Helsinki University of Technology
Publications in Computer and Information Science

Report E4
April 2006

BUILDING BLOCKS FOR VARIATIONAL BAYESIAN LEARNING OF LATENT VARIABLE MODELS

Tapani Raiko Harri Valpola Markus Harva Juha Karhunen

TEKNILLINEN KORKEAKOULU
TEKNISKA HÖGSKOLAN
HELSINKI UNIVERSITY OF TECHNOLOGY
TECHNISCHE UNIVERSITÄT HELSINKI
UNIVERSITE DE TECHNOLOGIE D'HELSINKI

Distribution:
Helsinki University of Technology
Department of Computer Science and Engineering
Laboratory of Computer and Information Science
P.O. Box 5400
FI-02015 TKK, Finland
Tel. +358-9-451 3267
Fax +358-9-451 3277

This report is downloadable at
http://www.cis.hut.fi/Publications/

ISBN 951-22-8191-0
ISSN 1796-2803

Building Blocks for Variational Bayesian Learning of Latent Variable Models

Tapani Raiko, Harri Valpola, Markus Harva, and Juha Karhunen
Helsinki University of Technology, Adaptive Informatics Research Centre
P.O.Box 5400, FI-02015 HUT, Espoo, FINLAND
email: `firstname.lastname@tkk.fi`
URL: `http://www.cis.hut.fi/projects/bayes/`
Fax: +358-9-451 3277

April 26, 2006

Abstract

We introduce standardised building blocks designed to be used with variational Bayesian learning. The blocks include Gaussian variables, summation, multiplication, nonlinearity, and delay. A large variety of latent variable models can be constructed from these blocks, including variance models and nonlinear modelling, which are lacking from most existing variational systems. The introduced blocks are designed to fit together and to yield efficient update rules. Practical implementation of various models is easy thanks to an associated software package which derives the learning formulas automatically once a specific model structure has been fixed. Variational Bayesian learning provides a cost function which is used both for updating the variables of the model and for optimising the model structure. All the computations can be carried out locally, resulting in linear computational complexity. We present experimental results on several structures, including a new hierarchical nonlinear model for variances and means. The test results demonstrate the good performance and usefulness of the introduced method.

1 Introduction

Various generative modelling approaches have provided powerful statistical learning methods for neural networks and graphical models during the last years. Such methods aim at finding an appropriate model which explains the internal structure or regularities found in the observations. It is assumed that these regularities are caused by certain latent variables (also called factors, sources, hidden variables, or hidden causes) which have generated the observed data through an unknown mapping [9]. In unsupervised learning, the goal is to identify both the unknown latent variables and generative mapping, while in supervised learning it suffices to estimate the generative mapping.

The expectation-maximisation (EM) algorithm has often been used for learning latent variable models [8, 9, 39, 41]. The distribution for the latent variables is modelled, but the model parameters are found using maximum likelihood or maximum a posteriori estimators. However, with such point estimates, determination of the correct model order and overfitting are ubiquitous and often difficult problems. Therefore, full Bayesian approaches making use of the complete posterior distribution have recently gained a lot of attention. Exact treatment of the posterior distribution is intractable except in simple toy problems, and hence one must resort to suitable approximations. So-called Laplacian approximation method [46, 8] employs a Gaussian approximation around the peak of the posterior distribution. However, this method still suffers from overfitting. In real-world problems, it often does not perform adequately, and has therefore largely given way for better alternatives. Among them, Markov Chain Monte Carlo (MCMC) techniques [49, 51, 56] are popular in supervised learning tasks, providing good estimation results. Unfortunately, the computational load is high, which restricts the use of MCMC in large scale unsupervised learning problems where the parameters and variables to be estimated are numerous. For instance, [68] has a case study in unsupervised learning from brain imaging data. He used MCMC for a scaled down toy example but resorted to point estimates with real data.

Ensemble learning [26, 47, 51, 5, 45], which is one of the variational Bayesian methods [40, 3, 41], has gained increasing attention during the last years. This is because it largely avoids overfitting, allows for estimation of the model order and structure, and its computational load is reasonable compared to the MCMC methods. Variational Bayesian learning was first employed in supervised problems [80, 26, 47, 5], but it has now become popular also in unsupervised modelling. Recently, several authors have successfully applied such techniques to linear factor analysis, independent component analysis (ICA) [36, 66, 12, 27], and their various extensions. These include linear independent factor analysis [2], several other extensions of the basic linear ICA model [4, 11, 53, 67], as well as MLP networks for modelling nonlinear observation mappings [44, 36] and nonlinear dynamics of the latent variables (source signals) [38, 74, 75]. Variational Bayesian learning has also been applied to large discrete models [54] such as nonlinear belief networks [15] and hidden Markov models [48].

In this paper, we introduce a small number of basic blocks for building latent variable models which are learned using variational Bayesian learning. The blocks have been introduced earlier in two conference papers [77, 23] and their applications in [76, 33, 30, 65, 62, 63]. [71] studied hierarchical models for variance sources from signal-processing point of view. This paper is the first comprehensive presentation about the block framework itself. Our approach is most suitable for unsupervised learning tasks which are considered in this paper, but in principle at least, it could be applied to supervised learning, too. A wide variety of factor-analysis-type latent-variable models can be constructed by combining the basic blocks suitably. Variational Bayesian learning then provides a cost function which can be used for updating the variables as well as for optimising the model structure. The blocks are designed so as to fit together and yield efficient update rules. By using a maximally factorial posterior approximation, all the required computations can be performed locally. This results in linear computational complexity as a function of the number of connections in the model. The Bayes Blocks software package by [72] is an open-source C++/Python implementation that can freely be downloaded.

The basic building block is a Gaussian variable (node). It uses as its input values both mean and variance. The other building blocks include addition and multiplication nodes, delay, and a Gaussian variable followed by a nonlinearity. Several known model structures can be constructed using these blocks. We also introduce some novel model structures by extending known linear structures using nonlinearities and variance modelling. Examples will be presented later on in this paper.

The key idea behind developing these blocks is that after the connections between the blocks in the chosen model have been fixed (that is, a particular model has been selected and specified), the cost function and the updating rules needed in learning can be computed automatically. The user does not need to understand the underlying mathematics since the derivations are done within the software package. This allows for rapid prototyping. The Bayes Blocks can also be used to bring different methods into a unified framework, by implementing a corresponding structure from blocks and by using results of these methods for initialisation. Different methods can then be compared directly using the cost function and perhaps combined to find even better models. Updates that minimise a global cost function are guaranteed to converge, unlike algorithms such as loopy belief propagation [60], extended Kalman smoothing [1], or expectation propagation [52].

[81] have introduced a general purpose algorithm called variational message passing. It resembles our framework in that it uses variational Bayesian learning and factorised approximations. The VIBES framework allows for discrete variables but not nonlinearities or nonstationary variance. The posterior approximation does not need to be fully factorised which leads to a more accurate model. Optimisation proceeds by cycling through each factor and revising the approximate posterior distribution. Messages that contain certain expectations over the posterior approximation are sent through the network.

[7, 17], and [6] view variational Bayesian learning as an extension to

the EM algorithm. Their algorithms apply to combinations of discrete and linear Gaussian models. In the experiments, the variational Bayesian model structure selection outperformed the Bayesian information criterion [69] at relatively small computational cost, while being more reliable than annealed importance sampling even with the number of samples so high that the computational cost is hundredfold.

A major difference of our approach compared to the related methods by [81] and by [7] is that they concentrate mainly on situations where there is a handy conjugate prior [16] of the posterior distributions available. This makes life easier, but on the other hand our blocks can be combined more freely, allowing richer model structures. For instance, the modelling of variance in a way described in Section 5.1, would not be possible using the gamma distribution for the precision parameter in the Gaussian node. The price we have to pay for this advantage is that the minimum of the cost function must be found iteratively, while it can be solved analytically when conjugate distributions are applied. The cost function can always be evaluated analytically in the Bayes Blocks framework as well. Note that the different approaches would fit together.

Similar graphical models can be learned with sampling based algorithms instead of variational Bayesian learning. For instance, the BUGS software package by [70] uses Gibbs sampling for Bayesian inference. It supports mixture models, nonlinearities, and nonstationary variance. There are also many software packages concentrated on discrete Bayesian networks. Notably, the Bayes Net toolbox by [55] can be used for Bayesian learning and inference of many types of directed graphical models using several methods. It also includes decision-theoretic nodes. Hence it is in this sense more general than our work. A limitation of the Bayes net toolbox [55] is that it supports latent continuous nodes only with Gaussian or conditional Gaussian distributions.

Autobayes [20] is a system that generates code for efficient implementations of algorithms used in Bayes networks. Currently the algorithm schemas include EM, k-means, and discrete model selection. This system does not yet support continuous hidden variables, nonlinearities, variational methods, MCMC, or temporal models. One of the greatest strengths of the code generation approach compared to a software library is the possibility of automatically optimising the code using domain information.

In the independent component analysis community, traditionally, the observation noise has not been modelled in any way. Even when it is modelled, the noise variance is assumed to have a constant value which is estimated from the available observations when required. However, more flexible variance models would be highly desirable in a wide variety of situations. It is well-known that many real-world signals or data sets are nonstationary, being roughly stationary on fairly short intervals only. Quite often the amplitude level of a signal varies markedly as a function of time or position, which means that its variance is nonstationary. Examples include financial data sets, speech signals, and natural images.

Recently, [59] have demonstrated that several higher-order statistical properties of natural images and signals are well explained by a stochastic model in which an otherwise stationary Gaussian process has a nonstationary variance. Variance models are also useful in explaining volatility of financial time series and in detecting outliers in the data. By utilising the nonstationarity of variance it is possible to perform blind source separation on certain conditions [36, 61].

Several authors have introduced hierarchical models related to those discussed in this paper. These models use subspaces of dependent features instead of single feature components. This kind of models have been proposed at least in context with independent component analysis [10, 35, 34, 58], and topographic or self-organising maps [43, 18]. A problem with these methods is that it is difficult to learn the structure of the model or to compare different model structures.

The remainder of this paper is organised as follows. In the following section, we briefly present basic concepts of variational Bayesian learning. In Section 3, we introduce the building blocks (nodes), and in Section 4 we discuss variational Bayesian computations with them. In the next section, we show examples of different types of models which can be constructed using the building blocks. Section 6 deals with learning and potential problems related with it, and in Section 7 we present experimental results on several structures given in Section 5. This is followed by a short discussion as well as conclusions in the last section of the paper.

2 Variational Bayesian learning

In Bayesian data analysis and estimation methods [51, 16, 39, 56], all the uncertain quantities are modelled 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 contains all the relevant information on the unknown variables and parameters.

Denote by $\boldsymbol{\theta}$ the set of all model parameters and other unknown variables that we wish to estimate from a given data set $\boldsymbol{X}$. The posterior probability density $p(\boldsymbol{\theta}|\boldsymbol{X})$ of the parameters $\boldsymbol{\theta}$ given the data $\boldsymbol{X}$ is obtained from Bayes rule[1]

$$p(\boldsymbol{\theta}|\boldsymbol{X}) = \frac{p(\boldsymbol{X}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\boldsymbol{X})} \tag{1}$$

Here $p(\boldsymbol{X}|\boldsymbol{\theta})$ is the likelihood of the parameters $\boldsymbol{\theta}$ given the data $\boldsymbol{X}$, $p(\boldsymbol{\theta})$ is the prior pdf of these parameters, and

$$p(\boldsymbol{X}) = \int_{\boldsymbol{\theta}} p(\boldsymbol{X}|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta} \tag{2}$$

is a normalising term which is called the evidence. The evidence can be directly understood as the marginal probability of the observed data $\boldsymbol{X}$ assuming the chosen model $\mathcal{H}$. By evaluating the evidences $p(\boldsymbol{X})$ for different models $\mathcal{H}_i$, one can therefore choose the model which describes the observed data with the highest probability[2] [8, 51].

Variational Bayesian learning [5, 26, 45, 47, 51] is a fairly recently introduced [26, 47] approximate fully Bayesian method, which has become popular because of its good properties. Its key idea is to approximate the exact posterior distribution $p(\boldsymbol{\theta}|\boldsymbol{X})$ by another distribution $q(\boldsymbol{\theta})$ that is computationally easier to handle. The approximating distribution is usually chosen to be a product of several independent distributions, one for each parameter or a set of similar parameters.

Variational Bayesian learning employs the Kullback-Leibler (KL) information (divergence) between two probability distributions $q(v)$ and $p(v)$. The KL information is defined by the cost function [24]

$$\mathcal{J}_{\mathrm{KL}}(q \parallel p) = \int_v q(v) \ln \frac{q(v)}{p(v)} dv \tag{3}$$

which measures the difference in the probability mass between the densities $q(v)$ and $p(v)$. Its minimum value 0 is achieved when the densities $q(v)$ and $p(v)$ are the same.

The KL information is used to minimise the misfit between the actual posterior pdf $p(\boldsymbol{\theta}|\boldsymbol{X})$ and its parametric approximation $q(\boldsymbol{\theta})$. However, the exact KL information $\mathcal{J}_{\mathrm{KL}}(q(\boldsymbol{\theta}) \parallel p(\boldsymbol{\theta}|\boldsymbol{X}))$ between these two densities does not yet yield a practical cost function, because the normalising term $p(\boldsymbol{X})$ needed in computing $p(\boldsymbol{\theta}|\boldsymbol{X})$ cannot usually be evaluated.

Therefore, the cost function used in variational Bayesian learning is defined [26, 47]

$$\mathcal{C}_{\mathrm{KL}} = \mathcal{J}_{\mathrm{KL}}(q(\boldsymbol{\theta}) \parallel p(\boldsymbol{\theta}|\boldsymbol{X})) - \ln p(\boldsymbol{X}) \tag{4}$$

[1]The subscripts of all pdf's are assumed to be the same as their arguments, and are omitted for keeping the notation simpler.

[2]More accurately, one could show the dependence on the chosen model $\mathcal{H}$ by conditioning all the pdf's in (1) by $\mathcal{H}$: $p(\boldsymbol{\theta}|\boldsymbol{X}, \mathcal{H})$, $p(\boldsymbol{X}|\mathcal{H})$, etc. We have here dropped also the dependence on $\mathcal{H}$ out for notational simplicity. See [74] for a somewhat more complete discussion of Bayesian methods and ensemble learning.

After slight manipulation, this yields

$$\mathcal{C}_{\mathrm{KL}} = \int_{\boldsymbol{\theta}} q(\boldsymbol{\theta}) \ln \frac{q(\boldsymbol{\theta})}{p(\boldsymbol{X}, \boldsymbol{\theta})} d\boldsymbol{\theta} \tag{5}$$

which does not require $p(\boldsymbol{X})$ any more. The cost function $\mathcal{C}_{\mathrm{KL}}$ consists of two parts:

$$\mathcal{C}_q = \langle \ln q(\boldsymbol{\theta}) \rangle = \int_{\boldsymbol{\theta}} q(\boldsymbol{\theta}) \ln q(\boldsymbol{\theta}) d\boldsymbol{\theta} \tag{6}$$

$$\mathcal{C}_p = \langle - \ln p(\boldsymbol{X}, \boldsymbol{\theta}) \rangle = - \int_{\boldsymbol{\theta}} q(\boldsymbol{\theta}) \ln p(\boldsymbol{X}, \boldsymbol{\theta}) d\boldsymbol{\theta} \tag{7}$$

where the shorthand notation $\langle \cdot \rangle$ denotes expectation with respect to the approximate pdf $q(\boldsymbol{\theta})$.

In addition, the cost function $\mathcal{C}_{\mathrm{KL}}$ provides a bound for the evidence $p(\boldsymbol{X})$. Since $\mathcal{J}_{\mathrm{KL}}(q \parallel p)$ is always nonnegative, it follows directly from (4) that

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

This shows that the negative of the cost function bounds the log-evidence from below.

It is worth noting that variational Bayesian ensemble learning can be derived from information-theoretic minimum description length coding as well [26]. Further considerations on such arguments, helping to understand several common problems and certain aspects of learning, have been presented in a recent paper [31].

The dependency structure between the parameters in our method is the same as in Bayesian networks [60]. Variables are seen as nodes of a graph. Each variable is conditioned by its parents. The difficult part in the cost function is the expectation $\langle \ln p(\boldsymbol{X}, \boldsymbol{\theta}) \rangle$ which is computed over the approximation $q(\boldsymbol{\theta})$ of the posterior pdf. The logarithm splits the product of simple terms into a sum. If each of the simple terms can be computed in constant time, the overall computational complexity is linear.

In general, the computation time is constant if the parents are independent in the posterior pdf approximation $q(\boldsymbol{\theta})$. This condition is satisfied if the joint distribution of the parents in $q(\boldsymbol{\theta})$ decouples into the product of the approximate distributions of the parents. That is, each term in $q(\boldsymbol{\theta})$ depending on the parents depends only on one parent. The independence requirement is violated if any variable receives inputs from a latent variable through multiple paths or from two latent variables which are dependent in $q(\boldsymbol{\theta})$, having a non-factorisable joint distribution there. Figure 1 illustrates the flow of information in the network in these two qualitatively different cases.

Our choice for $q(\boldsymbol{\theta})$ is a multivariate Gaussian density with a diagonal covariance matrix. Even this crude approximation is adequate for finding

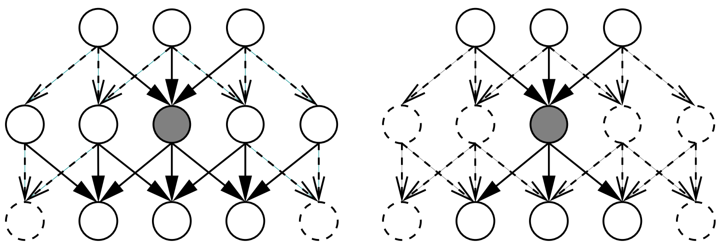

Figure 1: The dash-lined nodes and connections can be ignored while updating the shadowed node. *Left:* In general, the whole Markov blanket needs to be considered. *Right:* A completely factorial posterior approximation with no multiple computational paths leads to a decoupled problem. The nodes can be updated locally.

the region where the mass of the actual posterior density is concentrated. The mean values of the components of the Gaussian approximation provide reasonably good point estimates of the corresponding parameters or variables, and the respective variances measure the reliability of these estimates. However, occasionally the diagonal Gaussian approximation can be too crude. This problem has been considered in context with independent component analysis in [37], giving means to remedy the situation.

Taking into account posterior dependencies makes the posterior pdf approximation $q(\boldsymbol{\theta})$ more accurate, but also usually increases the computational load significantly. We have earlier considered networks with multiple computational paths in several papers, for example [44, 73, 74, 75]. The computational load of variational Bayesian learning then becomes roughly quadratically proportional to the number of unknown variables in the MLP network model used in [44, 75, 32].

The building blocks (nodes) introduced in this paper together with associated structural constraints provide effective means for combating the drawbacks mentioned above. Using them, updating at each node takes place locally with no multiple paths. As a result, the computational load scales linearly with the number of estimated quantities. The cost function and the learning formulas for the unknown quantities to be estimated can be evaluated automatically once a specific model has been selected, that is, after the connections between the blocks used in the model have been fixed. This is a very important advantage of the proposed block approach.

3 Node types

In this section, we present different types of nodes that can be easily combined together. Variational Bayesian inference algorithm for the nodes is then discussed in Section 4.

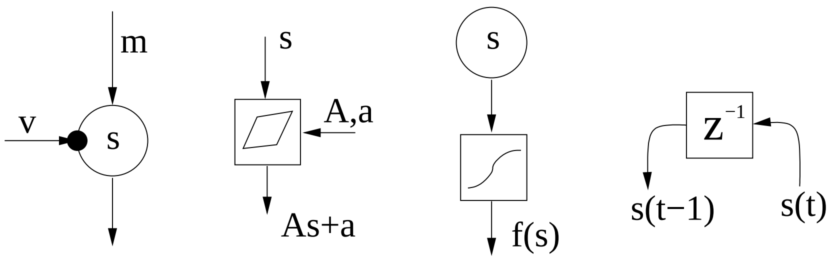

Figure 2: *First subfigure from the left:* The circle represents a Gaussian node corresponding to the latent variable s conditioned by mean m and variance $\exp(-v)$. *Second subfigure:* Addition and multiplication nodes are used to form an affine mapping from s to $As + a$. *Third subfigure:* A nonlinearity f is applied immediately after a Gaussian variable. *The rightmost subfigure:* Delay operator delays a time-dependent signal by one time unit.

In general, the building blocks can be divided into variable nodes, computation nodes, and constants. Each variable node corresponds to a random variable, and it can be either observed or hidden. In this paper we present only one type of variable node, the Gaussian node, but others can be used in the same framework. The computation nodes are the addition node, the multiplication node, a nonlinearity, and the delay node.

In the following, we shall refer to the inputs and outputs of the nodes. For a variable node, its inputs are the parameters of the conditional distribution of the variable represented by that node, and its output is the value of the variable. For computation nodes, the output is a fixed function of the inputs. The symbols used for various nodes are shown in Figure 2. Addition and multiplication nodes are not included, since they are typically combined to represent the effect of a linear transformation, which has a symbol of its own. An output signal of a node can be used as input by zero or more nodes that are called the children of that node. Constants are the only nodes that do not have inputs. The output is a fixed value determined at creation of the node.

Nodes are often structured in vectors or matrices. Assume for example that we have a data matrix $\mathbf{X} = [\mathbf{x}(1), \mathbf{x}(2), \dots, \mathbf{x}(T)]$, where $t = 1, 2, \dots T$ is called the time index of an n-dimensional observation vector. Note that t does not have to correspond to time in the real world, e.g. different t could point to different people. In the implementation, the nodes are either vectors so that the values indexed by t (e.g. observations) or scalars so that the values are constants w.r.t. t (e.g. weights). The data $\mathbf{X}$ would be represented with n vector nodes. A scalar node can be a parent of a vector node, but not a child of a vector node.

3.1 Gaussian node

The Gaussian node is a variable node and the basic element in building hierarchical models. Figure 2 (leftmost subfigure) shows the schematic diagram of the Gaussian node. Its output is the value of a Gaussian random variable s, which is conditioned by the inputs m and v. Denote generally by $\mathcal{N}(x; m_x, \sigma_x^2)$ the probability density function of a Gaussian random variable x having the mean m_x and variance σ_x^2. Then the conditional probability function (cpf) of the variable s is $p(s \mid m, v) = \mathcal{N}(s; m, \exp(-v))$. As a generative model, the Gaussian node takes its mean input m and adds to it Gaussian noise (or innovation) with variance $\exp(-v)$.

Variables can be latent or observed. Observing a variable means fixing its output s to the value in the data. Section 4 is devoted to inferring the distribution over the latent variables given the observed variables. Inferring the distribution over variables that are independent of t is also called learning.

3.2 Computation nodes

The addition and multiplication nodes are used for summing and multiplying variables. These standard mathematical operations are typically used to construct linear mappings between the variables. This task is automated in the software, but in general, the nodes can be connected in other ways, too. An addition node that has n inputs denoted by $s_1, s_2, \ldots, s_n$, gives the sum of its inputs as the output $\sum_{i=1}^n s_i$. Similarly, the output of a multiplication node is the product of its inputs $\prod_{i=1}^n s_i$.

A nonlinear computation node can be used for constructing nonlinear mappings between the variable nodes. The nonlinearity

$$f(s) = \exp(-s^2) \tag{9}$$

is chosen because the required expectations can be solved analytically for it. Another implemented nonlinearity for which the computations can be carried out analytically is the cut function $g(s) = \max(s, 0)$. Other possible nonlinearities are discussed in Section 4.3.

3.3 Delay node

The delay operation can be used to model dynamics. The node operates on time-dependent signals. It transforms the inputs $s(1), s(2), \ldots, s(T)$ into outputs $s_0, s(1), s(2), \ldots, s(T-1)$ where s_0 is a scalar parameter that provides a starting distribution for the dynamical process. The symbol z^{-1} in the rightmost subfigure of Fig. 2 illustrating the delay node is the standard notation for the unit delay in signal processing and temporal neural networks [24]. Models containing the delay node are called dynamic, and the other models are called static.

4 Variational Bayesian inference in Bayes blocks

In this section we give equations needed for computation with the nodes introduced in Section 3. Generally speaking, each node propagates to the forward direction a distribution of its output given its inputs. In the backward direction, the dependency of the cost function (5) of the children on the output of their parent is propagated. These two potentials are combined to form the posterior distribution of each variable. There is a direct analogy to Bayes rule (1): the prior (forward) and the likelihood (backward) are combined to form the posterior distribution. We will show later on that the potentials in the two directions are fully determined by a few values, which consist of certain expectations over the distribution in the forward direction, and of gradients of the cost function w.r.t. the same expectations in the backward direction.

In the following, we discuss in more detail the properties of each node. Note that the delay node does actually not process the signals, it just rewires them. Therefore no formulas are needed for its associated the expectations and gradients.

4.1 Gaussian node

Recall the Gaussian node in Section 3.1. The variance is parameterised using the exponential function as $\exp(-v)$. This is because then the mean $\langle v \rangle$ and expected exponential $\langle \exp v \rangle$ of the input v suffice for evaluating the cost function, as will be shown shortly. Consequently the cost function can be minimised using the gradients with respect to these expectations. The gradients are computed backwards from the children nodes, but otherwise our learning method differs clearly from standard back-propagation [24].

Another important reason for using the parameterisation $\exp(-v)$ for the prior variance of a Gaussian random variable s is that the posterior distribution of s then becomes approximately Gaussian, provided that the prior mean m of s is Gaussian, too (see for example Section 7.1 or [45]). The conjugate prior distribution of the inverse of the prior variance of a Gaussian random variable is the gamma distribution [16]. Using such gamma prior pdf causes the posterior distribution to be gamma, too, which is mathematically convenient. However, the conjugate prior pdf of the second parameter of the gamma distribution is something quite intractable. Hence gamma distribution is not suitable for developing hierarchical variance models. The logarithm of a gamma distributed variable is approximately Gaussian distributed [16], justifying the adopted parameterisation $\exp(-v)$. However, it should be noted that both the gamma and $\exp(-v)$ distributions are used as prior pdfs mainly because they make the estimation of the posterior pdf mathematically tractable [45]; one cannot claim that either of these choices would be correct.

4.1.1 Cost function

Recall now that we are approximating the joint posterior pdf of the random variables s, m, and v in a maximally factorial manner. It then decouples into the product of the individual distributions: $q(s,m,v) = q(s)q(m)q(v)$. Hence s, m, and v are assumed to be statistically independent a posteriori. The posterior approximation $q(s)$ of the Gaussian variable s is defined to be Gaussian with mean $\overline{s}$ and variance $\widetilde{s}$: $q(s) = \mathcal{N}(s; \overline{s}, \widetilde{s})$. Utilising these, the part $\mathcal{C}_p$ of the Kullback-Leibler cost function arising from the data, defined in Eq. (7), can be computed in closed form. For the Gaussian node of Figure 2, the cost becomes

$$\begin{aligned}\mathcal{C}_{s,p} &= -\langle \ln p(s|m,v)\rangle \\ &= \frac{1}{2}\Big\{ \langle \exp v\rangle \Big[(\overline{s} - \langle m\rangle)^2 + \mathrm{Var}\{m\} + \widetilde{s}\Big] - \langle v\rangle + \ln 2\pi \Big\} \end{aligned} \qquad (10)$$

The derivation is presented in Appendix B of [74] using slightly different notation. For the observed variables, this is the only term arising from them to the cost function $\mathcal{C}_{\mathrm{KL}}$.

However, latent variables contribute to the cost function $\mathcal{C}_{\mathrm{KL}}$ also with the part $\mathcal{C}_q$ defined in Eq. (6), resulting from the expectation $\langle \ln q(s)\rangle$. This term is

$$\mathcal{C}_{s,q} = \int_s q(s) \ln q(s) ds = -\frac{1}{2}[\ln(2\pi\widetilde{s}) + 1] \qquad (11)$$

which is the negative entropy of Gaussian variable with variance $\widetilde{s}$. The parameters defining the approximation $q(s)$ of the posterior distribution of s, namely its mean $\overline{s}$ and variance $\widetilde{s}$, are to be optimised during learning.

The output of a latent Gaussian node trivially provides the mean and the variance: $\langle s\rangle = \overline{s}$ and $\mathrm{Var}\{s\} = \widetilde{s}$. The expected exponential can be easily shown to be [45, 74]

$$\langle \exp s\rangle = \exp(\overline{s} + \widetilde{s}/2) \qquad (12)$$

The outputs of the nodes corresponding to the observations are known scalar values instead of distributions. Therefore for these nodes $\langle s\rangle = s$, $\mathrm{Var}\{s\} = 0$, and $\langle \exp s\rangle = \exp s$. An important conclusion of the considerations presented this far is that the cost function of a Gaussian node can be computed analytically in a closed form. This requires that the posterior approximation is Gaussian and that the mean $\langle m\rangle$ and the variance $\mathrm{Var}\{m\}$ of the mean input m as well as the mean $\langle v\rangle$ and the expected exponential $\langle \exp v\rangle$ of the variance input v can be computed. To summarise, we have shown that Gaussian nodes can be connected together and their costs can be evaluated analytically.

We will later on use derivatives of the cost function with respect to some expectations of its mean and variance parents m and v as messages from children to parents. They are derived directly from Eq. (10), taking the form

$$\frac{\partial \mathcal{C}_{s,p}}{\partial \langle m \rangle} = \langle \exp v \rangle (\langle m \rangle - \overline{s}) \tag{13}$$

$$\frac{\partial \mathcal{C}_{s,p}}{\partial \mathrm{Var}\{m\}} = \frac{\langle \exp v \rangle}{2} \tag{14}$$

$$\frac{\partial \mathcal{C}_{s,p}}{\partial \langle v \rangle} = -\frac{1}{2} \tag{15}$$

$$\frac{\partial \mathcal{C}_{s,p}}{\partial \langle \exp v \rangle} = \frac{(\overline{s} - \langle m \rangle)^2 + \mathrm{Var}\{m\} + \widetilde{s}}{2}. \tag{16}$$

4.1.2 Updating the posterior distribution

The posterior distribution $q(s)$ of a latent Gaussian node can be updated as follows.

1. The distribution $q(s)$ affects the terms of the cost function $\mathcal{C}_s$ arising from the variable s itself, namely $\mathcal{C}_{s,p}$ and $\mathcal{C}_{s,q}$, as well as the $\mathcal{C}_p$ terms of the children of s, denoted by $\mathcal{C}_{\mathrm{ch}(s),p}$. The gradients of the cost $\mathcal{C}_{\mathrm{ch}(s),p}$ with respect to $\langle s \rangle$, $\mathrm{Var}\{s\}$, and $\langle \exp s \rangle$ are computed according to Equations (13–16).

2. The terms in $\mathcal{C}_p$ which depend on $\overline{s}$ and $\widetilde{s}$ can be shown (see Appendix B.2) to be of the form [3]

$$\mathcal{C}_p = \mathcal{C}_{s,p} + \mathcal{C}_{\mathrm{ch}(s),p} = M\overline{s} + V[(\overline{s} - \overline{s}_{\mathrm{current}})^2 + \widetilde{s}] + E \langle \exp s \rangle , \tag{17}$$

   where
$$M = \frac{\partial \mathcal{C}_p}{\partial \overline{s}} , \qquad V = \frac{\partial \mathcal{C}_p}{\partial \widetilde{s}} , \qquad \text{and } E = \frac{\partial \mathcal{C}_p}{\partial \langle \exp s \rangle}. \tag{18}$$

3. The minimum of $\mathcal{C}_s = \mathcal{C}_{s,p} + \mathcal{C}_{s,q} + \mathcal{C}_{\mathrm{ch}(s),p}$ is solved. This can be done analytically if $E = 0$, corresponding to the case of so-called free-form solution (see [45] for details):

$$\overline{s}_{\mathrm{opt}} = \overline{s}_{\mathrm{current}} - \frac{M}{2V} , \qquad \widetilde{s}_{\mathrm{opt}} = \frac{1}{2V}. \tag{19}$$

   Otherwise the minimum is obtained iteratively. Iterative minimisation can be carried out efficiently using Newton's method for the posterior mean $\overline{s}$ and a fixed-point iteration for the posterior variance $\widetilde{s}$. The minimisation procedure is discussed in more detail in Appendix A.

[3]Note that constants are dropped out since they do not depend on $\overline{s}$ or $\widetilde{s}$.

4.2 Addition and multiplication nodes

Consider first the addition node. The mean, variance and expected exponential of the output of the addition node can be evaluated in a straightforward way. Assuming that the inputs s_i are statistically independent, these expectations are respectively given by

$$\left\langle \sum_{i=1}^{n} s_i \right\rangle = \sum_{i=1}^{n} \langle s_i \rangle \tag{20}$$

$$\operatorname{Var}\left\{ \sum_{i=1}^{n} s_i \right\} = \sum_{i=1}^{n} \operatorname{Var}\{s_i\} \tag{21}$$

$$\left\langle \exp\left(\sum_{i=1}^{n} s_i \right) \right\rangle = \prod_{i=1}^{n} \langle \exp s_i \rangle \tag{22}$$

The proof has been given in Appendix B.1.

Consider then the multiplication node. Assuming independence between the inputs s_i, the mean and the variance of the output take the form (see Appendix B.1)

$$\left\langle \prod_{i=1}^{n} s_i \right\rangle = \prod_{i=1}^{n} \langle s_i \rangle \tag{23}$$

$$\operatorname{Var}\left\{ \prod_{i=1}^{n} s_i \right\} = \prod_{i=1}^{n} \left[\langle s_i \rangle^2 + \operatorname{Var}\{s_i\} \right] - \prod_{i=1}^{n} \langle s_i \rangle^2 \tag{24}$$

For the multiplication node the expected exponential cannot be evaluated without knowing the exact distribution of the inputs.

The formulas (20)–(24) are given for n inputs because of generality, but in practice we have carried out the needed calculations pairwise. When using the general formula (24), the variance might otherwise occasionally take a small negative value due to minor imprecisions appearing in the computations. This problem does not arise in pairwise computations. Now, the propagation in the forward direction is covered.

The form of the cost function propagating from children to parents is assumed to be of the form (17). This is true even in the case, where there are addition and multiplication nodes in between (see Appendix B.2 for proof). Therefore only the gradients of the cost function with respect to the different expectations need to be propagated backwards to identify the whole cost function w.r.t. the parent. The required formulas are obtained in a straightforward manner from Eqs. (20)–(24). The gradients for the addition node are:

$$\frac{\partial C}{\partial \langle s_1 \rangle} = \frac{\partial C}{\partial \langle s_1 + s_2 \rangle} \tag{25}$$

$$\frac{\partial C}{\partial \mathrm{Var}\{s_1\}} = \frac{\partial C}{\partial \mathrm{Var}\{s_1 + s_2\}} \tag{26}$$

$$\frac{\partial C}{\partial \langle \exp s_1 \rangle} = \langle \exp s_2 \rangle \frac{\partial C}{\partial \langle \exp(s_1 + s_2) \rangle}. \tag{27}$$

For the multiplication node, they become

$$\frac{\partial C}{\partial \langle s_1 \rangle} = \langle s_2 \rangle \frac{\partial C}{\partial \langle s_1 s_2 \rangle} + 2\mathrm{Var}\{s_2\} \frac{\partial C}{\partial \mathrm{Var}\{s_1 s_2\}} \langle s_1 \rangle \tag{28}$$

$$\frac{\partial C}{\partial \mathrm{Var}\{s_1\}} = \left(\langle s_2 \rangle^2 + \mathrm{Var}\{s_2\} \right) \frac{\partial C}{\partial \mathrm{Var}\{s_1 s_2\}}. \tag{29}$$

As a conclusion, addition and multiplication nodes can be added between the Gaussian nodes whose costs still retain the form (17). Proofs can be found in Appendices B.1 and B.2.

4.3 Nonlinearity node

A serious problem arising here is that for most nonlinear functions it is impossible to compute the required expectations analytically. Here we describe a particular nonlinearity in detail and discuss the options for extending to other nonlinearities, for which the implementation is underway.

Ghahramani and Roweis have shown [19] that for the nonlinear function $f(s) = \exp(-s^2)$ in Eq. (9), the mean and variance have analytical expressions, to be presented shortly, provided that it has Gaussian input. In our graphical network structures this condition is fulfilled if we require that the nonlinearity must be inserted immediately after a Gaussian node. The same type of exponential function (9) is frequently used in standard radial-basis function networks [8, 24, 19], but in a different manner. There the exponential function depends on the Euclidean distance from a center point, while in our case it depends on the input variable s directly.

The first and second moments of the function (9) with respect to the distribution $q(s)$ are [19]

$$\langle f(s) \rangle = \exp\left(-\frac{\overline{s}^2}{2\widetilde{s} + 1} \right) (2\widetilde{s} + 1)^{-\frac{1}{2}} \tag{30}$$

$$\left\langle [f(s)]^2 \right\rangle = \exp\left(-\frac{2\overline{s}^2}{4\widetilde{s} + 1} \right) (4\widetilde{s} + 1)^{-\frac{1}{2}} \tag{31}$$

The formula (30) provides directly the mean $\langle f(s) \rangle$, and the variance is obtained from (30) and (31) by applying the familiar formula $\mathrm{Var}\{f(s)\} = \left\langle [f(s)]^2 \right\rangle - \langle f(s) \rangle^2$. The expected exponential $\langle \exp f(s) \rangle$ cannot be evaluated analytically, which limits somewhat the use of the nonlinear node.

The updating of the nonlinear node following directly a Gaussian node takes place similarly as the updating of a plain Gaussian node. The gradients of $\mathcal{C}_p$ with respect to $\langle f(s) \rangle$ and $\mathrm{Var}\{f(s)\}$ are evaluated assuming that they arise from a quadratic term. This assumption holds since the nonlinearity can only propagate to the mean of Gaussian nodes. The update formulas are given in Appendix C.

Another possibility is to use as the nonlinearity the error function $f(s) = \int_{-\infty}^{s} \exp(-r^2) dr$, because its mean can be evaluated analytically and variance approximated from above [15]. Increasing the variance increases the value of the cost function, too, and hence it suffices to minimise the upper bound of the cost function for finding a good solution. [15] apply the error function in MLP (multilayer perceptron) networks [8, 24] but in a manner different from ours.

Finally, [54] has applied the hyperbolic tangent function $f(s) = \tanh(s)$, approximating it iteratively with a Gaussian. [32] approximate the same sigmoidal function with a Gauss-Hermite quadrature. This alternative could be considered here, too. A problem with it is, however, that the cost function (mean and variance) cannot be computed analytically.

4.4 Other possible nodes

One of the authors has recently implemented two new variable nodes [21, 23] into the Bayes Blocks software library. They are the mixture-of-Gaussians (MoG) node and the rectified Gaussian node. MoG can be used to model any sufficiently well behaving distribution [8]. In the independent factor analysis (IFA) method introduced in [2], a MoG distribution was used for the sources, resulting in a probabilistic version of independent component analysis (ICA) [36].

The second new node type, the rectified Gaussian variable, was introduced in [53]. By omitting negative values and retaining only positive ones of a variable which is originally Gaussian distributed, this block allows modelling of variables having positive values only. Such variables are commonplace for example in digital image processing, where the picture elements (pixels) have always non-negative values. The cost functions and update rules of the MoG and rectified Gaussian node have been derived in [21]. We postpone a more detailed discussion of these nodes to forthcoming papers to keep the length of this paper reasonable.

In the early conference paper [77] where we introduced the blocks for the first time, two more blocks were proposed for handling discrete models and variables. One of them is a switch, which picks up its k-th continuous valued input signal as its output signal. The other one is a discrete variable k, which has a soft-max prior derived from the continuous valued input signals c_i of the node. However, we have omitted these two nodes from the present paper, because their performance has not turned out to be adequate. The reason might be that assuming all parents of all nodes independent is too restrictive. For instance, building a mixture-of-Gaussians from discrete and Gaussian variables with switches is possible, but the construction loses

Node type	$\langle\cdot\rangle$	$\mathrm{Var}\{\cdot\}$	$\langle\exp\cdot\rangle$
Gaussian node	$\overline{s}$	$\widetilde{s}$	(12)
Addition node	(20)	(21)	(22)
Multiplication node	(23)	(24)	-
Nonlinearity	(30)	(30),(31)	-
Constant	c	0	$\exp c$

Table 1: The forward messages or expectations that are provided by the output of different types of nodes. The numbers in parentheses refer to defining equations. The multiplication and nonlinearity cannot provide the expected exponential.

Input type	$\frac{\partial\mathcal{C}}{\partial\langle\cdot\rangle}$	$\frac{\partial\mathcal{C}}{\partial\mathrm{Var}\{\cdot\}}$	$\frac{\partial\mathcal{C}}{\partial\langle\exp\cdot\rangle}$
Mean of a Gaussian node	(13)	(14)	0
Variance of a Gaussian node	(15)	0	(16)
Addendum	(25)	(26)	(27)
Factor	(28)	(29)	0

Table 2: The backward messages or the gradients of the cost function w.r.t. certain expectations. The numbers in parentheses refer to defining equations. The gradients of the Gaussian node are derived from Eq. (10). The Gaussian node requires the corresponding expectations from its inputs, that is, $\langle m\rangle$, $\mathrm{Var}\{m\}$, $\langle v\rangle$, and $\langle\exp v\rangle$. Addition and multiplication nodes require the same type of input expectations that they are required to provide as output. Communication of a nonlinearity with its Gaussian parent node is described in Appendix C.

out to a specialised MoG node that makes fewer assumptions. In [63], the discrete node is used without switches.

Action and utility nodes [60, 55] would extend the library into decision theory and control. In addition to the messages about the variational Bayesian cost function, the network would propagate messages about utility. [64] describe such a system in a slightly different framework.

5 Combining the nodes

The expectations provided by the outputs and required by the inputs of the different nodes are summarised in Tables 1 and 2, respectively. One can see that the variance input of a Gaussian node requires the expected exponential of the incoming signal. However, it cannot be computed for the nonlinear and multiplication nodes. Hence all the nodes cannot be combined freely.

When connecting the nodes, the following restrictions must be taken into account:

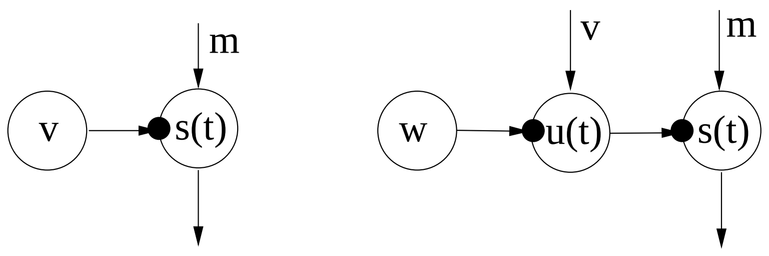

Figure 3: *Left:* The Gaussian variable $s(t)$ has a a constant variance $\exp(-v)$ and mean m. *Right:* A variance source is added for providing a non-constant variance input $u(t)$ to the output (source) signal $s(t)$. The variance source $u(t)$ has a prior mean v and prior variance $\exp(-w)$.

1. In general, the network has to be a directed acyclic graph (DAG). The delay nodes are an exception because the past values of any node can be the parents of any other nodes. This violation is not a real one in the sense that if the structure were unfolded in time, the resulting network would again be a DAG.

2. The nonlinearity must always be placed immediately after a Gaussian node. This is because the output expectations, Equations (30) and (31), can be computed only for Gaussian inputs. The nonlinearity also breaks the general form of the likelihood (17). This is handled by using special update rules for the Gaussian followed by a nonlinearity (Appendix C).

3. The outputs of multiplication and nonlinear nodes cannot be used as variance inputs for the Gaussian node. This is because the expected exponential cannot be evaluated for them. These restrictions are evident from Tables 1 and 2.

4. There should be only one computational path from a latent variable to a variable. Otherwise, the independency assumptions used in Equations (10) and (21)–(24) are violated and variational Bayesian learning becomes more complicated (recall Figure 1).

Note that the network may contain loops, that is, the underlying undirected network can be cyclic. Note also that the second, third, and fourth restrictions can be circumvented by inserting mediating Gaussian nodes. A mediating Gaussian node that is used as the variance input of another variable, is called the variance source and it is discussed in the following.

5.1 Nonstationary variance

In most currently used models, only the means of Gaussian nodes have hierarchical or dynamical models. In many real-world situations the variance

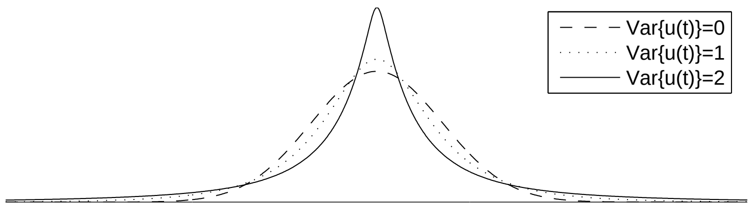

Figure 4: The distribution of $s(t)$ is plotted when $s(t) \sim \mathcal{N}(0, \exp[-u(t)])$ and $u(t) \sim \mathcal{N}(0, \cdot)$. Note that when $\mathrm{Var}\,\{u(t)\} = 0$, the distribution of $s(t)$ is Gaussian. This corresponds to the right subfigure of Fig. 3 when $m = v = 0$ and $\exp(-w) = 0, 1, 2$.

is not a constant either but it is more difficult to model it. For modelling the variance, too, we use the variance source [71] depicted schematically in Figure 3. Variance source is a regular Gaussian node whose output $u(t)$ is used as the input variance of another Gaussian node. Variance source can convert prediction of the mean into prediction of the variance, allowing to build hierarchical or dynamical models for the variance.

The output $s(t)$ of a Gaussian node to which the variance source is attached (see the right subfigure of Fig. 3) has in general a super-Gaussian distribution. Such a distribution is typically characterised by long tails and a high peak, and it is formally defined as having a positive value of kurtosis (see [36] for a detailed discussion). This property has been proved for example in [59], where it is shown that a nonstationary variance (amplitude) always increases the kurtosis. The output signal $s(t)$ of the stationary Gaussian variance source depicted in the left subfigure of Fig. 3 is naturally Gaussian distributed with zero kurtosis. The variance source is useful in modelling natural signals such as speech and images which are typically super-Gaussian, and also in modelling outliers in the observations.

5.2 Linear independent factor analysis

In many instances there exist several nodes which have quite similar role in the chosen structure. Assuming that i^{th} such node corresponds to a scalar variable y_i, it is convenient to use the vector $\mathbf{y} = (y_1, y_2, \ldots, y_n)^T$ to jointly denote all the corresponding scalar variables $y_1, y_2, \ldots, y_n$. This notation is used in Figures 5 and 6 later on. Hence we represent the scalar source nodes corresponding to the variables $s_i(t)$ using the source vector $\mathbf{s}(t)$, and the scalar nodes corresponding to the observations $x_i(t)$ using the observation vector $\mathbf{x}(t)$.

The addition and multiplication nodes can be used for building an affine

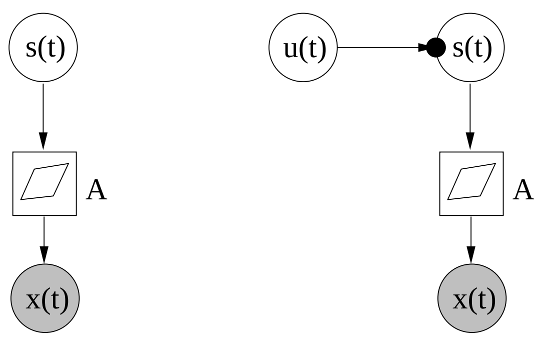

Figure 5: Model structures for linear factor analysis (FA) (left) and independent factor analysis (IFA) (right).

transformation

$$\mathbf{x}(t) = \mathbf{A}\mathbf{s}(t) + \mathbf{a} + \mathbf{n}_x(t) \tag{32}$$

from the Gaussian source nodes $\mathbf{s}(t)$ to the Gaussian observation nodes $\mathbf{x}(t)$. The vector $\mathbf{a}$ denotes the bias and vector $\mathbf{n}_x(t)$ denotes the zero-mean Gaussian noise in the Gaussian node $\mathbf{x}(t)$. This model corresponds to standard linear factor analysis (FA) assuming that the sources $s_i(t)$ are mutually uncorrelated; see for example [36].

If instead of Gaussianity it is assumed that each source $s_i(t)$ has some non-Gaussian prior, the model (32) describes linear independent factor analysis (IFA). Linear IFA was introduced by [2], who used variational Bayesian learning for estimating the model except for some parts which he estimated using the expectation-maximisation (EM) algorithm. Attias used a mixture-of-Gaussians source model, but another option is to use the variance source to achieve a super-Gaussian source model. Figure 5 depicts the model structures for linear factor analysis and independent factor analysis.

5.3 A hierarchical variance model

Figure 6 (right subfigure) presents a hierarchical model for the variance, and also shows how it can be constructed by first learning simpler structures shown in the left and middle subfigures of Fig. 6. This is necessary, because learning a hierarchical model having different types of nodes from scratch in a completely unsupervised manner would be too demanding a task, ending quite probably into an unsatisfactory local minimum.

The final rightmost variance model in Fig. 6 is somewhat involved in that it contains both nonlinearities and hierarchical modelling of variances. Before going into its mathematical details and into the two simpler models in Fig. 6, we point out that we have considered in our earlier papers related but simpler block models. In [76], a hierarchical nonlinear model for the data $\mathbf{x}(t)$ is discussed without modelling the variance. Such a model can be

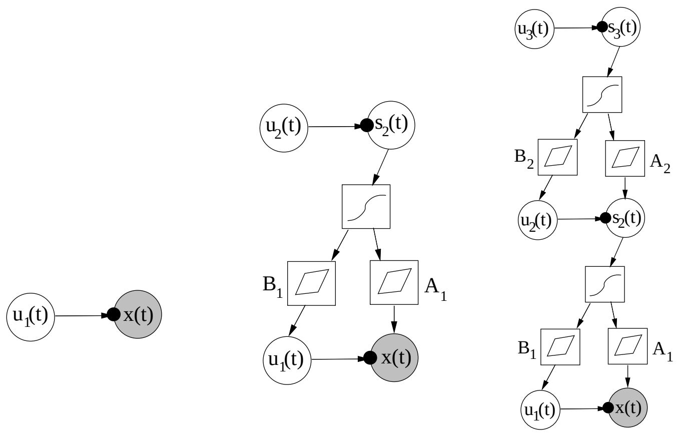

Figure 6: Construction of a hierarchical variance model in stages from simpler models. *Left:* In the beginning, a variance source is attached to each Gaussian observation node. The nodes represent vectors. *Middle:* A layer of sources with variance sources attached to them is added. They layers are connected through a nonlinearity and an affine mapping. *Right:* Another layer is added on the top to form the final hierarchical variance model.

applied for example to nonlinear ICA or blind source separation. Experimental results [76] show that this block model performs adequately in the nonlinear BSS problem, even though the results are slightly poorer than for our earlier computationally more demanding model [44, 75, 32] with multiple computational paths.

In another paper [71], we have considered hierarchical modelling of variance using the block approach without nonlinearities. Experimental results on biomedical MEG (magnetoencephalography) data demonstrate the usefulness of hierarchical modelling of variances and existence of variance sources in real-world data.

Learning starts from the simple structure shown in the left subfigure of Fig. 6. There a variance source is attached to each Gaussian observation node. The nodes represent vectors, with $\mathbf{u}_1(t)$ being the output vector of the variance source and $\mathbf{x}(t)$ the t^{th} observation (data) vector. The vectors $\mathbf{u}_1(t)$ and $\mathbf{x}(t)$ have the same dimension, and each component of the variance vector $\mathbf{u}_1(t)$ models the variance of the respective component of the observation vector $\mathbf{x}(t)$.

Mathematically, this simple first model obeys the equations

$$\mathbf{x}(t) = \mathbf{a}_1 + \mathbf{n}_x(t) \tag{33}$$

$$\mathbf{u}_1(t) = \mathbf{b}_1 + \mathbf{n}_{u_1}(t) \tag{34}$$

Here the vectors $\mathbf{a}_1$ and $\mathbf{b}_1$ denote the constant means (bias terms) of the data vector $\mathbf{x}(t)$ and the variance variable vector $\mathbf{u}_1(t)$, respectively. The additive "noise" vector $\mathbf{n}_x(t)$ determines the variances of the components of $\mathbf{x}(t)$. It has a Gaussian distribution with a zero mean and variance $\exp[-\mathbf{u}_1(t)]$:

$$\mathbf{n}_x(t) \sim \mathcal{N}(\mathbf{0}, \exp[-\mathbf{u}_1(t)]) \tag{35}$$

More precisely, the shorthand notation $\mathcal{N}(\mathbf{0}, \exp[-\mathbf{u}_1(t)])$ means that each component of $\mathbf{n}_x(t)$ is Gaussian distributed with a zero mean and variance defined by the respective component of the vector $\exp[-\mathbf{u}_1(t)]$. The exponential function $\exp(\cdot)$ is applied separately to each component of the vector $-\mathbf{u}_1(t)$. Similarly,

$$\mathbf{n}_{u_1}(t) \sim \mathcal{N}(\mathbf{0}, \exp[-\mathbf{v}_1]) \tag{36}$$

where the components of the vector $\mathbf{v}_1$ define the variances of the zero mean Gaussian variables $\mathbf{n}_{u_1}(t)$.

Consider then the intermediate model shown in the middle subfigure of Fig. 6. In this second learning stage, a layer of sources with variance sources attached to them is added. These sources are represented by the source vector $\mathbf{s}_2(t)$, and their variances are given by the respective components of the variance vector $\mathbf{u}_2(t)$ quite similarly as in the left subfigure. The (vector) node between the source vector $\mathbf{s}_2(t)$ and the variance vector $\mathbf{u}_1(t)$ represents an affine transformation with a transformation matrix $\mathbf{A}_1$ including a bias term. Hence the prior mean inputted to the Gaussian variance source having the output $\mathbf{u}_1(t)$ is of the form $\mathbf{B}_1\mathbf{f}(\mathbf{s}_2(t)) + \mathbf{b}_1$, where $\mathbf{b}_1$ is the bias vector, and $\mathbf{f}(\cdot)$ is a vector of componentwise nonlinear functions (9). Quite similarly, the vector node between $\mathbf{s}_2(t)$ and the observation vector $\mathbf{x}(t)$ yields as its output the affine transformation $\mathbf{A}_1\mathbf{f}(\mathbf{s}_2(t)) + \mathbf{a}_1$, where $\mathbf{a}_1$ is a bias vector. This in turn provides the input prior mean to the Gaussian node modelling the observation vector $\mathbf{x}(t)$.

The mathematical equations corresponding to the model represented graphically in the middle subfigure of Fig. 6 are:

$$\mathbf{x}(t) = \mathbf{A}_1\mathbf{f}(\mathbf{s}_2(t)) + \mathbf{a}_1 + \mathbf{n}_x(t) \tag{37}$$
$$\mathbf{u}_1(t) = \mathbf{B}_1\mathbf{f}(\mathbf{s}_2(t)) + \mathbf{b}_1 + \mathbf{n}_{u_1}(t) \tag{38}$$
$$\mathbf{s}_2(t) = \mathbf{a}_2 + \mathbf{n}_{s_2}(t) \tag{39}$$
$$\mathbf{u}_2(t) = \mathbf{b}_2 + \mathbf{n}_{u_2}(t) \tag{40}$$

Compared with the simplest model (33)–(34), one can observe that the source vector $\mathbf{s}_2(t)$ of the second (upper) layer and the associated variance vector $\mathbf{u}_2(t)$ are of quite similar form, given in Eqs. (39)–(40). The models (37)–(38) of the data vector $\mathbf{x}(t)$ and the associated variance vector $\mathbf{u}_1(t)$ in the first (bottom) layer differ from the simple first model (33)–(34) in that they contain additional terms $\mathbf{A}_1\mathbf{f}(\mathbf{s}_2(t))$ and $\mathbf{B}_1\mathbf{f}(\mathbf{s}_2(t))$, respectively. In these terms, the nonlinear transformation $\mathbf{f}(\mathbf{s}_2(t))$ of the source vector $\mathbf{s}_2(t)$ coming from the upper layer have been multiplied by the linear mixing matrices $\mathbf{A}_1$ and $\mathbf{B}_1$. All the “noise” terms $\mathbf{n}_x(t)$, $\mathbf{n}_{u_1}(t)$, $\mathbf{n}_{s_2}(t)$, and $\mathbf{n}_{u_2}(t)$

in Eqs. (37)–(40) are modelled by similar zero mean Gaussian distributions as in Eqs. (35) and (36).

In the last stage of learning, another layer is added on the top of the network shown in the middle subfigure of Fig. 6. The resulting structure is shown in the right subfigure. The added new layer is quite similar as the layer added in the second stage. The prior variances represented by the vector $\mathbf{u}_3(t)$ model the source vector $\mathbf{s}_3(t)$, which is turn affects via the affine transformation $\mathbf{B}_2\mathbf{f}(\mathbf{s}_3(t)) + \mathbf{b}_2$ to the mean of the mediating variance node $\mathbf{u}_2(t)$. The source vector $\mathbf{s}_3(t)$ provides also the prior mean of the source $\mathbf{s}_2(t)$ via the affine transformation $\mathbf{A}_2\mathbf{f}(\mathbf{s}_3(t)) + \mathbf{a}_2$.

The model equations (37)–(38) for the data vector $\mathbf{x}(t)$ and its associated variance vector $\mathbf{u}_1(t)$ remain the same as in the intermediate model shown graphically in the middle subfigure of Fig. 6. The model equations of the second and third layer sources $\mathbf{s}_2(t)$ and $\mathbf{s}_3(t)$ as well as their respective variance vectors $\mathbf{u}_2(t)$ and $\mathbf{u}_3(t)$ in the rightmost subfigure of Fig. 6 are given by

$$\mathbf{s}_2(t) = \mathbf{A}_2\mathbf{f}(\mathbf{s}_3(t)) + \mathbf{a}_2 + \mathbf{n}_{s_2}(t) \tag{41}$$

$$\mathbf{u}_2(t) = \mathbf{B}_2\mathbf{f}(\mathbf{s}_3(t)) + \mathbf{b}_2 + \mathbf{n}_{u_2}(t) \tag{42}$$

$$\mathbf{s}_3(t) = \mathbf{a}_3 + \mathbf{n}_{s_3}(t) \tag{43}$$

$$\mathbf{u}_3(t) = \mathbf{b}_3 + \mathbf{n}_{u_3}(t) \tag{44}$$

Again, the vectors $\mathbf{a}_2$, $\mathbf{b}_2$, $\mathbf{a}_3$, and $\mathbf{b}_3$ represent the constant means (biases) in their respective models, and $\mathbf{A}_2$ and $\mathbf{B}_2$ are mixing matrices with matching dimensions. The vectors $\mathbf{n}_{s_2}(t)$, $\mathbf{n}_{u_2}(t)$, $\mathbf{n}_{s_3}(t)$, and $\mathbf{n}_{u_3}(t)$ have similar zero mean Gaussian distributions as in Eqs. (35) and (36).

It should be noted that in the resulting network the number of scalar-valued nodes (size of the layers) can be different for different layers. Additional layers could be appended in the same manner. The final network of the right subfigure in Fig. 6 utilises variance nodes in building a hierarchical model for both the means and variances. Without the variance sources the model would correspond to a nonlinear model with latent variables in the hidden layer. As already mentioned, we have considered such a nonlinear hierarchical model in [76]. Note that computation nodes as hidden nodes would result in multiple paths from the latent variables of the upper layer to the observations. This type of structure was used in [44], and it has a quadratic computational complexity as opposed to linear one of the networks in Figure 6.

5.4 Linear dynamic models for the sources and variances

Sometimes it is useful to complement the linear factor analysis model

$$\mathbf{x}(t) = \mathbf{A}\mathbf{s}(t) + \mathbf{a} + \mathbf{n}_x(t) \tag{45}$$

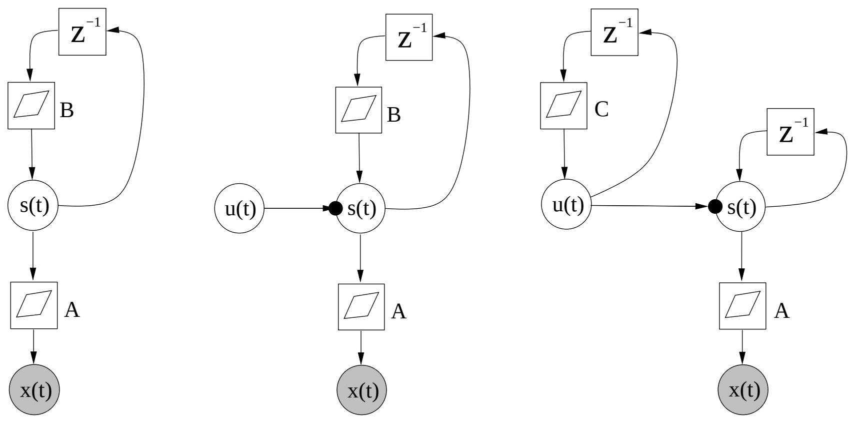

Figure 7: Three model structures. A linear Gaussian state-space model (left); the same model complemented with a super-Gaussian innovation process for the sources (middle); and a dynamic model for the variances of the sources which also have a recurrent dynamic model (right).

with a recursive one-step prediction model for the source vector $\mathbf{s}(t)$:

$$\mathbf{s}(t) = \mathbf{B}\mathbf{s}(t-1) + \mathbf{b} + \mathbf{n}_s(t) \tag{46}$$

The noise term $\mathbf{n}_s(t)$ is called the innovation process. The dynamic model of the type (45), (46) is used for example in Kalman filtering [24, 25], but other estimation algorithms can be applied as well [24]. The left subfigure in Fig. 7 depicts the structure arising from Eqs. (45) and (46), built from the blocks.

A straightforward extension is to use variance sources for the sources to make the innovation process super-Gaussian. The variance signal $\mathbf{u}(t)$ characterises the innovation process of $\mathbf{s}(t)$, in effect telling how much the signal differs from the predicted one but not in which direction it is changing. The graphical model of this extension is depicted in the middle subfigure of Fig. 7. The mathematical equations describing this model can be written in a similar manner as for the hierarchical variance models in the previous subsection.

Another extension is to model the variance sources dynamically by using one-step recursive prediction model for them:

$$\mathbf{u}(t) = \mathbf{C}\mathbf{u}(t-1) + \mathbf{c} + \mathbf{n}_u(t). \tag{47}$$

This model is depicted graphically in the rightmost subfigure of Fig. 7. In context with it, we use the simplest possible identity dynamical mapping for $\mathbf{s}(t)$:

$$\mathbf{s}(t) = \mathbf{s}(t-1) + \mathbf{n}_s(t). \tag{48}$$

The latter two models introduced in this subsection will be tested experimentally later on in this paper.

5.5 Hierarchical priors

It is often desirable that the priors of the parameters should not be too restrictive. A common type of a vague prior is the hierarchical prior [16]. For example the priors of the elements a_{ij} of a mixing matrix $\mathbf{A}$ can be defined via the Gaussian distributions

$$p(a_{ij} \mid v_i^a) = \mathcal{N}(a_{ij}; 0, \exp(-v_i^a)) \tag{49}$$

$$p(v_i^a \mid m^{va}, v^{va}) = \mathcal{N}(v_i^a; m^{va}, \exp(-v^{va})) \,. \tag{50}$$

Finally, the priors of the quantities m^{va} and v^{va} have flat Gaussian distributions $\mathcal{N}(\cdot; 0, 100)$ (the constants depending on the scale of the data). When going up in the hierarchy, we use the same distribution for each column of a matrix and for each component of a vector. On the top, the number of required constant priors is small. Thus very little information is provided and needed a priori. This kind of hierarchical priors are used in the experiments later on this paper.

6 Learning

Let us now discuss the overall learning procedure, describing also briefly how problems related with learning can be handled.

6.1 Updating of the network

The nodes of the network communicate with their parents and children by providing certain expectations in the feedforward direction (from parents to children) and gradients of the cost function with respect to the same expectations in the feedback direction (from children to parents). These expectations and gradients are summarised in Tables 1 and 2.

The basic element for updating the network is the update of a single node assuming the rest of the network fixed. For computation nodes this is simple: each time when a child node asks for expectations and they are out of date, the computational node asks from its parents for their expectations and updates its own ones. And vice versa: when parents ask for gradients and they are out of date, the node asks from its children for the gradients and updates its own ones. These updates have analytical formulas given in Section 4.

For a variable node to be updated, the input expectations and output gradients need to be up-to-date. The posterior approximation $q(s)$ can then be adjusted to minimise the cost function as explained in Section 4. The minimisation is either analytical or iterative, depending on the situation. Signals propagating outwards from the node (the output expectations and the input gradients) of a variable node are functions of $q(s)$ and are thus updated in the process. Each update is guaranteed not to increase the cost function.

One sweep of updating means updating each node once. The order in which this is done is not critical for the system to work. It would not be useful to update a variable twice without updating some of its neighbours in between, but that does not happen with any ordering when updates are done in sweeps. We have used an ordering where each variable node is updated only after all of its descendants have been updated. Basically when a variable node is updated, its input gradients and output expectations are labeled as outdated and they are updated only when another node asks for that information.

It is possible to use different measures to improve the learning process. Measures for avoiding local minima are described in the next subsection. Another enhancement can be used for speeding up learning. The basic idea is that after some time, the parameters describing $q(s)$ are changing fairly linearly between consecutive sweeps. Therefore a line search in that direction provides faster learning, as discussed in [28, 33]. We apply this line search only at every tenth sweep for allowing the consecutive updates to become fairly linear again.

Learning a model typically takes thousands of sweeps before convergence. The cost function decreases monotonically after every update. Typically this decrease gets smaller with time, but not always monotonically. Therefore care should be taken in selecting the stopping criterion. We have chosen to stop the learning process when the decrease in the cost during the previous 200 sweeps is lower than some predefined threshold.

6.2 Structural learning and local minima

The chosen model has a pre-specified structure which, however, has some flexibility. The number of nodes is not fixed in advance, but their optimal number is estimated using variational Bayesian learning, and unnecessary connections can be pruned away.

A factorial posterior approximation, which is used in this paper, often leads to automatic pruning of some of the connections in the model. When there is not enough data to estimate all the parameters, some directions remain ill-determined. This causes the posterior distribution along those directions to be roughly equal to the prior distribution. In variational Bayesian learning with a factorial posterior approximation, the ill-determined directions tend to get aligned with the axes of the parameter space because then the factorial approximation is most accurate.

The pruning tendency makes it easy to use for instance sparsely connected models, because the learning algorithm automatically selects a small amount of well-determined parameters. But at the early stages of learning, pruning can be harmful, because large parts of the model can get pruned away before a sensible representation has been found. This corresponds to the situation where the learning scheme ends up into a local minimum of the cost function [50]. A posterior approximation which takes into account the posterior dependences has the advantage that it has far less local minima than a factorial posterior approximation. It seems that Bayesian learning algorithms which have linear time complexity cannot avoid local minima in general.

However, suitable choices of the model structure and countermeasures included in the learning scheme can alleviate the problem greatly. We have used the following means for avoiding getting stuck into local minima:

- Learning takes place in several stages, starting from simpler structures which are learned first before proceeding to more complicated hierarchic structures. An example of this technique was presented in Section 5.3.

- New parts of the network are initialised appropriately. One can use for instance principal component analysis (PCA), independent component analysis (ICA), vector quantisation, or kernel PCA [29]. The best option depends on the application. Often it is useful to try different methods and select the one providing the smallest value of the cost function for the learned model. There are two ways to handle initialisation: either to fix the sources for a while and learn the weights of the model, or to fix the weights for a while and learn the sources corresponding to the observations. The fixed variables can be released gradually (see Section 5.1 of [76]).

- Automatic pruning is discouraged initially by omitting the term

  $$2\mathrm{Var}\left\{s_2\right\} \frac{\partial C}{\partial \mathrm{Var}\left\{s_1 s_2\right\}} \left\langle s_1 \right\rangle$$

  in the multiplication nodes (Eq. (28)). This effectively means that the mean of s_1 is optimistically adjusted as if there were no uncertainty about s_2. In this way the cost function may increase at first due to overoptimism, but it may pay off later on by escaping early pruning.

- New sources $s_i(t)$ (components of the source vector $\mathbf{s}(t)$ of a layer) are generated, and pruned sources are removed from time to time.

- The activations of the sources are reset a few times. The sources are re-adjusted to their places while keeping the mapping and other parameters fixed. This often helps if some of the sources are stuck into a local minimum.

7 Experimental results

The Bayes Blocks software [72] has been applied to several problems.

[71] considered several models of variance. The main application was the analysis of MEG measurements from a human brain. In addition to features corresponding to brain activity the data contained several artifacts such as muscle activity induced by the patient biting his teeth. Linear ICA applied to the data was able to separate the original causes to some degree but still many dependencies remained between the sources. Hence an additional layer of so-called variance sources was used to find correlations between the variances of the innovation processes of the ordinary sources. These were able to capture phenomena related to the biting artifact as well as to rhythmic activity.

An astrophysical problem of separating young and old star populations from a set of elliptical galaxy spectra has been studied by one of the authors in [57]. Since the observed quantities are energies and thus positive and since the mixing process is also known to be positive, it is necessary for the subsequent astrophysical analysis to be feasible to include these constraints to the model as well. The standard technique of putting a positive prior on the sources was found to have the unfortunate technical shortcoming of inducing sparsely distributed factors, which was deemed inappropriate in that specific application. To get rid of the induced sparsity but to still keep the positivity constraint, the nonnegativity was forced by rectification nonlinearities [22]. In addition to finding an astrophysically meaningful factorisation, several other specifications were needed to be met related to handling of missing values, measurements errors and predictive capabilities of the model.

In [63], a nonlinear model for relational data is applied to the analysis of the boardgame Go. The difficult part of the game state evaluation is to determine which groups of stones are likely to get captured. A model similar to the one that will be described in Section 7.2, is built for features of pairs of groups, including the probability of getting captured. When the learned model is applied to new game states, the estimates propagate through a network of such pairs. The structure of the network is thus determined by the game state. The approach can be used for inference in relational databases.

The following three sets of experiments are given as additional examples. The first one is a difficult toy problem that illustrates hierarchy and variance modelling, the second one studies the inference of missing values in speech spectra, and the third one has a dynamical model for image sequences.

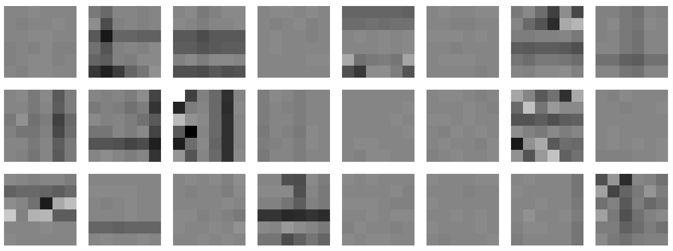

Figure 8: Samples from the 1000 image patches used in the extended bars problem. The bars include both standard and variance bars in horizontal and vertical directions. For instance, the patch at the bottom left corner shows the activation of a standard horizontal bar above the horizontal variance bar in the middle.

7.1 Bars problem

The first experimental problem studied was testing of the hierarchical non-linear variance model in Figure 6 in an extension of the well-known bars problem [13]. The data set consisted of 1000 image patches each having 6×6 pixels. They contained both horizontal and vertical bars. In addition to the regular bars, the problem was extended to include horizontal and vertical variance bars, characterized and manifested by their higher variance. Samples of the image patches used are shown in Figure 8.

The data were generated by first choosing whether vertical, horizontal, both, or neither orientations were active, each with probability 1/4. Whenever an orientation is active, there is a probability 1/3 for a bar in each row or column to be active. For both orientations, there are 6 regular bars, one for each row or column, and 3 variance bars which are 2 rows or columns wide. The intensities (grey level values) of the bars were drawn from a normalised positive exponential distribution having the pdf $p(z) = \exp(-z), z \geq 0$, $p(z) = 0, z < 0$. Regular bars are additive, and variance bars produce additive Gaussian noise having the standard deviation of its intensity. Finally, Gaussian noise with a standard deviation 0.1 was added to each pixel.

The network was built up following the stages shown in Figure 6. It was initialised with a single layer with 36 nodes corresponding to the 36 dimensional data vector. The second layer of 30 nodes was created at the sweep 20, and the third layer of 5 nodes at the sweep 100. After creating a layer only its sources were updated for 10 sweeps, and pruning was discouraged for 50 sweeps. New nodes were added twice, 3 to the second layer and 2 to the third layer, at sweeps 300 and 400. After that, only the sources were updated for 5 sweeps, and pruning was again discouraged for 50 sweeps. The source activations were reset at the sweeps 500, 600 and 700, and only the sources were updated for the next 40 sweeps. Dead nodes were removed every 20 sweeps. The multistage training procedure was designed to avoid suboptimal local solutions, as discussed in Section 6.2.

Figure 9 demonstrates that the algorithm finds a generative model that is quite similar to the generation process. The two sources on the third layer correspond to the horizontal and vertical orientations and the 18 sources on the second layer correspond to the bars. Each element of the weight matrices is depicted as a pixel with the appropriate grey level value in Fig. 9. The pixels of $\mathbf{A}_2$ and $\mathbf{B}_2$ are ordered similarly as the patches of $\mathbf{A}_1$ and $\mathbf{B}_1$, that is, vertical bars on the left and horizontal bars on the right. Regular bars, present in the mixing matrix $\mathbf{A}_1$, are reconstructed accurately, but the variance bars in the mixing matrix $\mathbf{B}_1$ exhibit some noise. The distinction between horizontal and vertical orientations is clearly visible in the mixing matrix $\mathbf{A}_2$.

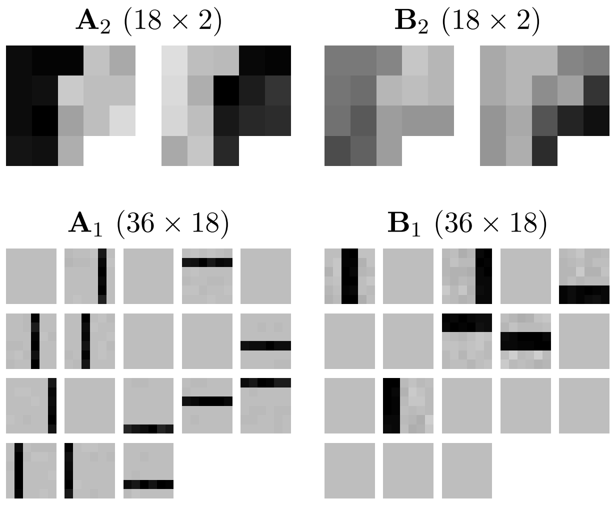

Figure 9: Results of the extended bars problem: Posterior means of the weight matrices after learning. The sources of the second layer have been ordered for visualisation purposes according to the weight (mixing) matrices $\mathbf{A}_2$ and $\mathbf{B}_2$. The elements of the matrices have been depicted as pixels having corresponding grey level values. The 18 pixels in the weight matrices $\mathbf{A}_2$ and $\mathbf{B}_2$ correspond to the 18 patches in the weight matrices $\mathbf{A}_1$ and $\mathbf{B}_1$.

A comparison experiment with a simplified learning procedure was run to demonstrate the importance of local optima. The creation and pruning of layers were done as before, but other methods for avoiding local minima (addition of nodes, discouraging pruning and resetting of sources) were disabled. The resulting weights can be seen in Figure 10. This time the learning ends up in a suboptimal local optimum of the cost function. One

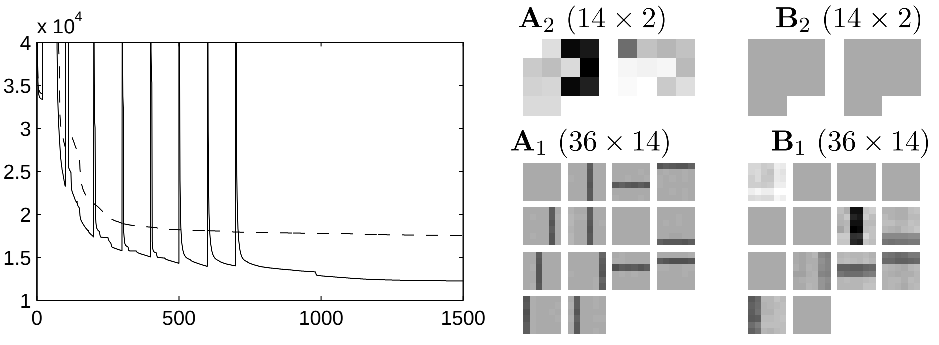

Figure 10: *Left:* Cost function plotted against the number of learning sweeps. Solid curve is the main experiment and the dashed curve is the comparison experiment. The peaks appear when nodes are added. *Right:* The resulting weights in the comparison experiment are plotted like in Fig-

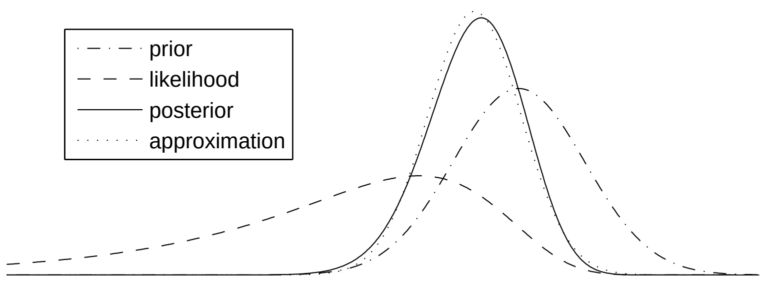

Figure 11: A typical example illustrating the posterior approximation of a variance source.

of the bars was not found (second horizontal bar from the bottom), some were mixed up in a same source (most variance bars share a source with a regular bar), fourth vertical bar from the left appears twice, and one of the sources just suppresses variance everywhere. The resulting cost function (5) is worse by 5292 compared to the main experiment. The ratio of the model evidences is thus roughly exp(5292).

Figure 11 illustrates the formation of the posterior distribution of a typical single variable. It is the first component of the variance source $\mathbf{u}_1(1)$ in the comparison experiment. The prior means here the distribution given its parents (especially $\mathbf{s}_2(1)$ and $\mathbf{B}_1$) and the likelihood means the potential given its children (the first component of $\mathbf{x}(1)$). Assuming the posteriors of other variables accurate, we can plot the true posterior of this variable and compare it to the Gaussian posterior approximation. Their difference is only 0.007 measured by Kullback-Leibler divergence.

7.2 Missing values in speech spectra

In hierarchical nonlinear factor analysis (HNFA) [76], there are a number of layers of Gaussian variables, the bottom-most layer corresponding to the data. There is a nonlinearity and a linear mixture mapping from each layer to all the layers below it.

HNFA resembles the model structure in Section 5.3. The model structure is depicted in the left subfigure of Fig. 12. Model equations are

$$\mathbf{h}(t) = \mathbf{A}\mathbf{s}(t) + \mathbf{a} + \mathbf{n}_h(t) \tag{51}$$

$$\mathbf{x}(t) = \mathbf{B}\boldsymbol{\phi}[\mathbf{h}(t)] + \mathbf{C}\mathbf{s}(t) + \mathbf{b} + \mathbf{n}_x(t)\,, \tag{52}$$

where $\mathbf{n}_h(t)$ and $\mathbf{n}_x(t)$ are Gaussian noise terms and the nonlinearity $\phi(\xi) = \exp(-\xi^2)$ again operates on each element of its argument vector separately. Note that we have included a short-cut mapping $\mathbf{C}$ from sources to observations. This means that hidden nodes only need to model the deviations from linearity.

HNFA is compared against three other methods. Factor analysis (FA) is a linear method described in Section 5.2. It is a special case of HNFA where the dimensionality of $\mathbf{h}(t)$ is zero. Nonlinear factor analysis (NFA) [44, 32] differs from HNFA in that it does not use mediating variables $\mathbf{h}(t)$:

$$\mathbf{x}(t) = \mathbf{B}\tanh[\mathbf{A}\mathbf{s}(t) + \mathbf{a}] + \mathbf{b} + \mathbf{n}_x(t). \tag{53}$$

Note that NFA has multiple computational paths between $\mathbf{s}(t)$ and $\mathbf{x}(t)$, which leads to a higher computational complexity compared to HNFA.

The self-organising map SOM [42] differs most from the other methods. A rectangular map has a number of map units with associated model vectors that are points in the data space. Each data point is matched to the closest map unit. The model vectors of the best-matching unit and its neighbours in the map are moved slightly towards the data point. See [42, 24] for details.

The data set consisted of speech spectrograms from several Finnish subjects. Short term spectra were windowed to 30 dimensions with a standard preprocessing procedure for speech recognition. It is clear that a dynamic source model would give better reconstructions, but in this case the temporal information was left out to ease the comparison of the models. Half of the about 5000 samples were used as test data with some missing values. Missing values were set in four different ways to measure different properties of the algorithms (Figure 13):

1. 38 percent of the values are set to miss randomly in 4×4 patches. (Right subfigure of Figure 12)
2. Training and testing sets are randomly permuted before setting missing values in 4×4 patches as in Setting 1.
3. 10 percent of the values are set to miss randomly independent of any neighbours. This is an easier setting, since simple smoothing using nearby values would give fine reconstructions.

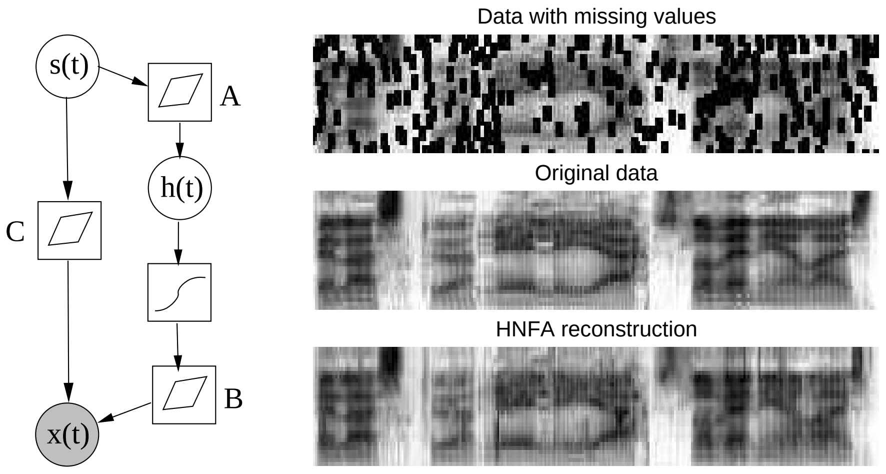

Figure 12: *Left:* The model structure for hierarchical nonlinear factor analysis (HNFA). *Right:* Some speech data with and without missing values (Setting 1) and the reconstruction given by HNFA.

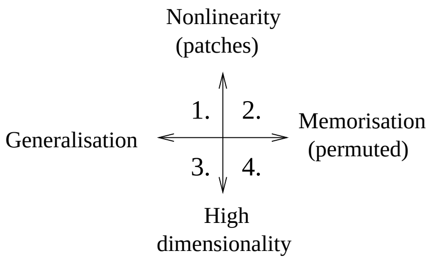

Figure 13: Four different experimental settings with the speech data used for measuring different properties of the algorithms.

4. Training and testing sets are permuted and 10 percent of the values are set to miss independently of any neighbours.

We tried to optimise each method and in the following, we describe how we got the best results. The self-organising map was run using the SOM Toolbox [79] with long learning time, 2500 map units and random initialisations. In other methods, the optimisation was based on minimising the cost function or its approximation. NFA was learned for 5000 sweeps through data using a Matlab implementation. Varying number of sources were tried out and the best ones were used as the result. The optimal number of sources was around 12 to 15 and the size used for the hidden layer was 30. A large enough number should do, since the algorithm can effectively prune out parts that are not needed.

In factor analysis (FA), the number of sources was 28. In hierarchical nonlinear factor analysis (HNFA), the number of sources at the top layer was varied and the best runs according to the cost function were selected. In those runs, the size of the top layer varied from 6 to 12 and the size of the middle layer, which is determined during learning, turned out to vary from 12 to 30. HNFA was run for 5000 sweeps through data. Each experiment with NFA or HNFA took about 8 hours of processor time, while FA and SOM were faster.

Several runs were conducted with different random initialisations but with the same data and the same missing value pattern for each setting and for each method. The number of runs in each cell is about 30 for HNFA, 4 for NFA and 20 for the SOM. FA always converges to the same solution. The mean and the standard deviation of the mean square reconstruction error are:

	FA	HNFA	NFA	SOM
Setting 1	1.87	1.80 ± 0.03	1.74 ± 0.02	1.69 ± 0.02
Setting 2	1.85	1.78 ± 0.03	1.71 ± 0.01	1.55 ± 0.01
Setting 3	0.57	$0.55 \pm .005$	$0.56 \pm .002$	0.86 ± 0.01
Setting 4	0.58	$0.55 \pm .008$	$0.58 \pm .004$	0.87 ± 0.01

The order of results of the Setting 1 follow our expectations on the nonlinearity of the models. The SOM with highest nonlinearity gives the best reconstructions, while NFA, HNFA and finally FA follow in that order. The results of HNFA vary the most - there is potential to develop better learning schemes to find better solutions more often. The sources $\mathbf{h}(t)$ of the hidden layer did not only emulate computation nodes, but they were also active themselves. Avoiding this situation during learning could help to find more nonlinear and thus perhaps better solutions.

In the Setting 2, due to the permutation, the test set contains vectors very similar to some in the training set. Therefore, generalisation is not as important as in the Setting 1. The SOM is able to memorise details corresponding to individual samples better due to its high number of parameters. Compared to the Setting 1, SOM benefits a lot and makes clearly the best reconstructions, while the others benefit only marginally.

The Settings 3 and 4, which require accurate expressive power in high dimensionality, turned out not to differ from each other much. The basic SOM has only two intrinsic dimensions[4] and therefore it was clearly poorer in accuracy. Nonlinear effects were not important in these settings, since HNFA and NFA were only marginally better than FA. HNFA was better than NFA perhaps because it had more latent variables when counting both $\mathbf{s}(t)$ and $\mathbf{h}(t)$.

To conclude, HNFA lies between FA and NFA in performance. HNFA is applicable to high dimensional problems and the middle layer can model

[4]Higher dimensional SOMs become quickly intractable due to exponential number of parameters.

part of the nonlinearity without increasing the computational complexity dramatically. FA is better than SOM when expressivity in high dimensions is important, but SOM is better when nonlinear effects are more important. The extensions of FA, NFA and HNFA, expectedly performed better than FA in each setting. It may be possible to enhance the performance of NFA and HNFA by new learning schemes whereas especially FA is already at its limits. On the other hand, FA is best if low computational complexity is the determining factor.

7.3 Variance model of image sequences

In this section an experiment with a dynamical model for variances applied to image sequence analysis is reported. The motivation behind modelling variances is that in many natural signals, there exists higher order dependencies which are well characterised by correlated variances of the signals [59]. Hence we postulate that we should be able to better catch the dynamics of a video sequence by modelling the variances of the features instead of the features themselves. This indeed is the case as will be shown.

The model considered can be summarised by the following set of equations:

$$\begin{aligned}
\mathbf{x}(t) &\sim \mathcal{N}(\mathbf{A}\mathbf{s}(t), \mathrm{diag}(\exp[-\mathbf{v}_x])) \\
\mathbf{s}(t) &\sim \mathcal{N}(\mathbf{s}(t-1), \mathrm{diag}(\exp[-\mathbf{u}(t)])) \\
\mathbf{u}(t) &\sim \mathcal{N}(\mathbf{B}\mathbf{u}(t-1), \mathrm{diag}(\exp[-\mathbf{v}_u]))
\end{aligned}$$

We will use the acronym DynVar in referring to this model. The linear mapping $\mathbf{A}$ from sources $\mathbf{s}(t)$ to observations $\mathbf{x}(t)$ is constrained to be sparse by assigning each source a circular region on the image patch outside of which no connections are allowed. These regions are still highly overlapping. The variances $\mathbf{u}(t)$ of the innovation process of the sources have a linear dynamical model. It should be noted that modelling the variances of the sources in this manner is impossible if one is restricted to use conjugate priors.

The sparsity of $\mathbf{A}$ is crucial as the computational complexity of the learning algorithm depends on the number of connections from $\mathbf{s}(t)$ to $\mathbf{x}(t)$. The same goal could have been reached with a different kind of approach as well. Instead of constraining the mapping to be sparse from the very beginning of learning it could have been allowed to be full for a number of iterations and only after that pruned based on the cost function as explained in Section 6.2. But as the basis for image sequences tends to get sparse anyway, it is a waste of computational resources to wait while most of the weights in the linear mapping tend to zero.

For comparison purposes, we postulate another model where the dynamical relations are sought directly between the sources leading to the following model equations:

$$\begin{aligned}\mathbf{x}(t) &\sim \mathcal{N}(\mathbf{A}\mathbf{s}(t), \mathrm{diag}(\exp[-\mathbf{v}_x])) \\ \mathbf{s}(t) &\sim \mathcal{N}(\mathbf{B}\mathbf{s}(t-1), \mathrm{diag}(\exp[-\mathbf{u}(t)]))\end{aligned}$$

We shall refer to this model as DynSrc.

The data $\mathbf{x}(t)$ was a video image sequence [78] of dimensions $16 \times 16 \times 4000$. That is, the data consisted of 4000 subsequent digital images of the size 16×16. A part of the data set is shown in Figure 14.

Both models were learned by iterating the learning algorithm 2000 times at which stage a sufficient convergence was attained. The first hint of the superiority of the DynVar model was provided by the difference of the cost between the models which was 28 bits/frame (for the coding interpretation, see [31]). To further evaluate the performance of the models, we considered a simple prediction task where the next frame was predicted based on the previous ones. The predictive distributions, $p(\mathbf{x}(t+1)|\mathbf{x}(1), ..., \mathbf{x}(t))$, for the models can be approximately computed based on the posterior approximation. The means of the predictive distributions are very similar for both of the models. Figure 15 shows the means of the DynVar model for the same sequence as in Figure 14. The means themselves are not very interesting, since they mainly reflect the situation in the previous frame. However, the DynVar model provides also a rich model for the variances. The standard deviations of its predictive distribution are shown in Figure 16. White stands for a large variance and black for a small one. Clearly, the model is able to increase the predicted variance in the area of high motion activity and hence provide better predictions. We can offer quantitative support for this claim by computing the predictive perplexities for the models. Predictive perplexity is widely used in language modelling and it is defined as

$$\mathrm{perplexity}(t) = \exp\left\{ -\frac{1}{256} \sum_{i=1}^{256} \log p(x_i(t+1)|\mathbf{x}(1), ..., \mathbf{x}(t)) \right\}.$$

The predictive perplexities for the same sequence as in Figure 14 are shown in Figure 17. Naturally the predictions get worse when there is movement in the video. However, DynVar model is able to handle it much better than the compared DynSrc model. The same difference can also be directly read by comparing the cost functions (3).

The possible applications for a model of image sequences include video compression, motion detection, early stages of computer vision, and making hypotheses on biological vision.

8 Discussion

One of the distinctive factors between different Bayesian approaches is the type of posterior approximation. We have concentrated on large unsuper-

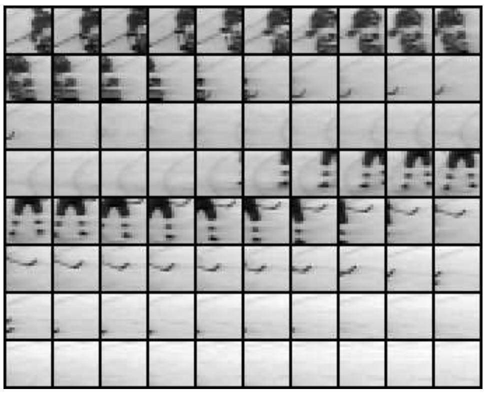
Figure 14: A sequence of 80 frames from the data used in the experiment.

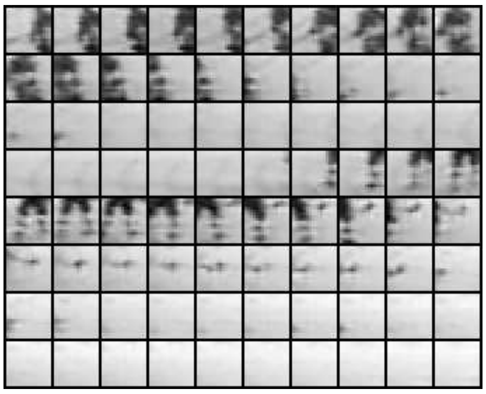
Figure 15: The means of the predictive distribution for the DynVar model.

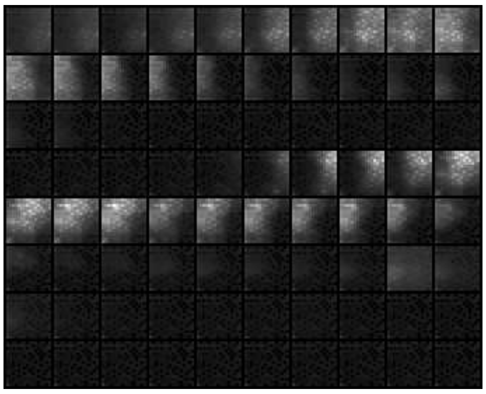
Figure 16: The standard deviations of the predictive distribution for the DynVar model.

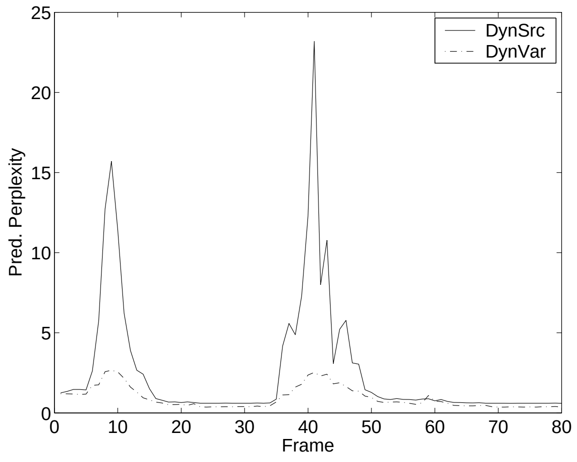

Figure 17: Predictive perplexities.

vised learning tasks, where point estimates are too prone to overfitting and sampling used in MCMC methods and particle filters, is often too slow. The problems tackled with particle filters in [14] vary from 1 to 10 in dimensionality, whereas the latent space in Section 7.3 is 128 dimensional. The variational Bayesian learning seems to provide a good compromise between point estimates and sampling methods.

Often the posterior distribution consists of clusters or solution modes. It depends on the posterior approximation again, whether only one of the clusters, or all of them are modelled. In our case, the expectation in $\mathcal{J}_{\mathrm{KL}}(q \parallel p)$ is taken over the approximate distribution q, which in practice leads to modelling a single mode. In expectation propagation [52], the Kullback-Leibler divergence is formed differently, leading to modelling of all the modes. Also, sampling is supposed to take place in all the modes. For the purpose of finding a single good representative of the posterior probability mass, the first approach should be better. In fact, the expectation over the true posterior, also known as the Bayes estimate, is often degenerate due to symmetry. For instance in factor analysis type models, the posterior is symmetric to the permutation of factors. The number of permutations also gives a hint of the infeasibility of accurately modelling all the modes in a high-dimensional problem. Perhaps it would be best to find one mode for the parameters, but all modes for the time-dependent variables when feasible.

The variational Bayesian methods vary further depending on the posterior approximation. In this paper, all variables are assumed to be independent a posteriori. We have chosen to model individual distributions as Gaussians. Often different conjugate distributions are used instead, for instance, the variance of a Gaussian variable is modelled with a Gamma distribution. Conjugate distributions are accurate and in some sense practical, but by restricting to Gaussians, the nodes can be connected more freely allowing for example hierarchical modelling of variances. It should be noted that the effect of assuming independencies is far more significant compared to the effect of approximations in modelling individual distributions.

The scope of this paper has been restricted to models which can be learned using purely local computations. This is possible, if the parents of each node are independent a posteriori. This can be accomplished by using a factorial posterior approximation and by not allowing multiple computational paths between variables. Purely local computations result in a computational complexity that is linear w.r.t. the number of connections in the model. In small models, one could afford to take all the dependencies into account. In larger models, it might be desirable to model posterior dependencies within disjoint groups of variables, but to assume the groups statistically independent, as is done by [81].

According to our experience, almost maximally factorial posterior pdf approximation $q(\boldsymbol{\theta})$ suffices in many cases. It seems that a good model structure is usually more important than a good approximation of the posterior pdf of the model. Therefore the available computation time is often better invested in a larger model using a simple posterior approximation. In any case, density estimates of continuous valued latent variables offer an important advantage over point estimates, because they are robust against overfitting and provide a cost function suitable for learning model structures. With variational Bayesian learning employing a factorial posterior pdf approximation $q(\boldsymbol{\theta})$ the density estimates are almost as efficient as point estimates. Moreover, latent variable models often exhibit rotational and other invariances which variational Bayesian learning can utilise by choosing a solution where the factorial approximation is most accurate.

The basic algorithm for learning and inference is based on updating a variable at a time while keeping other variables fixed. It has the benefits of being completely local and guaranteed to converge. A drawback is that the flow of information through time can be slow while performing inference in a dynamical model. There are alternative inference algorithms, where updates are carried out in forward and backward sweeps. These include particle smoothing [14], extended Kalman smoothing [1], and expectation propagation [52]. When the model needs to be learned at the same time, one needs to iterate a lot anyway, so the variational Bayesian algorithm that makes small but consistent improvements at every sweep might be preferable.

It is an important design choice that each node is updated while keeping the other nodes fixed. If new node types are added later on, there is no need to change the global learning algorithm, but it suffices to design an update rule for the new node type. Also, there is an option to update some nodes more often than others. When different parameters are coupled and cyclic updating is slow, it can be sped up by line search as described by [33]. Note that all the updates are done in order to minimise a global cost function, that is, a cost function over all the variables. Expectation propagation [52] updates one approximation at a time, too. An important difference is that there updates are done using a local cost function, i.e. the local approximation is fitted to the local true posterior assuming that the rest of the approximation is accurate. This is the reason why expectation propagation may diverge.

Large nonlinear problems often have numerous suboptimal local solutions that should be avoided. We have used many tricks to avoid them, as discussed in Section 6.2. It depends on the application which tricks work best. It is an important aspect of future work to make the procedure as simple as possible for the user.

9 Conclusions

In this paper, we have introduced standardised nodes (blocks) for constructing generative latent variable models. These nodes include a Gaussian node, addition, multiplication, a nonlinearity following directly a Gaussian node, and a delay node. The nodes have been designed so that they fit together, allowing construction of many types of latent variable models, including both known and novel structures. Constructing new prototype models is rapid since the user does not need to take care of the learning formulas. The nodes have been implemented in an open source software package called the Bayes Blocks [72].

The models built from these blocks are taught using variational Bayesian (ensemble) learning. This learning method essentially uses as its cost function the Kullback-Leibler information between the true posterior density and its approximation. The cost function is used for updating the unknown variables in the model, but it also allows optimisation of the number of nodes in the chosen model type. By using a factorial posterior density approximation, all the required computations can be carried out locally by propagating means, variances, and expected exponentials instead of full distributions. In this way, one can achieve a linear computational complexity with respect to the number of connections in the chosen model. However, initialisation to avoid premature pruning of nodes and local minima require special attention in each application for achieving good results.

In this paper, we have tested the introduced method experimentally in three separate unsupervised learning problems with different types of models. The results demonstrate the good performance and usefulness of the method. First, hierarchical nonlinear factor analysis (HNFA) with variance modelling was applied to an extension of the bars problem. The presented algorithm could find a model that is essentially the same as the complicated way in which the data were generated. Secondly, HNFA was used to reconstruct missing values in speech spectra. The results were consistently better than with linear factor analysis, and were generally best in cases requiring accurate representation in high dimensionality. The third experiment was carried out using real-world video image data. We compared the linear dynamical model for the means and for the variances of the sources. The results demonstrate that finding strong dependencies between different sources was considerably easier when the variances were modelled, too.

Acknowledgments

We thank Antti Honkela, Tomas Östman, and Alexander Ilin for work on Bayes blocks software library and for useful comments, and Hans van Hateren for supplying the video data [78] used in the 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.

APPENDICES

A Updating $q(s)$ for the Gaussian node

Here we show how to minimise the function

$$\mathcal{C}(m, v) = Mm + V[(m - m_0)^2 + v] + E \exp(m + v/2) - \frac{1}{2} \ln v, \qquad (54)$$

where M,V,E, and m_0 are scalar constants. A unique solution exists when $V > 0$ and $E \geq 0$. This problem occurs when a Gaussian posterior with mean m and variance v is fitted to a probability distribution whose logarithm has both a quadratic and exponential part resulting from Gaussian prior and log-Gamma likelihoods, respectively, and Kullback-Leibler divergence is used as the measure of the misfit.

In the special case $E = 0$, the minimum of $\mathcal{C}(m, v)$ can be found analytically and it is $m = m_0 - \frac{M}{2V}$, $v = \frac{1}{2V}$. In other cases where $E > 0$, minimisation is performed iteratively. At each iteration, one Newton iteration for the mean m and one fixed-point iteration for the variance v are carried out as explained in more detail in the following.

A.1 Newton iteration for the mean m

The Newton iteration for m is obtained by

$$\begin{aligned} m_{i+1} &= m_i - \frac{\partial\mathcal{C}(m_i, v_i)/\partial m_i}{\partial^2\mathcal{C}(m_i, v_i)/\partial m_i^2} \\ &= m_i - \frac{M + 2V(m_i - m_0) + E\exp(m + v/2)}{2V + E\exp(m + v/2)}. \end{aligned} \tag{55}$$

The Newton iteration converges in one step if the second derivative remains constant. The step is too short if the second derivative decreases and too long if the second derivative increases. For stability, it is better to take too short than too long steps.

In this case, the second derivative always decreases if the mean m decreases and vice versa. For stability it is therefore useful to restrict the growth of m because it is consistently over-estimated.

A.2 Fixed-point iteration for the variance v

A simple fixed-point iteration rule is obtained for the variance v by solving the zero of the derivative:

$$\begin{aligned} 0 &= \frac{\partial\mathcal{C}(m, v)}{\partial v} = V + \frac{E}{2}\exp(m + v/2) - \frac{1}{2v} \Leftrightarrow \\ v &= \frac{1}{2V + E\exp(m + v/2)} \overset{def}{=} g(v) \end{aligned} \tag{56}$$

$$v_{i+1} = g(v_i) \tag{57}$$

In general, fixed-point iterations are stable around the solution v_{opt} if $|g'(v_{\text{opt}})| < 1$ and converge best when the derivative $g'(v_{\text{opt}})$ is near zero. In our case $g'(v_i)$ is always negative and can be less than -1. In this case the solution can be an unstable fixed-point. This can be avoided by taking a weighted average of (57) and a trivial iteration $v_{i+1} = v_i$:

$$v_{i+1} = \frac{\xi(v_i)g(v_i) + v_i}{\xi(v_i) + 1} \overset{def}{=} f(v_i) \tag{58}$$

The weight ξ should be such that the derivative of f is close to zero at the optimal solution v_{opt} which is achieved exactly when $\xi(v_{\text{opt}}) = -g'(v_{\text{opt}})$.

It holds

$$\begin{aligned} &g'(v) = -\frac{(E/2)\exp(m + v/2)}{[2V + E\exp(m + v/2)]^2} = g^2(v)\left[V - \frac{1}{2g(v)}\right] = g(v)\left[Vg(v) - \frac{1}{2}\right] \Rightarrow \\ &g'(v_{\text{opt}}) = v_{\text{opt}}\left[Vv_{\text{opt}} - \frac{1}{2}\right] \Rightarrow \xi(v_{\text{opt}}) = v_{\text{opt}}\left[\frac{1}{2} - Vv_{\text{opt}}\right] \end{aligned} \tag{59}$$

The last steps follow from the fact that $v_{\mathrm{opt}} = g(v_{\mathrm{opt}})$ and from the requirement that $f'(v_{\mathrm{opt}}) = 0$. We can assume that v is close to v_{opt} and use

$$\xi(v) = v \left[\frac{1}{2} - V v_{\mathrm{opt}} \right] . \tag{60}$$

Note that the iteration (57) can only yield estimates with $0 < v_{i+1} < 1/2V$ which means that $\xi(v_{i+1}) > 0$. Therefore the use of ξ always shortens the step taken in (58). If the initial estimate $v_0 > 1/2V$, we can set it to $v_0 = 1/2V$.

A.3 Summary of the updating method for $q(s)$

1. Set $v_0 \leftarrow \min(v_0, 1/2V)$.
2. Iterate:
   (a) Solve the new estimate of the mean m from Eq. (55) under the restriction that the maximum step is 4;
   (b) Solve the new estimate of the variance v from Eqs. (60) and (58) under the restriction that the maximum step is 4.
3. Stop the iteration after the corrections become small enough, being under some suitable predefined threshold value.

B Addition and multiplication nodes

Equations (20)–(24) for the addition and multiplication nodes are proven in the following section. Only the Equation (20) applies in general, the others assume that the incoming signals are independent a posteriori. That is, $q(s_1, s_2, \ldots, s_n) = q(s_1)q(s_2)\ldots q(s_n)$. Also the proof of the form of the cost function mostly concerns propagation through addition and multiplication nodes, so it is presented here. Finally, the formulas of propagating the gradients of the cost function w.r.t. the expectations are derived.

B.1 Expectations

Equation (20) follows directly from the linearity of the expectation operation, or can be proven analogously to the proof of Equation (23):

$$\begin{aligned}\left\langle \prod_{i=1}^{n} s_i \right\rangle &= \int \left(\prod_{i=1}^{n} s_i \right) q(s_1, s_2, \ldots, s_n) d\mathbf{s} \\ &= \int \prod_{i=1}^{n} s_i q(s_i) d\mathbf{s} = \prod_{i=1}^{n} \int s_i q(s_i) ds_i = \prod_{i=1}^{n} \langle s_i \rangle .\end{aligned}$$

Equation (21) states that the variance of a sum of independent variables is the sum of their variances. This fact can be found in basic probability theory books. It can be proven with simple manipulation by using Equations (20) and (23).

Equation (22) can be proven by applying (23) to $\exp s_i$:

$$\left\langle \exp\left(\sum_{i=1}^{n} s_i\right)\right\rangle = \left\langle \prod_{i=1}^{n} \exp s_i \right\rangle = \prod_{i=1}^{n} \langle \exp s_i \rangle .$$

Equation (24) can be proven by applying Equation (23) to both s_i and s_i^2:

$$\begin{aligned}\operatorname{Var}\left\{\prod_{i=1}^{n} s_i\right\} &= \left\langle \left(\prod_{i=1}^{n} s_i\right)^2 \right\rangle - \left\langle \prod_{j=1}^{n} s_j \right\rangle^2 = \left\langle \prod_{i=1}^{n} s_i^2 \right\rangle - \left(\prod_{j=1}^{n} \langle s_j \rangle\right)^2 \\ &= \prod_{i=1}^{n} \langle s_i^2 \rangle - \prod_{j=1}^{n} \langle s_j \rangle^2 = \prod_{i=1}^{n} \left[\langle s_i \rangle^2 + \operatorname{Var}\{s_i\}\right] - \prod_{j=1}^{n} \langle s_j \rangle^2 .\end{aligned}$$

B.2 Form of the cost function

The form of the part of the cost function that an output of a node affects is shown to be of the form

$$\mathcal{C}_p = M\langle\cdot\rangle + V[(\langle\cdot\rangle - \langle\cdot\rangle_{\text{current}})^2 + \operatorname{Var}\{\cdot\}] + E\langle\exp\cdot\rangle + C \qquad (61)$$

where $\langle\cdot\rangle$ denotes the expectation of the quantity in question. If the output is connected directly to another variable, this can be seen from Eq. (10) by substituting

$$\begin{aligned} M &= \langle \exp v \rangle (\langle s \rangle_{\text{current}} - \langle m \rangle) \\ V &= \frac{1}{2}\langle \exp v \rangle \\ E &= 0 \\ C &= \frac{1}{2}\left[\langle \exp v \rangle \left(\operatorname{Var}\{m\} + \langle m \rangle^2 - \langle s \rangle^2_{\text{current}}\right) - \langle v \rangle + \ln 2\pi\right] . \end{aligned}$$

If the output is connected to multiple variables, the sum of the affected costs is of the same form. Now one has to prove that this form remains the same when the signals are fed through the addition and multiplication nodes. [5]

If the cost function is of the predefined form (61) for the sum $s_1 + s_2$, it has the same form for s_1, when s_2 is regarded as a constant. This can be

[5] Note that delay node only rewires connections so it does not affect the formulas.

shown using Eqs. (20), (21), and (22):

$$\begin{aligned}
\mathcal{C}_p &= M \langle s_1 + s_2 \rangle + V \left[(\langle s_1 + s_2 \rangle - \langle s_1 + s_2 \rangle_{\text{current}})^2 + \mathrm{Var}\{s_1 + s_2\} \right] \\
&\quad + E \langle \exp(s_1 + s_2) \rangle + C \qquad (62) \\
&= M \langle s_1 \rangle + V \left[(\langle s_1 \rangle - \langle s_1 \rangle_{\text{current}})^2 + \mathrm{Var}\{s_1\} \right] \\
&\quad + (E \langle \exp s_2 \rangle) \langle \exp s_1 \rangle + (C + M \langle s_2 \rangle + V \mathrm{Var}\{s_2\})
\end{aligned}$$

It can also be seen from (62) that when $E = 0$ for the sum $s_1 + s_2$, it is zero for the addend s_1, that is $E' = E \langle \exp s_2 \rangle = 0$. This means that the outputs of product and nonlinear nodes can be fed through addition nodes.

If the cost function is of the predefined form (61) with $E = 0$ for the product $s_1 s_2$, it is similar for the variable s_1, when the variable s_2 is regarded as a constant. This can be shown using Eqs. (23) and (24):

$$\begin{aligned}
\mathcal{C}_p &= M \langle s_1 s_2 \rangle + V \left[(\langle s_1 s_2 \rangle - \langle s_1 s_2 \rangle_{\text{current}})^2 + \mathrm{Var}\{s_1 s_2\} \right] + C \qquad (63) \\
&= (M \langle s_2 \rangle + 2V \mathrm{Var}\{s_2\} \langle s_1 \rangle_{\text{current}}) \langle s_1 \rangle \\
&\quad + \left[V \left(\langle s_2 \rangle^2 + \mathrm{Var}\{s_2\} \right) \right] \left[(\langle s_1 \rangle - \langle s_1 \rangle_{\text{current}})^2 + \mathrm{Var}\{s_1\} \right] \\
&\quad + \left(C - V \mathrm{Var}\{s_2\} \langle s_1 \rangle_{\text{current}}^2 \right)
\end{aligned}$$

C Updating $q(s)$ for the Gaussian node followed by a nonlinearity

A Gaussian variable has its own terms in the cost function and it affects the cost function of its children. In case there is a nonlinearity attached to it, only the latter is changed. The cost function of the children can be written in the form

$$\mathcal{C}_{\mathrm{ch}(s),p} = M \langle f(s) \rangle + V[(\langle f(s) \rangle - \langle f(s) \rangle_{\text{current}})^2 + \mathrm{Var}\{f(s)\}] \qquad (64)$$

where $\langle f(s) \rangle_{\text{current}}$ stands for the expectation using the current posterior estimate $q(s)$, and M and V are constants.

The posterior $q(s) = \mathcal{N}(s; \overline{s}, \widetilde{s})$ is updated to minimise the cost function. For $\widetilde{s}$ we get a fixed point iteration for the update candidate:

$$\begin{aligned}
v\widetilde{s}_{new} = \Bigg[\langle \exp v \rangle &+ \frac{4V \left(1 - 2\overline{s}^2 + 2\widetilde{s}\right) (\langle f(s) \rangle - \frac{M}{2V}) \langle f(s) \rangle}{(2\widetilde{s} + 1)^2} \\
&- \frac{4V \left(1 - 4\overline{s}^2 + 4\widetilde{s}\right) \left\langle [f(s)]^2 \right\rangle}{(4\widetilde{s} + 1)^2} \Bigg]^{-1} \qquad (65)
\end{aligned}$$

And for $\overline{s}$ we have an approximated Newton's iteration update candidate

$$\overline{s}_{new} = \overline{s} - \widetilde{s}_{new} \left[\langle \exp v \rangle \left(\overline{s} - \langle m \rangle \right) + 4V\overline{s} \left(\frac{\left(\langle f(s) \rangle - \frac{M}{2V} \right) \langle f(s) \rangle}{2\widetilde{s}+1} - \frac{\left\langle [f(s)]^2 \right\rangle}{4\widetilde{s}+1} \right) \right] \tag{66}$$

These candidates guarantee a direction, in which the cost function decreases locally. As long as the cost function is about to increase in value, the step size is halved. This guarantees the convergence to a stable point.

D Example where point estimates fail

The following example illustrates what can go wrong with point estimates. Three dimensional data vectors $\mathbf{x}(t)$ are modelled with the linear factor analysis model $\mathbf{x}(t) = \mathbf{a}s(t) + \mathbf{n}(t)$, using a scalar source signal $s(t)$ and a Gaussian noise vector $\mathbf{n}(t)$ with zero mean and parameterised variance $p(n_k) = \mathcal{N}(0, \sigma_k^2)$. Here $\mathbf{a}$ is a three-dimensional weight vector.

The weight vector $\mathbf{a}$ might get a value $\mathbf{a} = [1\ \ 0\ \ 0]^T$, while the source can just copy the values of the first dimension of $\mathbf{x}(t)$, that is, $s(t) = x_1(t)$. When the reconstruction error or the noise term is evaluated: $\mathbf{n}(t) = \mathbf{x}(t) - \mathbf{a}s(t) = [0\ \ x_2(t)\ \ x_3(t)]^T$, one can see that problems will arise with the first variance parameter σ_1^2. The likelihood goes to infinity as σ_1^2 goes to zero. The same applies to the posterior density, since it is basically just the likelihood multiplied by a finite factor.

The found model is completely useless and still, it is rated as infinitely good using point estimates. These problems are typical for models with estimates of the noise level or products. They can be sometimes avoided by fixing the noise level or using certain normalisations [4]. When the noise model is nonstationary (see Section 5.1), the problem becomes even worse, since the infinite likelihood appears if the any of the variances goes to zero.

References

[1] B. Anderson and J. Moore. *Optimal Filtering.* Prentice-Hall, Englewood Cliffs, NJ, 1979.

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

[3] H. Attias. A variational Bayesian framework for graphical models. In T. L. et al., editor, *Advances in Neural Information Processing Systems 12*, pages 209–215, Cambridge, 2000. MIT Press.

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

[5] D. Barber and C. Bishop. Ensemble learning in Bayesian neural networks. In C. Bishop, editor, *Neural Networks and Machine Learning*, pages 215–237. Springer, Berlin, 1998.

[6] M. Beal. *Variational Algorithms for Approximate Bayesian Inference.* PhD thesis, University of London, UK, 2003.

[7] M. Beal and Z. Ghahramani. The variational Bayesian EM algorithm for incomplete data: with application to scoring graphical model structures. *Bayesian Statistics 7*, pages 453–464, 2003.

[8] C. Bishop. *Neural Networks for Pattern Recognition.* Clarendon Press, 1995.

[9] C. Bishop. Latent variable models. In M. Jordan, editor, *Learning in Graphical Models*, pages 371–403. The MIT Press, Cambridge, MA, USA, 1999.

[10] J.-F. Cardoso. Multidimensional independent component analysis. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP'98)*, pages 1941–1944, Seattle, Washington, USA, May 12–15, 1998.

[11] K. Chan, T.-W. Lee, and T. Sejnowski. Variational learning of clusters of undercomplete nonsymmetric independent components. In *Proc. Int. Conf. on Independent Component Analysis and Signal Separation (ICA2001)*, pages 492–497, San Diego, USA, 2001.

[12] R. Choudrey, W. Penny, and S. Roberts. An ensemble learning approach to independent component analysis. In *Proc. of the IEEE Workshop on Neural Networks for Signal Processing, Sydney, Australia, December 2000*, pages 435–444. IEEE Press, 2000.

[13] P. Dayan and R. Zemel. Competition and multiple cause models. *Neural Computation*, 7(3):565–579, 1995.

[14] A. Doucet, N. de Freitas, and N. J. Gordon. *Sequential Monte Carlo Methods in Practice.* Springer Verlag, 2001.

[15] B. J. Frey and G. E. Hinton. Variational learning in nonlinear Gaussian belief networks. *Neural Computation*, 11(1):193–214, 1999.

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

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

[18] Z. Ghahramani and G. E. Hinton. Hierarchical non-linear factor analysis and topographic maps. In M. I. Jordan, M. J. Kearns, and S. A. Solla, editors, *Advances in Neural Information Processing Systems 10*, pages 486–492. The MIT Press, Cambridge, MA, USA, 1998.

[19] Z. Ghahramani and S. Roweis. Learning nonlinear dynamical systems using an EM algorithm. In M. Kearns, S. Solla, and D. Cohn, editors, *Advances in Neural Information Processing Systems 11*, pages 431–437. The MIT Press, Cambridge, MA, USA, 1999.

[20] A. Gray, B. Fischer, J. Schumann, and W. Buntine. Automatic derivation of statistical algorithms: The EM family and beyond. In *Advances in Neural Information Processing Systems 15*, 2002.

[21] M. Harva. *Hierarchical Variance Models of Image Sequences.* Helsinki Univ. of Technology, Dept. of Computer Science and Eng., Espoo, Finland, March 2004. Master of Science (Dipl.Eng.) thesis. Available at `http://www.cis.hut.fi/mha`.

[22] M. Harva and A. Kabán. A variational Bayesian method for rectified factor analysis. In *Proc. 2005 IEEE International Joint Conference on Neural Networks (IJCNN 2005)*, pages 185–190, Montreal, Canada, 2005.

[23] M. Harva, T. Raiko, A. Honkela, H. Valpola, and J. Karhunen. Bayes Blocks: An implementation of the variational Bayesian building blocks framework. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence, UAI 2005*, pages 259–266, Edinburgh, Scotland, July 2005.

[24] S. Haykin. *Neural Networks – A Comprehensive Foundation, 2nd ed.* Prentice-Hall, 1998.

[25] S. Haykin, editor. *Kalman Filtering and Neural Networks.* Wiley, New York, 2001.

[26] G. E. Hinton and D. van Camp. 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, 1993.

[27] P. Højen-Sørensen, O. Winther, and L. Hansen. Mean-field approaches to independent component analysis. *Neural Computation*, 14(4):889–918, 2002.

[28] A. Honkela. Speeding up cyclic update schemes by pattern searches. In *Proc. of the 9th Int. Conf. on Neural Information Processing (ICONIP'02)*, pages 512–516, Singapore, 2002.

[29] A. Honkela, S. Harmeling, L. Lundqvist, and H. Valpola. Using kernel PCA for initialisation of variational Bayesian nonlinear blind source separation method. In C. Puntonet and A. Prieto, editors, *Proc. of the Fifth Int. Conf. on Independent Component Analysis and Blind Signal Separation (ICA 2004)*, volume 3195 of *Lecture Notes in Computer Science*, pages 790–797, Granada, Spain, 2004. Springer-Verlag, Berlin.

[30] A. Honkela, T. Östman, and R. Vigário. Empirical evidence of the linear nature of magnetoencephalograms. In *Proc. 13th European Symposium on Artificial Neural Networks (ESANN 2005)*, pages 285–290, Bruges, Belgium, 2005.

[31] A. Honkela and H. Valpola. Variational learning and bits-back coding: an information-theoretic view to Bayesian learning. *IEEE Transactions on Neural Networks*, 15(4):800–810, 2004.

[32] A. Honkela and H. Valpola. Unsupervised variational Bayesian learning of nonlinear models. In L. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*. MIT Press, Cambridge, MA, USA, 2005. To appear.

[33] A. Honkela, H. Valpola, and J. Karhunen. Accelerating cyclic update algorithms for parameter estimation by pattern searches. *Neural Processing Letters*, 17(2):191–203, 2003.

[34] A. Hyvärinen and P. Hoyer. Emergence of phase and shift invariant features by decomposition of natural images into independent feature subspaces. *Neural Computation*, 12(7):1705–1720, 2000.

[35] A. Hyvärinen and P. Hoyer. Emergence of topography and complex cell properties from natural images using extensions of ICA. In S. A. Solla, T. K. Leen, and K.-R. Mller, editors, *Advances in Neural Information Processing Systems 12*, pages 827–833. The MIT Press, Cambridge, MA, USA, 2000.

[36] A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. J. Wiley, 2001.

[37] A. Ilin and H. Valpola. On the effect of the form of the posterior approximation in variational learning of ICA models. In *Proc. of the 4th Int. Symp. on Independent Component Analysis and Blind Signal Separation (ICA2003)*, pages 915–920, Nara, Japan, 2003.

[38] A. Ilin, H. Valpola, and E. Oja. Nonlinear dynamical factor analysis for state change detection. *IEEE Trans. on Neural Networks*, 15(3):559–575, May 2003.

[39] M. Jordan, editor. *Learning in Graphical Models.* The MIT Press, Cambridge, MA, USA, 1999.

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

[41] M. Jordan and T. Sejnowski, editors. *Graphical Models: Foundations of Neural Computation.* The MIT Press, Cambridge, MA, USA, 2001.

[42] T. Kohonen. *Self-Organizing Maps.* Springer, 3rd, extended edition, 2001.

[43] T. Kohonen, S. Kaski, and H. Lappalainen. Self-organized formation of various invariant-feature filters in the Adaptive-Subspace SOM. *Neural Computation*, 9(6):1321–1344, 1997.

[44] 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.

[45] H. Lappalainen and J. Miskin. Ensemble learning. In M. Girolami, editor, *Advances in Independent Component Analysis*, pages 75–92. Springer-Verlag, Berlin, 2000.

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

[47] D. MacKay. 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, 1995.

[48] D. MacKay. Ensemble learning for hidden Markov models. Available at `http://wol.ra.phy.cam.ac.uk/mackay/`, 1997.

[49] D. MacKay. Introduction to Monte Carlo methods. In M. Jordan, editor, *Learning in Graphical Models*, pages 175–204. The MIT Press, Cambridge, MA, USA, 1999.

[50] D. MacKay. Local minima, symmetry-breaking, and model pruning in variational free energy minimization. Available at `http://www.inference.phy.cam.ac.uk/mackay/`, 2001.

[51] D. MacKay. *Information Theory, Inference, and Learning Algorithms.* Cambridge University Press, 2003.

[52] T. Minka. Expectation propagation for approximate Bayesian inference. In *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence, UAI 2001*, pages 362–369, Seattle, Washington, USA, 2001.

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

[54] K. Murphy. A variational approximation for Bayesian networks with discrete and continuous latent variables. In *Proc. of the 15th Annual Conf. on Uncertainty in Artificial Intelligence (UAI–99)*, pages 457–466, Stockholm, Sweden, 1999.

[55] K. Murphy. The Bayes net toolbox for Matlab. *Computing Science and Statistics*, 33:331–350, 2001.

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

[57] L. Nolan, M. Harva, A. Kabán, and S. Raychaudhury. A data-driven Bayesian approach to finding young stellar populations in early-type galaxies from their UV-optical spectra. *Monthly Notices of the Royal Astronomical Society*, 2005. To appear. Available at `http://www.cis.hut.fi/mha/`.

[58] H.-J. Park and T.-W. Lee. A hierarchical ICA method for unsupervised learning of nonlinear dependencies in natural images. In C. Puntonet and A. Prieto, editors, *Proc. of the 5th Int. Conf. on Independent Component Analysis and Blind Signal Separation (ICA2004)*, pages 1253–1261, Granada, Spain, 2004.

[59] L. Parra, C. Spence, and P. Sajda. Higher-order statistical properties arising from the non-stationarity of natural signals. In T. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 786–792. The MIT Press, Cambridge, MA, USA, 2001.

[60] J. Pearl, editor. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann Publishers, San Francisco, California, 1988.

[61] D.-T. Pham and J.-F. Cardoso. Blind separation of instantaneous mixtures of nonstationary sources. *IEEE Trans. on Signal Processing*, 49(9):1837–1848, 2001.

[62] T. Raiko. Partially observed values. In *Proc. Int. Joint Conf. on Neural Networks (IJCNN'04)*, pages 2825–2830, Budapest, Hungary, 2004.

[63] T. Raiko. Nonlinear relational Markov networks with an application to the game of Go. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN 2005)*, pages 989–996, Warsaw, Poland, September 2005.

[64] T. Raiko and M. Tornio. Learning nonlinear state-space models for control. In *Proc. Int. Joint Conf. on Neural Networks (IJCNN'05)*, pages 815–820, Montreal, Canada, 2005.

[65] T. Raiko, H. Valpola, T. Östman, and J. Karhunen. Missing values in hierarchical nonlinear factor analysis. In *Proc. of the Int. Conf. on Artificial Neural Networks and Neural Information Processing (ICANN/ICONIP 2003)*, pages 185–189, Istanbul, Turkey, 2003.

[66] S. Roberts and R. Everson, editors. *Independent Component Analysis: Principles and Practice.* Cambridge Univ. Press, 2001.

[67] S. Roberts, E. Roussos, and R. Choudrey. Hierarchy, priors and wavelets: structure and signal modelling using ICA. *Signal Processing*, 84(2):283–297, February 2004.

[68] D. Rowe. *Multivariate Bayesian Statistics: Models for Source Separation and Signal Unmixing.* Chapman & Hall/CRC, Medical College of Wisconsin, 2003.

[69] G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.

[70] D. Spiegelhalter, A. Thomas, N. Best, and W. Gilks. BUGS: Bayesian inference using Gibbs sampling, version 0.50. Available at `http://www.mrc-bsu.cam.ac.uk/bugs/`, 1995.

[71] H. Valpola, M. Harva, and J. Karhunen. Hierarchical models of variance sources. *Signal Processing*, 84(2):267–282, 2004.

[72] H. Valpola, A. Honkela, M. Harva, A. Ilin, T. Raiko, and T. Östman. Bayes Blocks software library, 2003. Available at `http://www.cis.hut.fi/projects/bayes/software/`.

[73] H. Valpola, A. Honkela, and J. Karhunen. An ensemble learning approach to nonlinear dynamic blind source separation using state-space models. In *Proc. Int. Joint Conf. on Neural Networks (IJCNN'02)*, pages 460–465, Honolulu, Hawaii, USA, 2002.

[74] H. Valpola and J. Karhunen. An unsupervised ensemble learning method for nonlinear dynamic state-space models. *Neural Computation*, 14(11):2647–2692, 2002.

[75] H. Valpola, E. Oja, A. Ilin, A. Honkela, and J. Karhunen. Nonlinear blind source separation by variational Bayesian learning. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E86-A(3):532–541, 2003.

[76] H. Valpola, T. Östman, and J. Karhunen. Nonlinear independent factor analysis by hierarchical models. In *Proc. 4th Int. Symp. on Independent Component Analysis and Blind Signal Separation (ICA2003)*, pages 257–262, Nara, Japan, 2003.

[77] H. Valpola, T. Raiko, and J. Karhunen. Building blocks for hierarchical latent variable models. In *Proc. 3rd Int. Conf. on Independent Component Analysis and Signal Separation (ICA2001)*, pages 710–715, San Diego, USA, 2001.

[78] J. H. van Hateren and D. L. Ruderman. Independent component analysis of natural image sequences yields spatio-temporal filters similar to simple cells in primary visual cortex. *Proceedings of the Royal Society of London B*, 265(1412):2315–2320, 1998.

[79] J. Vesanto, J. Himberg, E. Alhoniemi, and J. Parhankangas. Self-organizing map in Matlab: the SOM toolbox. In *Proceedings of the Matlab DSP Conference*, pages 35–40, Espoo, Finland, November 1999. Available at `http://www.cis.hut.fi/projects/somtoolbox/`.

[80] C. S. Wallace. Classification by minimum-message-length inference. In S. G. Aki, F. Fiala, and W. W. Koczkodaj, editors, *Advances in Computing and Information – ICCI '90*, volume 468 of *Lecture Notes in Computer Science*, pages 72–81. Springer, Berlin, 1990.

[81] J. Winn and C. M. Bishop. Variational message passing. *Journal of Machine Learning Research*, 6:661–694, April 2005.