Unsupervised Learning of Nonlinear Dynamic State-Space Models

Harri Valpola
Neural Networks Research Centre
Helsinki University of Technology
P.O.Box 5400, 02015 HUT, Finland
Email: Harri.Valpola@hut.fi
URL: http://www.cis.hut.fi/harri/

Abstract

This technical report describes how the nonlinear factor analysis algorithm introduced in [4] can be extended by taking into account the dynamics of the factors. The resulting algorithm represents the observations as having been generated by a nonlinear dynamical system. The internal states and the nonlinear mappings describing the nonlinear system are assumed to be unknown and they are estimated from the observations. Experiments with artificial data verify that the algorithm is able to reconstruct the dynamics of the underlying process which has generated the observations.

1 Introduction

This technical report describes how the nonlinear factor analysis (NLFA) algorithm [4] can be extended by modelling the dynamics of the factors. The goal is to find factors which not only represent the observations compactly but are also predictable. Only those parts which differ from the NLFA algorithm are explained and this report should therefore be read together with [4].

Like in NLFA, learning is unsupervised and is based on uncovering regularities in the observations. NLFA can capture static regularities within observation vectors $\mathbf{x}(t)$ but discards any temporal structure present in the sequence of observations. Including a model of the dynamics of the factors results in a nonlinear dynamic factor analysis (NDFA) model which can capture both static and temporal structure of the observations.

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

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

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

The observations $\mathbf{x}(t)$ are assumed to have been generated by the factors $\mathbf{s}(t)$ through a nonlinear mapping $\mathbf{f}$. It is reasonable to assume that the model misses some of the factors affecting the observations and that the nonlinear mapping is somewhat inaccurate. The error caused by these imperfections is modelled by i.i.d. Gaussian noise $\mathbf{n}(t)$. The nonlinear mapping $\mathbf{f}$ is modelled by a multi-layer perceptron (MLP) network.

The model for the dynamics of the factors has almost the same structure as the observation model. The observations are assumed to have been generated by the factors at the previous time instant. This means that the factors can be interpreted as the states of a dynamical system. The noise $\mathbf{m}(t)$ of the dynamic model is often called process noise or innovation process. Similar models have been proposed in [2, 7] where the nonlinear mappings are modelled by radial basis functions [6] and in [1] where the nonlinearities are modelled by MLP networks.

As the dynamic model (2) has the same functional form as observation model (1), similar sets of hyperparameters can be used for both models. Learning could also be achieved with the same algorithm which was used for NLFA in [4]. Some minor changes are made, however, which take into account the fact that the dynamic mapping $\mathbf{g}$ can usually be expected to be closer to identity mapping than zero and the model of the dynamics induces posterior correlations to the factors.

Section 2 discusses the properties of the state-space model used in this report and some of its alternatives. Section 3 introduces the modifications to the learning algorithm. Results of simulations are reported in section 4.

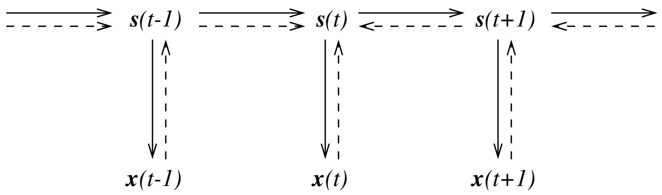


Figure 1: The causal relations assumed by the model are depicted by solid arrows. Observations $\mathbf{x}(1), \ldots, \mathbf{x}(t), \ldots$ give information about the values of the factors. The flow of information to the factor $\mathbf{s}(t)$ is represented by dashed arrows. Bayes' rule is used for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

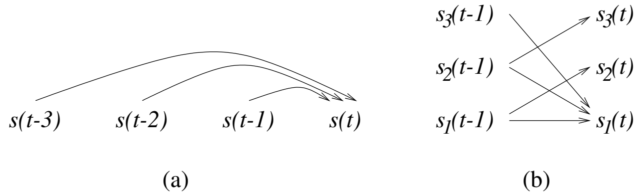


Figure 2: (a) The factor $s(t)$ depends on the three previous values $s(t-3)$, $s(t-2)$ and $s(t-1)$. (b) This dynamics can be transformed into an equivalent state representation where $s_1(t)$ corresponds to $s(t)$ while $s_2(t)$ and $s_3(t)$ store the values $s(t-1)$ and $s(t-2)$, respectively.

2 Predictable factors and state-space models

Models are tools which enable making inferences based on observations. One of their most important applications is prediction. The model can be used to infer the expected state of the world in the future or predict the expected consequences of various actions.

In unsupervised learning, the goal is to find a compact representation for the observations. The benefit is that it is often easier to find the connection between two subsets of observations using the compact representations than directly between the observations. In this case we are interested in the connection which can be made in the temporal domain.

NLFA can find compact representation for the observations. It does not take into account the temporal behaviour of the observations, but it can be expected that it is often easier to predict the future factors from the past factors than directly the future observations from the past observations. It would therefore be possible to first use NLFA to find a compact representation for observations $\mathbf{x}(t)$ in terms of factors $\mathbf{s}(t)$ and then find the mapping from the past factors $\mathbf{s}(t-1)$, $\mathbf{s}(t-2)$, $\ldots$, to the current factor $\mathbf{s}(t)$.

The drawback of this approach would be that in the first stage, learning does not explicitly aim at finding factors which facilitate the prediction. This can be remedied simply by combining the stages and letting the learning of factors take into account both the accuracy of description of the observations and the prediction of future factors.

The model defined by (1) and (2) does exactly this. Learning of factors $\mathbf{s}(t)$ 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 in figure 1.

As (2) shows, the model assumes that the factors $\mathbf{s}(t)$ can be predicted from the immediately preceding factors $\mathbf{s}(t-1)$ without knowing the factors in the more distant past. This does not restrict the class of dynamical processes which can be modelled because any model with long range dependencies can be converted into an equivalent model with only one step dependencies but larger number of factors. This means that the factors store all the information needed for predicting the dynamic behaviour of the process and therefore the factors can be interpreted as the state of the dynamical system.

Figure 2 gives an example about how a model with three step delays can be transformed into an equivalent state representation with only one step delays but more factors. In this case the two extra factors $s_2(t)$ and $s_3(t)$ store the values $s(t-1)$ and $s(t-2)$, but they could, for instance, store the first and second time derivatives of $s(t)$ as well. The benefit of using the state representation is that the model can learn the structure of the memory.

3 Model and learning algorithm

The functional form and learning algorithm for the nonlinear dynamic factor analysis (NDFA) model proposed in this report are very close to the ones introduced for NLFA in [4]: the mappings $\mathbf{f}$ and $\mathbf{g}$ are modelled by MLP networks and a gradient based learning algorithm is used for iteratively updating the approximation of the posterior probability of the unknown variables. There are some modifications, however, which take into account the fact that the mapping $\mathbf{g}$ models the dynamics of the factors.

First of all, it can be expected that the values of the factors $\mathbf{s}(t)$ are close to the values at the previous time step $\mathbf{s}(t-1)$. This prior knowledge can be taken into account by using the MLP network to model only the change in factors instead of trying to learn the whole mapping from $\mathbf{s}(t-1)$ to $\mathbf{s}(t)$. The functional form of the dynamic mapping $\mathbf{g}$ is thus

$$\mathbf{g}(s(t-1)) = \mathbf{s}(t-1) + \mathbf{D}\tanh[\mathbf{C}\mathbf{s}(t-1) + \mathbf{c}] + \mathbf{d}\,, \tag{3}$$

while the observation mapping is

$$\mathbf{f}(s(t)) = \mathbf{B}\tanh[\mathbf{A}\mathbf{s}(t) + \mathbf{a}] + \mathbf{b}\,. \tag{4}$$

It should be noted that the MLP network modelling the dynamics can learn to overrule the term $\mathbf{s}(t-1)$ if it turns out that the value of the factor at time $t-1$ does not bear information about the value of the factor at time t.

3.1 Improved approximation of the posterior probability

In [4], the posterior probability of the unknown variables was approximated as Gaussian distribution with diagonal covariance matrix. This means that the unknown variables were approximated to be independent given the observations. Notice that this assumption is false even if the unknown variables are assumed to be independent a priori. For instance, the factors are assumed to be independent a priori but observations induce dependencies between them. These dependencies are strongest for factors at the same time instant, that is, $s_i(t)$ and $s_j(t)$ have a posterior dependence but $s_i(t_1)$ and $s_j(t_2)$ can be nearly independent when t_1 is far from t_2.

The NLFA model has an indeterminacy of the rotation of the factors and the model can utilise this by choosing the rotation which makes the factors independent not only a priori but also a posteriori. However, the inclusion of the dynamic model causes posterior dependencies between factors at different time steps and these dependencies do not vanish for any rotation or any other mapping of the factor space. The smaller the process noise the stronger the dependence will be.

Taking into account the full posterior covariance between $\mathbf{s}(t-1)$ and $\mathbf{s}(t)$ is computationally costly if the dimension of the factor space is large. In practice, the most significant posterior correlations are the posterior autocorrelations of the factors, that is, the correlations between $s_i(t-1)$ and $s_i(t)$. They can be taken into account without increasing the computational complexity significantly.

In [4], the approximation of the posterior probability of the factors had the factorial form

$$q(\mathbf{S}|\mathbf{X}) = \prod_{i,t} q(s_i(t)|\mathbf{X})\,, \tag{5}$$

where $\mathbf{S}$ denotes the factors and $\mathbf{X}$ the observations. The approximations $q(s_i(t)|\mathbf{X})$ were Gaussian with mean $\bar{s}_i(t)$ and variance $\tilde{s}_i(t)$. For the dynamical model introduced in this report, the approximation has the form

$$q(\mathbf{S}|\mathbf{X}) = \prod_{i,t} q(s_i(t)|s_i(t-1), \mathbf{X})\,, \tag{6}$$

where $q(s_i(t)|s_i(t-1), \mathbf{X})$ is Gaussian and depends linearly on $s_i(t-1)$:

$$q(s_i(t)|s_i(t-1), \mathbf{X}) = N(s_i(t)|\bar{s}_i(t) + \breve{s}_i(t,t-1)[s_i(t-1) - \bar{s}_i(t-1)], \mathring{s}_i(t))\,. \tag{7}$$

Here $N(\xi|\mu, \sigma^2)$ denotes a Gaussian distribution over ξ with mean μ and variance σ^2.

Given $s_i(t-1)$, the posterior variance of $s_i(t)$ is $\mathring{s}_i(t)$ and the posterior mean is $\bar{s}_i(t) + \breve{s}_i(t,t-1)[s_i(t-1) - \bar{s}_i(t-1)]$. The approximate posterior $q(s_i(t)|s_i(t-1), \mathbf{X})$ is thus parametrised by the mean $\bar{s}_i(t)$, linear dependence $\breve{s}_i(t,t-1)$ and variance $\mathring{s}_i(t)$ as defined by (7), whereas in NLFA the posterior of the factor was parametrised by mean and variance alone.

It is easy to see by induction that if the past values of the factors are marginalised out, the posterior mean of $s_i(t)$ is $\bar{s}_i(t)$ and posterior variance $\tilde{s}_i(t)$ is

$$\tilde{s}_i(t) = \mathring{s}_i(t) + \breve{s}_i^2(t,t-1)\tilde{s}_i(t-1)\,. \tag{8}$$

Notice that the marginalised variances $\tilde{s}_i(t)$ are computed recursively in a sweep forward in time.

3.2 Feedforward and backward computations

The feedforward computations start with the parameters of the posterior approximation of the unknown variables of the model. For the factors, the parameters of the posterior approximation are the posterior mean $\bar{s}_i(t)$, the posterior variance $\mathring{s}_i(t)$ and the dependence $\breve{s}_i(t,t-1)$. The end result of the feedforward computations is the value of the cost function C.

The first stage of the computations is the iteration of (8) to obtain the marginalised posterior mean $\bar{s}_i(t)$ and variance $\tilde{s}_i(t)$ of the factors. Thereafter the computations proceed like in the NLFA algorithm: the means and variances are propagated through the MLP networks. The final stage, the computation of the cost function, differs only in the terms $\int q(s_i(t)|s_i(t-1),\mathbf{X})\ln q(s_i(t)|s_i(t-1),\mathbf{X})ds_i(t)$ and $-\int q(\boldsymbol{\theta}|\mathbf{X})\ln p(s_i(t)|\boldsymbol{\theta})d\boldsymbol{\theta}$. In the NLFA algorithm, the former had the form

$$\ln 2\pi e\tilde{s}_i(t)\,, \tag{9}$$

but now they have the form

$$\ln 2\pi e\,\mathring{s}_i(t)\,. \tag{10}$$

The latter terms can be shown to yield

$$\frac{1}{2}\left[(\bar{s}_i(t)-\bar{g}_i(t))^2+\mathring{s}_i(t)+\left(\breve{s}_i(t,t-1)-\frac{\partial g_i(t)}{\partial s_i(t-1)}\right)^2\tilde{s}_i(t-1)+\tilde{g}_i^*(t)\right]e^{2\tilde{v}_i-2\bar{v}_i}+\bar{v}_i+\frac{1}{2}\ln 2\pi\,, \tag{11}$$

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, and by $\tilde{g}_i^*(t)$, the posterior variance of $g_i(t)$ without the contribution of $s_i(t-1)$, that is, assuming $s_i(t-1)$ fixed. Notice that if $\breve{s}_i(t,t-1)$ is zero, the term inside the square brackets takes the form $(\bar{s}_i(t)-\bar{g}_i(t))^2+\mathring{s}_i(t)+\tilde{g}_i(t)$ because $\tilde{g}_i^*(t)$ is defined to be $\tilde{g}_i(t)-[\partial g_i(t)/\partial s_i(t-1)]^2\tilde{s}_i(t-1)$.

In the feedbackward phase, the gradient of the cost function C w.r.t. the parameters of the posterior approximation is computed by the back-propagation algorithm, that is, the steps of the feedforward computations are reversed and the gradient of the cost function is propagated backwards to the parameters of the posterior approximation. Since the essential modification to the feedforward phase of NLFA algorithm is (8), this is also the essential modification in the backward computations.

The cost function is a function of parameters of the posterior approximation. In the computation of the cost function, the marginalised posterior variances $\tilde{s}_i(t)$ of the factors are used as intermediate variables and hence the gradient is also computed through these variables. Let us use the notation $C(\tilde{s}_i(t))$ to mean that C is considered to be a function of the intermediate variables $\tilde{s}_i(1), \ldots, \tilde{s}_i(t)$ in addition to the parameters of the posterior approximation. The gradient computations resulting from (8) by the chain rule are then as follows:

$$\frac{\partial C}{\partial \mathring{s}_i(t)} = \frac{\partial C(\tilde{s}_i(t))}{\partial \mathring{s}_i(t)} + \frac{\partial C(\tilde{s}_i(t))}{\partial \tilde{s}_i(t)}\frac{\partial \tilde{s}_i(t)}{\partial \mathring{s}_i(t)} = \frac{\partial C(\tilde{s}_i(t))}{\partial \mathring{s}_i(t)} + \frac{\partial C(\tilde{s}_i(t))}{\partial \tilde{s}_i(t)} \tag{12}$$

$$\frac{\partial C}{\partial \breve{s}_i(t,t-1)} = \frac{\partial C(\tilde{s}_i(t))}{\partial \breve{s}_i(t,t-1)} + \frac{\partial C(\tilde{s}_i(t))}{\partial \tilde{s}_i(t)}\frac{\partial \tilde{s}_i(t)}{\partial \breve{s}_i(t,t-1)} = \frac{\partial C(\tilde{s}_i(t))}{\partial \breve{s}_i(t,t-1)} + 2\frac{\partial C(\tilde{s}_i(t))}{\partial \tilde{s}_i(t)}\breve{s}_i(t,t-1)\tilde{s}_i(t-1) \tag{13}$$

$$\frac{\partial C(\tilde{s}_i(t))}{\partial \tilde{s}_i(t)} = \frac{\partial C(\tilde{s}_i(t+1))}{\partial \tilde{s}_i(t)} + \frac{\partial C(\tilde{s}_i(t+1))}{\partial \tilde{s}_i(t+1)}\frac{\partial \tilde{s}_i(t+1)}{\partial \tilde{s}_i(t)} = \frac{\partial C(\tilde{s}_i(t+1))}{\partial \tilde{s}_i(t)} + \frac{\partial C(\tilde{s}_i(t+1))}{\partial \tilde{s}_i(t+1)}\breve{s}_i^2(t+1,t) \tag{14}$$

The terms $\partial C(\tilde{s}_i(t))/\partial \mathring{s}_i(t)$ and $\partial C(\tilde{s}_i(t))/\partial \breve{s}_i(t, t-1)$ can be computed from (10) and (11) while $\partial C(\tilde{s}_i(t+1))/\partial \tilde{s}_i(t)$ also includes terms originating from the mappings $\mathbf{f}$ and $\mathbf{g}$ as their feedforward computation starts with the posterior means $\bar{s}_i(t)$ and variances $\tilde{s}_i(t)$.

In the adaptation, the posterior means $\bar{s}_i(t)$ of the factors are treated as in the NLFA algorithm except for the correction in the step size which is discussed in section 3.3. The variances $\mathring{s}_i(t)$ are adapted like $\tilde{s}_i(t)$ in the NLFA. The posterior dependence $\breve{s}_i(t, t-1)$ is adapted by solving $\partial C/\partial \breve{s}_i(t, t-1) = 0$ which yields

$$\breve{s}_i(t, t-1) = \frac{\frac{\partial g_i(t)}{\partial s_i(t-1)} e^{2\tilde{v}_i - 2\bar{v}_i}}{2\frac{\partial C(\tilde{s}_i(t))}{\partial \tilde{s}_i(t)} + e^{2\tilde{v}_i - 2\bar{v}_i}} \,. \tag{15}$$

Equation (15) shows that $\breve{s}_i(t, t-1)$ depends on $\partial C(\tilde{s}_i(t))/\partial \tilde{s}_i(t)$ which in turn depends on $\breve{s}_i(t+1, t)$ as (14) shows. This means that the update of the dependencies $\breve{s}_i(t, t-1)$ and the computation of the gradient w.r.t. the marginalised variance $\tilde{s}_i(t)$ are done recursively backward in time which is the counterpart of (8) where the marginalised variances are computed recursively forward in time.

3.3 Correction of the step size

In the NLFA algorithm, the step sizes of the updates of the posterior means $\bar{s}_i(t)$ of the factors were corrected by taking into account the correlated effects of the updates. In NLFA the update of the posterior means $\bar{s}_i(t)$ affect the cost function through terms which result from the prior for factor $s_i(t)$ and likelihood $p(\mathbf{x}(t)|\mathbf{s}(t))$. The likelihood results in several terms, one for each component of the observation vector $\mathbf{x}(t)$. The problem was that the update rule gives the optimal value for each $\bar{s}_i(t)$ assuming that otherwise the posterior approximation stays fixed but this assumption is violated because for efficiency, all posterior means and variances are updated at once. This was remedied by utilising the Jacobian matrix of $\mathbf{f}(\mathbf{s}(t))$ w.r.t. $\mathbf{s}(t)$.

In the NDFA algorithm the situation is slightly more complex as the "prior" $p(\mathbf{s}(t)|\mathbf{s}(t-1))$ of the factors $\mathbf{s}(t)$ is also affected by the factors at the previous time step, but essentially the same procedure can be used: first the effect of all the proposed updates of the posterior means of factors on $\mathbf{x}(t)$ and $\mathbf{s}(t)$ is computed, then the step is projected back to the factors. This is compared with the result which would be obtained assuming that each of the posterior means of the factors is the only parameter of the posterior approximation to change. The size of the update is then corrected by the ratio of the assumed and actual effect of the proposed updates just as in the NLFA algorithm.

3.4 Initialisation of the factors

In the NLFA algorithm, the factors were initialised to the principal components of the observations. This could also be used in NDFA, but the drawback would be that the initialisation would only aim at factors which give a good representation for the observations but would not explicitly aim at factors which can predict the future factors and can themselves be predicted from the past factors.

Phase space embedding methods are standard techniques in the analysis of nonlinear dynamical systems and they can also be applied here. In short, the idea is that the internal state of a (deterministic) dynamical system is embedded in the sequence of observations. It may be impossible to deduce the state $\mathbf{s}(t)$ of the system from one measurement $\mathbf{x}(t)$ alone. Under suitable conditions, however, a sequence $[\mathbf{x}(t)\, \mathbf{x}(t-1)\, \mathbf{x}(t-2)\, \ldots\, \mathbf{x}(t-D)]$ of observations contains all the information needed to reconstruct the original state if the number D of delays is large enough [8].

The solution used here is to initialise the factors to principal components of a sequence of observations. In general, the state of a dynamical system is nonlinearly embedded in the sequence, but principal component analysis can nevertheless find a good starting point for the factors. To be more specific, instead of computing the principal components from $\mathbf{x}(t)$, they are computed from $\mathbf{y}(t) = [\mathbf{x}^T(t)\, \mathbf{x}^T(t-1)\, \ldots\, \mathbf{x}^T(t-D)]^T$.

The NLFA algorithm could be used[1] for extracting a state which is nonlinearly embedded in $\mathbf{y}(t)$. This would amount to computing the principal components of $\mathbf{y}(t)$, then using the NLFA algorithm to further refine the extracted factors and finally learning the dynamic model for $\mathbf{x}(t)$ starting from the factors given by the NLFA algorithm. However, here the dynamic model is included already in the second phase: the NDFA algorithm is used for finding factors which can represent the concatenated observations $\mathbf{y}(t)$.

The benefit of this procedure is that using the concatenated observations $\mathbf{y}(t)$ as observations promotes the algorithm to find factors which can represent not only the original observations $\mathbf{x}(t)$ but also their time behaviour.

[1] This was suggested by Dr. J. P. Barnard (personal communication).

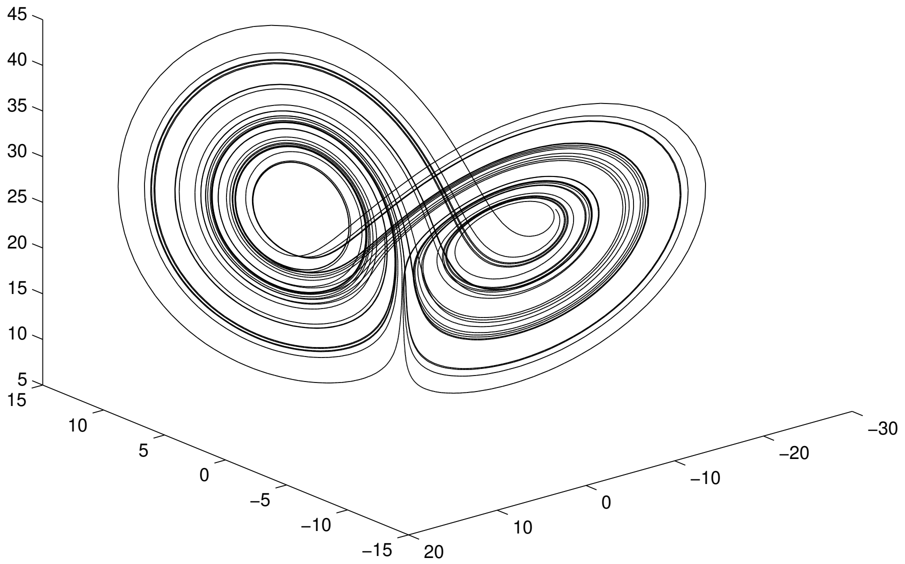


Figure 3: Time evolution of the state resulting from the Lorenz system with $\sigma = 3$, $\rho = 26.5$ and $\beta = 1$.

Once these factors have appeared and the dynamic mapping $\mathbf{g}(t)$ has adapted, the factors representing the dynamics have support from the dynamic mapping and the time lagged part of $\mathbf{y}(t)$ can be dropped away leaving only $\mathbf{x}(t)$. If the learning starts directly with $\mathbf{x}(t)$, there is the danger that some of the factors describing the dynamics will be effectively pruned away.

4 Results

4.1 Problem setting

The working hypothesis to be tested was that the NDFA algorithm should be able to find a representation for the observations which facilitates the prediction of the observations, i.e., a representation where modelling the dynamics of the underlying data generating process is easier than directly for the original observations.

Data was generated by first defining the underlying dynamic process and then mapping some of the states of the process onto observations by one of the random MLP networks which were used in [4].

The underlying dynamic process was constructed by combining three independent dynamic processes. One of the processes was harmonic oscillator with angular velocity $1/3$. The harmonic oscillator has a two-dimensional state representation and linear dynamics. Notice, however, that if the harmonic oscillator is modelled with linear dynamics, the gain has to be exactly one. Otherwise the oscillations will either decay or grow exponentially for gains less than or greater than one, respectively. A robust model for a harmonic oscillator with a constant amplitude therefore needs to be nonlinear.

The two other processes were chosen to be Lorenz processes [5]. The Lorenz system has a three-dimensional state space whose dynamics is governed by the following set of differential equation:

$$\frac{dz_1}{dt} = \sigma(z_2 - z_1) \tag{16}$$

$$\frac{dz_2}{dt} = \rho z_1 - z_2 - z_1 z_3 \tag{17}$$

$$\frac{dz_3}{dt} = z_1 z_2 - \beta z_3 \tag{18}$$

The parameter vectors $[\sigma \;\; \rho \;\; \beta]$ for the processes were chosen to be $[3 \;\; 26.5 \;\; 1]$ and $[4 \;\; 30 \;\; 1]$. The time evolution of the state for the first Lorenz process is shown in figure 3.

The ability to find the underlying factors which have generated the observations was already demonstrated in [4]. To make the problem more difficult, one dimension of each of the three elementary processes was hidden. As the original process has a total of eight states, five states are left from which the data was generated nonlinearly. In [4], some of the experiments used a random MLP network with inverse hyperbolic sine activation functions to generate ten-dimensional observations from five inputs. The same MLP network was used here. Like in [4], the data set consisted of 1000 observation.

Figure 4 shows the eight states of the underlying dynamical system, the five projections made from them and the noisy observations obtained from the projections by the random MLP network. The standard deviation of the observation noise was 0.1 while the standard deviation of the signal was normalised to unity.

4.2 Finding the underlying process

The NDFA algorithm was used for learning a dynamic model of the observations. Several different random initialisations of the MLP networks and structures of the model were tested. For the first 500 iterations, the concatenated vector $\mathbf{y}(t) = [\mathbf{x}(t-2)^T \ \dots \ \mathbf{x}(t+2)^T]^T$ was used instead of $\mathbf{x}(t)$ as the observation vectors. After that, $\mathbf{y}(t)$ was replaced by $\mathbf{x}(t)$ and the observation MLP was reduced accordingly. The cost function was found to be minimised by a model where there were ten factors and both the observation MLP network and the factor dynamics MLP network had one hidden layer of 30 neurons.

After 7500 iterations the model had learned a dynamic process which was able to represent the observations. 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. In order to test the quality of the dynamic model learned by the algorithm, 1000 new values were predicted for the factors using the estimated mapping $\mathbf{g}$. The factors are shown in the upper part of figure 5.

The experiments with NLFA reported in [4] indicated that 7500 iterations were sufficient for learning the factors and the observation mapping. It turned out that more iterations are needed to fully learn the underlying dynamical process. Most of the learning was finished after 100,000 iterations, but some progress was observed even after 600,000 iterations. The simulation was not continued beyond that, however. In any case, the experiment confirms that ensemble learning is robust against overlearning, i.e., there is no need to control the complexity of the resulting mappings by early stopping of learning. The lower part of figure 5 shows the factors in the end of learning.

Visual inspection of the plots in figure 5 confirms that the NDFA algorithm has been able to capture the characteristics of the dynamics of the data-generating process. It also shows that only nine out of ten factors are actually used in the end. However, it is difficult to compare the estimated dynamics with the original by looking only at the predicted factors $\mathbf{s}(t)$. This is because the model learned by the NDFA uses a state representation which differs from the original.

Two processes can be considered equivalent if their state representations differ only by an invertible nonlinear transformation. As the original underlying states of the 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 ten-dimensional factors to the original eight-dimensional states. The mapping was then used for visualising the dynamics in the original state space.

Figure 6 shows the reconstruction of the original states made from the predicted states $\mathbf{s}(t)$. First of all, it is evident that the factors contain all the required information about the state of the underlying process because the reconstructions are quite good for $1 \leq t \leq 1000$ even after 7500 iterations. Initially the dynamics is not modelled accurately enough to simulate the long term behaviour of the process, but in the end, the dynamics of all three underlying subprocesses are captured.

4.3 Prediction accuracy

An obvious way to assess the quality of the learned model is to see on how long term the predictions given by the model are accurate. It should be noted that since the Lorenz process is chaotic, it is numerically impossible to predict it exactly in the distant future. The best that can be hoped for is to capture the overall aspects of the long-term behaviour. Figure 6 confirms that this is accomplished by the NDFA algorithm.

For comparison, the prediction of the future observations was tested with a nonlinear auto-regressive (NAR) model

$$\mathbf{x}(t) = \mathbf{h}(\mathbf{x}(t-1), \mathbf{x}(t-2), \dots, \mathbf{x}(t-D)) + \mathbf{n}(t) . \tag{19}$$

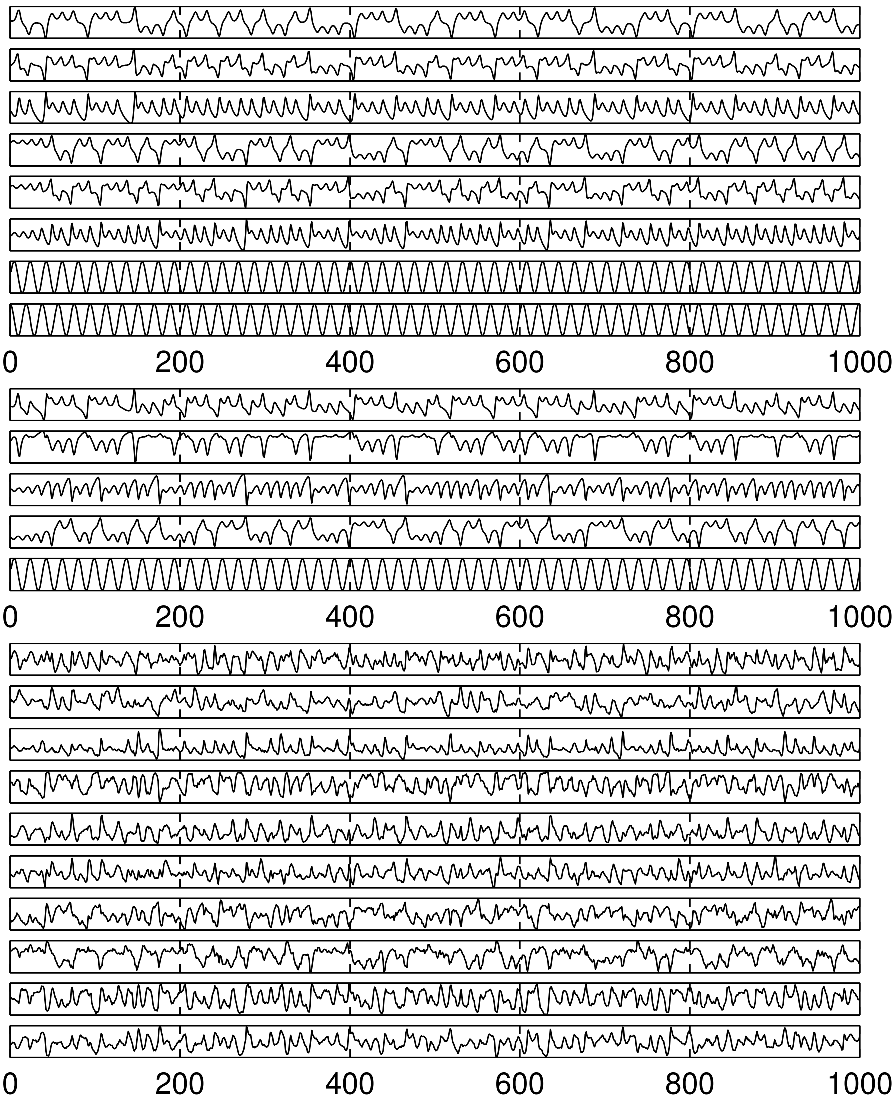


Figure 4: Each plot shows a time series. The eight plots on the top show the eight states on which the dynamics of the underlying process is defined. The process is composed of two Lorenz processes, each of which has three states, and a harmonic oscillator which has two states. Five projections from the eight states are used for generating the observations. These are shown in the middle. The ten plots on the bottom of the figure are the noisy observations which are obtained by instantaneous nonlinear mapping from the five projections.

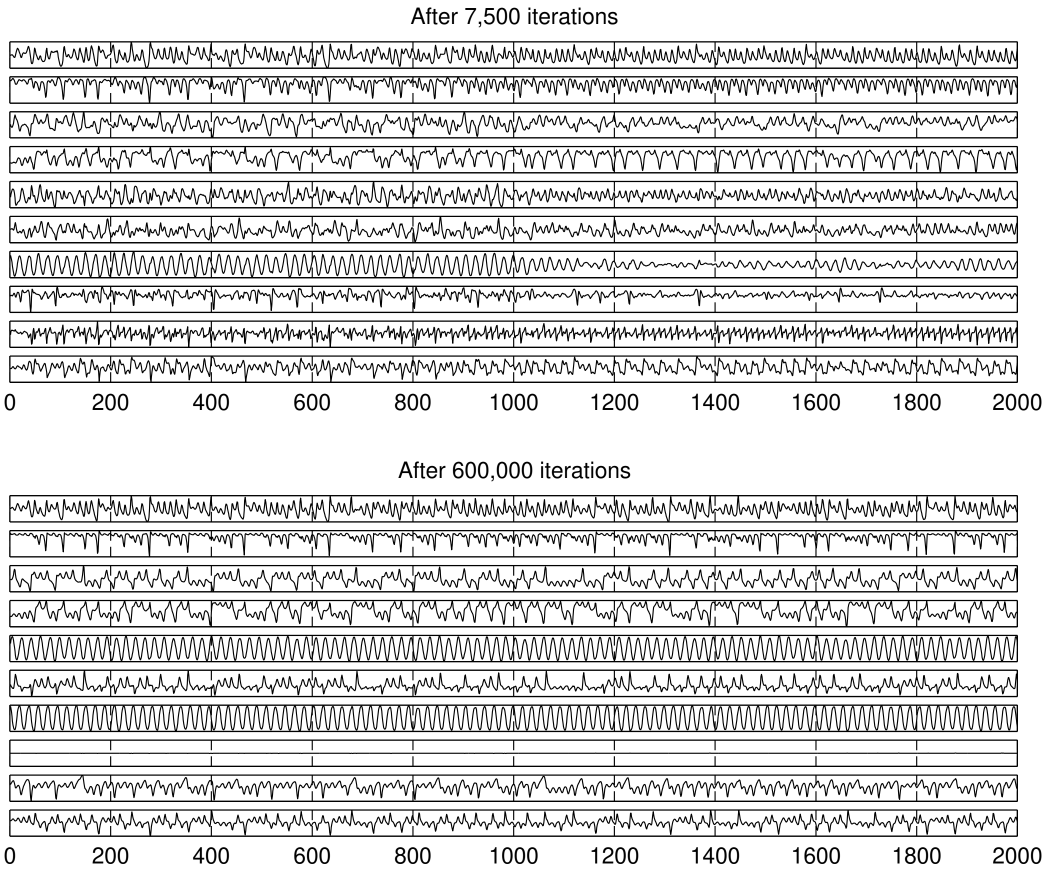
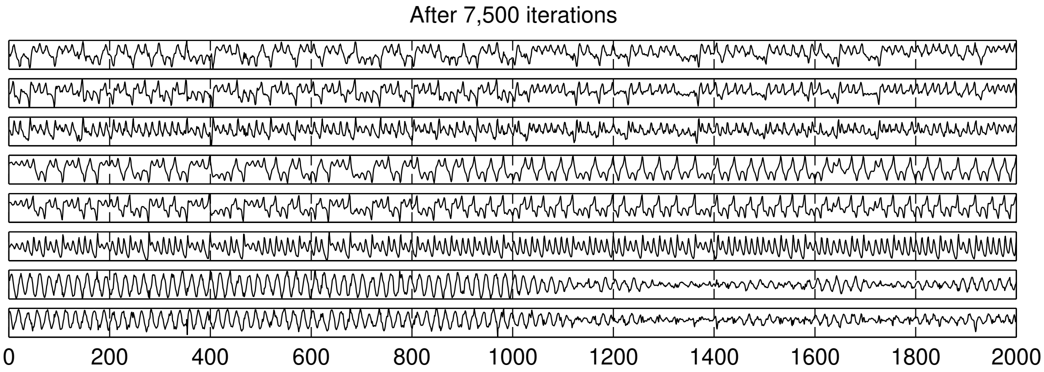


Figure 5: The ten plots on the top show the factors $\mathbf{s}(t)$ after 7500 iterations and the ten plots on the bottom show the factors after 600,000 iterations. The first 1000 values have been estimated based on the observations and the following 1000 values have been predicted using $\mathbf{s}(t) = \mathbf{g}(\mathbf{s}(t-1))$, i.e., without the innovation process.



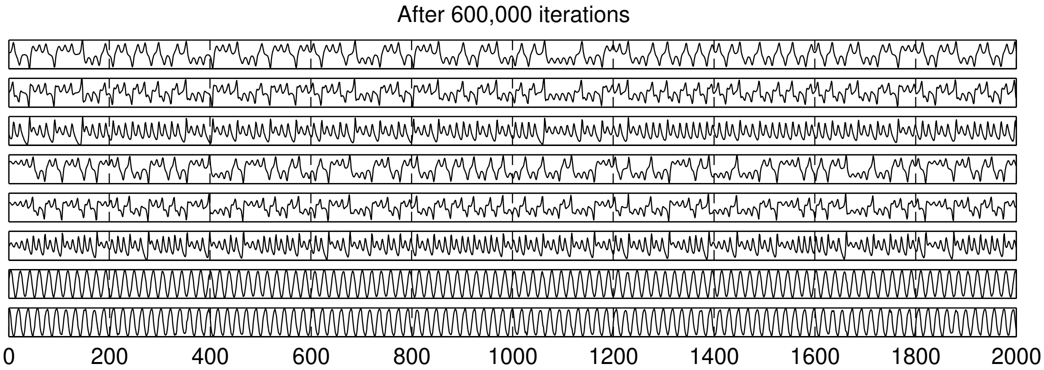


Figure 6: The original eight-dimensional state of the underlying process used for generating the observations is reconstructed from the predicted factors $\mathbf{s}(t)$ shown in figure 5. After 7500 iterations the model can follow the dynamics for a while and after 600,000 iterations the essential characteristics of the dynamics are captured with high fidelity as can be seen by comparing the original latent variables ($1 \leq t \leq 1000$) to the following 1000 predicted states.

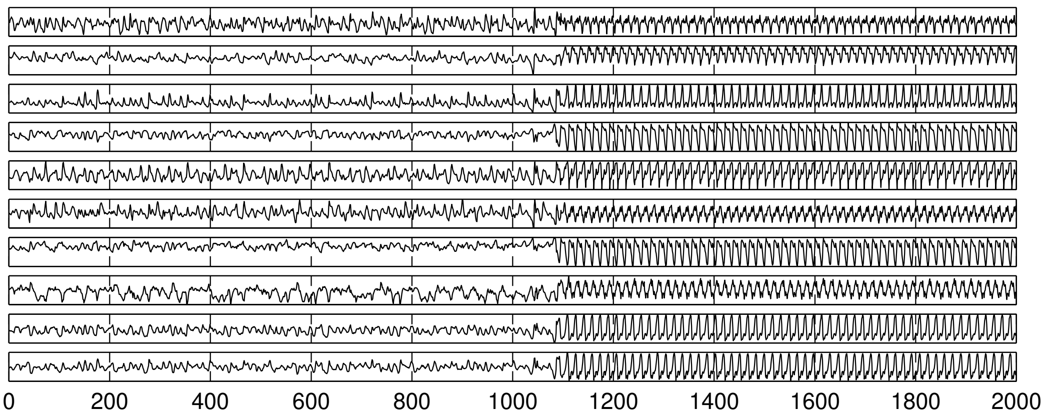


Figure 7: The ten plots show the original observations $\mathbf{x}(t)$ ($1 \leq t \leq 1000$) and the result of prediction made by a nonlinear auto-regressive model ($t \geq 1001$). The noise process was omitted like in the results shown in figures 5 and 6.

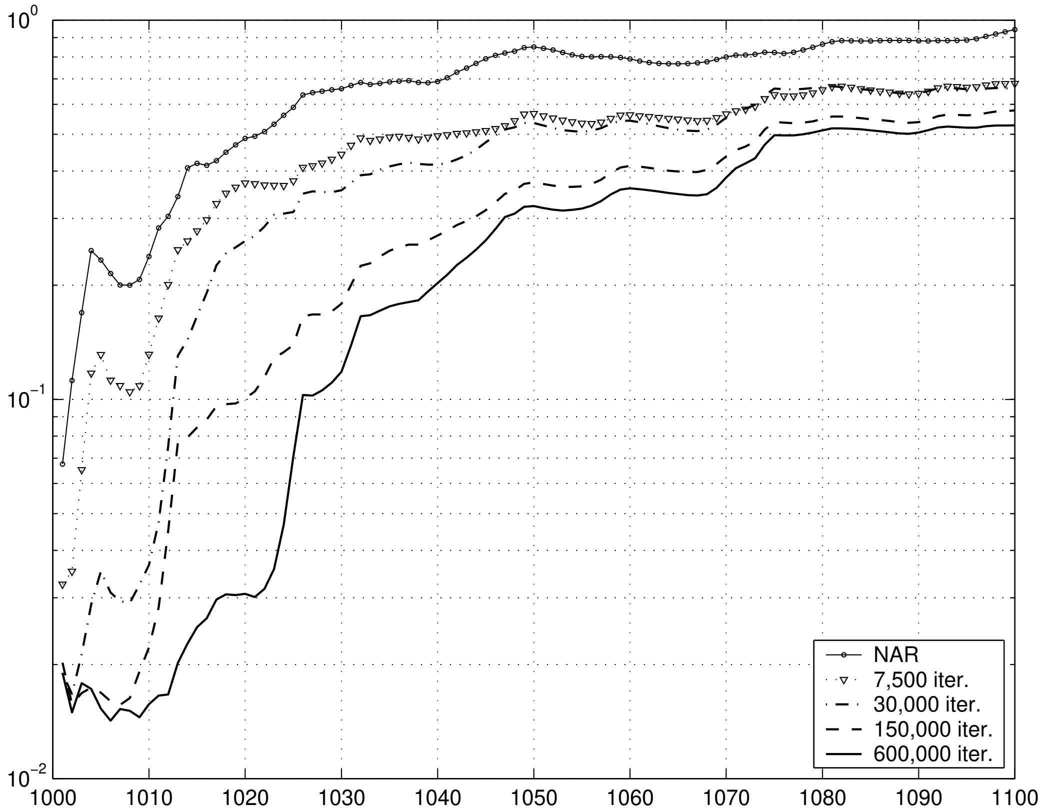


Figure 8: The average cumulative squared prediction error is computed for the predictions made using the NDFA algorithm after 7500 (dotted with triangles), 30,000 (dash-dotted), 150,000 (dashed) and and 600,000 iterations (solid) as well as by a nonlinear auto-regressive model (solid with dots). The predictions are based on 100 Monte-Carlo simulations of the estimated dynamics. Taking into consideration the observation noise whose variance is 0.01, the prediction obtained by the NDFA algorithm in the end of learning is excellent up to $t = 1010$ and fairly good up to $t = 1022$. The NAR model is quite inaccurate already after $t \geq 1003$.

Several strategies were tested and the best performance was given by an MLP network with 20 inputs and one hidden layer of 30 neurons. The inputs to the MLP network were the principal components extracted from the sequence of ten past observations, that is, D was ten. Figure 7 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 good enough for modelling the long-term behaviour of the observations.

Both the NAR model and the model used in NDFA algorithm contain noise process which can be taken into account in the prediction by Monte-Carlo simulation. Figure 8 shows the results obtained by 100 runs of Monte-Carlo simulation. 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 the NDFA algorithm has been able to model the dynamics of the process as 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.

Figures 9–12 show the results of the Monte-Carlo simulations for individual time series. Each plot shows the averages μ and the range $\mu \pm \sigma$ as obtained from the simulations. Here σ stands for the standard deviation. Figure 9 depicts the predicted continuation of the factors $\mathbf{s}(t)$. In figure 10, these are mapped to the original state space while figures 11 and 12 show the predicted observations obtained by the NDFA algorithm and NAR model, respectively, together with the true continuation.

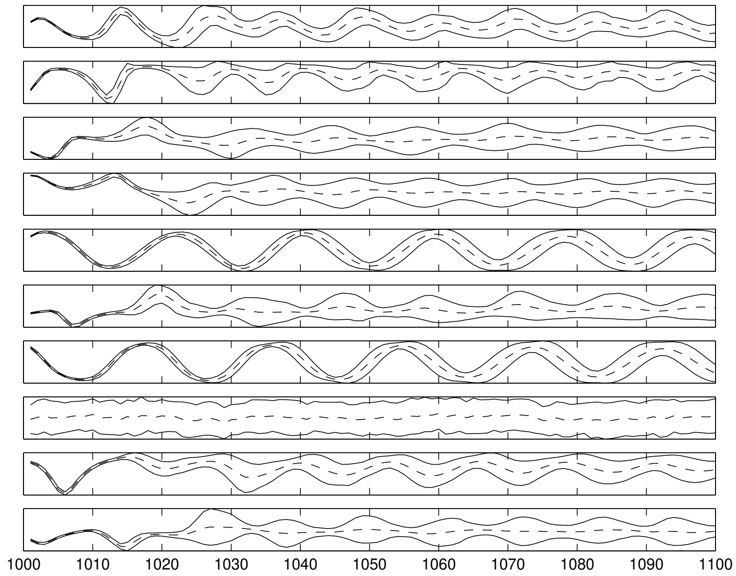


Figure 9: The factors $\mathbf{s}(t)$ have been predicted by Monte-Carlo simulations. The mean μ (dashed) and range $\mu \pm \sigma$ (solid) obtained from the simulation runs are shown.

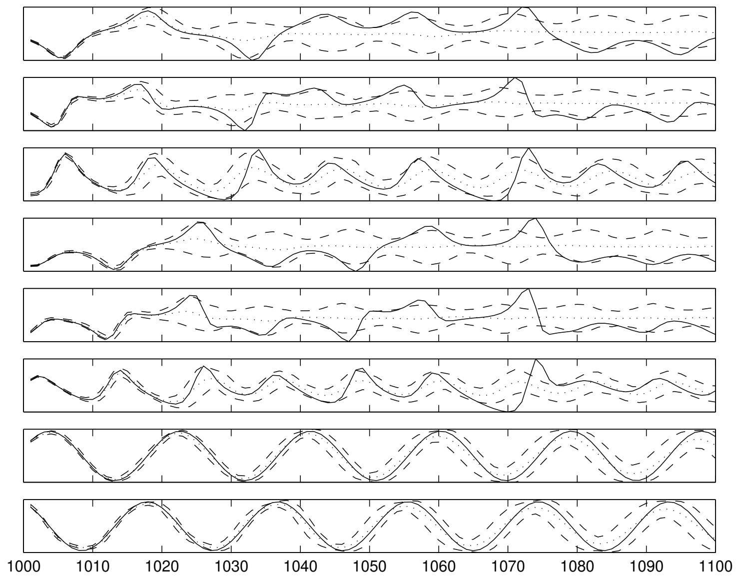


Figure 10: The factors in figure 9 are mapped on the original state space. The mean (dotted) and range (dashed) together with the true continuation (solid) are shown. Due to the chaotic nature of the Lorenz system, the state cannot be predicted for distant future. Notice, however, how the model has been able to predict the timing of the oscillations for the first and fourth states although the sign is uncertain.

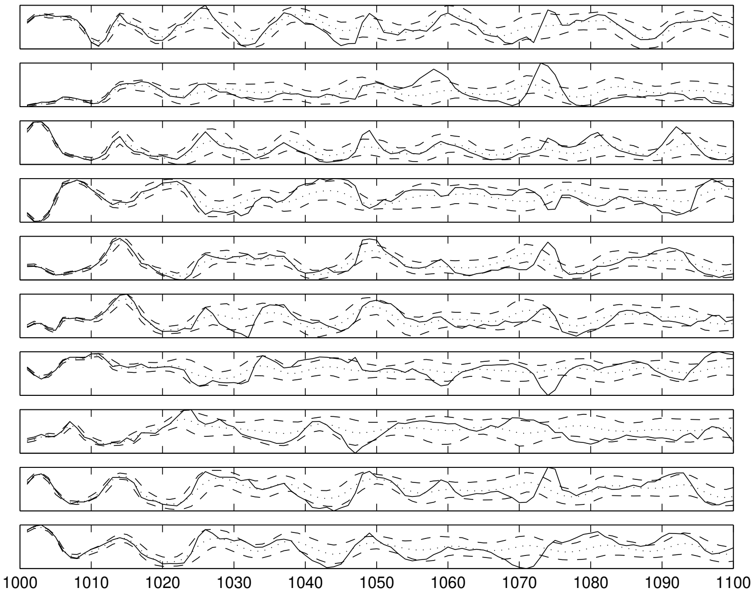


Figure 11: The factors in figure 9 are projected onto the observations by the estimated mapping **f**. The mean (dotted) and range (dashed) together with the true continuation (solid) are shown.

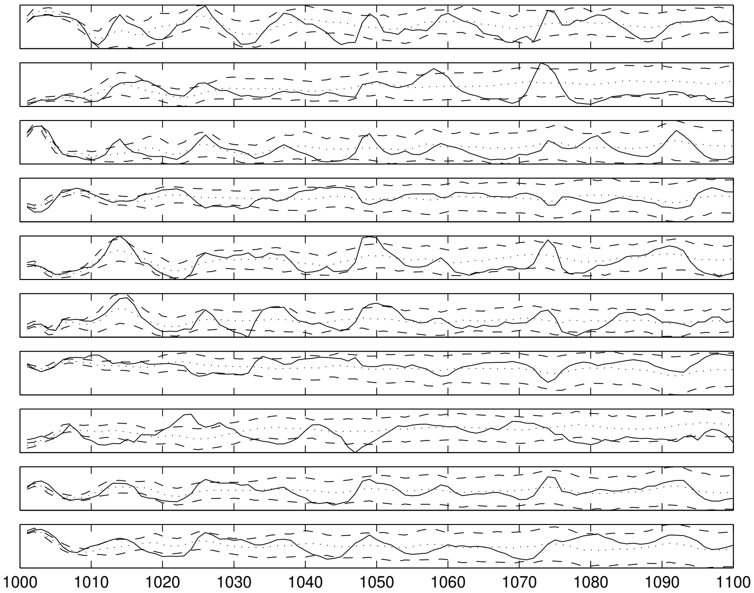


Figure 12: Monte-Carlo simulations with the NAR model have been used for predicting the observations. The mean (dotted) and range (dashed) together with the true continuation (solid) are shown.

5 Discussion

5.1 Independent and weakly coupled processes

The experiments show that the NDFA algorithm is able to extract predictable factors from the observations. The 1000 observations given for the algorithm can span the ten dimensional factor space only sparsely which indicates that the MLP network must have been able to generalise very well. This is possible since the observations were generated by lower-dimensional independent processes and the dynamics of the whole system can thus be expressed as a sum of the individual simple dynamics. This type of a mapping is easy to model with an MLP network.

The innovation process has a Gaussian prior which means that there is a rotational degeneracy in the model as certain rotations map the diagonal Gaussian density onto another diagonal Gaussian density. The rotation can be absorbed in the linear mappings of the MLP networks thus resulting in an equivalent model. However, the factors extracted by the algorithm are clearly very close to the original undelying time series used in generating the data. Each factor can be attributed to one process and none of the factors is a mixture of the states of different elementary processes as would be expected if the algorithm would randomly settle to one of the equivalent rotations of the factor space.

The reason is that the approximation of the posterior probability of the factors cannot represent all the degenerate solutions equally well. The approximation assumes each factor to be independent of the other factors given the observations. The dynamic mapping induces posterior correlations which violate this assumption but the learning algorithm finds the solution which is closest to the assumption. It turns out that this is achieved by separating the underlying processes, thus yielding sparse dynamic couplings between different factors.

5.2 Future directions

In this work, adaptation of the approximation of the posterior density was done in batches, i.e., all the observations were processed before adaptation. It would be straight-forward to derive an on-line version of learning, however. For each new sample the new posterior approximation $q(\boldsymbol{\theta}|\mathbf{X}_{t+1})$ would be adapted by minimising the misfit between the new approximation and $p(\mathbf{x}(t+1)|\boldsymbol{\theta})q(\boldsymbol{\theta}|\mathbf{X}_t)/p(\mathbf{x}(t+1)|\mathbf{X}_t)$. Here $\boldsymbol{\theta}$ denotes all the unknown variables of the model, including factors, parameters of the mappings and hyperparameters. This type of on-line learning would be essentially a version of Kalman filtering [3].

The nonlinear dynamic model used here is universal in the sense that in principle any data generating process can be modelled by it with any given precision. The significance of this property is mostly theoretical since in practice the amount of observations limit the complexity of models which can be reliably learned from the data. The generalisation ability of the models can be improved by prior knowledge about the model structure. This should also make it easier to identify the factors with physical quantities of the observed system.

Like the NLFA algorithm, the proposed NDFA algorithm scales quadratically as a function of the dimension of factor space and several times larger factor spaces than in the experiments reported here are thus computationally feasible. However, with high-dimensional nonlinear models the interpretation of the results can be rather difficult. Due to the approximation made in the posterior, the algorithm was shown to extract factors each of which can be identified with one of the underlying independent processes. This should happen more reliably if the prior model would also promote sparse temporal couplings between the underlying factors, e.g., by using a sparse prior on the weight matrices of the MLP networks. Use of non-Gaussian model for the innovation process $\mathbf{m}(t)$ should also aid at the interpretation of the results in a similar way as non-Gaussian model of factors in linear factor analysis.

Another example of using prior knowledge could be implementation of memories with specific structure. The state representation can in principle learn any memory, but learning complex structures can be difficult in practice. It is also possible that some of the states or control signals of the underlying dynamical process are directly measured and they can then be included in the model as known external inputs.

6 Acknowledgements

This work was funded by the EU project BLISS. The author wishes to acknowledge the contribution of Mr. Juha Reunanen who assisted in doing many of the preliminary simulations which led to this work. The author would also like to thank Dr. J. P. Barnard for suggesting the use of NLFA for extracting the embedded state of a dynamical system.

References

[1] T. Briegel and V. Tresp. Fisher scoring and a mixture of modes approach for approximate inference and learning in nonlinear state space models. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems 11*, pages 403–409, Denver, Colorado, USA, Nov. 30–Dec. 5, 1998, 1999. The MIT Press.

[2] Z. Ghahramani and S. T. Roweis. Learning nonlinear dynamical systems using an EM algorithm. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems 11*, pages 599–605, Denver, Colorado, USA, Nov. 30–Dec. 5, 1998, 1999. The MIT Press.

[3] M. S. Grewal and A. P. Andrews. *Kalman Filtering.* Prentice-Hall, Englewood Cliffs, New Jersey, 1993.

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

[5] E. N. Lorenz. Deterministic nonperiodic flow. *Journal of Atmospheric Sciences*, 20:130–141, 1963.

[6] J. Moody and C. Darken. Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1(2):281–294, 1989.

[7] S. T. Roweis and Z. Ghahramani. An EM algorithm for identification of nonlinear dynamical systems. In S. Haykin, editor, *Kalman Filtering and Neural Networks.* To appear.

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