Fast Variational Bayesian Linear State-Space Model

Jaakko Luttinen

Aalto University, Espoo, Finland
jaakko.luttinen@aalto.fi

Abstract. This paper presents a fast variational Bayesian method for linear state-space models. The standard variational Bayesian expectation-maximization (VB-EM) algorithm is improved by a parameter expansion which optimizes the rotation of the latent space. With this approach, the inference is orders of magnitude faster than the standard method. The speed of the proposed method is demonstrated on an artificial dataset and a large real-world dataset, which shows that the standard VB-EM algorithm is not suitable for large datasets because it converges extremely slowly. In addition, the paper estimates the temporal state variables using a smoothing algorithm based on the block LDL decomposition. This smoothing algorithm reduces the number of required matrix inversions and avoids a model augmentation compared to previous approaches.

Keywords: variational Bayesian methods, linear state-space models, parameter expansion

1 Introduction

Linear state-space models (LSSM) are widely used in time-series analysis [1, 2]. They assume that the observations are generated linearly from a latent linear dynamical system. Although many real-world processes are non-linear, the linearity makes the model easy to analyze and efficient to estimate. In addition, many non-linear systems can be approximated using linear models, thus the LSSM is an important tool for time-series applications.

The Bayesian framework offers a principled way to estimate the model parameters from data. However, the estimation is analytically intractable making approximations necessary. This paper focuses on the variational Bayesian (VB) approximation, which can be computed using the variational Bayesian expectation-maximization (VB-EM) algorithm. The VB-EM algorithm assumes that the variables are independent and updates the approximate posterior distributions of the variables one at a time [3, 4].

The standard VB-EM algorithm may converge extremely slowly if the variables are strongly coupled. Because the variables are updated one at a time, the updates to each variable may be small and this results in zigzagging. This effect can be reduced by using parameter expansion to add auxiliary variables which reduce the coupling between some variables [5, 6]. This can be seen as a parameterized joint optimization of multiple variables. Because the expansion and the effect on the speed of the algorithm depends on the model, it is important to examine efficient parameter expansions for different models (see, e.g., [7–9]).

This paper derives a parameter expansion for the linear state-space model and shows experimentally that the VB-EM algorithm can be unusable for large datasets if the expansion is not used. The proposed parameter expansion is based on the rotation of the latent space, thus reducing the coupling between the states and the system parameters. Similar parameter expansion has been applied to canonical correlation and factor analysis models [8, 7]. However, those results cannot be applied directly to the LSSM because the rotation of the dynamics adds extra complexity.

In addition to convergence speed problems, the estimation of the state variables is not trivial, because the standard Rauch-Tung-Striebel [10] smoother cannot be applied directly as noted in [11]. This has been previously solved by using a parallel variant of the smoother in [11] and a model augmentation in [12]. This paper provides another perspective on the smoothing algorithm by deriving it from the Cholesky, or LDL, decomposition of a block-banded matrix. This results in a smoothing algorithm which requires less matrix inversions than the previous approaches, avoids the cost of the model augmentation and can be extended to other Markov random fields with different graph structure.

The paper is organized as follows: Section 2 defines the linear state-space model used in the paper. Section 3 briefly summarizes the standard VB-EM algorithm for the model. Section 4 derives the proposed smoothing algorithm. Section 5 presents the parameter expansion for the model. Section 6 presents experimental results that show the effect of the parameter expansion. Section 7 ends the paper with conclusions.

2 Model

In linear state-space models a sequence of M-dimensional observations $\mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_N)$ is assumed to be generated from latent D-dimensional states $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$ which follow a first-order Markov process:

$$\mathbf{x}_n = \mathbf{A}\mathbf{x}_{n-1} + \text{noise}, \tag{1}$$

$$\mathbf{y}_n = \mathbf{C}\mathbf{x}_n + \text{noise}, \tag{2}$$

where the noise is Gaussian, $\mathbf{A}$ is the $D \times D$ state dynamics matrix and $\mathbf{C}$ is the $M \times D$ loading matrix. Usually, the latent space dimensionality D is assumed to be much smaller than the observation space dimensionality M in order to model the dependencies of high-dimensional observations efficiently.

The equations defining the linear state-space model can be used to construct a Bayesian model [11]. The likelihood function is

$$p(\mathbf{Y}|\mathbf{C}, \mathbf{X}, \boldsymbol{\tau}) = \prod_{n=1}^{N} \mathcal{N}(\mathbf{y}_n|\mathbf{C}\mathbf{x}_n, \operatorname{diag}(\boldsymbol{\tau})^{-1}), \tag{3}$$

where $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the probability density function of the Gaussian distribution of variable $\mathbf{x}$ with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$. The covariance matrix in (3) is diagonal, that is, the noise is independent for each observed element of $\mathbf{y}_n$. The probability of the states is given as

$$p(\mathbf{X}|\mathbf{A}) = \mathcal{N}(\mathbf{x}_0|\mathbf{m}_0, \boldsymbol{\Lambda}_0^{-1}) \prod_{n=1}^{N} \mathcal{N}(\mathbf{x}_n|\mathbf{A}\mathbf{x}_{n-1}, \mathbf{I}), \tag{4}$$

where $\mathbf{x}_0$ is an auxiliary initial state with mean $\mathbf{m}_0$ and precision $\boldsymbol{\Lambda}_0$. The noise of the process in (4) has unit covariance matrix without loss of generality, because the latent space can be rotated arbitrarily by compensating it in the parameters $\mathbf{A}$ and $\mathbf{C}$. The parameters of the process are given the following priors:

$$p(\mathbf{A}|\boldsymbol{\alpha}) = \prod_{i=1}^{D}\prod_{j=1}^{D} \mathcal{N}(a_{ij}|0, \alpha_j^{-1}), \qquad p(\boldsymbol{\alpha}) = \prod_{d=1}^{D} \mathcal{G}(\alpha_d|a_\gamma, b_\gamma), \tag{5}$$

$$p(\mathbf{C}|\boldsymbol{\gamma}) = \prod_{m=1}^{M}\prod_{d=1}^{D} \mathcal{N}(c_{md}|0, \gamma_d^{-1}), \qquad p(\boldsymbol{\gamma}) = \prod_{d=1}^{D} \mathcal{G}(\gamma_d|a_\gamma, b_\gamma), \tag{6}$$

$$p(\boldsymbol{\tau}) = \prod_{m=1}^{M} \mathcal{G}(\tau_m|a_\tau, b_\tau), \tag{7}$$

where a_{ij} is the element on the i-th row and j-th column of the matrix $\mathbf{A}$, α_d is the d-th element of the vector $\boldsymbol{\alpha}$, and $\mathcal{G}(\alpha|a, b)$ is the probability density function of the gamma distribution with shape a and rate b. The variables $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$ are automatic relevance determination (ARD) parameters, which prune out components that are not significant enough. The hyperparameters a_α, b_α, a_γ, b_γ, a_τ and b_τ can be set to small values (e.g., 10^{-5}) to give broad priors. The above model definition is similar to [11, 12] and the details can be modified without affecting the main results of this paper.

3 Posterior Approximation

As the posterior distribution of the variables is analytically intractable, it is approximated using variational Bayesian (VB) framework [4]. The approximation is assumed to factorize with respect to the variables as

$$p(\mathbf{X}, \mathbf{A}, \boldsymbol{\alpha}, \mathbf{C}, \boldsymbol{\gamma}, \boldsymbol{\tau}|\mathbf{Y}) \approx q(\mathbf{X}, \mathbf{A}, \boldsymbol{\alpha}, \mathbf{C}, \boldsymbol{\gamma}, \boldsymbol{\tau}) = q(\mathbf{X})q(\mathbf{A})q(\boldsymbol{\alpha})q(\mathbf{C})q(\boldsymbol{\gamma})q(\boldsymbol{\tau}). \tag{8}$$

The approximation is optimized by minimizing the Kullback-Leibler divergence from the true posterior, which is equivalent to maximizing the lower bound of the marginal log likelihood

$$\begin{aligned}\mathcal{L}(\mathbf{Y}) = \langle \log p(\mathbf{Y}|\mathbf{C}, \mathbf{X}, \boldsymbol{\tau}) \rangle + \left\langle \log \frac{p(\mathbf{X}|\mathbf{A})}{q(\mathbf{X})} \right\rangle + \left\langle \log \frac{p(\mathbf{A}|\boldsymbol{\alpha})}{q(\mathbf{A})} \right\rangle \\ + \left\langle \log \frac{p(\boldsymbol{\alpha})}{q(\boldsymbol{\alpha})} \right\rangle + \left\langle \log \frac{p(\mathbf{C}|\boldsymbol{\gamma})}{q(\mathbf{C})} \right\rangle + \left\langle \log \frac{p(\boldsymbol{\gamma})}{q(\boldsymbol{\gamma})} \right\rangle + \left\langle \log \frac{p(\boldsymbol{\tau})}{q(\boldsymbol{\tau})} \right\rangle ,\end{aligned} \tag{9}$$

where $\langle\cdot\rangle$ is the expectation with respect to the approximate posterior distribution q.

For conjugate-exponential models, the approximation can be optimized by using the variational Bayesian expectation-maximization (VB-EM) algorithm [3]. In VB-EM, the posterior approximation is updated for the variables one at a time and iterated until convergence. The approximate distributions have the following forms:

$$q(\mathbf{X}) = \mathcal{N}([\mathbf{X}]_:|\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x), \qquad q(\boldsymbol{\tau}) = \prod_{m=1}^{M} \mathcal{G}(\tau_m|\bar{a}_\tau^{(m)}, \bar{b}_\tau^{(m)}), \tag{10}$$

$$q(\mathbf{A}) = \prod_{d=1}^{D} \mathcal{N}(\mathbf{a}_d|\boldsymbol{\mu}_a^{(d)}, \boldsymbol{\Sigma}_a^{(d)}), \qquad q(\boldsymbol{\alpha}) = \prod_{d=1}^{D} \mathcal{G}(\alpha_d|\bar{a}_\alpha^{(d)}, \bar{b}_\alpha^{(d)}), \tag{11}$$

$$q(\mathbf{C}) = \prod_{m=1}^{M} \mathcal{N}(\mathbf{c}_m|\boldsymbol{\mu}_c^{(m)}, \boldsymbol{\Sigma}_c^{(m)}), \qquad q(\boldsymbol{\gamma}) = \prod_{d=1}^{D} \mathcal{G}(\gamma_d|\bar{a}_\gamma^{(d)}, \bar{b}_\gamma^{(d)}), \tag{12}$$

where $\mathbf{a}_d$ and $\mathbf{c}_m$ are the row vectors of $\mathbf{A}$ and $\mathbf{C}$, respectively, and $[\mathbf{X}]_:$ is a vector obtained by stacking the vectors $\mathbf{x}_n$. It is straightforward to derive the following update equations of the variational parameters:

$$\boldsymbol{\Sigma}_a^{(d)} = \left(\langle \mathrm{diag}(\boldsymbol{\alpha}) \rangle + \sum_{n=1}^{N} \langle \mathbf{x}_{n-1}\mathbf{x}_{n-1}^{\mathrm{T}} \rangle \right)^{-1}, \qquad \boldsymbol{\mu}_a^{(d)} = \boldsymbol{\Sigma}_a^{(d)} \sum_{n=1}^{N} \langle x_{dn}\mathbf{x}_{n-1} \rangle, \tag{13}$$

$$\bar{a}_\alpha^{(d)} = a_\alpha + \frac{D}{2}, \qquad \bar{b}_\alpha^{(d)} = b_\alpha + \frac{1}{2}\sum_{i=1}^{D} \langle a_{id}^2 \rangle, \tag{14}$$

$$\boldsymbol{\Sigma}_c^{(m)} = \left(\langle \mathrm{diag}(\boldsymbol{\gamma}) \rangle + \sum_{n \in \mathcal{O}_{m:}} \langle \tau_m \rangle \langle \mathbf{x}_n\mathbf{x}_n^{\mathrm{T}} \rangle \right)^{-1}, \quad \boldsymbol{\mu}_c^{(m)} = \boldsymbol{\Sigma}_c^{(m)} \sum_{n \in \mathcal{O}_{m:}} y_{mn} \langle \tau_m \rangle \langle \mathbf{x}_n \rangle, \tag{15}$$

$$\bar{a}_\gamma^{(d)} = a_\gamma + \frac{M}{2}, \qquad \bar{b}_\gamma^{(d)} = b_\gamma + \frac{1}{2}\sum_{m=1}^{M} \langle c_{md}^2 \rangle, \tag{16}$$

$$\bar{a}_\tau^{(m)} = a_\tau + \frac{N_m}{2}, \qquad \bar{b}_\tau^{(m)} = b_\tau + \frac{1}{2}\sum_{n \in \mathcal{O}_{m:}} \xi_{mn}, \tag{17}$$

where $\mathcal{O}_{m:}$ is the set of time instances n for which the observation y_{mn} is not missing, N_m is the size of the set $\mathcal{O}_{m:}$, and $\xi_{mn} = \left\langle (y_{mn} - \mathbf{c}_m^{\mathrm{T}}\mathbf{x}_n)^2 \right\rangle$. Gaussian and gamma distributed variables have the following expectations:

$$\text{For } \mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \qquad \langle \mathbf{x} \rangle = \boldsymbol{\mu} \quad \text{and} \quad \langle \mathbf{x}\mathbf{x}^{\mathrm{T}} \rangle = \boldsymbol{\mu}\boldsymbol{\mu}^{\mathrm{T}} + \boldsymbol{\Sigma}. \tag{18}$$

$$\text{For } \boldsymbol{\alpha} \sim \mathcal{G}(a, b), \qquad \langle \alpha \rangle = \frac{a}{b} \quad \text{and} \quad \langle \log \alpha \rangle = \psi(a) - \log(b). \tag{19}$$

The formula for updating $q(\mathbf{X})$ is discussed in the following section.

4 Smoothing Algorithm

The approximate posterior distribution $q(\mathbf{X})$ can be updated using filtering and smoothing algorithms. However, the standard Rauch-Tung-Striebel (RTS) smoother cannot be applied straightforwardly because the required expectations under $q(\mathbf{A})$ are difficult to compute [11]. Previous approaches have solved this by using a parallel variant of the smoother [11] or augmenting the model to be able to apply standard RTS smoother [12]. Although these are working methods, this section presents another view on the smoothing problem and derives an algorithm which can be applied easily and efficiently in the VB framework.

Instead of trying to apply standard filters and smoothers directly, the smoothing can be computed equivalently from a Cholesky decomposition perspective [13]. The smoothing can be seen as a multiplication by the inverse of a large block-banded matrix. Utilizing the block-banded structure of the matrix, the computational complexity of the inversion is $\mathcal{O}(D^3N)$ instead of $\mathcal{O}(D^3N^3)$, where $D \ll N$. The inverse is computed using the block LDL decomposition and inverting this decomposition in two parts. The resulting smoothing algorithm is similar to the standard Kalman filter [14] and RTS smoother although not exactly identical.

The smoothing algorithm can be derived from the standard update equations of $q(\mathbf{X})$. Computational aspects aside, the update equation of the covariance matrix $\mathbf{\Sigma}_x$ is

$$\mathbf{\Sigma}_x = \begin{bmatrix} \mathbf{\Sigma}_{1,1} & \cdots & \mathbf{\Sigma}_{1,N} \\ \vdots & \ddots & \vdots \\ \mathbf{\Sigma}_{N,1} & \cdots & \mathbf{\Sigma}_{N,N} \end{bmatrix} = \begin{bmatrix} \mathbf{\Psi}_{0,0} & \mathbf{\Psi}_{0,1} & & \\ \mathbf{\Psi}_{0,1}^{\mathrm{T}} & \mathbf{\Psi}_{1,1} & \ddots & \\ & \ddots & \ddots & \mathbf{\Psi}_{N-1,N} \\ & & \mathbf{\Psi}_{N-1,N}^{\mathrm{T}} & \mathbf{\Psi}_{N,N} \end{bmatrix}^{-1} = \mathbf{\Psi}^{-1}, \qquad (20)$$

where the block-banded matrix $\mathbf{\Psi}$ is defined as

$$\begin{aligned} \mathbf{\Psi} = & \begin{bmatrix} \mathbf{\Lambda}_0 + \langle \mathbf{A}^{\mathrm{T}}\mathbf{A} \rangle & \langle \mathbf{A}_1 \rangle^{\mathrm{T}} & & \\ \langle \mathbf{A} \rangle & \mathbf{I} - \langle \mathbf{A}^{\mathrm{T}}\mathbf{A} \rangle & \ddots & \\ & \ddots & \ddots & \langle \mathbf{A} \rangle^{\mathrm{T}} \\ & & \langle \mathbf{A} \rangle & \mathbf{I} - \langle \mathbf{A}^{\mathrm{T}}\mathbf{A} \rangle \end{bmatrix} \\ & + \begin{bmatrix} \mathbf{0} & & & \\ & \sum_{m \in \mathcal{O}_{:1}} \langle \tau_m \rangle \langle \mathbf{c}_m \mathbf{c}_m^{\mathrm{T}} \rangle & & \\ & & \ddots & \\ & & & \sum_{m \in \mathcal{O}_{:N}} \langle \tau_m \rangle \langle \mathbf{c}_m \mathbf{c}_m^{\mathrm{T}} \rangle \end{bmatrix}, \end{aligned} \qquad (21)$$

and $\mathcal{O}_{:n}$ is the set of dimensions m for which the observation y_{mn} is not missing. The first matrix term in the sum (21) comes from the prior (4) and the second matrix term comes from the likelihood (3). Note that although $\mathbf{\Psi}$ is block-

Algorithm 1 Forward pass.

```
Input: $\{\mathbf{\Psi}_{n,n}\}_{n=0}^{N}$, $\{\mathbf{\Psi}_{n,n+1}\}_{n=0}^{N-1}$, $\{\mathbf{v}_n\}_{n=0}^{N}$
  $\tilde{\mathbf{\Sigma}}_{0,0} \leftarrow \mathbf{\Psi}_{0,0}^{-1}$
  $\tilde{\boldsymbol{\mu}}_0 \leftarrow \tilde{\mathbf{\Sigma}}_{0,0}\mathbf{v}_0$
  for $n = 0 \to N-1$ do
    $\tilde{\mathbf{\Sigma}}_{n,n+1} \leftarrow \tilde{\mathbf{\Sigma}}_{n,n}\mathbf{\Psi}_{n,n+1}$
    $\tilde{\mathbf{\Sigma}}_{n+1,n+1} \leftarrow \left(\mathbf{\Psi}_{n+1,n+1} - \tilde{\mathbf{\Sigma}}_{n,n+1}^{\mathrm{T}}\mathbf{\Psi}_{n,n+1}\right)^{-1}$
    $\tilde{\boldsymbol{\mu}}_{n+1} \leftarrow \tilde{\mathbf{\Sigma}}_{n+1,n+1}\left(\mathbf{v}_{n+1} - \tilde{\mathbf{\Sigma}}_{n,n+1}^{\mathrm{T}}\tilde{\boldsymbol{\mu}}_n\right)$
  end for
Output: $\{\tilde{\mathbf{\Sigma}}_{n,n}\}_{n=0}^{N}$, $\{\tilde{\mathbf{\Sigma}}_{n,n+1}\}_{n=0}^{N-1}$, $\{\tilde{\boldsymbol{\mu}}_n\}_{n=0}^{N}$
```

banded, $\mathbf{\Sigma}$ is dense in general. The posterior mean parameter is updated as

$$\boldsymbol{\mu}_x = \begin{bmatrix} \boldsymbol{\mu}_o \\ \boldsymbol{\mu}_1 \\ \vdots \\ \boldsymbol{\mu}_N \end{bmatrix} = \mathbf{\Sigma}_x \begin{bmatrix} \mathbf{v}_0 \\ \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_N \end{bmatrix} = \mathbf{\Sigma}_x\mathbf{v}, \quad \text{where } \mathbf{v} = \begin{bmatrix} \mathbf{\Lambda}_0\mathbf{m}_0 \\ \sum_{m\in\mathcal{O}_{:1}} y_{mn}\langle\tau_m\rangle\langle\mathbf{c}_m\rangle \\ \vdots \\ \sum_{m\in\mathcal{O}_{:N}} y_{mn}\langle\tau_m\rangle\langle\mathbf{c}_m\rangle \end{bmatrix}. \quad (22)$$

The vector $\mathbf{v}_0$ comes from the prior and the vectors $\{\mathbf{v}_n\}_{n=1}^{N}$ come from the likelihood.

Instead of computing the full matrix $\mathbf{\Sigma}_x$, it is sufficient for the VB-EM algorithm to compute only the diagonal blocks $\mathbf{\Sigma}_{n,n}$, the first super-diagonal blocks $\mathbf{\Sigma}_{n,n+1}$, and the mean $\boldsymbol{\mu}_x$. These terms can be computed efficiently by writing the parameters as $\mathbf{\Sigma}_x = \mathbf{\Psi}^{-1}\mathbf{I}$ and $\boldsymbol{\mu}_x = \mathbf{\Psi}^{-1}\mathbf{v}$, and utilizing the block-banded structure of $\mathbf{\Psi}$. Because $\mathbf{\Psi}$ is a symmetric positive-definite matrix, it can be decomposed using the block LDL decomposition $\mathbf{\Psi} = \mathbf{LDL}^{\mathrm{T}}$, where $\mathbf{D}$ is a block-diagonal matrix and $\mathbf{L}$ is a lower-triangular matrix with identity matrices on the diagonal. Thus, multiplying on the left by $\mathbf{\Psi}^{-1}$ is equivalent to multiplying on the left by $\mathbf{L}^{-\mathrm{T}}\mathbf{D}^{-1}\mathbf{L}^{-1}$. This can be computed in two phases: First, multiplying on the left by $\mathbf{D}^{-1}\mathbf{L}^{-1}$ results in the forward pass shown in Algorithm 1. Second, multiplying on the left by $\mathbf{L}^{-\mathrm{T}}$ results in the backward pass shown in Algorithm 2. Note that both algorithms can be implemented as in-place algorithms by overwriting the inputs with the outputs.

This smoothing algorithm has a few benefits compared to the previous methods [11, 12]. First, the algorithm needs to compute only one matrix inversion per time instance, whereas the parallel and the augmented variants require three and two inversions, respectively. Second, the augmented variant requires the Cholesky decomposition of $\langle\mathbf{A}^{\mathrm{T}}\mathbf{A}\rangle$ and $\langle\mathbf{C}^{\mathrm{T}}\operatorname{diag}(\boldsymbol{\tau})\mathbf{C}\rangle$ for each time instance if $\mathbf{A}$, $\mathbf{C}$ or $\boldsymbol{\tau}$ varies in time or if the data $\mathbf{Y}$ contains missing values. Third, the proposed Cholesky approach makes it straightforward to modify the algorithm if one changes the graph structure of the Markov random field to something else than a Markov chain. Fourth, if $\mathbf{\Psi}$ is modified directly, for instance, if optimizing the natural parameters, the covariance and the mean can be computed without needing to solve what would be the parameters of the corresponding Markov

Algorithm 2 Backward pass.

```
Input: $\{\tilde{\mathbf{\Sigma}}_{n,n}\}_{n=0}^{N}$, $\{\tilde{\mathbf{\Sigma}}_{n,n+1}\}_{n=0}^{N-1}$, $\{\tilde{\boldsymbol{\mu}}_n\}_{n=0}^{N}$
  $\mathbf{\Sigma}_{N,N} \leftarrow \tilde{\mathbf{\Sigma}}_{N,N}$
  $\boldsymbol{\mu}_N \leftarrow \tilde{\boldsymbol{\mu}}_N$
  for $n = N-1 \to 0$ do
    $\mathbf{\Sigma}_{n,n+1} \leftarrow -\tilde{\mathbf{\Sigma}}_{n,n+1}\mathbf{\Sigma}_{n+1,n+1}$
    $\mathbf{\Sigma}_{n,n} \leftarrow \tilde{\mathbf{\Sigma}}_{n,n} - \tilde{\mathbf{\Sigma}}_{n,n+1}\mathbf{\Sigma}_{n,n+1}^{\mathrm{T}}$
    $\boldsymbol{\mu}_n \leftarrow \tilde{\boldsymbol{\mu}}_n - \tilde{\mathbf{\Sigma}}_{n,n+1}\boldsymbol{\mu}_{n+1}$
  end for
Output: $\{\mathbf{\Sigma}_{n,n}\}_{n=0}^{N}$, $\{\mathbf{\Sigma}_{n,n+1}\}_{n=0}^{N-1}$, $\{\boldsymbol{\mu}_n\}_{n=0}^{N}$
```

chain. However, whatever smoothing algorithm is used, significant speeding up can be obtained by using the parameter expansion discussed in the next section.

5 Speeding Up the Inference

The variational Bayesian EM algorithm may converge extremely slowly because it updates only one variable at a time resulting in zigzagging for strongly coupled variables. It may be possible to speed up the algorithm using parameter expansion which reduces the coupling between the variables [5, 6]. For instance, parameter expanded VB-EM has been used for factor analysis [7], canonical correlation analysis [8], and common spatial patterns [9].

In state-space models, the states $\mathbf{x}_n$ and the loadings $\mathbf{C}$ are coupled through a dot product $\mathbf{C}\mathbf{x}_n$, which is unaltered if the latent space is rotated arbitrarily:

$$\mathbf{y}_n = \mathbf{C}\mathbf{x}_n = \mathbf{C}\mathbf{R}^{-1}\mathbf{R}\mathbf{x}_n. \tag{23}$$

Thus, one intuitive transformation would be $\mathbf{C} \to \mathbf{C}\mathbf{R}^{-1}$ and $\mathbf{X} \to \mathbf{R}\mathbf{X}$. In order to keep the dynamics of the latent states unaffected by the transformation, the state dynamics matrix $\mathbf{A}$ must be transformed accordingly:

$$\mathbf{R}\mathbf{x}_n = \mathbf{R}\mathbf{A}\mathbf{R}^{-1}\mathbf{R}\mathbf{x}_{n-1}, \tag{24}$$

resulting in a transformation $\mathbf{A} \to \mathbf{R}\mathbf{A}\mathbf{R}^{-1}$.

The parameter expansion is performed by parameterizing the posterior distributions with $\mathbf{R}$ and maximizing the lower bound of the marginal log likelihood (9) with respect to $\mathbf{R}$. Thus, the method optimizes the posterior distributions of several variables jointly instead of one at a time. In general, the optimal value for the parameter $\mathbf{R}$ is found using numerical optimization methods, although for a simple factor analysis model, the solution can be found analytically [7]. The optimal transformation is guaranteed not to decrease the lower bound if the initial value $\mathbf{R} = \mathbf{I}$ recovers the original posterior unaffected. The optimization is computationally efficient because the lower bound terms affected by $\mathbf{R}$ are low-dimensional.

The rotation can be optimized using nonlinear conjugate gradient (CG) algorithm. It is sufficient to find only a rough estimate of the optimal rotation

$\mathbf{R}$ to speed up the algorithm significantly, thus 10 iterations of CG was used in this paper, and CG was run after each iteration of the VB-EM algorithm. The gradients required by CG are not given in order to keep the paper concise but the derivations are straightforward.

The work required deriving the cost function for $\mathbf{R}$ for different models may be reduced by deriving the cost function for small general blocks that appear in several models. These general results may be used directly if a similar transformation for a similar block appears in another model. In order to provide modular results that can be applied to other models, the following subsections consider the following transformations of small blocks: rotating a Gaussian with an ARD prior ($\mathbf{C}$ and $\boldsymbol{\gamma}$), rotating a Gaussian with an ARD prior from left and right ($\mathbf{A}$ and $\boldsymbol{\alpha}$), and rotating a Gaussian Markov chain ($\mathbf{X}$).

5.1 Rotation of a Gaussian Variable with an ARD Prior

Let us examine the rotation of $\mathbf{C}$ as $\mathbf{C}\mathbf{R}^{-1}$ in our linear state-space model. This transformation corresponds to a trivial rotation of the posterior mean and covariance of $\mathbf{C}$. However, recall that $\mathbf{C}$ has an ARD prior with hyperparameters $\boldsymbol{\gamma}$ as defined in (6). It would be possible to simply rotate $\mathbf{C}$ without changing $\boldsymbol{\gamma}$ but it is more efficient to also transform the hyperparameters $\boldsymbol{\gamma}$. This allows $q(\mathbf{C})$ and $q(\boldsymbol{\gamma})$ to be optimized jointly. The transformation of $\boldsymbol{\gamma}$ is motivated by the VB-EM update equation (16).

The rotation of $\mathbf{C}$ can be seen as the following transformation of $q(\mathbf{C})$ and $q(\boldsymbol{\gamma})$:

$$q_*(\mathbf{C}) = \prod_{m=1}^{M} \mathcal{N}\left(\mathbf{c}_m \middle| \mathbf{R}^{-\mathrm{T}}\boldsymbol{\mu}_c^{(m)}, \mathbf{R}^{-\mathrm{T}}\boldsymbol{\Sigma}_c^{(m)}\mathbf{R}^{-1}\right), \tag{25}$$

$$q_*(\boldsymbol{\gamma}) = \prod_{d=1}^{D} \mathcal{G}(\gamma_d | \bar{a}_\gamma^{(d)}, \beta_\gamma^{(d)}), \tag{26}$$

where $\boldsymbol{\mu}_c^{(m)}$, $\boldsymbol{\Sigma}_c^{(m)}$ and $\bar{a}_\gamma^{(d)}$ are the parameters of the original distributions defined in (12),

$$\beta_\gamma^{(d)} = b_\gamma + \frac{1}{2}\left[\mathbf{R}^{-\mathrm{T}}\langle\mathbf{C}^{\mathrm{T}}\mathbf{C}\rangle\mathbf{R}^{-1}\right]_{dd}, \tag{27}$$

$\langle\cdot\rangle$ is the expectation with respect to the original posterior distribution, and $[\cdot]_{ij}$ is the element on the i-th row and j-th column. Note that the original posterior distributions $q(\mathbf{C})$ and $q(\boldsymbol{\gamma})$ are recovered by setting $\mathbf{R} = \mathbf{I}$, thus the optimal transformation is guaranteed not to worsen the posterior approximation.

The transformation affects only a small number of lower bound terms in (9) making the optimization of the rotation efficient. The transformation of $\mathbf{C}$ affects the likelihood term $\langle\log p(\mathbf{Y}|\mathbf{C},\mathbf{X},\boldsymbol{\tau})\rangle_*$ but this effect is cancelled by the transformation of $\mathbf{X}$ and can thus be ignored. The remaining terms are affected as

$$\langle \log q(\mathbf{C}) \rangle_* = -M \log |\mathbf{R}^{\text{-T}}| + \text{const}, \tag{28}$$

$$\langle \log p(\mathbf{C}|\boldsymbol{\gamma}) \rangle_* = -\frac{1}{2} \operatorname{tr} \left(\langle \mathbf{C}^{\mathrm{T}} \mathbf{C} \rangle_* \langle \operatorname{diag}(\boldsymbol{\gamma}) \rangle_* \right) + \frac{M}{2} \sum_{d=1}^{D} \langle \log \gamma_d \rangle_* + \text{const}, \tag{29}$$

$$\langle \log q(\boldsymbol{\gamma}) \rangle_* = \sum_{d=1}^{D} \langle \log \gamma_d \rangle_* + \text{const}, \tag{30}$$

$$\langle \log p(\boldsymbol{\gamma}) \rangle_* = (a_\gamma - 1) \sum_{d=1}^{D} \langle \log \gamma_d \rangle_* - b_\gamma \sum_{d=1}^{D} \langle \gamma_d \rangle_d + \text{const}, \tag{31}$$

where const is the part that is constant with respect to $\mathbf{R}$, $\langle \cdot \rangle_*$ is the expectation with respect to the transformed posterior distribution q_*, and

$$\langle \mathbf{C}^{\mathrm{T}} \mathbf{C} \rangle_* = \mathbf{R}^{\text{-T}} \langle \mathbf{C}^{\mathrm{T}} \mathbf{C} \rangle \mathbf{R}^{-1}. \tag{32}$$

The expectations $\langle \gamma_d \rangle_*$ and $\langle \log \gamma_d \rangle_*$ are computed as shown in (19). When optimizing the rotation $\mathbf{R}$, it is not necessary to compute the rotated covariance matrix nor the rotated mean of $\mathbf{C}$ because the cost function only requires $\langle \mathbf{C}^{\mathrm{T}} \mathbf{C} \rangle_*$ which can be computed efficiently using (32). After the optimization, the parameters of the posterior distribution are transformed using the optimal rotation in (25).

5.2 Double Rotation of a Gaussian Variable with an ARD Prior

The state dynamics matrix $\mathbf{A}$ should be rotated as $\mathbf{RAR}^{-1}$. However, performing this transformation exactly would make the rows of $\mathbf{A}$ dependent in the posterior approximation causing the VB-EM algorithm to be computationally much more intensive. Thus, the posterior distribution is transformed in such a way that the rows remain independent but that the transformation resembles the "true" transformation. The idea is to use a transformation which gives true values for the relevant expectations $\langle \mathbf{A} \rangle$ and $\langle \mathbf{A}^{\mathrm{T}} \mathbf{A} \rangle$ although the covariance of $\mathbf{A}$ is transformed "incorrectly". Note that the transformation of the posterior distribution is not really incorrect even if it does not correspond to $\mathbf{RAR}^{-1}$ because, in principle, the transformation can be chosen arbitrarily.

In addition to rotating $\mathbf{A}$, the ARD parameter $\boldsymbol{\alpha}$ in (5) is also transformed in order to improve the effect of the transformation. Thus, the transformation of $q(\mathbf{A})$ and $q(\boldsymbol{\alpha})$ is

$$q_*(\mathbf{A}) = \prod_{d=1}^{D} \mathcal{N} \left(\mathbf{a}_d \middle| \sum_{j=1}^{D} r_{dj} \mathbf{R}^{\text{-T}} \boldsymbol{\mu}_a^{(d)}, \left(\sum_{i=1}^{D} r_{id} \right)^2 \mathbf{R}^{\text{-T}} \boldsymbol{\Sigma}_a^{(d)} \mathbf{R}^{\text{-T}} \right) \tag{33}$$

$$q_*(\boldsymbol{\alpha}) = \prod_{d=1}^{D} \mathcal{G}(\alpha_d | \bar{a}_\alpha^{(d)}, \beta_\alpha^{(d)}) \tag{34}$$

where $\boldsymbol{\mu}_a^{(d)}$, $\boldsymbol{\Sigma}_a^{(d)}$ and $\bar{a}_\alpha^{(d)}$ are the parameters of the original distributions in (11), r_{ij} is the element $[\mathbf{R}]_{ij}$, and

$$\beta_\alpha^{(d)} = b_\alpha + \frac{1}{2}\left[\langle \mathbf{A}^{\mathrm{T}}\mathbf{A}\rangle_*\right]_{dd} . \tag{35}$$

This transformation differs from the exact transformation $\mathbf{RAR}^{-1}$ in that the cross-covariances between the rows of $\mathbf{A}$ are kept zero and the covariance of each row is not a weighted sum of the covariances of all rows but only a rotated and scaled version of the covariance of the same row. Although this transformation does not exactly correspond to $\mathbf{RAR}^{-1}$, it has the nice property that the expectations $\langle \mathbf{A}\rangle_*$ and $\langle \mathbf{A}^{\mathrm{T}}\mathbf{A}\rangle_*$ are transformed "correctly". Also, note that setting $\mathbf{R} = \mathbf{I}$ recovers the original posterior approximation unaffected.

The terms in the lower bound of the marginal log likelihood (9) are affected as

$$\langle \log q(\mathbf{A})\rangle_* = -D \log |\mathbf{R}^{\text{-T}}| - D\sum_{j=1}^{D} \log \left| \sum_{i=1}^{D} r_{ij} \right| + \text{const}, \tag{36}$$

$$\langle \log p(\mathbf{A}|\boldsymbol{\alpha})\rangle_* = -\frac{1}{2}\operatorname{tr}\left(\langle \mathbf{A}^{\mathrm{T}}\mathbf{A}\rangle_* \langle \operatorname{diag}(\boldsymbol{\alpha})\rangle_*\right) + \frac{D}{2}\sum_{d=1}^{D}\langle \log \alpha_d\rangle_* + \text{const}, \tag{37}$$

$$\langle \log q(\boldsymbol{\alpha})\rangle_* = \sum_{d=1}^{D}\langle \log \alpha_d\rangle_* + \text{const}, \tag{38}$$

$$\langle \log p(\boldsymbol{\alpha})\rangle_* = (a_\alpha - 1)\sum_{d=1}^{D}\langle \log \alpha_d\rangle_* - b_\alpha \sum_{d=1}^{D}\langle \alpha_d\rangle_* + \text{const} . \tag{39}$$

In addition, the transformation of $q(\mathbf{A})$ affects the lower bound term $\langle \log p(\mathbf{X}|\mathbf{A})\rangle_*$ but that is examined in the next subsection. The transformed expectations of $\mathbf{A}$ are

$$\langle \mathbf{A}\rangle_* = \mathbf{R}\langle \mathbf{A}\rangle\mathbf{R}^{-1}, \tag{40}$$

$$\langle \mathbf{A}^{\mathrm{T}}\mathbf{A}\rangle_* = \mathbf{R}^{\text{-T}}\left[\langle \mathbf{A}\rangle^{\mathrm{T}}\mathbf{R}^{\mathrm{T}}\mathbf{R}\langle \mathbf{A}\rangle + \sum_{d=1}^{D}\left(\sum_{i=1}^{D} r_{id}\right)^2 \boldsymbol{\Sigma}_a^{(d)}\right]\mathbf{R}^{-1}. \tag{41}$$

The expectations $\langle \alpha_d\rangle_*$ and $\langle \log \alpha_d\rangle_*$ are computed as shown in (19).

5.3 Rotation of a Gaussian Markov Chain

The state variables $\mathbf{x}_n$ are rotated as $\mathbf{R}\mathbf{x}_n$. Because the states $\mathbf{x}_n$ are not independent in the posterior approximation, the rotation is written equivalently for all the states as $(\mathbf{I}\otimes\mathbf{R})[\mathbf{X}]_:$. This results in the transformed posterior

$$q_*(\mathbf{X}) = \mathcal{N}\left([\mathbf{X}]_: \middle| (\mathbf{I}\otimes\mathbf{R})\boldsymbol{\mu}_x, (\mathbf{I}\otimes\mathbf{R})\boldsymbol{\Sigma}_x(\mathbf{I}\otimes\mathbf{R})^{\mathrm{T}}\right), \tag{42}$$

where $\boldsymbol{\mu}_x$ and $\boldsymbol{\Sigma}_x$ are the original posterior mean and covariance parameters in (10). Note that it is not necessary to compute nor store the full covariance matrix $\boldsymbol{\Sigma}_x$, because the cost function of the rotation requires only the cross-covariance $\operatorname{cov}(\mathbf{x}_{n-1}, \mathbf{x}_n)$ between consecutive states and the covariance $\operatorname{cov}(\mathbf{x}_n, \mathbf{x}_n)$.

The transformation affects only a few lower bound terms in (9). The effect on the likelihood term $\langle \log p(\mathbf{Y}|\mathbf{C}, \mathbf{X}, \boldsymbol{\tau})\rangle_*$ is cancelled by the transformation of $\mathbf{C}$. Assuming that $q(\mathbf{A})$ is transformed as described in the previous subsection, the lower bound terms are affected as

$$\langle \log q(\mathbf{X})\rangle_* = -(N+1)\log|\mathbf{R}| + \text{const}, \tag{43}$$

$$\begin{aligned}\langle \log p(\mathbf{X}|\mathbf{A})\rangle_* = \operatorname{tr}\Bigg(&-\frac{1}{2}\boldsymbol{\Lambda}_0\langle \mathbf{x}_0\mathbf{x}_0^{\mathrm{T}}\rangle_* + \boldsymbol{\Lambda}_0\mathbf{m}_0\langle \mathbf{x}_0\rangle_*^{\mathrm{T}} + \sum_{n=1}^{N}\Bigg[-\frac{1}{2}\langle \mathbf{x}_n\mathbf{x}_n^{\mathrm{T}}\rangle_* \\ &-\frac{1}{2}\langle \mathbf{A}^{\mathrm{T}}\mathbf{A}\rangle_*\langle \mathbf{x}_{n-1}\mathbf{x}_{n-1}^{\mathrm{T}}\rangle_* + \langle \mathbf{A}\rangle_*\langle \mathbf{x}_{n-1}\mathbf{x}_n^{\mathrm{T}}\rangle_*\Bigg]\Bigg),\end{aligned} \tag{44}$$

where $\langle \mathbf{A}\rangle_*$ and $\langle \mathbf{A}^{\mathrm{T}}\mathbf{A}\rangle_*$ are defined in (40) and (41), respectively, and

$$\langle \mathbf{x}_n\rangle_* = \mathbf{R}\langle \mathbf{x}_n\rangle, \tag{45}$$

$$\langle \mathbf{x}_n\mathbf{x}_n^{\mathrm{T}}\rangle_* = \mathbf{R}\langle \mathbf{x}_n\mathbf{x}_n^{\mathrm{T}}\rangle\mathbf{R}^{\mathrm{T}}, \tag{46}$$

$$\langle \mathbf{A}\rangle_*\langle \mathbf{x}_{n-1}\mathbf{x}_n^{\mathrm{T}}\rangle_* = \mathbf{R}\langle \mathbf{A}\rangle\langle \mathbf{x}_{n-1}\mathbf{x}_n^{\mathrm{T}}\rangle\mathbf{R}^{\mathrm{T}}, \tag{47}$$

$$\begin{aligned}\operatorname{tr}\left(\langle \mathbf{A}^{\mathrm{T}}\mathbf{A}\rangle_*\langle \mathbf{x}_{n-1}\mathbf{x}_{n-1}^{\mathrm{T}}\rangle_*\right) &= \operatorname{tr}\left(\langle \mathbf{A}\rangle^{\mathrm{T}}\mathbf{R}^{\mathrm{T}}\mathbf{R}\langle \mathbf{A}\rangle\langle \mathbf{x}_{n-1}\mathbf{x}_{n-1}^{\mathrm{T}}\rangle\right) \\ &\quad + \sum_{d=1}^{D}\left(\sum_{k=1}^{D} r_{kd}\right)^2 \operatorname{tr}\left(\boldsymbol{\Sigma}_{\mathbf{A}}^{(d)}\langle \mathbf{x}_{n-1}\mathbf{x}_{n-1}^{\mathrm{T}}\rangle\right).\end{aligned} \tag{48}$$

Note that the sum over n in $\langle \log p(\mathbf{X}|\mathbf{A})\rangle_*$ can be computed independently of $\mathbf{R}$ before starting the optimization of $\mathbf{R}$ in order to reduce the computational cost.

6 Experiments

6.1 Artificial Data

The method was tested on an artificial dataset, which was generated using the model with known parameter values. We generated $N = 400$ latent states $\mathbf{x}_n$ by using the following state dynamics matrix:

$$\mathbf{A} = \begin{bmatrix} \cos(\omega) & -\sin(\omega) & 0 & 0 \\ \sin(\omega) & \cos(\omega) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \tag{49}$$

where $\omega = 0.3$. Thus, the four latent signals in $\mathbf{X}$ are as follows: the first and the second signals are noisy oscillators, the third signal is Gaussian random walk, and the fourth signal is Gaussian white noise. The loading matrix $\mathbf{C}$ for projecting the observations with dimensionality $M = 30$ was sampled from the Gaussian distribution with zero mean and unit variance for each element. The variance of the observation noise was set to $\tau^{-1} = 9$.

The dataset was used to estimate the parameters with the variational Bayesian learning method. The learning was performed both with and without the rotations in order to compare the effect of the rotations on the performance. Both methods used the same model and initialization. The dimensionality of the latent states was set to $D = 8$, which is larger than the true dimensionality, to let the ARD prior prune out any irrelevant dimensions. The hyperparameters were given broad priors by setting $a_\alpha = b_\alpha = a_\gamma = b_\gamma = a_\tau = b_\tau = 10^{-5}$, $\boldsymbol{\mu}_0 = \mathbf{0}$ and $\boldsymbol{\Lambda}_o = 10^{-3} \cdot \mathbf{I}$.

The approximate posterior distributions of the variables were initialized by using the VB-EM update formulas (13)–(17) but ignoring the variational messages from the child nodes, that is, taking into account only the parent nodes. This means roughly that the variables were initialized from the prior. For instance, the mean of $\boldsymbol{\alpha}$, $\boldsymbol{\gamma}$ and $\boldsymbol{\tau}$ were set to ones, the mean of $\mathbf{A}$ to zeros and the covariance of $\mathbf{A}$ to the identity matrix. In order to get some initial latent space spanned, the mean of the loading matrix $\mathbf{C}$ was initialized randomly from the standard Gaussian distribution and the covariance was set to zero. The latent states $\mathbf{X}$ were not initialized because they were updated first in the VB-EM algorithm.

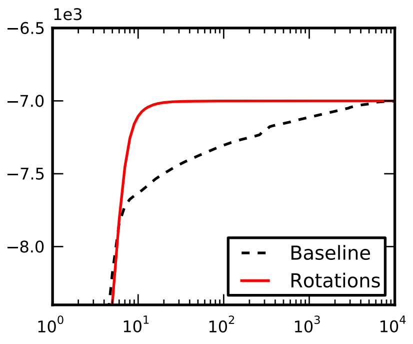
The performance of the methods was measured by monitoring the lower bound of the marginal log likelihood (9) and root-mean-square error (RMSE) on the training and test sets. The test set was created by removing training data y_{mn} randomly with probability 0.8, thus approximately 20% of the data was used for training.

Figure 1 shows the performance of both methods as a function of iterations. The number of iterations is used for simplicity as the computational cost of the rotations is negligible. The standard learning method converges in approximately 10000 iterations whereas the method with rotations converges in 10–20 iterations based on the lower bound of the marginal log likelihood in Fig. 1a. The rotations do not only affect the lower bound but also the reconstruction of the data. The standard method overfits at the beginning of the learning phase as can be seen from Fig. 1b and the predictions are improved very slowly as the test error shows in Fig. 1c. In comparison, the method with rotations finds the solution orders of magnitude faster.

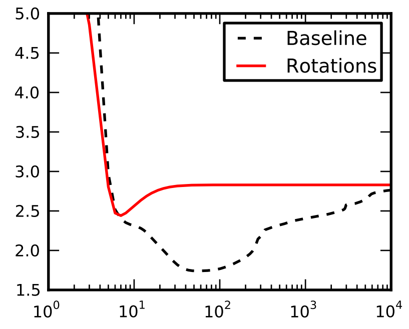
6.2 Weather Data

The methods were tested on a real-world weather dataset provided by the Helsinki Testbed project of the Finnish Meteorological Institute (FMI). From the large dataset, we used temperature measurements in Southern Finland over a period of almost two years with an interval of ten minutes resulting in $N = 89202$ time instances.[1] Measurements from some weather stations were badly corrupted

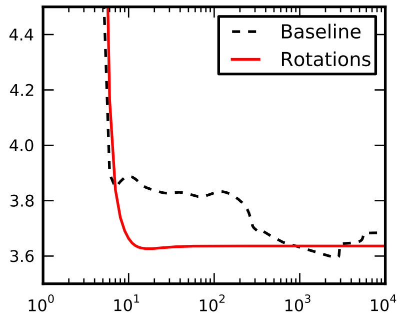
[1] The data is available at `http://users.ics.aalto.fi/jluttine/ecml2013/` under the FMI Open Data License.



(a) Lower bound



(b) Training RMSE



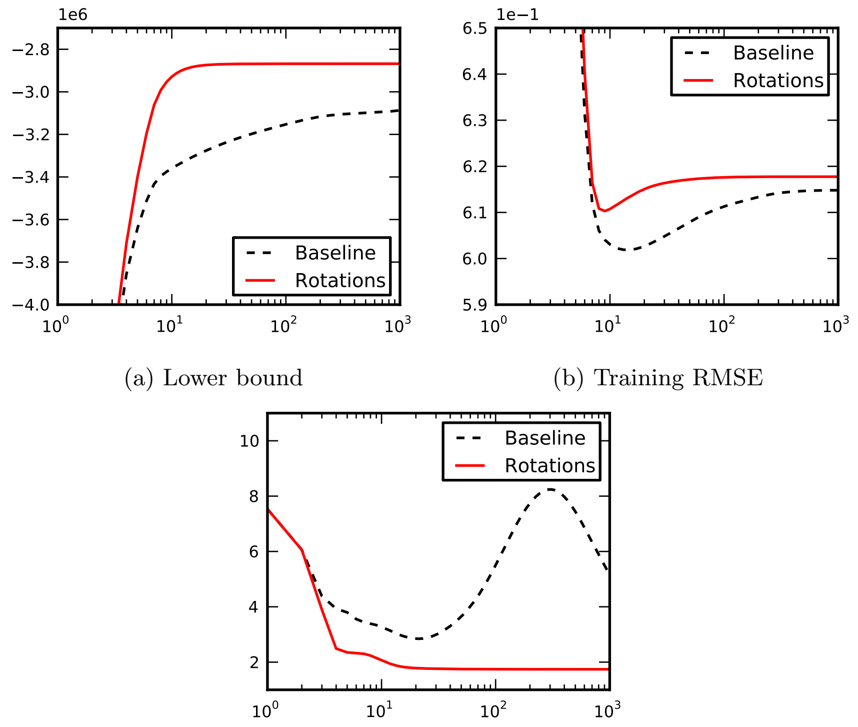
(c) Test RMSE

Fig. 1. Results of the artificial experiment. (a) The marginal log likelihood lower bound, (b) training error, and (c) testing error shown as a function of iterations for the standard learning method (baseline) and the proposed method using rotations. Note that the x-axis has a logarithmic scale.

as discussed in [15] and were therefore discarded. Thus, $M = 66$ stations remained for the analysis. From the remaining data, approximately 35% of the measurements were missing.

The standard method and the proposed method using rotations were used to estimate the linear state-space model from the data. The model used latent space dimensionality $D = 10$ and broad hyperpriors as in the artificial experiment. The initialization was done similarly as in the artificial experiment. Test data was formed by removing training data randomly with probability 0.2 and completely for periods of one day at ten day intervals resulting in a large number of short gaps in the training data.

Figure 2 shows the lower bound of the marginal log likelihood, the training error and test error for both methods. Based on the lower bound in Fig. 2a, the standard method has not converged in 1000 iterations and the progress is extremely slow, whereas the method with rotations converges in 20–30 iterations.



(a) Lower bound

(b) Training RMSE

(c) Test RMSE

Fig. 2. Results of the weather experiment. (a) The marginal log likelihood lower bound, (b) training error, and (c) testing error shown as a function of iterations. Note that the x-axis has a logarithmic scale.

The computational cost of the rotations is again negligible and can be ignored for simplicity. Similarly to the artificial experiment, the standard method overfits at the beginning as can be seen in Fig. 2b. The performance difference of the learning methods can be seen clearly from the test error in Fig. 2c as the test error for the standard method is significantly larger and decreasing very slowly. From these measures it is evident that the standard learning method has not yet converged in 1000 iterations and it might require several thousands iterations more to reach convergence.

7 Conclusions

The paper presented a parameter expansion for improving the speed of the variational Bayesian inference of linear state-space models. The expansion was based on optimizing the rotation of the latent space, which corresponds to a parameterized joint optimization of multiple variables. The transformations improved the speed of the inference by orders of magnitude compared to the standard VB-EM algorithm as shown in the experiments for artificial and real-world data. Thus, the proposed parameter expansion should be a standard technique if using VB inference for variants of linear state-space models in problems that are not very small.

The paper also gave a new perspective on estimating the posterior distribution of the states in the VB-EM algorithm. The states were estimated by using a smoothing algorithm based on the block LDL decomposition instead of the standard filters and smoothers used in previous papers. This approach reduced the number of required matrix inversions and avoided a model augmentation. In addition, as the algorithm is based on the LDL decomposition, one can utilize existing sparse LDL algorithms in order to generalize the smoothing algorithm, for instance, to other Gaussian Markov random fields with different graph structure.

A Python implementation of the presented method is published as a part of the Bayesian Python (BayesPy) package under the GNU General Public License.[2] In addition, the data and the scripts for running the experiments shown in the paper are available under open licenses.[3]

Acknowledgements. This work was supported by the Helsinki Doctoral Programme in Computer Science (Hecse) and the Academy of Finland (Finnish Centre of Excellence in Computational Inference Research COIN, 251170). The author would like to thank the Finnish Meteorological Institute and the Helsinki Testbed project for providing the dataset and sharing it under the FMI Open Data License.

References

1. Bar-Shalom, Y., Li, X.R., Kirubarajan, T.: Estimation with Applications to Tracking and Navigation. Wiley-Interscience (2001)
2. Shumway, R.H., Stoffer, D.S.: Time Series Analysis and Its Applications. Springer (2000)
3. Beal, M.J., Ghahramani, Z.: The variational Bayesian EM algorithm for incomplete data: with application to scoring graphical model structures. Bayesian Statistics **7** (2003) 453–464
4. Bishop, C.M.: Pattern Recognition and Machine Learning. 2nd edn. Information Science and Statistics. Springer, New York (2006)
5. Liu, C., Rubin, D.B., Wu, Y.N.: Parameter expansion to accelerate EM: the PX-EM algorithm. Biometrika **85** (1998) 755–770
6. Qi, Y.A., Jaakkola, T.S.: Parameter expanded variational Bayesian methods. [16] 1097–1104
7. Luttinen, J., Ilin, A.: Transformations in variational Bayesian factor analysis to speed up learning. Neurocomputing **73** (2010) 1093–1102

[2] BayesPy is available at `https://github.com/jluttine/bayespy`.

[3] The material is available at `http://users.ics.aalto.fi/jluttine/ecml2013/`.

8. Klami, A., Virtanen, S., Kaski, S.: Bayesian canonical correlation analysis. Journal of Machine Learning Research **14** (2013) 899–937
9. Kang, H., Choi, S.: Probabilistic models for common spatial patterns: Parameter-expanded EM and variational Bayes. In: Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence. (2012)
10. Rauch, H.E., Tung, F., Striebel, C.T.: Maximum likelihood estimates of linear dynamic systems. AIAA Journal **3**(8) (1965) 1445–1450
11. Beal, M.J.: Variational algorithms for approximate Bayesian inference. PhD thesis, Gatsby Computational Neuroscience Unit, University College London (2003)
12. Barber, D., Chiappa, S.: Unified inference for variational Bayesian linear Gaussian state-space models. [16]
13. Eubank, R.L., Wang, S.: The equivalence between the Cholesky decomposition and the Kalman filter. The American Statistician **56**(1) (2002) 39–43
14. Kalman, R.E., Bucy, R.S.: New results in linear filtering and prediction theory. Journal of Basic Engineering **85** (1961) 95–108
15. Luttinen, J., Ilin, A., Karhunen, J.: Bayesian robust PCA of incomplete data. Neural Processing Letters **36**(2) (2012) 189–202
16. Schölkopf, B., Platt, J., Hoffman, T., eds.: Advances in Neural Information Processing Systems 19. MIT Press, Cambridge, MA (2007)