Supervised Learning in Dynamic Bayesian Networks

Shamim Nemati
School of Engineering and Applied Sciences
Harvard University
shamim@seas.harvard.edu

Ryan P. Adams
School of Engineering and Applied Sciences
Harvard University
rpa@seas.harvard.edu

Abstract

Dynamic Bayesian networks (DBNs) are powerful tools for unsupervised discovery of structure in time series, using generative learning algorithms such as expectation maximization (EM). While in many problems, the DBN latent variables are the primary quantities of interest for estimation purposes, unsupervised learning is often a way to perform automatic feature learning for a downstream supervised task, such as time series classification and sequential labeling. We show that the DBNs model structure allow for combination of the two aforementioned objectives using backpropagation. Our proposed architecture is motivated by the problem of nonstationary multivariate time series classification and sequential labeling. We derive supervised learning algorithms for parameter estimation and inference of latent variables in two commonly used DBN models for such time series, namely the switching vector autoregressive (SVAR) model and the switching Kalman filter (SKF). The result of our procedure can be interpreted as a compactly-parameterized recurrent neural network, which can be pre-trained using EM. Using examples from three nonstationary neurophysiological time series cohorts, we show how discriminatively-discovered dynamics in time series lead to a significant improvement in classification performance over methods based on feature learning via EM, as well as other commonly used time series classifiers.

1 Introduction

Dynamic Bayesian networks (DBNs) are a probabilistic framework for modeling shared temporal structure (or *dynamics*) in multivariate time series cohorts [19]. The modeling power and computational efficiency of DBNs have made them popular for many unsupervised time series problems [7, 15]. DBN models typically include one or more layers of discrete (e.g., hidden Markov model) or continuous (e.g., Kalman filter) latent variables, or a combination thereof (e.g., switching Kalman filter), arranged in a directed acyclic graph (DAG) structure. This structure allows for parametric (or deterministic) approximate inference of latent variables using variants of the belief propagation (BP) algorithm [19, 9]. Learning model parameters for this class of models is most often done using maximum likelihood estimation. The marginal likelihood is often challenging to compute, however, and so maximum likelihood learning is most often performed using a variant of the expectation maximization (EM) algorithm [3]. EM relies on estimates of the marginal distributions over latent variables. In some cases — notably HMMs and Gaussian linear dynamical systems — these marginals are exactly computable using dynamic programming; effective approximate procedures have been developed for other popular DBN models [19].

While in many problems, model fitting and prediction of future time series values are the primary tasks of interest, unsupervised learning is often used to learn *features* for a downstream supervised (classification) task [14, 15]. In these cases, likelihood-based learning may be suboptimal; the latent dynamics that are important to the supervised target may only be weakly related to those that are best for explaining the raw statistics of the time series. Additionally, such methods may be hamstrung by the shortcomings of approximate inference, or the model may be underspecified with respect to the nuanced features associated with the outcomes of interest. For instance, in a neurophysiological experiment involving EEG recordings, it may be the case that only a single low amplitude oscillation

is the distinguishing feature of successful trials, and therefore a reduced-model specifically trained to capture that oscillation may provide a more parsimonious solution to the problem of predicting outcomes of each trial. It is therefore desirable to learn models of time series dynamics in which the latent variables are directly tuned towards the supervised task of interest.

In this work, we propose a method for learning latent dynamics that result in discriminatively-useful features of time series. Rather than depending on label-free unsupervised learning to discover relevant features of the time series, we build a system that expressly learns the dynamics that are most relevant for classifying time series labels. Our goal is to obtain compact representations of nonstationary and multivariate time series (*representation learning*)[1]. Our approach is based on the idea of using the inferred marginals of hidden variables as inputs to a gradient-based supervised learner such as logistic regression or a multi-layer neural network. If the loss function is differentiable, it will be possible to compute the gradient in terms of these marginals, and then backpropagate this loss through the message passing inference procedure. To accomplish this we use a connection between DBNs and artificial neural networks (ANNs) to perform inference and learning in a manner analogous to backpropagation in neural networks [21]. This connection stems from the observation that the directed acyclic graph structure of a state-space model can be unrolled both as a function of time and inference steps to yield a deterministic neural network with parameters that are tied across time. Thus, the parameters governing the DBN model can be learned in a manner analogous to that of a neural network. Indeed, the resulting system can be viewed as a compactly-parameterized recurrent neural network (RNN) [25]. Although the standard use of RNNs has been for time series prediction (network output is the predicted input time series in the future) or sequential labeling (when output is a label sequence associated with the input data sequence), with additional processing layers one may obtain a time series classifier from this class of models [8]. Nevertheless, RNNs have proven hard to train, since the optimization surface tends to include multiple local minima and is sensitive to gradient explosion and decay [25]. Moreover, standard RNNs are "black box" models that make it difficult to incorporate mechanistic models of the underlying systems. The framework proposed here helps address both of these shortcomings. First, knowledge of the underlying system dynamics can be directly incorporated into the state-space models that constitute the basic building blocks of a dynamic Bayesian network. Secondly, equipped with a generative model, we can rely on unsupervised pre-training via EM to systematically initialize the parameters of the equivalent RNN; this is analogous to pre-training very large multi-layered neural networks [6].

Discriminative approaches to learning in graphical models can be broadly classified into discrete versus continuous latent variable models. Some of the recent works within the first category include: *structured output classification* [18] where the hidden discrete states of an HMM are designed to correspond to target labels which are observed in the training data and thus can be learned using outcome-discriminative learning, and *approximate marginal inference* in conditional random fields [5, 24, 4]. Supervised learning techniques for learning of HMMs and related conditional random fields have been shown to outperform generative maximum likelihood learning in many tasks [17, 13, 26, 2]. More recently, It has empirically been shown that *marginalization-based* learning via empirical risk minimization gives better results than likelihood based approximations in the presence of model mis-specification [4]. In the continuous domain, Kim and Pavlovic [12] used a gradient based approach to learning parameters of a conditional state-space model. They assumed ground truth for continuous latent state is known during the learning phase, and provided analytical gradients for the conditional likelihood of the latent state variables with respect to the state-space model parameters. In contrast, here we propose a framework for gradient-based learning in hybrid discrete and continuous state-space models, and given differentiable but otherwise arbitrary cost functions.

2 Unrolling Dynamic Bayesian Networks into Deep Neural Networks

Assume we are given a collection of N multivariate time series and the associated outcome variables: $\{(y^{(1)}, O^{(1)}), (y^{(2)}, O^{(2)}), \cdots, (y^{(N)}, O^{(N)})\}$, where each of the time series is M-dimensional and the n-th time series $y^{(n)}$ is of length T_n. The corresponding label $O^{(n)}$ can be a scalar, such as a discrete label, or may itself be a length-T_n time series vector that assigns a label to each instant. Such cohort time series arise in many areas of science and engineering, including large clinical databases of patient vital sign time series [15], multi-trial neurophysiological experiments involving multichannel recordings of neuronal activities, and in problems involving audio transcription and recursive handwriting recognition [8]. Our objective is to find shared dynamic features across the different time series that are predictive of the outcomes of interest.

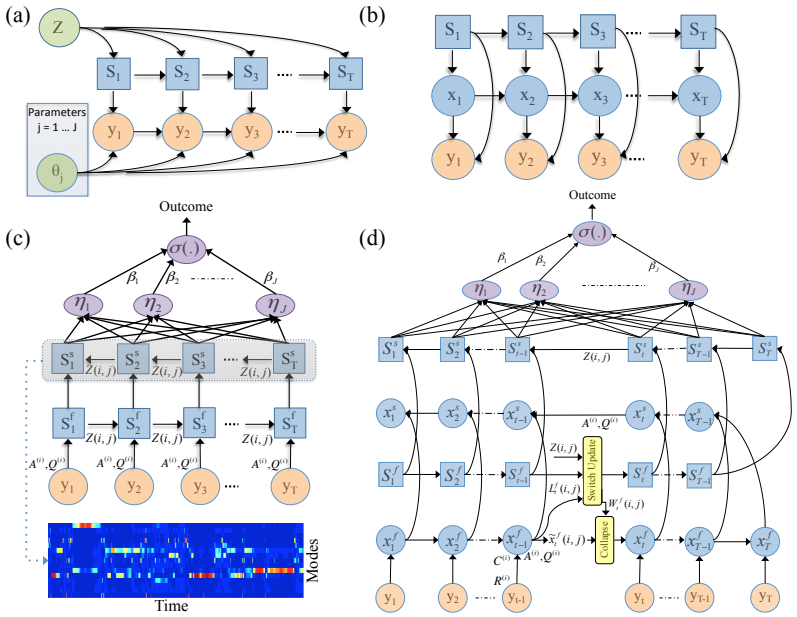

Figure 1: Supervised learning in dynamic Bayesian networks. (a) Graphical model representation of the switching vector autoregressive (switching VAR), and the switching Kalman filter (SKF) are shown in panels (a) and (b), respectively. Panel (c) and (d) show the unrolled representation (with respect to time and inference steps) of the two models, with an added logistic regression layer (elliptic nodes) which utilize the marginals over the discrete latent variables as features for time series classification (an example of inferred marginals is shown at the bottom of the panel (c)). These unrolled structures, which resemble recurrent neural networks, allow for efficient supervised learning and inference via error backpropagation (see appendix for the analytic expressions of the gradients).

2.1 Dynamic Bayesian Networks

The DAG structure of a DBN model represents conditional dependence structure within a discrete time step and causal conditional dependence structure across time steps (see Fig. 1), ultimately producing the observed time series. This graphical structure results in a factorization of the overall joint distribution that ideally would have tractable local distributions. At time slice t, let the graph consist of one or more (discrete or continuous) observed variables $\mathbf{y}_t^{(n)}$, and continuous and/or discrete latent variables $\mathbf{x}_t^{(n)}$ and $s_t^{(n)}$, respectively. We focus on cases where an efficient, deterministic exact or approximate algorithm exists for inference of marginals $P_{\mathbf{\Theta}}(x_t^{(n)} \mid \{\mathbf{y}_t^{(n)}\})$ and $P_{\mathbf{\Theta}}(s_t^{(n)} \mid \{\mathbf{y}_t^{(n)}\})$. Typically, such efficiency arises from a dynamic programming or *message passing* procedure, such as the forward-backward algorithm or the Kalman filter/smoother. Here $\mathbf{\Theta}$ denotes the set of all parameters defining the DBN model. Without loss of generality, let us assume we are interested in minimizing a differentiable loss function of the marginals over the discrete latent variables. We construct this loss by assuming that there exists a differentiable parametric function $\boldsymbol{\mu}_\beta(P_{\mathbf{\Theta}}(s_1^{(n)} \mid \{\mathbf{y}_t^{(n)}\}), \ldots, P_{\mathbf{\Theta}}(s_{T_n}^{(n)} \mid \{\mathbf{y}_t^{(n)}\}))$ that produces an estimate $\hat{O}^{(n)}$ of the global label given these marginals. This can be motivated, for example, by the idea that the label of a switching dynamical system would be determined by proportion of time spent in different states. The overall learning objective, in terms of $\mathbf{\Theta}$ and β, can then be written

$$\mathbf{\Theta}^\star, \beta^\star = \underset{\mathbf{\Theta}, \beta}{\operatorname{argmin}} \sum_{n=1}^{N} \ell(\boldsymbol{\mu}_\beta(P_{\mathbf{\Theta}}(s_1^{(n)} \mid \{\mathbf{y}_t^{(n)}\}), \ldots, P_{\mathbf{\Theta}}(s_{T_n}^{(n)} \mid \{\mathbf{y}_t^{(n)}\})), O^{(n)}), \tag{1}$$

where $\ell(\hat{O}, O)$ is the loss function. We will write the overall objective function in terms of $\mathbf{\Theta}$ and β (for a given data set) as $\mathcal{L}(\mathbf{\Theta}, \beta)$. As a notational shortcut, we will write $\boldsymbol{\mu}^{(n)}(\mathbf{\Theta}, \beta)$ for the nth marginal-based label predictor as function of $\mathbf{\Theta}$ and β, with the underlying time series data implied. Below we examine marginal-based supervised learning in two DBN examples.

2.1.1 Switching Linear Dynamical System

The switching linear dynamical system (SLDS) [19] models time series using two layers of hidden state evolution (see Fig. 1(b)). In the high-level layer, the time series evolves through a set of J latent states according to a Markov chain. In the lower level, each of these states corresponds to a unique linear dynamical system that evolves a continuous hidden state and produces the observed time series. The generative model is as follows: a latent process for each time series $s_t^{(n)} \in \{1, \cdots, J\}$ evolves as a Markov chain with initial distribution $\pi^{(n)}$ and $J \times J$ transition matrix Z. Each of the N series has an unobserved continuous state variable $\mathbf{x}_t^{(n)} \in \mathbb{R}^D$ that evolves according to linear dynamics which are determined by the current latent state $s_t^{(n)}$, producing observations $\mathbf{y}_t^{(n)}$. The jth linear system has state dynamics $A^{(j)}$, observation matrix $C^{(j)}$, state noise covariance $Q^{(j)}$, and observation noise covariance $R^{(j)}$:

$$\mathbf{x}_t^{(n)} = A^{(s_t^{(n)})}\mathbf{x}_{t-1}^{(n)} + \mathbf{v}_t^{(n)} \qquad \mathbf{v}_t^{(n)} \sim \mathcal{N}(\mathbf{0}, Q^{(s_t^{(n)})}) \tag{2}$$

$$\mathbf{y}_t^{(n)} = C^{(s_t^{(n)})}\mathbf{x}_t^{(n)} + \mathbf{w}_t^{(n)} \qquad \mathbf{w}_t^{(n)} \sim \mathcal{N}(\mathbf{0}, R^{(s_t^{(n)})}). \tag{3}$$

We refer to these state-specific dynamics together as $\Delta^{(j)} = \{A^{(j)}, Q^{(j)}, C^{(j)}, R^{(j)}\}$ (also known as a *mode*).

2.1.2 Switching Vector Autoregression Model

The switching vector autoregression (VAR) model is an important special case of the switching linear dynamical system, where the latent continuous dynamics describe a VAR process of order P, i.e., the continuous state vectors correspond to lagged versions of the observations. Assuming a library of J latent states, the resulting switching linear dynamical model describes an equivalent switching VAR model of the form:

$$\mathbf{y}_t^{(n)} = \sum_{p=1}^{P} \mathbf{a}_p^{(s_t^{(n)})}\mathbf{y}_{t-p}^{(n)} + \mathbf{w}_t^{(n)} \qquad \mathbf{w}_t^{(n)} \sim \mathcal{N}(0, Q^{(s_t^{(n)})}), \tag{4}$$

with the multivariate autoregressive model coefficient matrices $\mathbf{a}_p^{(j)}$ of size $M \times M$, with maximal time lag $p = 1 \ldots P$, and noise term $\mathbf{w}_t^{(j)}$ with covariance $Q^{(j)}$. Here the state-specific parameters are $\Delta^{(j)} = \{\mathbf{a}_1^{(j)}, \cdots, \mathbf{a}_P^{(j)}, Q^{(j)}\}$. Fig. 1(a) depicts the graphical model representation of a switching VAR model, which is equivalent to an HMM with continuous-valued autoregressive observations. Henceforth we use $\mathbf{\Theta} = \{\{\Delta^{(j)}\}_{j=1}^{J}, Z, \pi^{(n)}\}$ to denote the set of all model parameters defining the DBN.

2.2 Marginal-based Label Predictions

Essentially any standard supervised learning algorithm can incorporate hidden-state marginals as features used to produce a label. Here we examine two significant cases of interest: where there is a global label for the time series, and where the supervised target is itself an aligned time series. We describe the classification setting, but these approaches would generalize directly to continuous labels and more structured settings.

Global Label from Hidden State Proportions We assume that each label $O^{(n)}$ can take on one of K possible outcomes, and can be modeled using a softmax classifier with parameters β. The inputs to the logistic regressor are the marginal estimates of expected proportion of the time that is spent in each of the latent discrete states[1]:

$$\mu_k^{(n)}(P_{\mathbf{\Theta}}(s_1^{(n)}), \cdots, P_{\mathbf{\Theta}}(s_{T_n}^{(n)})) = \frac{\exp\{\beta_{k,0} + \beta_k^{\mathsf{T}}\eta^{(n)}\}}{\sum_{k'=1}^{K}\exp\{\beta_{k',0} + \beta_{k'}^{\mathsf{T}}\eta^{(n)}\}}, \quad \eta_j^{(n)} = \frac{1}{T_n}\sum_{t=1}^{T_n} P_{\mathbf{\Theta}}(s_t^{(n)} = j)$$

where the β_k are length J weight vectors, $\beta_{k,0}$ are biases, and the $\eta^{(n)}$ are length J vectors of hidden state proportions, which are weighed in a softmax function with row vector parameters β_k. We take the classification cost function to be the negative log likelihood (*negentropy*) of the outcome labels, given the time series:

[1] In the remainder of the paper, we will write the data-conditional marginals as $P_{\Theta}(s_t^{(n)})$ for compactness.

$$-\log \Pr(\mathbf{O}|\boldsymbol{\mu}(\Theta,\beta)) = -\sum_{n=1}^{N}\sum_{k=1}^{K} \mathbf{O}_k^{(n)} \log \mu_k^{(n)}(\Theta,\beta). \tag{5}$$

Training can then be performed using the gradient of the logistic regression log likelihood, learning the β_k as well as backpropagating through the $\eta^{(n)}$ to fit the dynamics parameters $\boldsymbol{\Theta}$ proxied by the marginals .

Sequential Labels from Local Marginals Some tasks require a time-aligned sequence of labels, in a similar fashion to a conditional random field, i.e., $O^{(n)}$ is a sequence of size T_n, with each label taking one of K discrete values. Here, the marginal-based predictor produces a label at each time step, which is the result of a softmax applied to the marginal estimates:

$$\mu_{t,k}^{(n)}(P_{\boldsymbol{\Theta}}(s_1^{(n)}),\cdots,P_{\boldsymbol{\Theta}}(s_{T_n}^{(n)})) = \frac{\exp\{\beta_{k,0} + \beta_k^{\mathsf{T}}\eta_t^{(n)}\}}{\sum_{k'=1}^{K}\exp\{\beta_{k',0} + \beta_{k'}^{\mathsf{T}}\eta_t^{(n)}\}}, \qquad \eta_{t,j}^{(n)} = P_{\boldsymbol{\Theta}}(s_t^{(n)} = j)\,.$$

Here the $\eta_t^{(n)}$ are length-J marginal estimates at each time t, being weighed in a softmax classifier with parameters β_k. We take the classification objective to be the negative log likelihood (*negentropy*) of the outcome labels, given the time series:

$$-\log \Pr(\mathbf{O}|\boldsymbol{\mu}(\Theta,\beta)) = -\sum_{n=1}^{N}\sum_{t=1}^{T_n}\sum_{k=1}^{K} \mathbf{O}_{t,k}^{(n)} \log \mu_{t,k}^{(n)}(\Theta,\beta). \tag{6}$$

Again, the standard logistic regression likelihood can be used for training, with gradients of β directly available and gradients of $\boldsymbol{\Theta}$ available via backpropagation.

2.3 Optimization

2.3.1 Gradient Calculations via Backpropagation

The analytic gradients of any of the above supervised learners in terms of β and $\boldsymbol{\Theta}$ can be calculated using a two-pass algorithm. The forward pass involves running inference to approximate the marginal distributions over the latent variables, and subsequently evaluating the predictor $\boldsymbol{\mu}(\cdot)$. The backward pass utilizes the chain rule via reverse mode differentiation (backpropagation) to obtain the gradients of the overall loss in terms of parameters. Since DBN inference algorithms involve a sequence of differentiable operations, the derivative of the loss function with respect to the discrete and continuous marginals, and finally model parameters $\boldsymbol{\Theta}$ can be calculated efficiently. To accomplish this, it helps to visualize an unrolled version of the dynamic Bayesian network forward-backward inference procedure, in which snapshots of a random variable at times t and $t+1$ are distinct deterministic (fixed at the values determined by the inference step) nodes in a feedforward neural network (see Fig. 1(c) and (d)). Note that since the overall gradient over a time series cohort is the sum of the individual gradients, gradient calculations can be done in parallel for each time series. (Analytic expressions for the gradients with respect to parameters of the switching VAR and the SKF are given in the appendix.)

The above gradient can be directly plugged into an optimization routine [22, 25] to optimize $\boldsymbol{\Theta}$ and β. When optimizing over large time series cohorts, we found that an stochastic optimization approach—using mini-batches with a few iterations per batch and a momentum term—yielded improved generalization performance with significant speed up. We also found it useful to implement an early stopping criteria based on classification performance on a held-out validation set.

2.3.2 Initialization via Expectation Maximization

One reason for the resurgence of neural networks as architectures for supervised learning is due to the realization that probabilistic unsupervised learning can often lead to fruitful parameter initialization [6]. Although such unsupervised pre-training has fallen out of favor for visual object recognition, we find it to be a natural fit for the discriminative learning we present here. Specifically, good initial parameters can easily be found using the expectation maximization algorithm [19] for unsupervised learning from the time series, in the absence of label information. This observation supports the intuition that although likelihood based learning and the resulting features may not necessarily be good for discriminating between classes, they nevertheless capture the structure of the input data and can provide a good starting point for discriminative fine-tuning to make rapid progress [6].

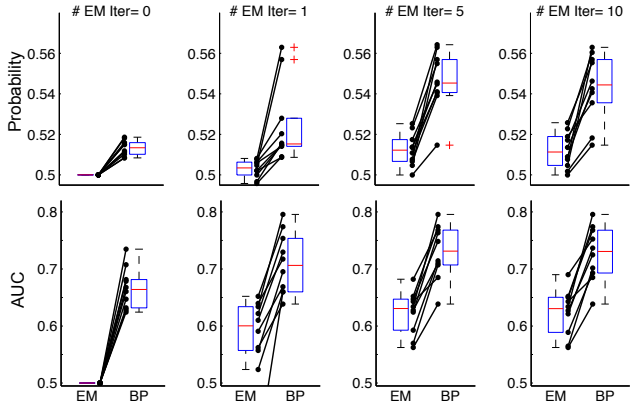

Figure 2: Effects of EM-based pretraining on the performance of supervised learning on the LFP decoding experiment. Each panel shows classification performance over ten folds (testing set performance) based on 0, 1, 5, 10, and 20 iterations of expectation maximization (EM) followed by 30 iterations of the supervised learning via backpropagation (BP). The cost function that is being optimized (Bernoulli probability of outcomes) is shown in the top row, and the area under receiver operating curve (AUC) is presented in the bottom row.

3 Experiments and Results

For comparison purposes, we considered four algorithms that have been shown to outperform classical techniques such as dynamic time warping (DTW) [27], in applications involving long sequences of noisy multivariate time series [16]. The linear dynamical systems (LDS) and the complex linear dynamical systems (CLDS) [16] approaches to time series classification are based on the assumption that all time series in a cohort share a set of latent factors with Gaussian linear dynamics. In practice, all time series are stacked to form the observation sequence for a linear state-space model, where the rows of the observation matrix describe the proportions of latent dynamics included in each time series; these row-vectors are then taken as features for a downstream classification task. The CLDS model allows for complex entries in the observation matrix, which has the additional advantage of including a phase term for time series-specific alignment of latent dynamics (typically, absolute value of the observation matrix constitutes the final feature set). An EM algorithm is used to learn all the parameters of the dynamical system on the training data, followed by learning the parameters of a classifier. Evaluation of a testing fold involves learning of the observation matrix, keeping all other model parameters constant, followed by application of the classifier from the training step. Our third time series method was based on a mixture of Gaussian (MOG). We learned K MOG models on subsets of time series associated with each of the classes. A testing fold time series was then assigned to the class corresponding to the MOG with the greatest posterior probability. As our fourth method, we considered two RNN architectures; the Layer Recurrent Neural Networks (LRNN) implementations in MATLAB®'s artificial neural network package, and the Long Short-Term Memory (LSTM) model implemented within the RNNLib package [8]. Hyper parameters of these models (including the input block length, number of hidden layers, parameters of adaptive weight noise regularization, magnitude of the momentum and the learning rate for optimization) were optimized using the Bayesian Optimization technique for global optimization [23]. For the purpose of time series classification, the target sequence for the RNN was taken to be the one-hot coded class label for the time series. The final classification was then based on summing over the inputs to each of the K output units at all time points in the output sequence, then applying the softmax function [8].

In all relevant cases, logistic regression was used as the final classifier. All the results presented here are 10-fold cross-validated, and the median (and interquartiles) performance over the testing-folds are presented. We report the area under the receiver operating curve (AUC) and classification accuracy for binary and muticlass classification tasks, respectively.

3.1 Decoding Local Field Potentials

Our first example is a binary time series classification task (decoding brain activity), involving bivariate time series of local field potentials (LFP) recorded from the visual area V4 and inferior temporal (IT) cortex of a rhesus macaque while performing an attention task. Each of the 420 trials lasted for 2.6 seconds, starting with the animal gazing at an illuminated location at the center of a computer screen in a dark room. An arrow then appeared (cue onset) to indicate the location of a target to appear on the screen (one of two possible locations: bottom versus top). Within roughly 500 milliseconds a target object appeared (stimuli onset). After a variable amount of time the target changed color (target change), indicating that the subject should make a saccade within a few tens of milliseconds. LFP time series were recorded at 1000Hz and were downsampled to 200Hz, yielding

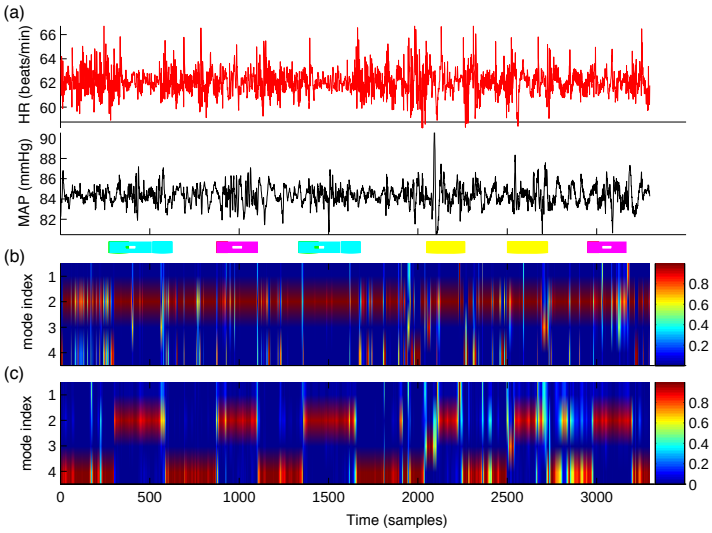

Figure 3: An example of a filtered time series of heart rate (HR) and mean blood pressure (MAP) from the Tilt-Table experiment is shown in panel (a). Non-supine segments are marked by horizontal bars under panel (a). The inferred marginal probabilities of each of the four modes using the EM and the backpropagation approaches are shown in panels (b) and (c), respectively (darker red colors indicate a higher posterior probability for a mode).

roughly 520 samples per time series. Selective attention requires communication among multiple brain regions in a timely manner, and consequently the resulting LFP time series are nonstationary. We modeled the data using a switching VAR model with five hidden states, each corresponding to an AR model of order three. Fig. 2 shows the decoding performance based on EM versus the proposed supervised approach. Notably, five iterations of EM pre-training seems to provide a good starting point for fine-tuning by the supervised algorithm. Only the MOG classifier was able to achieve comparable performances in terms of AUC. Table 1 provides a comparison of various time series classification methods on this dataset.

3.2 Detecting Postural State in a Tilt-Table Experience

Our second example involves a sequential labeling task including four labels. Time series of heart rate (HR) and mean arterial blood pressure (MAP) were acquired from 10 healthy subjects undergoing a Tilt-Table experiment (see [15] for more info). Briefly, subjects were placed in the supine position and secured to a table. Tilting of the table was performed at various speeds from the horizontal position to the vertical position and back to supine, generating four postural categories of (1) supine, (2) slow-tilt, (3) fast tilt, and (4) standing (see Fig. 3). Each experiment lasted for over 45 minutes, resulting in bivariate time series of roughly 3500 samples duration. (With the advent of wearable health monitoring technologies, analysis and quantification of such time series are becoming increasingly important.) We used a SKF with four dynamical modes ($J = 4$, $D = 3$) to model this data, and constructed a sequential labeling/classification task to infer the postural position of the subjects based on their physiological time series dynamics. The supervised learning algorithm was initialized using five iterations of the EM algorithm, followed by supervised learning with early stopping. The results shown in Fig. 3 and Table 1 indicate that the joint supervised learning significantly improves the accuracy of classification.

3.3 Predicting Hospital Mortality in Patients with Sepsis

For our last example, we utilized retrospective heart-rate time series from 118 (survived=73, expired=45) patients with sepsis from the publicly available MIMIC-II intensive care unit (ICU) database. Every year sepsis, a medical condition characterized by whole-body inflammation, strikes over 750,000 people in the US alone, killing one in four patients affected. The existing literature suggest that pathological fluctuations in heart rate (HR) time series can be predictive of onset of sepsis in ICU patients. Early assessment of risk for sepsis within the ICU can guide interventional strategies, and thus change the current standard of care. We designed a 10-fold cross-validation study to assess risk for mortality (a binary classification task) in septic patients, based on 10 seconds resolution HR time series during the first 12 hours of ICU stay. HR time series within the ICU is highly nonstationary and the switching VAR model has been previously applied to model such time series [15]. We utilized a switching VAR model with AR order of three and 10 modes to model this cohort time series. The results presented in Table 1 indicate that the supervised learning approach gives better prediction performance than the competing techniques. Interestingly, for many choices

	LFP (Classification)	Sepsis (Classification)	Tilt Table (Sequential Labeling)
LDS	0.48 [0.44, 0.54]	0.53 [0.50, 0.55]	–
CLDS	0.50 [0.45, 0.54]	0.54 [0.52, 0.56]	–
MOG	0.70 [0.68, 0.70]	0.48 [0.44, 0.51]	–
LRNN	0.52 [0.51, 0.54]	0.50 [0.50, 0.51]	60 [54, 67]
RNNLIB	–	0.55 [0.50 0.62]	–
EM	0.63 [0.59, 0.65]	0.62 [0.57, 0.68]	60 [56, 62]
BP	0.74 [0.70, 0.77]	0.67 [0.63, 0.70]	68 [64, 69]

Table 1: Comparison of time series classification algorithms (see text). MOG, BP, LRNN, and RNNLIB are supervised and the remaining algorithms are unsupervised. Performance metric for the Tilt-Table dataset is classification accuracy (chance is at 25%), and for LFP and sepsis is AUC (chance is at 0.50).

of parameter values and initial weights, the RNN models tended to make rapid initial progress by assigning negative labels to all time series, which impeded further progress via optimization.

4 Discussion

Pattern recognition in time series data has a broad range of applications from finance to medical informatics, however robust algorithms for finding predictive patterns in long sequences of nonstationary multivariate time series are sparse [27]. We have presented a novel technique for identifying discriminative parameter settings in dynamic Bayesian networks, based on inferred estimates of marginals. We showed that supervised learning of the model parameters and the marginals yields better features than EM for multivariate time series classification and sequential labeling. The main idea of our approach was to present the learning algorithm with the outcomes (or labels) corresponding to each time-series, in order to learn time-series features that are most relevant to the discriminative task of distinguishing among the labels. We showed that if the loss function defined on the marginals is differentiable, it will be possible to compute the gradient in terms of these marginals, and then backpropagate the loss gradient through the message passing inference procedure. The resulting algorithm allows for combining unsupervised pre-training with supervised fine-tuning to design and initialize a new class of RNNs for time series classification and sequential labeling. The technique developed here is significant from a theoretical point of view, since one may apply the the backpropagation-based learning described here to any probabilistic model, defined on a directed acyclic graph structure.

The resulting model addresses two shortcomings of the competing neural networks, namely the black box nature of the RNNs and lack of a systematic approach to initialization of network weights in the classic RNN architectures. Considering the first issue, state-space models that constitute the basic building blocks of the DBN models provide a convenient framework for incorporation of mechanistic models of the systems generating the observed time series. Therefore, the discovered time series features can be further interpreted in terms of the underlying mechanisms. For instance, in the Tilt-Table example, tilting is known to disrupt the sympathovagal balance in the direction of increased sympathetic activation [20]. Notably, analysis [20] of the dynamic modes that were most probable during the tilting events reveal a significant increase in sympathetic modulation. Concerning the issue of initialization of RNN models, the class of RNNs presented here enjoy the benefit of generative pre-training via EM. This results mirrors similar findings within the Deep Belief Networks literature, where unsupervised pre-training is known to significantly improve the predictive performance of discriminative neural networks [11, 6]. The intuition is that pre-training puts the network at a region within the parameter space that allows the discriminative learning to rapidly progress. Moreover, since the input is high-dimension it is harder to overfit the input data versus since the labels are low dimensional it is much easier to overfit [10].

Another advantage of the supervised learning approach to inference and learning in DBN models is the ease of imposing additional constraints on the network structure during learning. For instance, the objective loss function of the SKF model can be augmented to impose max-pooling constraints on the discrete latent variables, by allowing only the most probable mode at each time step to pass its gradient information. This will reduce the computational complexity of backpropagation through the supervised SKF architecture which typically require J^2 evaluations of the Kalman filter per time step, and imposes a form of sparsity on the discrete switching variables.

Our ongoing and future works involve learning more expressive representations of time series from the inferred marginals. For instance, a convolutional neural network layer may replace the logistic classifier employed here, to extract additional features pertaining to rate of transition among dynamical modes, as well as, features that may represent long-range trends in evolution of the dynamical modes in nonstationary time series.

References

[1] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE TPAMI*, 35(8):1798–1828, 2013.

[2] Philémon Brakel, Dirk Stroobandt, and Benjamin Schrauwen. Training energy-based models for time-series imputation. *The Journal of Machine Learning Research*, 14(1):2771–2797, 2013.

[3] Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977.

[4] Justin Domke. Learning graphical model parameters with approximate marginal inference. *IEEE TPAMI*, 35(10):2454–2467, 2013.

[5] Frederick Eaton and Zoubin Ghahramani. Choosing a variable to clamp: approximate inference using conditioned belief propagation. *Proceedings of AISTATS*, 5:145–152, 2009.

[6] D. Erhan, Y. Bengio, A. Courville, P.A. Manzagol, P. Vincent, and S. Bengio. Why does unsupervised pre-training help deep learning? *JMLR*, 11:625–660, 2010.

[7] E.B. Fox, E.B. Sudderth, M.I. Jordan, and A.S. Willsky. Sharing features among dynamical systems with beta processes. *Proceedings of NIPS*, 22:549–557, 2009.

[8] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of ICML*, pages 369–376. ACM, 2006.

[9] Tom Heskes and Onno Zoeter. Expectation propagation for approximate inference in dynamic bayesian networks. In *Proceedings of UAI*, pages 216–223, 2002.

[10] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[11] G.E. Hinton, S. Osindero, and Y.W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[12] Minyoung Kim and Vladimir Pavlovic. Discriminative learning for dynamic state prediction. *IEEE TPAMI*, 31(10):1847–1861, 2009.

[13] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. pages 282–289, 2001.

[14] Thomas A Lasko, Joshua C Denny, and Mia A Levy. Computational phenotype discovery using unsupervised feature learning over noisy, sparse, and irregular clinical data. *PLOS ONE*, 8(6):e66341, 2013.

[15] L. Lehman, R. Adams, L. Mayaud, G. Moody, A Malhotra, R. Mark, and S. Nemati. A physiological time series dynamics-based approach topatient monitoring and outcome prediction. *IEEE Journal of BHI*, PP (99):1–1, 2014.

[16] Lei Li and B Aditya Prakash. Time series clustering: Complex is simpler! In *Proceedings of ICML*, pages 185–192, 2011.

[17] Andrew McCallum, Dayne Freitag, and Fernando C. N. Pereira. Maximum entropy Markov models for information extraction and segmentation. In *ICML Proceedings*, pages 591–598, 2000.

[18] R. Memisevic. An introduction to structured discriminative learning. Technical report, University of Toronto, 2006.

[19] Kevin P. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, University of California, 2002.

[20] Shamim Nemati, Li-wei H Lehman, and Ryan P Adams. Learning outcome-discriminative dynamics in multivariate physiological cohort time series. In *Proceedings of IEEE EMBC*, pages 7104–7107, 2013.

[21] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 1988.

[22] M Schmidt. minfunc: unconstrained differentiable multivariate optimization in matlab. 2012.

[23] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian optimization of machine learning algorithms. In *Proceedings of NIPS*, pages 2951–2959, 2012.

[24] V. Stoyanov, A. Ropson, and J. Eisner. Empirical risk minimization of graphical model parameters given approximate inference, decoding, and model structure. In *Proceedings of AISTATS*, 2011.

[25] Ilya Sutskever. *Training recurrent neural networks*. PhD thesis, University of Toronto, 2013.

[26] PC Woodland and D. Povey. Large scale discriminative training for speech recognition. In *ASR2000-Automatic Speech Recognition: Challenges for the new Millenium ISCA Research Workshop*, 2000.

[27] Zhengzheng Xing, Jian Pei, and Eamonn Keogh. A brief survey on sequence classification. *ACM SIGKDD*, 12(1):40–48, 2010.

A Appendix

Hereafter we will use parentheses to index individual states, and we will use brackets to indicate the individual elements of a vector or a matrix. Thus, $A(i)[m,n]$ refers to the m-th row, n-th column element of the matrix of state dynamics for the i-th mode. We will use the symbol $\odot$ to denote the Frobenius inner product of two matrices (or vectors) defined as $A \odot B = \sum_i \sum_j A_{ij} B_{ij}$. Moreover, for a matrix B, indexed by i, j, the colon notation $B(i,:)$ denotes entries ranging over all j values. All the exponentiations involving matrices are element-wise. Finally, $\delta_{m,n}$ denotes a conformable matrix with the (m,n)-th element equal to one and zero elsewhere. Similarly, $\delta_{m,n}^{n,m}$ denotes a conformable matrix with ones at the (m,n)-th and (n,m)-th elements and zero elsewhere.

A.1 Removing the Constraints via Parameter Transformations

The objective of the proposed leaning algorithm is to minimize the cost function:

$$\mathbf{\Theta}^\star, Z^\star, \beta^\star = \underset{\mathbf{\Theta}, Z, \beta}{\operatorname{argmin}} \{-\log \Pr(\mathbf{O}|\boldsymbol{\mu}(\Theta, \beta))\} \tag{7}$$

subject to the constraints that covariance matrices $Q^{(j)}$ and $R^{(j)}$ remain positive definite, and all the elements of Z stay nonnegative and each row sums to one. We can convert this constrained optimization problem to an equivalent unconstrained problem by defining the following transformations:

Let $\bar{Z}(i,j)$ be such that $Z(i,j) = \frac{exp(\bar{Z}(i,j))}{\sum_{j'} exp(\bar{Z}(i,j'))}$, which results in the following gradient for $\bar{Z}$:

$$\frac{\partial E}{\partial \bar{Z}(i,j)} = \frac{\partial E}{\partial \bar{Z}} \odot \frac{\partial Z}{\partial \bar{Z}(i,j)}, \tag{8}$$

where

$$\frac{\partial Z(k,l)}{\partial \bar{Z}(i,j)} = \delta_{i,k} Z(i,j)(\delta_{j,l} - Z(k,l)). \tag{9}$$

Furhermore, to ensure we optimize over the space of positive semi-definite matrices, we use the Cholesky decomposition representation of the covariance matrices. For instance, in the case of the state-noise covariance matrices, we represent $Q(j) = \Gamma(j)\Gamma(j)^\intercal$, where ($\Gamma(j)$ is a lower diagonal matrix). Then $\frac{\partial E}{\partial L(j)} = \frac{\partial E}{\partial Q(j)} \frac{\partial Q(j)}{\partial \Gamma(j)}$, given by:

$$\frac{\partial E}{\partial \Gamma(j)[m,n]} = \frac{\partial E}{\partial Q(j)} \odot (\delta_{m,n} \Gamma(j)^\intercal + \Gamma(j) \delta_{m,n}^\intercal), \tag{10}$$

and the corresponding gradient vector includes only the lower diagonal elemnts of $\frac{\partial E}{\partial \Gamma(j)}$.

In what follows, we start with the regression layer shown in Fig. 1, and recursively estimate the error gradients all the way down to the filtering layer, and ultimately calculate the error gradient with respect to the parameters of the dynamical states.

A.2 Derivatives of the Regression Layer

Logistic regression is commonly used for predicting outcomes of categorical variable. For instance, each of the N time-series within a cohort may be associated with an outcome (or label) denoted by $\{O_n^{true}\}_{n=1}^N$). In this work, we use the logistic and multinomial regression methods to map the hidden state marginal proportions to the outcome variables of interest. Therefore, we provide the analytic gradients of the corresponding error functions with respect to the regression parameters and the mode proportions.

A.2.1 Binary Outcomes

Given a set of predictor variables (taken to be the mode proportions here), the binary logit function takes the following form:

$$\sigma(\bar{\eta}) = \frac{1}{1 + exp(-\bar{\eta})} \tag{11}$$

with $\bar{\eta} = \beta_0 + \beta_1 \eta_1 + \cdots + \beta_J \eta_J$. In Eq. (11), $\sigma(\bar{\eta})$ can be interpreted as the probability of a positive outcome, given the mode proportions (or more generally, model parameters). The Bernoulli probability of a given outcome O^{true} ($\in \{0,1\}$) is then parameterized by $\sigma(\eta)$ we follows:

$$P(O^{true}|\sigma(\bar{\eta})) = \sigma(\bar{\eta})^{O^{true}} (1 - \sigma(\bar{\eta})^{1 - O^{true}} \ . \tag{12}$$

The objective of parameter fitting (finding the β_j coefficients) is to minimize the negative log-likelihood of the outcomes, given the predictor variables:

$$E = -logProb(O^{true};\ \bar{\eta}) = -(O^{true} log(\sigma(\bar{\eta})) + (1 - O^{true}) log(1 - \sigma(\bar{\eta}))). \tag{13}$$

Note that the overall error is the sum over the individual errors, that is: $E_{all} = -\sum_{n=1}^{N} logProb(O_n^{true};\ \bar{\eta}^{(n)})$.

Error Gradient

The derivative of the error function in Eq. (13) with respect to $\bar{\eta}$ is given by

$$\frac{\partial E}{\partial \bar{\eta}} = (-\frac{O^{true}}{\sigma(\bar{\eta})} + \frac{1 - O^{true}}{1 - \sigma(\bar{\eta})}) \frac{exp(-\bar{\eta})}{(1 + exp(-\bar{\eta}))^2}. \tag{14}$$

Moreover, it follows from the Chain Rule that

$$\frac{\partial E}{\beta_{\mathbf{j}}} = \frac{\partial E}{\partial \bar{\eta}} \eta_j\ , \tag{15}$$

$$\frac{\partial E}{\eta_{\mathbf{j}}} = \frac{\partial E}{\partial \bar{\eta}} \beta_j\ . \tag{16}$$

A.2.2 Multinomial Outcomes

The multinomial probability of a given outcome O^{true} ($\in \{1, \cdots, K\}$), parameterized by μ_k, is given by

$$Prob(O^{true}|\mu_k) = \prod_{k=1}^{K} \mu_k^{O_k^{true}} = exp(\sum_{k=1}^{K} O_k^{true} log \mu_k) \tag{17}$$

where $\mu_k = \frac{exp(\theta_k)}{\sum_{k'} exp(\theta_{k'})}$, and $\theta_k = \beta_{k,0} + \sum_{j=1}^{J} \beta_{k,j} \eta_j$.

We take the error function to be the negative log-likelihood of the probability of an outcome given μ:

$$E = -logProb(O^{true}|\mu) = -\sum_{k=1}^{K} O_k^{true} log(\mu_k). \tag{18}$$

Note that the overall error is $E_{all} = -\sum_{n=1}^{N} logProb(O_n^{true}|\mu_k^{(n)}$.

Error Gradient

The error gradients with respect to the parameters $\beta_{k,j}$ of the multinomial regression and the mode proportions η_j are given by

$$\frac{\partial E}{\partial \beta_{k,j}} = \frac{\partial E}{\partial \theta_{\mathbf{k}}} \eta_j \tag{19}$$

$$\frac{\partial E}{\partial \eta_j} = \sum_{k=1}^{K} \frac{\partial E}{\partial \theta_{\mathbf{k}}} \beta_{k,j}\ , \tag{20}$$

where

$$\frac{\partial E}{\partial \theta_{\mathbf{k}}} = -\sum_{k'=1}^{K} O_{k'}^{true} (\delta_{k,k'} - \mu_k). \tag{21}$$

A.3 Derivatives of the Switching Kalman Filter

A.3.1 Filtering Step

For completeness we summarize the essential equations (pertinent to error backpropagation) of the filtering and smoothing steps of the SKF algorithm, as described in Murphy (1998) [19].

The filtering step of the SKF algorithm takes the following form:

$$(\tilde{\mu}_{t|t}(i,j), \tilde{V}_{t|t}(i,j), L_t(i,j)) = KalmanFilter(\mu_{t-1|t-1}(i), V_{t-1|t-1}(i), \mathbf{y}_t(i); A(j), C(j), Q(j), R(j))$$

$$\mu_{t|t-1} = A\mu_{t-1|t-1} \,, \tag{22}$$
$$V_{t|t-1} = AV_{t-1|t-1}A^\intercal + Q \,, \tag{23}$$
$$\mathbf{e}_t = \mathbf{y}_t - C\mu_{t|t-1} \,, \tag{24}$$
$$\mathcal{S}_t = CV_{t|t-1}C^\intercal + R \,, \tag{25}$$
$$G_t = V_{t|t-1}C^\intercal \mathcal{S}_t^{-1} \,, \tag{26}$$
$$\tilde{\mu}_{t|t} = \mu_{t|t-1} + G_t\mathbf{e}_t \,, \tag{27}$$
$$\tilde{V}_{t|t} = (I - G_tC)V_{t|t-1} \,, \tag{28}$$
$$L_t = \mathcal{N}(\mathbf{e}_t;\, 0,\, \mathcal{S}_t) \,. \tag{29}$$

$$(M_t^f(j), W_t^f(i,j)) = ForwardSwitchFilter(M_{t-1}^f, Z, L_t)$$

$$\alpha_t^f(i,j) = M_{t-1}^f(i)L_t^f(i,j)Z(i,j) \,, \tag{30}$$
$$a_t^f(j) = \sum_{i'} \alpha_t^f(i',j) \,, \tag{31}$$
$$M_t^f(j) = \frac{a_t^f(j)}{\sum_{j'} a_t^f(j')} \,, \tag{32}$$
$$W_t^f(i,j) = \frac{\alpha_t^f(i,j)}{\sum_{j'} a_t^f(j')} / M_t^f(j). \tag{33}$$

$$(\mu_{t|t}(j), V_{t|t}(j)) = Collapse(\tilde{\mu}_{t|t}, \tilde{V}_{t|t}, W_t^f)$$

$$\mu_{t|t}(j) = \sum_i W_t^f(i,j)\tilde{\mu}_{t|t}(i,j) \tag{34}$$
$$V_{t|t}(j) = \sum_i W_t^f(i,j)(\tilde{V}_{t|t}(i,j) + (\tilde{\mu}_{t|t}(i,j) - \mu_{t|t}(j))(\tilde{\mu}_{t|t}(i,j) - \mu_{t|t}(j))^\intercal) \,, \tag{35}$$

For $t = 1, \cdots, T$. Note, $M_t^f(i) = Prob(S_t = i|y_{1:t})$, with the initial condition $M_0^f = \pi$.

The analytical partial derivatives of the three operators above are presented in Appendix A, sections A.4.5, A.4.6, and A.4.7.

A.3.2 Smoothing Step

The smoothing step of the SKF algorithm for the switching variables takes the following form [19]:

$$(M_t^s(i)) = BackwardSwitchSmooth(M_t^f,\ M_{t+1}^s,\ Z)$$

$$a_t^s(i,j) = M_t^f(i)Z(i,j) \,, \tag{36}$$
$$b_t^s(i,j) = \frac{a_t^s(i,j)}{\sum_{i'} a_t^s(i',j)} \,, \tag{37}$$
$$M_t^s(i) = \sum_{j'} b_t^s(i,j')M_{t+1}^s(j') \,, \tag{38}$$

for $t = T-1, \cdots, 1$. Note, $M_t^s(i) = Prob(S_t = i|y_{1:T})$, with the initial condition $M_T^s = M_T^f$.

A.4 Error Gradient Calculations

We start from the filtering step of the SKF algorithm and calculate the analytical partial derivatives of each node output(s) with respect to its input(s), as we move forward in time. Next, smoothing of the switching variables is performed and the corresponding analytical gradients are calculated. The back-propagation algorithm starts from the reverse direction (from the output of the smoothed switching variables or the mode proportions) and propagates the gradient information backward (starting from time T) through the smoothed switching variables, and finally the filtered variables (end ing in time t=1).

A.4.1 Error Gradient with Respect to Smoothed switching Variables

Derivatives of the error with respect to the mode proportions η_i are given by Eq. (16) (in the case of binary outcomes) and Eq. (20) (in the case of multinomial outcomes). Next, the error is backpropagated through the smoothed switching variables, as follows:

$$\begin{aligned}\frac{\partial E}{\partial M_1^s(i)} &= \frac{1}{T}\frac{\partial E}{\partial \eta_i}\ , \\ \frac{\partial E}{\partial M_t^s(i)} &= \frac{1}{T}\frac{\partial E}{\partial \eta_i} + \sum_{j'}\frac{\partial E}{\partial M_{t-1}^s(j')} b_{t-1}^s(j',i)\ ,\ t = 2\cdots T \end{aligned} \tag{39}$$

We also compute the following derivatives:

$$\frac{\partial E}{\partial a_t^s(i,j)} = \sum_{k'}\frac{\partial E}{\partial M_t^s(k')} M_{t+1}^s(j)[\frac{\delta_{k',i}}{\sum_{i'} a_t^s(i',j)} - \frac{a_t^s(k',j)}{(\sum_{i'} a_t^s(i',j))^2}]\ ,\ t = 1\cdots T-1 \tag{40}$$

A.4.2 Error Gradient with Respect to Filtered switching variables

Next, derivatives of the error with respect to the filtered switching variables can be calculated as follows:

$$\begin{aligned}\frac{\partial E}{\partial M_T^f(i)} &= \frac{\partial E}{\partial M_T^s(i)}\ , \\ \frac{\partial E}{\partial M_t^f(i)} &= \sum_{j'}\frac{\partial E}{\partial a_t^s(i,j')} Z(i,j') + \frac{\partial E}{\partial W_{t+1}^f}\odot\frac{\partial W_{t+1}^f}{\partial M_t^f(i)} + \frac{\partial E}{\partial M_{t+1}^f}\odot\frac{\partial M_{t+1}^f}{\partial M_t^f(i)}\ ,\ t = T-1\cdots 1 \end{aligned} \tag{41}$$

where

$$\begin{aligned}\frac{\partial E}{\partial W_T^f(i,j)} &= 0,\ \forall\, i,j\ , \\ \frac{\partial E}{\partial W_t^f(i,j)} &= \frac{\partial E}{\partial \mu_t^f(j)}\odot\frac{\partial \mu_t^f(j)}{\partial W_t^f(i,j)} + \frac{\partial E}{\partial V_t^f(j)}\odot\frac{\partial V_t^f(j)}{\partial W_t^f(i,j)}\ ,\ t = T-1\cdots 1\ . \end{aligned} \tag{42}$$

A.4.3 Error Gradient with Respect to Filtered State Variables

Although the marginal distributions of the state variable are Gaussian random variables, during the error back-propagation they can be treated as real-valued means (vectors) and covariances (matrices). The following gradients of the error with respect to the state variables are recursively calculated:

$$\begin{aligned}\frac{\partial E}{\partial \mu_t^f(i)[m]} &= \sum_{j'}\frac{\partial E}{\partial L_{t+1}^f(i,j')}\frac{\partial L_{t+1}^f(i,j')}{\partial \mu_t^f(i)[m]} + \sum_{j'}\frac{\partial E}{\partial \tilde{\mu}_{t+1}^f(i,j')}\odot\frac{\partial \tilde{\mu}_{t+1}^f(i,j')}{\partial \mu_t^f(i)[m]}\ ,\ t = 1\cdots T-2\ , \\ \frac{\partial E}{\partial \mu_t^f(i)[m]} &= \sum_{j'}\frac{\partial E}{\partial L_{t+1}^f(i,j')}\frac{\partial L_{t+1}^f(i,j')}{\partial \mu_t^f(i)[m]}\ ,\ t = T-1\ , \end{aligned} \tag{43}$$

$$\begin{aligned}\frac{\partial E}{\partial V_t^f(i)[m,n]} &= \sum_{j'}\frac{\partial E}{\partial L_{t+1}^f(i,j')}\frac{\partial L_{t+1}^f(i,j')}{\partial V_t^f(i)[m,n]} + \sum_{j'}\frac{\partial E}{\partial \tilde{V}_{t+1}^f(i,j')}\odot\frac{\partial \tilde{V}_{t+1}^f(i,j')}{\partial V_t^f(i)[m,n]} \\ &+ \sum_{j'}\frac{\partial E}{\partial \tilde{\mu}_{t+1}^f(i,j')}\odot\frac{\partial \tilde{\mu}_{t+1}^f(i,j')}{\partial V_t^f(i)[m,n]}\ ,\ t = 1\cdots T-2\ , \\ \frac{\partial E}{\partial V_t^f(i)[m,n]} &= \sum_{j'}\frac{\partial E}{\partial L_{t+1}^f(i,j')}\frac{\partial L_{t+1}^f(i,j')}{\partial V_t^f(i)[m,n]}\ ,\ t = T-1\ , \end{aligned} \tag{44}$$

where

$$\frac{\partial E}{\partial L_t^f(i,j)} = \frac{\partial E}{\partial W_t^f}\odot\frac{\partial W_t^f}{\partial L_t^f(i,j)} + \frac{\partial E}{\partial M_t^f}\odot\frac{\partial M_t^f}{\partial L_t^f(i,j)}\ ,\ t = 1\cdots T\ , \tag{45}$$

and

$$\begin{aligned}\frac{\partial E}{\partial \tilde{\mu}_t^f(i,j)[m]} &= \frac{\partial E}{\partial \mu_t^f(j)}\odot\frac{\partial \mu_t^f(j)}{\partial \tilde{\mu}_t^f(i,j)[m]} + \frac{\partial E}{\partial V_t^f(j)}\odot\frac{\partial V_t^f(j)}{\partial \tilde{\mu}_t^f(i,j)[m]}\ ,\ t = 1\cdots T-1\ , \end{aligned} \tag{46}$$

$$\begin{aligned}\frac{\partial E}{\partial \tilde{V}_t^f(i,j)[m]} &= \frac{\partial E}{\partial V_t^f(j)}\odot\frac{\partial V_t^f(j)}{\partial \tilde{V}_t^f(i,j)[m,n]}\ ,\ t = 1\cdots T-1\ . \end{aligned} \tag{47}$$

Note that, $\partial \tilde{V}_{t+1}^f(i,j')/\partial \mu_t^f(i) = 0$ for all $i,\ j$, and $\partial \mu_t^f(j)/\partial \tilde{V}_t^f(i,j) = 0$ for all $i,\ j$.

A.4.4 Error Gradient with Respect to Model Parameters

We finally arrive at the error gradients with respect to the model parameters. The derivatives with respect to the Markov switching state transition matrix are given by:

$$\frac{\partial E}{\partial Z(i,j)} = \sum_{t=1}^{T} \frac{\partial E}{\partial M_t^f(j)} \odot \frac{\partial M_t^f(j)}{\partial Z(i,j)} + \sum_{t=1}^{T-1} \frac{\partial E}{\partial a_t^s(i,j)} M_t^f(i) \tag{48}$$

For the other model parameters the error gradients are as follows:

$$\begin{aligned}\frac{\partial E}{\partial A(j)[m,n]} &= \sum_{i'}[\sum_{t=1}^{T-1} \frac{\partial E}{\partial \tilde{\mu}_t^f(i',j)} \odot \frac{\partial \tilde{\mu}_t^f(i',j)}{\partial A(j)[m,n]} \\ &+ \sum_{t=1}^{T-1} \frac{\partial E}{\partial \tilde{V}_t^f(i',j)} \odot \frac{\partial \tilde{V}_t^f(i',j)}{\partial A(j)[m,n]} + \sum_{t=1}^{T} \frac{\partial E}{\partial L_t^f(i',j)} \frac{\partial L_t^f(i',j)}{\partial A(j)[m,n]}].\end{aligned} \tag{49}$$

$$\begin{aligned}\frac{\partial E}{\partial C(j)[m,n]} &= \sum_{i'}[\sum_{t=1}^{T-1} \frac{\partial E}{\partial \tilde{\mu}_t^f(i',j)} \odot \frac{\partial \tilde{\mu}_t^f(i',j)}{\partial C(j)[m,n]} \\ &+ \sum_{t=1}^{T-1} \frac{\partial E}{\partial \tilde{V}_t^f(i',j)} \odot \frac{\partial \tilde{V}_t^f(i',j)}{\partial C(j)[m,n]} + \sum_{t=1}^{T} \frac{\partial E}{\partial L_t^f(i',j)} \frac{\partial L_t^f(i',j)}{\partial C(j)[m,n]}].\end{aligned} \tag{50}$$

$$\begin{aligned}\frac{\partial E}{\partial Q(j)[m,n]} &= \sum_{i}[\sum_{t=1}^{T-1} \frac{\partial E}{\partial \tilde{\mu}_t^f(i',j)} \odot \frac{\partial \tilde{\mu}_t^f(i',j)}{\partial Q(j)[m,n]} \\ &+ \sum_{t=1}^{T-1} \frac{\partial E}{\partial \tilde{V}_t^f(i',j)} \odot \frac{\partial \tilde{V}_t^f(i',j)}{\partial Q(j)[m,n]} + \sum_{t=1}^{T} \frac{\partial E}{\partial L_t^f(i',j)} \frac{\partial L_t^f(i',j)}{\partial Q(j)[m,n]}].\end{aligned} \tag{51}$$

$$\begin{aligned}\frac{\partial E}{\partial R(j)[m,n]} &= \sum_{i'}[\sum_{t=1}^{T-1} \frac{\partial E}{\partial \tilde{\mu}_t^f(i',j)} \odot \frac{\partial \tilde{\mu}_t^f(i',j)}{\partial R(j)[m,n]} \\ &+ \sum_{t=1}^{T-1} \frac{\partial E}{\partial \tilde{V}_t^f(i',j)} \odot \frac{\partial \tilde{V}_t^f(i',j)}{\partial R(j)[m,n]} + \sum_{t=1}^{T} \frac{\partial E}{\partial L_t^f(i',j)} \frac{\partial L_t^f(i',j)}{\partial R(j)[m,n]}].\end{aligned} \tag{52}$$

A.4.5 Analytical Derivatives of the Kalman Filter

Recall the Kalman filter operator is defined as:
$(\tilde{\mu}_{t|t}(i,j), \tilde{V}_{t|t}(i,j), L_t(i,j)) = KalmanFilter(\mu_{t-1|t-1}(i), V_{t-1|t-1}(i), \mathbf{y}_t(i); A(j), C(j), Q(j), R(j))$.

The analytical derivatives of the Kalman filter operator is given by

$$\frac{\partial \tilde{\mu}_{t|t-1}}{\partial A[m,n]} = \delta_{m,n}\mu_{t-1|t-1} \tag{53}$$

$$\frac{\partial \tilde{\mu}_{t|t-1}}{\partial \mu_{t-1|t-1}(m)} = A\delta_m \tag{54}$$

$$\frac{\partial \tilde{V}_{t|t-1}}{\partial A[m,n]} = \delta_{m,n}V_{t-1|t-1}A^\intercal + AV_{t-1|t-1}\delta_{m,n}^\intercal \tag{55}$$

$$\frac{\partial \tilde{V}_{t|t-1}}{\partial Q[m,n]} = \delta_{m,n}^{n,m} \tag{56}$$

$$\frac{\partial \tilde{V}_{t|t-1}}{\partial V_{t-1|t-1}[m,n]} = A\delta_{m,n}^{n,m}A^\intercal \tag{57}$$

$$\frac{\partial \mathbf{e}_t}{\partial A[m,n]} = -C\delta_{m,n}\mu_{t-1|t-1} \tag{58}$$

$$\frac{\partial \mathbf{e}_t}{\partial C[m,n]} = -\delta_{m,n}A\mu_{t-1|t-1} \tag{59}$$

$$\frac{\partial \mathbf{e}_t}{\partial \mu_{t-1|t-1}[m]} = -CA\delta_m \tag{60}$$

$$\frac{\partial \mathcal{S}_t}{\partial A[m,n]} = C\frac{\partial \tilde{V}_{t|t-1}}{\partial A[m,n]}C^\intercal \tag{61}$$

$$\frac{\partial \mathcal{S}_t}{\partial C[m,n]} = \delta_{m,n}\tilde{V}_{t|t-1}C^\intercal + C\tilde{V}_{t|t-1}\delta_{m,n}^\intercal \tag{62}$$

$$\frac{\partial \mathcal{S}_t}{\partial Q[m,n]} = C\frac{\partial \tilde{V}_{t|t-1}}{\partial Q[m,n]}C^\intercal \tag{63}$$

$$\frac{\partial \mathcal{S}_t}{\partial R[m,n]} = \delta_{m,n}^{n,m} \tag{64}$$

$$\frac{\partial \mathcal{S}_t}{\partial V_{t-1|t-1}[m,n]} = C\frac{\partial \tilde{V}_{t|t-1}}{\partial V_{t-1|t-1}[m,n]}C^\intercal \tag{65}$$

$$\frac{\partial G_t}{\partial A[m,n]} = (\frac{\partial \tilde{V}_{t|t-1}}{\partial A[m,n]}C^\intercal - G_t\frac{\partial \mathcal{S}_t}{\partial A[m,n]})\mathcal{S}_t^{-1} \tag{66}$$

$$\frac{\partial G_t}{\partial C[m,n]} = (\tilde{V}_{t|t-1}\delta_{m,n}^\intercal - G_t\frac{\partial \mathcal{S}_t}{\partial C[m,n]})\mathcal{S}_t^{-1} \tag{67}$$

$$\frac{\partial G_t}{\partial Q[m,n]} = (\frac{\partial \tilde{V}_{t|t-1}}{\partial Q[m,n]}C^\intercal - G_t\frac{\partial \mathcal{S}_t}{\partial Q[m,n]})\mathcal{S}_t^{-1} \tag{68}$$

$$\frac{\partial G_t}{\partial R[m,n]} = (-G_t\frac{\partial \mathcal{S}_t}{\partial R[m,n]})\mathcal{S}_t^{-1} \tag{69}$$

$$\frac{\partial G_t}{\partial V_{t-1|t-1}[m,n]} = (\frac{\partial \tilde{V}_{t|t-1}}{\partial V_{t-1|t-1}[m,n]}C^\intercal - G_t\frac{\partial \mathcal{S}_t}{\partial V_{t-1|t-1}[m,n]})\mathcal{S}_t^{-1} \tag{70}$$

$$\frac{\partial \tilde{\mu}_{t|t}}{\partial A[m,n]} = \frac{\partial \tilde{\mu}_{t|t-1}}{\partial A[m,n]} + \frac{\partial G_t}{\partial A[m,n]}\mathbf{e}_t + G_t\frac{\partial \mathbf{e}_t}{\partial A[m,n]} \tag{71}$$

$$\frac{\partial \tilde{\mu}_{t|t}}{\partial C[m,n]} = \frac{\partial G_t}{\partial C[m,n]}\mathbf{e}_t + G_t\frac{\partial \mathbf{e}_t}{\partial C[m,n]} \tag{72}$$

$$\frac{\partial \tilde{\mu}_{t|t}}{\partial Q[m,n]} = \frac{\partial G_t}{\partial Q[m,n]}\mathbf{e}_t \tag{73}$$

$$\frac{\partial \tilde{\mu}_{t|t}}{\partial R[m,n]} = \frac{\partial G_t}{\partial R[m,n]}\mathbf{e}_t \tag{74}$$

$$\frac{\partial \tilde{\mu}_{t|t}}{\partial \mu_{t-1|t-1}[m]} = \frac{\partial \tilde{\mu}_{t|t-1}}{\partial \mu_{t-1|t-1}(m)} + G_t\frac{\partial \mathbf{e}_t}{\partial \mu_{t-1|t-1}[m]} \tag{75}$$

$$\frac{\partial \tilde{\mu}_{t|t}}{\partial V_{t-1|t-1}[m,n]} = \frac{\partial G_t}{\partial V_{t-1|t-1}[m,n]}\mathbf{e}_t \tag{76}$$

$$\frac{\partial \tilde{V}_{t|t}}{\partial A[m,n]} = -\frac{\partial G_t}{\partial A[m,n]}C\tilde{V}_{t|t-1} + (I - G_tC)\frac{\partial \tilde{V}_{t|t-1}}{\partial A[m,n]} \tag{77}$$

$$\frac{\partial \tilde{V}_{t|t}}{\partial C[m,n]} = -(\frac{\partial G_t}{\partial C[m,n]}C + G_t\delta_{m,n})\tilde{V}_{t|t-1} \tag{78}$$

$$\frac{\partial \tilde{V}_{t|t}}{\partial Q[m,n]} = -\frac{\partial G_t}{\partial Q[m,n]}C\tilde{V}_{t|t-1} + (I - G_tC)\frac{\partial \tilde{V}_{t|t-1}}{\partial Q[m,n]} \tag{79}$$

$$\frac{\partial \tilde{V}_{t|t}}{\partial R[m,n]} = -\frac{\partial G_t}{\partial R[m,n]}C\tilde{V}_{t|t-1} \tag{80}$$

$$\frac{\partial \tilde{V}_{t|t}}{\partial V_{t-1|t-1}[m,n]} = -\frac{\partial G_t}{\partial V_{t-1|t-1}[m,n]}C\tilde{V}_{t|t-1} + (I - G_tC)\frac{\partial \tilde{V}_{t|t-1}}{\partial V_{t-1|t-1}[m,n]} \tag{81}$$

$$\frac{\partial L_t}{\partial \mathbf{e}_t} = -L_t^f\mathcal{S}_t^{-1}\mathbf{e}_t \tag{82}$$

$$\frac{\partial L_t}{\partial \mathcal{S}_t} = -\frac{1}{2}L_t^f(\mathcal{S}_t^{-1} - \mathcal{S}_t^{-1}\mathbf{e}_t\mathbf{e}_t^\intercal\mathcal{S}_t^{-1}) \tag{83}$$

$$\frac{\partial L_t}{\partial A[m,n]} = \frac{\partial L_t}{\partial \mathbf{e}_t} \odot \frac{\partial \mathbf{e}_t}{\partial A[m,n]} + \frac{\partial L_t}{\partial \mathcal{S}_t} \odot \frac{\partial \mathcal{S}_t}{\partial A[m,n]} \tag{84}$$

$$\frac{\partial L_t}{\partial C[m,n]} = \frac{\partial L_t}{\partial \mathbf{e}_t} \odot \frac{\partial \mathbf{e}_t}{\partial C[m,n]} + \frac{\partial L_t}{\partial \mathcal{S}_t} \odot \frac{\partial \mathcal{S}_t}{\partial C[m,n]} \tag{85}$$

$$\frac{\partial L_t}{\partial Q[m,n]} = \frac{\partial L_t}{\partial \mathcal{S}_t} \odot \frac{\partial \mathcal{S}_t}{\partial Q[m,n]} \tag{86}$$

$$\frac{\partial L_t}{\partial R[m,n]} = \frac{\partial L_t}{\partial \mathcal{S}_t} \odot \frac{\partial \mathcal{S}_t}{\partial R[m,n]} \tag{87}$$

$$\frac{\partial L_t}{\partial \mu_{t-1|t-1}[m]} = \frac{\partial L_t}{\partial \mathbf{e}_t} \odot \frac{\partial \mathbf{e}_t}{\partial \mu_{t-1|t-1}[m]} \tag{88}$$

$$\frac{\partial L_t}{\partial V_{t-1|t-1}[m,n]} = \frac{\partial L_t}{\partial \mathcal{S}_t} \odot \frac{\partial \mathcal{S}_t}{\partial V_{t-1|t-1}[m,n]} \tag{89}$$

A.4.6 Analytical Derivatives of the Filtered switching variables

The analytic derivatives of the forward filtering operator:
$(M_t^f(j), W_t^f(i,j)) = ForwardSwitchFilter(M_{t-1}^f, Z, L_t)$, is given as follows:

$$\frac{\partial M_t^f(i)}{\partial M_{t-1}^f(j)} = \sum_{k'} L_t^f(j,i) Z(j,i) [\frac{\delta_{k',j}}{\sum_{i'} a_t^f(i')} - \frac{a_t^f(i)}{(\sum_{i'} a_t^f(i'))^2}] \ , \ t = 1 \cdots T. \tag{90}$$

$$\frac{\partial M_t^f(k)}{\partial Z(i,j)} = M_{t-1}^f(i) L_t^f(i,j) [\frac{\delta_{k',j}}{\sum_{k'} a_t^f(k')} - \frac{a_t^f(k)}{(\sum_{k'} a_t^f(k'))^2}]. \tag{91}$$

$$\frac{\partial M_t^f(k)}{\partial L(i,j)} = M_{t-1}^f(i) Z_t^f(i,j) [\frac{\delta_{k',j}}{\sum_{k'} a_t^f(k')} - \frac{a_t^f(k)}{(\sum_{k'} a_t^f(k'))^2}]. \tag{92}$$

$$\frac{\partial W_t^f(i,j)}{\partial M_{t-1}^f(k)} = (\delta_{i,k} L_t^f(i,j) Z(i,j) \tag{93}$$

$$- \ W_t^f(i,j) [\sum_{j'} a_t^f(j') \frac{\partial M_t^f(j)}{\partial M_{t-1}^f(k)} + M_t^f(j) \sum_{j'} L_t^f(k,j') Z(k,j')]) / (M_t^f(j) \sum_{j'} a_t^f(j')).$$

$$\frac{\partial W_t^f(i,j)}{\partial Z(k,l)} = (\delta_{i,k} \delta_{j,l} L_t^f(i,j) M_{t-1}^f(i) \tag{94}$$

$$- \ W_t^f(i,j) [\sum_{j'} a_t^f(j') \frac{\partial M_t^f(j)}{\partial Z(k,l)} + M_{t-1}^f(k) L_t^f(k,l) M_t^f(j)]) / (M_t^f(j) \sum_{j'} a_t^f(j')).$$

$$\frac{\partial W_t^f(i,j)}{\partial L(k,l)} = (\delta_{i,k} \delta_{j,l} Z(i,j) M_{t-1}^f(i) \tag{95}$$

$$- \ W_t^f(i,j) [\sum_{j'} a_t^f(j') \frac{\partial M_t^f(j)}{\partial L(k,l)} + M_{t-1}^f(k) Z_t^f(k,l) M_t^f(j)]) / (M_t^f(j) \sum_{j'} a_t^f(j')).$$

A.4.7 Analytical Derivatives of the Collapse Function

Recall the collapse operator: $(\mu_{t|t}(j), V_{t|t}(j)) = Collapse(\tilde{\mu}_{t|t}, \tilde{V}_{t|t}, W_t^f)$.
The corresponding derivatives are:

$$
\begin{aligned}
\frac{\partial \mu_{t|t}(i,j)}{\partial \tilde{\mu}_{t|t}[m]} &= W(i,j)\delta_m && (96)\\
\frac{\partial \mu_{t|t}(i,j)}{\partial W_t[i]} &= \tilde{\mu}_{t|t}(i,j) && (97)\\
\frac{\partial V_{t|t}(i,j)}{\partial \tilde{\mu}_{t|t}[m]} &= W(i,j)[(\delta_m - W(i,j)\delta_m)(\tilde{\mu}_{t|t}(i,j) - \mu_{t|t}(j))^\intercal && (98)\\
&+ (\tilde{\mu}_{t|t}(i,j) - \mu_{t|t}(j))(\delta_m - W(i,j)\delta_m)^\intercal]\\
&+ \sum_{k\neq i} W(k,j)[(-W(i,j)\delta_m)(\tilde{\mu}_{t|t}(k,j) - \mu_{t|t}(j))^\intercal\\
&+ (\tilde{\mu}_{t|t}(k,j) - \mu_{t|t}(j))(-W(i,j)\delta_m)^\intercal]\\
\frac{\partial V_{t|t}(i,j)}{\partial \tilde{V}_{t|t}[m,n]} &= \delta_{m,n}^{n,m} W(i,j) && (99)\\
\frac{\partial V_{t|t}(i,j)}{\partial W_t[i]} &= \tilde{V}_{t|t}(i,j) + (\tilde{\mu}_{t|t}(i,j) - \mu_{t|t}(j))^\intercal(\tilde{\mu}_{t|t}(i,j) - \mu_{t|t}(j)) && (100)
\end{aligned}
$$

B Appendix for switching VAR

Although the switching LDS model includes the switching VAR (SVAR) models, inference in the switching VAR models is exact. For the sake of completness we also provide the gradients for the switching VAR model. Let us start by reviewing the forward and backward inference steps

$L_t(j) = Liklihood(\mathbf{y}_{t-1}, \mathbf{y}_t;\ A(j),\ Q(j))$

$$\mathbf{e}_t = \mathbf{y}_t - A(j)\mathbf{y}_{t-1}\ , \tag{101}$$

$$L_t(j) = \mathcal{N}(\mathbf{e}_t;\ 0,\ Q(j))\ . \tag{102}$$

With the following derivatives:

$$\frac{\partial L_t}{\partial \mathbf{e}_t} = -L_t Q^{-1}\mathbf{e}_t \tag{103}$$

$$\frac{\partial L_t}{\partial A[m,n]} = \frac{\partial L_t}{\partial \mathbf{e}_t} \odot (-\delta_{m,n}\mathbf{y}_{t-1}) \tag{104}$$

$$\frac{\partial L_t}{\partial Q} = -\frac{1}{2}L_t(Q^{-1} - Q^{-1}(\mathbf{e}_t\mathbf{e}_t^\intercal)Q^{-1}) \tag{105}$$

(106)

$M_t^f(j) = ForwardSwitchFilter(M_{t-1}^f, L_t, Z)$

$$a_t^f(j) = L_t(j)M_{t-1}^f(j)Z(i,j)\ , \tag{107}$$

$$M_t^f(j) = \frac{a_t^f(j)}{\sum_{j'} a_t^f(j')}. \tag{108}$$

For $t = 1, \cdots, T$. Note, $M_t^f(i) = Prob(S_t = i|y_{1:t})$, with the initial condition $M_0^f = \pi$.
The partial derivatives are given by

$$\frac{\partial M_t^f(i)}{\partial M_{t-1}^f(j)} = \sum_{k'}[\frac{L_t(i)Z(j,i)\delta_{k',j}}{\sum_{i'} a_t^f(i')} - \frac{L_t(k')Z(j,k')a_t^f(i)}{(\sum_{i'} a_t^f(i'))^2}]\ ,\ t = 1\cdots T. \tag{109}$$

$$\frac{\partial M_t^f(k)}{\partial Z(i,j)} = M_{t-1}^f(i)L_t(j)[\frac{\delta_{k,j}}{\sum_{k'} a_t^f(k')} - \frac{a_t(k)}{(\sum_{k'} a_t^f(k'))^2}]. \tag{110}$$

$$\frac{\partial M_t^f(k)}{\partial L(j)} = \sum_{i'} M_{t-1}^f(i')Z_t^f(i',j)[\frac{\delta_{k,j}}{\sum_{k'} a_t^f(k)} - \frac{a_t^f(k)}{(\sum_{k'} a_t^f(k'))^2}]. \tag{111}$$

(112)

Note that, the backward-step and the corresponding derivatives for the Switching AR are identical to the switching Kalman filter.