\ln p(\mathbf{X}|\theta_1)$ does not depend, can be left out and the integration needs to be done only over the distribution of θ_1.

In some cases it is possible to use a free form approximation for $q(\boldsymbol{\theta}|\mathbf{X})$ where only the factorial structure is assumed but otherwise the function is chosen so that the cost function is minimised [3,5]. This approach is feasible for linear models and has been used for instance in [7–9].

Due to the complex nonlinear mappings $\mathbf{f}$ and $\mathbf{g}$, it is practically impossible to find any simple closed form solution for the distribution of the factors. We shall use a fixed form approximation where the functional form of the approximation is fixed and only the parameters of the approximation are optimised by minimising the cost function.

A natural choice for the functional form of the approximation is Gaussian because the posterior densities are at least asymptotically Gaussian but also because then the integrals are computationally tractable. The posterior is thus approximated by

$$q(\boldsymbol{\theta}|\mathbf{X}) = \prod_i q(\theta_i|\mathbf{X}) \, , \tag{6}$$

where each $q(\theta_i|\mathbf{X})$ is Gaussian. We shall denote the mean and variance of this distribution by $\bar{\theta}_i$ and $\tilde{\theta}_i$, respectively. The end result of learning is thus a Gaussian approximation of the posterior density of all unknown parameters, characterised by the posterior mean $\bar{\theta}_i$ and variance $\tilde{\theta}_i$ of each parameter.

The factorial form of (6) means that no posterior dependencies of the unknown variables are modelled, in other words, the posterior is approximated by Gaussian density with diagonal covariance matrix. Whether or not this is enough depends on the problem and it seems that for the problem at hand the diagonal approximation is good enough.

In fact, the factorial assumption has one positive side effect. In case the model has indeterminacies, that is, several different parameter values give exactly the same observation distribution, the minimisation of the misfit between the true posterior and its approximation can exploit the extra degrees of freedom and choose the solution which fits the factorial assumption best (actually the model needs not have exact indeterminacies, it suffices that the observation distributions are close enough). This has the consequence that posterior uncertainty is concentrated on a set of parameters and it is easy to spot these underdetermined parameters and prune them away. Without the factorial assumption the posterior uncertainty would have random directions in the parameter space and the directions would not be aligned with individual parameters except by chance.

3. Nonlinear factor analysis

This chapter gives an overview of the nonlinear factor analysis algorithm[1] and the results reported in [1].

3.1. MODEL DEFINITION

In the nonlinear factor analysis model presented in [5], the observations $\mathbf{x}(t)$ are modelled as having been generated by additive Gaussian i.i.d. noise $\mathbf{n}(t)$ and factors $\mathbf{s}(t)$ through a nonlinear mapping $\mathbf{f}$. The mapping $\mathbf{f}$ is modelled as an multilayer perceptron (MLP) network with tanh nonlinearities on one hidden layer. In other words, the observations are assumed to be generated as

$$\mathbf{x}(t) = \mathbf{f}(\mathbf{s}(t)) + \mathbf{n}(t) = \mathbf{B}\tanh(\mathbf{A}\mathbf{s}(t) + \mathbf{a}) + \mathbf{b} + \mathbf{n}(t)\,. \tag{7}$$

We use the notation where scalar functions operate on each element of the vector individually, that is, $\tanh[1\,2]^T = [\tanh 1 \; \tanh 2]^T$.

It has been shown that MLP networks have the universal approximation property [10] if there are enough hidden neurons[2]. This means that any nonlinearity can be modelled by the MLP network. In practice, of course, there are nonlinearities which are very difficult to model. The wide spread use of MLP networks is due to the fact that MLP networks are capable of representing many nonlinearities encountered in real world problems. It is easy to represent roughly linear mappings since tanh behaves linearly close to the origin. For this application, the important property of (7) is that the number of parameters needed for the model grows linearly with the dimension of $\mathbf{s}(t)$.

The factors $\mathbf{s}(t)$ and the noise $\mathbf{n}(t)$ have zero mean Gaussian i.i.d. models with individual variances. These variances are parametrised in logarithmic scale because then the assumption of Gaussian posterior density is more valid (corresponds to log-normal posterior).

The parameters of the model (noise parameters, matrix $\mathbf{A}$, vectors $\mathbf{a}$ and $\mathbf{b}$) are assigned hierarchical priors with Gaussian densities on each level and variances parametrised in logarithmic scale. On top of everything are 12 Gaussian prior distributions which each have large variances. This means that practically all the prior information is given in the structure of the hierarchical prior.

3.2. ALGORITHM

The learning algorithm is based on computing gradient of the cost function with respect to the posterior means $\bar{\theta}_i$ and variances $\tilde{\theta}_i$ of all the unknown variables and then solving for zero (we are using θ in the sense of a general unknown variable of the model). For efficiency, most parameters are updated simultaneously. The resulting fixed-point equations are efficient, but they have to be stabilised because they assume that other parameters are kept fixed (see [1] for details).

[1] A Matlab package is available at http://www.cis.hut.fi/projects/ica/bayes/.

[2] English translation of the neural nets jargon: the vector $\mathbf{A}\mathbf{s}(t)$ has enough dimensions.

As an example of the terms in the cost function, we shall give the term resulting from $p(x_j(t)|\mathbf{s}(t), \boldsymbol{\theta})$, one of the terms in the joint probability $p(\boldsymbol{\theta}, \mathbf{X})$. It can be shown that

$$-\int q(\boldsymbol{\theta}|\mathbf{X}) \ln p(x_j(t)|\mathbf{s}(t) d\boldsymbol{\theta} = \frac{1}{2}[(x_j(t) - \bar{f}_j(t))^2 + \tilde{f}_j(t)]e^{2\tilde{v}_j - 2\bar{v}_j} + \bar{v}_j + \frac{1}{2}\ln 2\pi \,, \tag{8}$$

where v_j denotes the log standard deviation of the noise and $\bar{f}_j(t)$ and $\tilde{f}_j(t)$ denote the posterior mean and variance of the nonlinear function f.

Computationally, the computation of the posterior mean and particularly the posterior variance of f are the most expensive parts of the algorithm. Ideally

$$\bar{f}_j(t) = \int q(\boldsymbol{\theta}|\mathbf{X}) f_j(\mathbf{s}(t), \dots) d\boldsymbol{\theta} \tag{9}$$

and

$$\tilde{f}_j(t) = \int q(\boldsymbol{\theta}|\mathbf{X})[f_j(\mathbf{s}(t), \dots) - \bar{f}_j(t)]^2 d\boldsymbol{\theta} \,. \tag{10}$$

These integrals are in practice intractable and they are approximated.

The computational efficiency of MLP networks is based on the fact that it is composed of a series of linear transformations and element-wise nonlinear transformations of the vectors. When the gradient is computed by the chain rule, it is possible to order the computations such that a vector of gradients is propagated in the opposite direction. This is known as error back-propagation algorithm in the neural networks circles.

In order to exploit the same computational efficiency, we have computed the posterior means and variances in an analogous manner. The only difference is that the linear transformations and element-wise nonlinearities are applied to distributions over vectors. At each level, the distribution is characterised by its posterior mean and variance and therefore it is possible to propagate the gradient of the cost function with respect to these intermediate values (posterior means and variances) up to the starting point, the posterior means and variances of the factors and the parameters of the MLP network.

The affect of the nonlinear transformation on the distribution is approximated by a second order Taylor's series expansion around the posterior mean when computing the transformed posterior mean and by a first order Taylor's series expansion when computing the transformed posterior variance. Care has been taken to assure that the posterior variance stays sufficiently low in order for the approximation to be accurate enough.

Evaluation of the posterior variance is computationally the most expensive part because a matrix instead of a vector needs to be propagated through the linear transformations. Due to the factorial assumption of $q(\boldsymbol{\theta}, \mathbf{X})$, the posterior covariance starts as diagonal but linear transformations introduce cross terms. If there were only one linear transformation it would be possible to do without the cross terms because they are not needed in (8). However, the cross terms resulting from the first linear transformation are needed because they affect the diagonal terms resulting after the second linear transformation. In effect, the computation of the posterior variance $\tilde{f}_j(t)$ requires the computation of $\partial f_j(t)/\partial s_k(t)$, that is, the Jacobian matrix of $\mathbf{f}(\mathbf{s}(t))$ with respect to $\mathbf{s}(t)$.

3.2.1. *Learning scheme*

Due to the factorial assumption of $q(\boldsymbol{\theta}|\mathbf{X})$, some parts of the model can be effectively pruned away during learning (the posterior distribution of parameters approaches the prior distribution). In the beginning of learning this can cause a problem. The usual thing with MLP networks is to initialise the matrices $\mathbf{A}$ and $\mathbf{B}$ and the vectors $\mathbf{a}$ and $\mathbf{b}$ randomly. However, it means that initially it seems that there is nothing useful the factors $\mathbf{s}(t)$ could represent and they will therefore get large posterior variances. This in turn means that the mapping $\mathbf{f}$ is not adapted because it cannot find any meaningful relation between the factors and the observations.

In [1], the problem was solved by initialising the posterior means of the factors to principal components of the observations and the posterior variances of the factors to small values. Only the distributions of the parameters of the mapping $\mathbf{f}$ were adapted for the first 50 iterations (one iteration meaning updating the distributions based on all observations). During another 50 iterations only the distributions of the factors and parameters of the MLP network were updated while keeping all noise parameters and hyperparameters fixed. After that all distributions were adapted for 7,400 iterations resulting in a total of 7,500 iterations. This amount was chosen conservatively so that most simulations would have converged.

Flexible nonlinear mappings, such as the MLP network, almost invariably have local minima of their parameters. For all simulations, several random initialisations of the MLP network were tested and the one giving the best fit to the posterior probability were chosen.

3.3. RESULTS

In [1], the feasibility of the algorithm was demonstrated on various artificial and a real data sets. It was shown that the algorithm is able to infer the dimension of the nonlinear data manifold embedded in a higher dimensional space. This is because the cost function used in ensemble gives a lower bound for the probability of the observations which can be used testing between hypotheses of different dimensionalities of the data manifold.

Due to rotation invariancies in the Gaussian prior distribution of the factors, it is impossible to retrieve the original factors actually used for generating the data, but it was shown that non-Gaussian model of the factors (mixture-of-Gaussians) is able to retrieve the original non-Gaussian factors.

Finally, the algorithm was shown to find a compact representation for 30 dimensional measurements taken from an industrial pulp process. The most probable model turned out to have a ten dimensional factor space and through the nonlinear mapping estimated by the algorithm these factors were able to represent as much of the data as over 20 factors using linear observation model.

4. Dynamical extension of nonlinear factor analysis

The problem of estimating a nonlinear dynamical factor analysis model has been considered easier than the problem of estimating a nonlinear factor analysis model without dynamics [11]. Given the success of the nonlinear factor analysis algorithm proposed in [1] to estimate factors of dimensionality up to ten, it is natural to try to extend the algorithm to take into account the dynamics of the factors. For instance the pulp process data presented in [1] was a time series which clearly had time dependencies between the factors. A natural extension is to consider the factors as the states of a nonlinear dynamical system and use a nonlinear model for the state dynamics.

Algorithm-wise this is very simple. In addition to the observation mapping (1), a nonlinear mapping is used to model the dynamics of the factors. The same MLP network structure can be used and consequently the same algorithms for computing the posterior means and variances of the nonlinear function and the derivatives with respect to the posterior means and variances of the arguments of the function.

The analogy for (7) will be

$$\mathbf{s}(t) = \mathbf{g}(\mathbf{s}(t-1)) + \mathbf{m}(t) = \mathbf{D} \tanh(\mathbf{C}\mathbf{s}(t-1) + \mathbf{c}) + \mathbf{d} + \mathbf{m}(t) \,, \tag{11}$$

and for (8),

$$\frac{1}{2}[(s_j(t) - \bar{g}_j(t))^2 + \tilde{s}_j(t) + \tilde{g}_j(t)]e^{2\tilde{u}_j - 2\bar{u}_j} + \bar{u}_j + \frac{1}{2}\ln 2\pi \,. \tag{12}$$

For the computation of the gradients with respect to the posterior mean and variance of the factors, the nonlinear mapping $\mathbf{g}$ introduces simply an extra additive term which propagates the information from $\mathbf{s}(t)$ to $\mathbf{s}(t-1)$.

It would be possible to use Kalman smoothing for updating the factor distributions as was done in [12,11]. This would mean that on each iteration, a forward and backward recursion of the factor distributions would be computed. On each iteration, the distribution of $\mathbf{s}(t)$ would be updated based on all the observations because the forward and backward recursion pass the information. However, we update the distribution of $\mathbf{s}(t)$ based only on the distribution of $\mathbf{s}(t-1)$, $\mathbf{s}(t+1)$ and $\mathbf{x}(t)$. This means that it takes $n+1$ iterations to pass information from $\mathbf{x}(t-n)$ and $\mathbf{x}(t+n)$ to $\mathbf{s}(t)$. We chose not to use Kalman smoothing because the adaptation of the MLP network requires many iterations in any case and the extra computational cost of Kalman smoothing would therefore be at least partially wasted.

4.1. INITIALISATION

In the nonlinear factor analysis algorithm, the factors were initialised using PCA. The same initialisation for the dynamical nonlinear factor analysis is not reasonable because it does not take time information into account. It turns out, however, that it is possible to utilise the nonlinear factor analysis algorithm very effectively to find a good initial guess for the factors.

Phase space embedding methods are standard techniques in the analysis of nonlinear dynamical systems. In short, the idea is that the internal state of a (deterministic) dynamical system is embedded in the sequence of observations. Take for instance the famous Lorenz equations which define a nonlinear dynamical system with a three dimensional state. Suppose only one of the states is measured. It is impossible to deduce the state $\mathbf{s}(t)$ of the system from one measurement $x(t)$ alone. However, a sequence $[x(t)\, x(t-h)\, x(t-2h)\, \ldots\, x(t-nh)]$ of observations contains all the information needed to reconstruct the original state [13]. In the case of Lorenz equations, the three dimensional space is nonlinearly embedded in the sequence.

Nonlinear factor analysis is well suited for extracting the underlying state from a sequence of observations and therefore we can use delay embedding and the algorithm presented in [1] to initialise the factors. However, nonlinear factor analysis alone is not enough for model comparison because it is not a generative model for the dynamical process. Therefore the dynamical nonlinear factor analysis is required at least for assessing the quality of the state space extracted by nonlinear factor analysis.

It would be possible to estimate the dynamics of the factors simply by taking the state estimates given by the nonlinear factor analysis and fitting an MLP network for the given factors. This can be used as the initialisation for the MLP network modelling the dynamics, but in practice updating the distributions of the factors improves the quality of the model.

5. Simplification for linear observation model

In some cases there is reason to believe that the observation mapping $\mathbf{f}(t)$ is linear, or a reason to test this hypothesis. Then we need an algorithm which uses the observation equation

$$\mathbf{x}(t) = \mathbf{f}(\mathbf{s}(t)) + \mathbf{n}(t) = \mathbf{A}\mathbf{s}(t) + \mathbf{a} + \mathbf{n}(t)\,. \tag{13}$$

Ensemble learning can be used in this case and since there is only one linear transformation, computation of the Jacobian matrix of $\mathbf{f}$ is trivial and the computation can be done efficiently.

If the MLP network (11) was used to model the dynamics of the factors in combination of (13), the algorithm would not be significantly faster because (11) would dominate. However, it is possible to drop one linear transformation from the MLP network and thus obtain a network whose computational cost is of the same order as for the linear mapping (13). This can be done by defining $\mathbf{s}(t) = \mathbf{D}\mathbf{s}'(t) + \mathbf{d}$ and rewriting (11) as

$$\mathbf{s}'(t) = \tanh(\mathbf{C}\mathbf{s}(t-1) + \mathbf{c}) + \mathbf{m}'(t) = \tanh(\mathbf{C}'\mathbf{s}'(t-1) + \mathbf{c}') + \mathbf{m}'(t)\,, \tag{14}$$

where $\mathbf{C}' = \mathbf{C}\mathbf{D}$, $\mathbf{c}' = \mathbf{c} + \mathbf{C}\mathbf{d}$ and $\mathbf{m}(t) = \mathbf{D}\mathbf{m}'(t)$. Similarly, (13) will look like

$$\mathbf{x}(t) = \mathbf{A}\mathbf{s}(t) + \mathbf{a} + \mathbf{n}(t) = \mathbf{A}'\mathbf{s}'(t) + \mathbf{a}' + \mathbf{n}(t)\,, \tag{15}$$

where $\mathbf{A} = \mathbf{D}\mathbf{A}'$ and $\mathbf{a}' = \mathbf{a} + \mathbf{A}\mathbf{d}$.

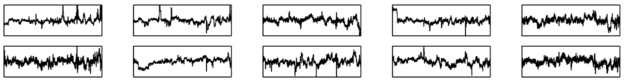

Figure 1. Posterior means of ten time series of factors estimated from the industrial pulp process by nonlinear factor analysis. Time increases form left to right.

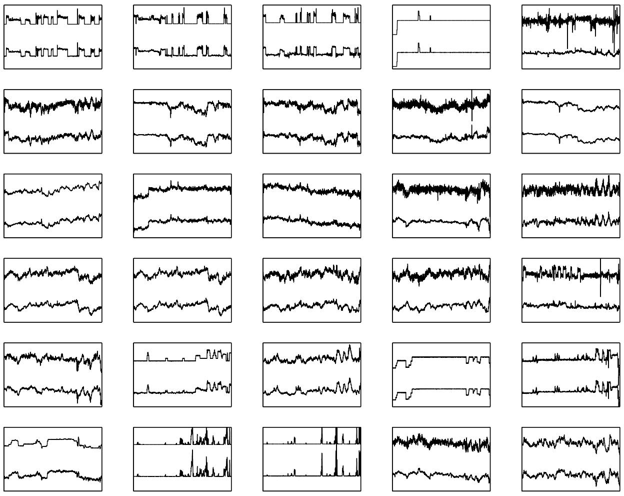

Figure 2. Each plot shows one of the thirty original time series on top of the nonlinear reconstruction made from the factors shown in figure 1.

In general, this transformation increases the dimension, but the speedup is significant and the modification is therefore useful. We are going to model the transformed innovation process $\mathbf{m}'(t)$ as i.i.d. Gaussian noise although this means that using (14) and (15) is no longer equivalent to using (11) and (13).

6. Results

We used the same industrial pulp process data as in [1]. The data set consists of 2480 samples of 30 dimensional measurements taken every 10 minutes. For comparison we show ten time series of factors extracted with the nonlinear factor analysis algorithm in figure 1. The original time series and the nonlinear reconstructions made from the ten factors are shown in figure 2. Dynamics is not taken into account and the results would be the same even if the measurements were shuffled in time.

The measurements clearly have dynamics and it is not surprising that the dynamical extension of nonlinear factor analysis model gives higher probability for the observations. Ten time series extracted with the dynamical model are shown

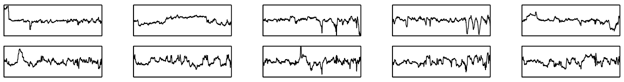

Figure 3. Fifteen time series of factors estimated from the industrial pulp process by dynamical nonlinear factor analysis. Time increases form left to right.

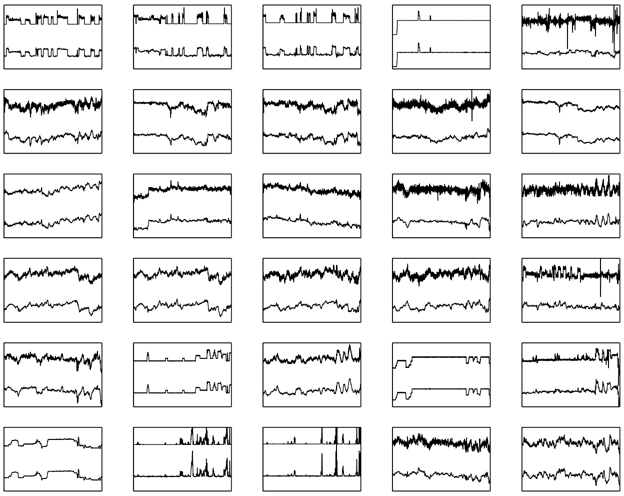

Figure 4. Each plot shows one of the thirty original time series on top of the nonlinear reconstruction made from the factors shown in figure 3.

in figure 3 and the reconstructions in figure 4.

The simplified model using (14) and (15) gives higher probability for the observations than the nonliner model without dynamics but lower than the nonlinear model with nonlinear dynamics. This is expected since the measurements are taken from a chemical process which is likely to have nonlinearities not only in the dynamics but also in the measurements. The linear observation model gives promising results with magnetoencephalographic data where the measurements can be expected to be linear functions of the underlying currents in the brain.

7. Discussion

The experiments show that the dynamical extension of the nonlinear factor analysis algorithm presented in [1] is feasible and produces interesting results. Here we presented results with 10 dimensional feature spaces but the method seems to scale quite well and it is possible to use several times larger feature spaces. However, with this high dimensional nonlinear models the interpretation of the results can be rather difficult. Some prior knowledge about the model structure should make it easier to identify the factors with physical quantities of the observed system. Also the use of non-Gaussian model for the innovation process $\mathbf{m}(t)$ should aid at the interpretation of the results in a similar way as non-Gaussian model of factors in linear factor analysis.

8. Acknowledgements

This work was funded by the EU project BLISS. The authors would like to thank Dr. J. P. Barnard for suggesting the use of nonlinear factor analysis for extracting the embedded state of a dynamical system.

References

1. H. Lappalainen and A. Honkela, "Bayesian nonlinear independent component analysis by multi-layer perceptrons," in *Advances in Independent Component Analysis*, M. Girolami, ed., pp. 93–121, Springer, Berlin, 2000.
2. G. E. Hinton and D. van Camp, "Keeping neural networks simple by minimizing the description length of the weights," in *Proceedings of the COLT'93*, (Santa Cruz, California), pp. 5–13, 1993.
3. D. J. C. MacKay, "Developments in probabilistic modelling with neural networks—ensemble learning," in *Neural Networks: Artificial Intelligence and Industrial Applications. Proceedings of the 3rd Annual Symposium on Neural Networks, Nijmegen, Netherlands, 14-15 September 1995*, (Berlin), pp. 191–198, Springer, 1995.
4. M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, "An introduction to variational methods for graphical models," in *Learning in Graphical Models*, M. I. Jordan, ed., pp. 105–161, The MIT Press, Cambridge, MA, 1999.
5. H. Lappalainen and J. W. Miskin, "Ensemble learning," in *Advances in Independent Component Analysis*, M. Girolami, ed., pp. 76–92, Springer, Berlin, 2000.
6. D. J. C. MacKay and M. N. Gibbs, "Density networks," in *Proceedings of Society for General Microbiology Edinburgh Meeting*, J. Kay, ed., 1997.
7. H. Attias, "Independent factor analysis," *Neural Computation*, **11**, (4), pp. 803–851, 1999.
8. C. M. Bishop, "Bayesian PCA," in *Advances in Neural Information Processing Systems 11*, M. S. Kearns, S. A. Solla, and D. A. Cohn, eds., pp. 382–388, MIT Press, 1999.
9. J. Miskin and D. J. C. MacKay, "Ensemble learning independent component analysis for blind separation and deconvolution of images," in *Advances in Independent Component Analysis*, M. Girolami, ed., pp. 123–141, Springer, Berlin, 2000.
10. K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, **6**, pp. 1069–1072, 1989.
11. S. T. Roweis and Z. Ghahramani, "An EM algorithm for identification of nonlinear dynamical systems," in *Kalman Filtering and Neural Networks*, S. Haykin, ed. To appear.
12. Z. Ghahramani and S. T. Roweis, "Learning nonlinear dynamical systems using an EM algorithm," in *Advances in Neural Information Processing Systems 11*, S. A. S. M. S. Kearns and D. A. Cohn, eds., pp. 599–605, MIT Press, 1999.
13. F. Takens, "Detecting strange attractors in turbulence," in *Dynamical Systems and Turbulence*, D. A. Rand and L.-S. Young, eds., pp. 366–381, Springer, Berlin, 1981.