A classification-based approach to monitoring the safety of dynamic systems

Shengtong Zhong [a], Helge Langseth [a], Thomas Dyhre Nielsen [b]

[a] *Department of Computer and Information Science, The Norwegian University of Science and Technology, Trondheim, Norway*

[b] *Department of Computer Science, Aalborg University, Aalborg, Denmark*

Abstract

Monitoring a complex process often involves keeping an eye on hundreds or thousands of sensors to determine whether or not the process is stable. We have been working with dynamic data from an oil production facility in the North sea, where unstable situations should be identified as soon as possible. Motivated by this problem setting, we propose a general model for classification in dynamic domains, and exemplify its use by showing how it can be employed for *activity detection*. We construct our model by using well known statistical techniques as building-blocks, and evaluate each step in the model-building process empirically. Exact inference in the proposed model is intractable, so in this paper we experiment with an approximate inference scheme.

1 Introduction

A typical task for the risk and reliability engineer is to monitor the status of a dynamic system, like, e.g., a chemical process. Doing so will often mean tending to a large number of sensors, each of them updating their readings on a regular basis. Real-life processes have their own natural dynamics when everything is running according to plan; "outliers" may on the other hand be seen as indications that the process is leaving its stable state, and thereby becoming more dangerous. Thus, the engineer would like to know if the system is unstable in order to ensure that the proper corrective actions are implemented as soon as the system becomes unsafe. Unfortunately, it may be difficult to measure the status of the system directly, and one will typically only have

Email addresses: shket@idi.ntnu.no (Shengtong Zhong), helgel@idi.ntnu.no (Helge Langseth), tdn@cs.aau.dk (Thomas Dyhre Nielsen).

access to indirect status indicators, which need to be analyzed and combined in a statistical model. Formally, detecting the instantaneous status of a system described by a collection $\boldsymbol{Y} = \{Y_1, Y_2, \ldots, Y_n\}$ of random variables is identical to *classification*, where an object described by a value assignment $\boldsymbol{y} = \{y_1, y_2, \ldots, y_n\}$ is mapped to one of a set of possible labels (or classes). The labels for an object is represented by a class variable C, and are denoted $sp\,(C)$. We will focus on real-valued attributes in this paper, meaning that $\boldsymbol{y} \in \mathbb{R}^n$. In a probabilistic framework, it is well-known that the optimal classifier will label an object $\boldsymbol{y}$ by the class label $\widehat{c}$, where

$$\widehat{c} = \arg \min_{c \in sp(C)} \sum_{c' \in sp(C)} L(c, c') P(c' | \boldsymbol{y}) \tag{1}$$

and $L(c, c')$ is the loss-function encoding the cost of mis-classification. Learning a classifier therefore amounts to estimating the probability distribution $P(C = c|\boldsymbol{y})$.

The engineer may not only want to assess the *instantaneous* status of a system, but rather to detect if the system is *about to become* unstable (that is, to predict future problems). This would give a system operator the chance to implement countermeasures before anyone is exposed to an increased level of risk. Classifiers that fail to take the dynamic aspect of a process into account will not be able to make accurate predictions, and will therefore not be able to recognize a problem under development. In *dynamic classification*, the task is to assign a class label to an object at each time step. To support the classification, objects are characterized by a new observation at each time step as well. We use $\boldsymbol{Y}^t = \{Y_1^t, Y_2^t, \ldots, Y_n^t\}$ to denote the random variables describing the object at time t, where $\boldsymbol{y}^t = \{y_1^t, y_2^t, \ldots, y_n^t\}$ is a specific value assignment to these variables. The collected observations from time $t = 1$ and up to time t is denoted as $\boldsymbol{y}^{1:t}$. The set of possible labels (or classes) for the time series at time t is represented by a class variable C^t, and denoted $sp\,(C^t)$. With the observations $\boldsymbol{y}^{1:t}$ from time step 1 to t, the optimal classifier will label $\boldsymbol{y}^{1:t}$ by the class label $\widehat{c}^t$ at time t

$$\widehat{c}^t = \arg \min_{c^t \in sp(C^t)} \sum_{c' \in sp(C^t)} L(c^t, c') P(c' | \boldsymbol{y}^{1:t});$$

confer also Equation (1).

In a risk and reliability setting, the desire to build efficient statistical models that are flexible yet easy to understand for domain experts has led to reduced focus on traditional frameworks like fault trees. On the other hand, the Bayesian network (BN) framework [28,17] has received increased attention from the community over the last decade [22], partly because BNs have proven to be an attractive alternative to classical reliability formalisms, see e.g., [33,18]. BNs have also been used extensively for classification [8,21,35].

The dynamic Bayesian network framework [12] supports the specification of dynamic processes, and has already found numerous applications in reliability engineering, see, e.g., [20,27]. A simple instance of this framework is the *hidden Markov model* (HMM), which has also been considered for classification purposes [15,31,6]; to this end the "hidden" node in the HMM is used as the classification node, and the attributes at time t are assumed to be independent of those at time $t+1$ given the class label at either of the two points in time. Further simplification can be obtained by assuming that all attributes at one time step are conditionally independent given the class label at that time step; the resulting model by [26] is known as a dynamic naïve Bayes (dNB) classifier. The dNB models can be efficiently estimated from data due to the relatively small number of parameters required to specify them.

To the best of our knowledge, there has been no systematic investigation into the properties of probabilistic classifiers and their applicability to real-life dynamic data. In this paper we will take a step in that direction by examining the underlying assumption of some well-known probabilistic classifiers and their natural extensions to dynamic domains. We do so by carefully linking our analysis back to a real-life dataset, and the result of this analysis is a classification model, which can be used to, e.g., help prevent unwanted events by automatically analyzing a data stream and raise an alert if the process is entering an unstable state. For the discussions to be concrete, we will tie the model development to the task of *activity recognition* in offshore oil drilling; this is further described in Section 2. In Section 3 we give a general overview of the dynamic classification scheme, and we also propose a specific classification model called a *dynamic latent classification model* (abbreviated to dLCM). Next, we look at inference and learning in dLCMs (Section 4), before reporting on their classification accuracy in Section 5. Finally, in Section 6 we conclude and give directions for future research.

2 The domain and the dataset

Offshore oil drilling is a complex process, potentially with major risks to the safety of the operators involved (see, e.g., [34]). Further, the drilling process in itself is extremely expensive, leading to a focus on cost efficient operation, including high demands wrt. the reliability of the equipment employed. This has again resulted in a plethora of data being collected – either for real-time analysis of the state of the ongoing operations or to enable investigations after an events has occurred. We will consider one such dataset from an oil production installation in the North Sea. Data, consisting of 62 variables, is captured every five seconds. The data is monitored in real time by experienced engineers, who have a number of tasks to perform ranging from understanding the situation on the platform in order to avoid a number of either dangerous

or costly situations, to optimization of the drilling operation. The variables that are collected in this dataset cover measurements taken both topside (like flow rates) and down-hole (like, for instance, gamma rate).

The overall drilling process can be broken down into a series of *activities* that are performed iteratively as the depth of the well increases. Recognizing which activity is performed at a given point in time is called *activity recognition*, and is the focus of the present paper. Out of the 62 attributes that are collected, domain experts have selected the following 9 attributes as the most important for activity detection: *Depth Bit Measured*, *Depth Hole Measured*, *Block Position*, *Hookload*, *Weight On Bit*, *Revolutions Per Minute*, *Torque*, *Mud Flow In*, and *Standpipe Pressure*.

In the *Wellsite Information Transfer Specification* (WITS), a total of 34 different activities with associated activity codes are defined. Each activity has its separate purpose and consists of a set of actions. Out of the 34 different drilling activities in total, only a handful are really important to recognize. The important activities in our analysis, which roughly correspond to those that constitute most of the total well drilling time, are described next:

WITS2 – Drilling: The activity occurs when the well is gaining depth by crushing rock at the bottom of the hole and removing the crushed pieces (*cuttings*) out of the well-bore. Thus, the drill string is rotating during this activity, and mud is circulated at low speed to transport out the cuttings. The activity is interrupted by other activities, but continues until the well reaches the reservoir and production of oil may commence.

WITS3 – Connection: This activity involves changing the length of the drill-string, by either adding or removing pieces of drill-pipe.

WITS8 – Tripping in: This is the act of running the drill string into the well hole.

WITS9 – Tripping out: Tripping out means pulling the drill string out of the well bore.

It what follows, the remaining activities will collectively be grouped under the label *Others*.

Knowing which activity is performed at any point in time is important in several contexts: Firstly, the operation of an offshore installation can be monitored by groups of experts located elsewhere (typically in on-shore control-rooms). These experts are shielded from the offshore-operation in that they only observe visualizations of streams of data. Important aggregations, like which activity is performed, helps them better understand the situation on-site.

Secondly, operators are consistently looking for more cost-efficient ways of drilling, and the sequencing of activities during an operation is important for hunting down potential time-sinks.

Thirdly, some undesired events can only happen during specific activities, and knowing the current activity is therefore of high importance. For instance, apparent early warnings of undesired events can be given more credence if that event can actually occur during the current activity and no weight if the event is impossible. From a safety perspective, this allows for a better early-warning system with a lower rate of false alarms.

Finally, it is worth mentioning that activity recognition is a task that also finds applications in areas as diverse as health care [32] and video analysis [23]. In this paper we develop a model for dynamic classification and exemplify the process in the oil drilling domain, but other safety and reliability applications of the developed model are readily available.

3 From static to dynamic Bayesian classifiers

In this section we develop a general framework for performing dynamic classification. The framework will be specified incrementally by examining its expressivity relative to the oil production data. In Section 5 we further justify the framework by setting up an empirical study using the oil production data. In the study we analyze the accuracy results for the sequence of models that are being considered in this section and which lead to the proposed modeling framework.

3.1 Static classifiers

Standard (static) classifiers like NB [5] or TAN [8] assume that the class variable and attributes at different time points are independent given the model. This independence assumption is clearly violated in many domains and, in particular, in domains that specify a process evolving over time. To validate the independence assumptions in practice, we can for instance compare the marginal distribution of the class variable with the conditional distribution of the class variable given its value at the previous time step. From the results using the oil production data, we see a considerable correlation between the class variable of consecutive time slices. In particular, if the system was in the *drilling* activity at time $t-1$, the probability of being in the *drilling* activity also at time t changes from 0.325 (static classifier) to 0.997 (dynamic classifier). The reason for this dramatic difference is that the system tends to remain in the *drilling* activity as soon as *drilling* has commenced, an effect that the static classifier is unable to represent. One way to capture this dependence is to explicitly take the dynamics of the process into account, i.e., to look at *dynamic classifiers*.

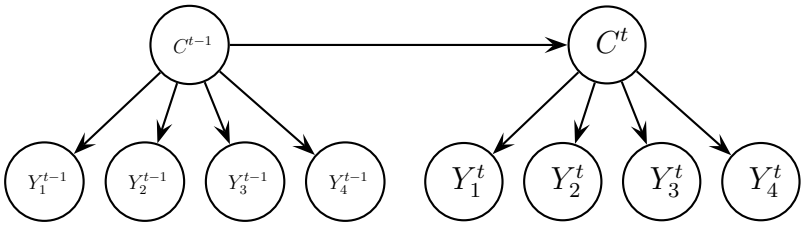

Fig. 1. Attributes are assumed to be conditionally independent given the class variable (equivalent structure for HMM) with $n = 4$.

3.2 A simple dynamic classifier

The temporal dynamics of the class variable can be described using, e.g., a first order Markov model, where $P(C^t|C^{1:t-1}) = P(C^t|C^{t-1})$ for all t. By combining this temporal model with the class-conditional observation model for the attributes we have the well-known hidden Markov model (HMM)[29]. The HMM model is described by a prior distribution over the class variable $P(C^0)$, a conditional observation distribution $P(\boldsymbol{Y}^t|C^t)$, and transition probabilities for the class variable $P(C^t|C^{t-1})$; we assume that the model is *stationary*, i.e., $P(\boldsymbol{Y}^t|C^t) = P(\boldsymbol{Y}^s|C^s)$ and $P(C^t|C^{t-1}) = P(C^s|C^{s-1})$, for all $s, t \geq 1$. HMMs have previously been used in reliability contexts. For example, Smyth [31] considers fault detection in dynamical systems, Durand and Gaudoin [6] applies HMMs for modeling the failure and debugging process of software, and Zamalieva et al. [36] use HMMs for online labeling of event sequences wrt. failure and non-failure scenarios.

With a continuous observation vector, the typical way of modeling the conditional distribution is to use a class-conditional multivariate Gaussian distribution with mean $\boldsymbol{\mu}_c$ and covariance matrix $\boldsymbol{\Sigma}_c$, i.e., $\boldsymbol{Y}|\{C = c\} \sim N(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$ [10]. Unfortunately, learning a full covariance matrix involves estimating a number of parameters that is quadratic in the number of attributes, which may result in over-fitting when data is scarce compared to the number of free parameters. One approach to alleviate this problem is to introduce additional independence assumptions about the domain being modeled. Specifically, by assuming that all variables are independent given the class variable, we will at each time step have a NB model defined by a diagonal covariance matrix, thus requiring only $O(n)$ parameters to be learned, where $n = |\boldsymbol{Y}|$ is the number of attributes in the model. This structure corresponds to the dNB model for dynamic domains. A graphical representation of the resulting independence assumptions can be seen in Fig. 1 in the form of a 2TBN [25].

As for the (static) NB model, the independence assumptions encoded in the dNB model are often violated in real-world settings. For example, if we consider the measured flow of drilling fluid going in to the well (*Mud Flow In*) and the observed pressure in the well (*Stand Pipe Pressure*), and plot their values conditioned on the class variable (activities *tripping in* and *tripping out*), it

is evident that there is a conditional correlation between the two attributes given the class, see Fig. 2; similar results are also obtained when considering other pairs of attributes.

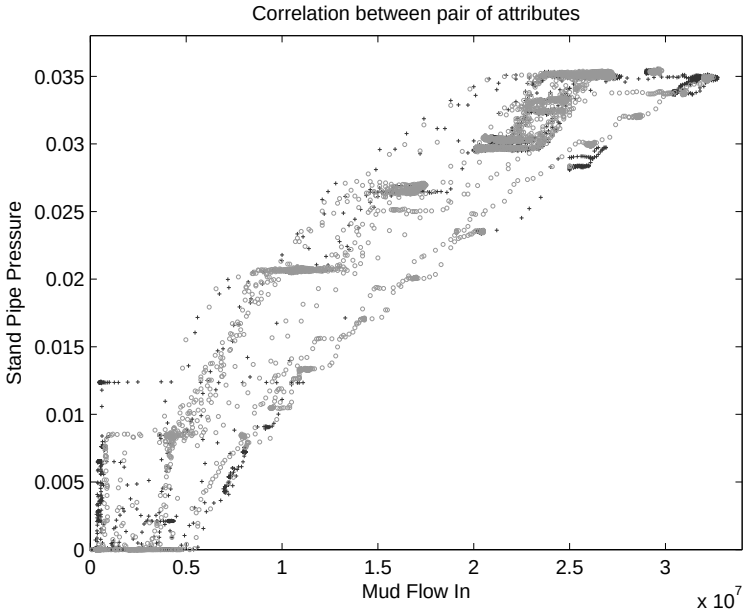

Fig. 2. Scatter plot of the *Mud Flow In*$_t$ (x-axis) and the *Stand Pipe Pressure*$_t$ (y-axis) for the two classes in the oil production data (black "+" is the *tripping in* class and grey "o" is the *tripping out* class). The conditional correlation between the two attributes is evident.

3.3 Modeling dependence between attributes

There are several approaches to model attribute dependence. For example, Friedman et al. [9] propose an extension of the TAN model [8] to facilitate continuous domains. In the TAN framework, each attribute is allowed to have at most one parent besides the class variable. As an alternative, [21] present the latent classification model (LCM), which can be seen as combining the NB model with a factor analysis model [7].

An LCM offers a natural extension of the NB model by introducing continuous latent variables $\boldsymbol{Z} = (Z_1, \ldots, Z_k)$ as children of the class variable C and parents of all the attributes $\boldsymbol{Y} = (Y_1, \ldots, Y_n)$. The latent variables and the attributes work as a factor analyzer focusing on modeling the correlation structure among the attributes.

Following the approach by [21], we introduce latent variables to encode conditional dependencies among the attributes. Specifically, for each time step t we have the vector $\boldsymbol{Z}^t = (Z_1^t, \ldots, Z_k^t)$ of latent variables that appear as children of the class variable and parents of all the attributes (see Fig. 3). The latent

variable $\boldsymbol{Z}^t$ is assigned a multivariate Gaussian distribution conditional on the class variable and the attribute vector $\boldsymbol{Y}$ is also assumed to be a multivariate Gaussian distribution conditional on the latent variables:

$$\boldsymbol{Z}^t|\{C^t = c^t\} \sim N(\boldsymbol{\mu}_{c^t}, \boldsymbol{\Sigma}_{c^t}),$$
$$\boldsymbol{Y}^t|\{\boldsymbol{Z}^t = \boldsymbol{z}^t\} \sim N(\boldsymbol{L}\boldsymbol{z}^t, \boldsymbol{\Theta}),$$

where $\boldsymbol{\Sigma}_{c^t}$ and $\boldsymbol{\Theta}$ are diagonal matrixes and $\boldsymbol{L}$ is the transition matrix; note that the stationarity assumption is encoded in the model.

In this model, the latent variables capture the dependencies between the attributes. They are conditionally independent given the class but marginally dependent. Furthermore, the same mapping, $\boldsymbol{L}$, from the latent space to the attribute space is used for all classes, and hence, the relation between the class and the attributes is conveyed by the latent variables only.

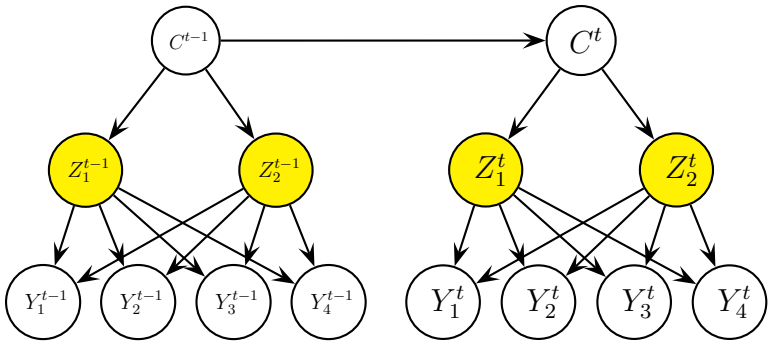

Fig. 3. In each time step, the conditional dependencies between the attributes are encoded by the latent variables (Z_1^t, Z_2^t).

The model in Fig. 3 assumes that the attributes in different time slices are independent given the class variable. This assumption implies that the temporal dynamics is captured at the class level only. When the state specification of the class variable is coarse, then this assumption will rarely hold (obviously, the finer the granularity of the state specification of the class variable, the more appropriate this assumption will be). For the oil production data, this assumption does not hold as we can see in Fig. 4, which show the conditional correlation of the *Stand Pipe Pressure* attribute in successive time slices in both *tripping in* and *tripping out* activities.

We propose to address this apparent short-coming by modeling the dynamics of the system at the level of the latent variables, which semantically can be seen as a compact representation of the "true" state of the system. At this abstraction level, the modeling approach is related to that of Kohda and Cui [20] who propose a factorial HMM, where latent/unobserved variables are used to model temporal dynamics in safety monitoring systems. To be more specific,

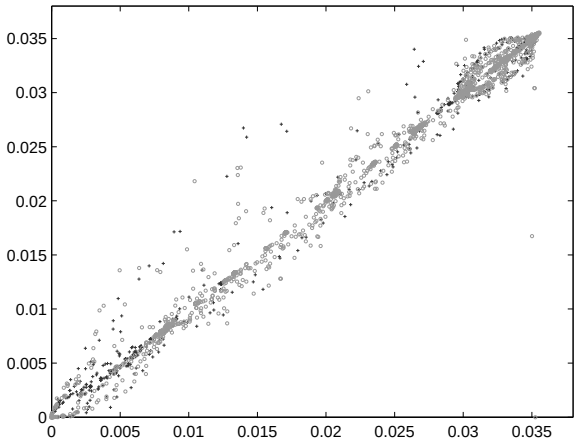

Fig. 4. Scatter plot of the *Stand Pipe Pressure* (x-axis is value of *Stand Pipe Pressure* at time t and y-axis is value of *Stand Pipe Pressure* at time $t+1$) for the two classes in the oil production data (black "+" is *tripping in* class and gray "o" is *tripping out* class). The conditional correlation of this attribute over time is evident.

we encode the state specific dynamics by assuming that the latent variable vector $\boldsymbol{Z}^t$ follows a linear multivariate Gaussian distribution conditioned on $\boldsymbol{Z}^{t-1}$:

$$\boldsymbol{Z}^t | \{\boldsymbol{Z}^{t-1} = \boldsymbol{z}^{t-1}, C^t = c^t\} \sim N(\boldsymbol{A}_{c^t} \boldsymbol{z}^{t-1}, \boldsymbol{\Sigma}_{c^t})$$

where $\boldsymbol{A}_{c^t}$ encodes the class conditional transition dynamics for the latent variables. A graphical representation of the model is given in Fig. 5, and will be referred to as a *dynamic latent classification model* (dLCM). Observe that conditional on the class variables, the state specific model dynamics is related to a factorized Kalman filter model.

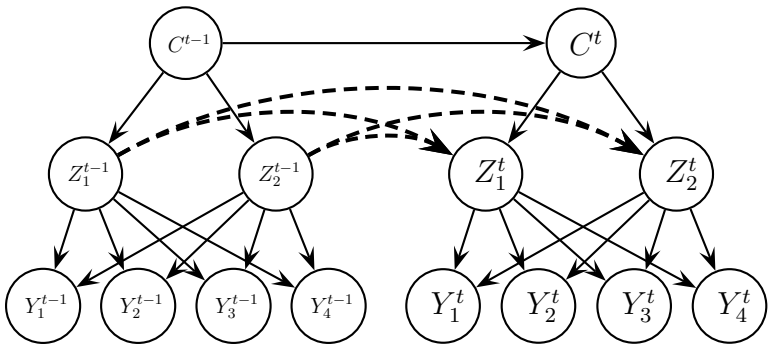

Fig. 5. The state specific dynamics are encoded at the level of the latent variables with $k = 2$ and $n = 4$.

3.4 Modeling non-linear systems

One of the main assumptions in the model above is that there is a linear mapping from the latent variables to the attribute space as well as a linear mapping from the latent variables in a given time slice to the latent variables in the succeeding time slice (i.e., that the state specific dynamics are linear). When the class variable takes a fixed value, the model is equivalent to a linear dynamical system (LDS) [1], also known as a linear state-space model.

Given sufficient dimension of the latent space, an LDS can represent any complex real-world processes, although the computational cost can make the LDS model infeasible in practice if the target process exhibit a complicated behavior (see Appendix C). In order to reduce the computational cost while maintaining the representational power, we introduce a discrete mixture variable M for each time slice as done by [21] for static domains (see Fig. 6). A related dynamic model is the *switching* state-space model (SSSM), which was proposed to combine discrete and continuous dynamics [13]. The representational power and computational efficiency of the SSSM have been well demonstrated [13,2]. The model we propose differs from the SSSM not only by the introduction of a class variable, but also by our model using the discrete class variables to carry the dynamics over time (whereas this is achieved by the latent mixture node for the SSSM). SSSMs focus on modeling non-linear real-world process with one single system state (class), whereas our model is intended to capture the non-linearity of multiple system states (classes) at the same time.

We call our model a dynamic latent classification model (dLCM), and note that we for each time slice can regard the model as combining a naïve Bayes model with a mixture of factor analyzers. In this case, the mixture variable follows a multinomial distribution conditioned on the class variable, and the attributes $\boldsymbol{Y}^t$ follow a multivariate Gaussian distribution conditioned on the latent variables and the discrete mixture variable, i.e.:

$$
\begin{aligned}
&M^t|\{C^t = c^t\} \sim P(M^t|C^t = c^t),\\
&\boldsymbol{Y}^t|\{\boldsymbol{Z}^t = \boldsymbol{z}^t, M^t = m^t\} \sim N(\boldsymbol{L}_{m^t}\boldsymbol{z}^t, \boldsymbol{\Theta}_{m^t}),
\end{aligned}
$$

where $1 \leq m^t \leq |sp\,(M)|$.

4 Learning and inference

In what follows we discuss algorithms for performing inference and learning in the proposed models.

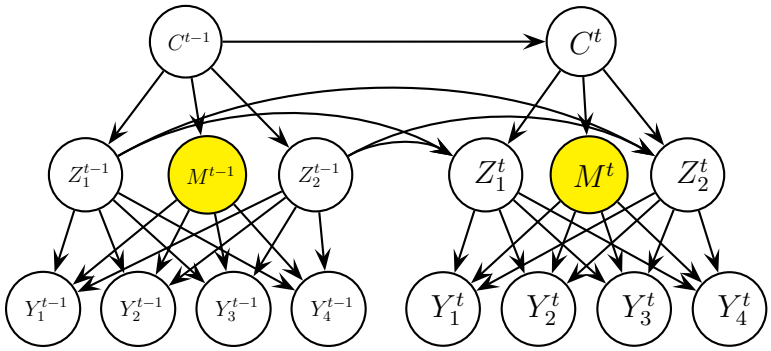

Fig. 6. A mixture variable M is introduced at each time slice to extend of the model, with $k = 2, n = 4$.

4.1 Inference

By inference we mean to calculate the conditional distribution over some variables of interest given observations of others, for instance calculating $P(c^t|\boldsymbol{y}^{1:t})$. As the number of time-slices for which data has been collected may be quite large, we are looking for an efficient way of calculating these probability distributions.

As our model is structurally similar to the linear dynamical systems model [1], we will find inspiration for our inference scheme from the inference algorithms associated with those models [30,1]. Therefore, we will consider two phases of inference, the *forward* (or *filtering*) phase and the *backward* (*smoothing*) phase. The results of the forward and backward calculations are given in the next subsections; further details can be found in Appendix A.

Filtering using forward recursion

The goal of the filtering phase is to quantify the uncertainty over the state of the system at time t given the observations we have up to and including time t. Primarily, the variable of interest is the class variable C^t, but simultaneously, the latent variables $(\boldsymbol{Z}^t, M^t)$ also convey information, and are therefore also of relevance. We thus calculate the probability distribution $p(c^t, \boldsymbol{z}^t, m^t|\boldsymbol{y}^{1:t})$ during filtering, and this is done recursively in t. This means that $p(c^t, \boldsymbol{z}^t, m^t|\boldsymbol{y}^{1:t})$ is found using the related results from the previous time-step, $p(c^{t-1}, \boldsymbol{z}^{t-1}, m^{t-1}|\boldsymbol{y}^{1:t-1})$, combined with information about how likely the new observation $\boldsymbol{y}^t$ is and how the system evolves over time (see also Appendix A):

$$p(\boldsymbol{z}^t, m^t, c^t, |\boldsymbol{y}^{1:t}) \;\propto\; p(\boldsymbol{y}^t|\boldsymbol{z}^t, m^t)\, p(m^t|c^t) \cdot \sum_{c^{t-1}} p(c^t|c^{t-1}) \int_{\boldsymbol{z}^{t-1}} p(\boldsymbol{z}^t|\boldsymbol{z}^{t-1}, c^t) \sum_{m^{t-1}} p(\boldsymbol{z}^{t-1}, m^{t-1}, c^{t-1}|\boldsymbol{y}^{1:t-1})\, \mathrm{d}\boldsymbol{z}^{t-1}. \quad (2)$$

However, by examining the inference rule above we see that exact filtering is intractable (scaling exponentially with t, see also [24,2]) because neither the class variables nor the mixture variables are observed: At time $t = 1$, $p(\boldsymbol{z}^1, m^1, c^1|\boldsymbol{y}^1)$ is built up by a single Gaussian. However, at time-step $t = 2$, due to the summation over the class variable C^1 and mixture variable M^1 in Equation (2), $p(\boldsymbol{z}^2, m^2, c^2|\boldsymbol{y}^{1:2})$ will contain a mixture of $|sp\,(C)| \cdot |sp\,(M)|$ Gaussians; the model contains a mixture of $|sp\,(C)|^2 \cdot |sp\,(M)|^2$ Gaussians at $t = 3$, and so on. To control this explosion in computational complexity, we will resort to Gaussian collapse method [3,2]. The Gaussian collapse guarantees that the distribution $p(\boldsymbol{z}^t, m^t, c^t, |\boldsymbol{y}^{1:t})$ is represented by a single Gaussian at any time-step t; details are given in Appendix A.

Smoothing using the backward recursion

Similar to the forward pass, the backward pass also relies on a recursive computation. Note that where the traditional forward-backward algorithm would make use of the backward-phase to calculate $p(\boldsymbol{y}^{t+1:T}|\boldsymbol{z}^t, m^t, c^t)$ [1], we rather follow [2] and calculate $p(\boldsymbol{z}^t, m^t, c^t|\boldsymbol{y}^{1:T})$ instead. The idea is to compute $p(\boldsymbol{z}^t, m^t, c^t|\boldsymbol{y}^{1:T})$ from the corresponding result of the previous recursive step, $p(\boldsymbol{z}^{t+1}, m^{t+1}, c^{t+1}|\boldsymbol{y}^{1:T})$. As the calculations are a bit involved they are not presented here, but can be found in Appendix A.2.

There are two approximations involved when completing the backward recursion: Firstly, similar to [2], we approximate $p(\boldsymbol{z}^{t+1}|m^{t:t+1}, c^{t:t+1}, \boldsymbol{y}^{1:T})$ by $p(\boldsymbol{z}^{t+1}|m^{t+1}, c^{t+1}, \boldsymbol{y}^{1:T})$. Secondly, we observe that $p(\boldsymbol{z}^t, m^t, c^t|\boldsymbol{y}^{1:T})$ will become a mixture of Gaussians, and the number of components increases exponentially in $T - t$ (see Equation (A.3)). We therefore resort to the same solution strategy as for the forward phase, and approximate $p(\boldsymbol{z}^t, m^t, c^t|\boldsymbol{y}^{1:T})$ by a single Gaussian at each time-point t; see Appendix A for more details.

4.2 Learning

Learning the dLCM model involves estimating the parameters of the model, the number of latent variables, and the number of mixture components. With the number of latent variables and the number of mixture components specified, parameter learning becomes simple: since the class variables are always observed during learning, the resulting model is similar to a linear state-space model for which an EM-algorithm [4] can be applied.

Following the approach of [21], we determine the number of latent variables, k, and the number of mixture components, $|sp\,(M)|$, by performing a systematic search over a selected subset of candidate structures (note that a model structure is completely specified by its number of latent variables and num-

ber of mixture components). For each candidate structure we learn the model parameters by applying the EM-algorithm, and the quality of the resulting model is then evaluated using the wrapper approach [19]. Finally, we select the model with the highest score.

Algorithm 1 Learn an dLCM classifier with the wrapper approach

Input: A dataset $\boldsymbol{D}^{1:T}$. Number of wrapper folds to use: γ.
Output: A dLCM classifier.

1: Partition the dataset into γ wrapper folds $\boldsymbol{W}_1, \ldots, \boldsymbol{W}_\gamma$ so that each $\boldsymbol{W}_w$ is a (time-)continuous part of $\boldsymbol{D}^{1:T}$.
2: **for** possible values of k and $|sp(M)|$ **do**
3: **for** $w = 1, \ldots, \gamma$ **do**
4: Learn a classifier from the $\boldsymbol{D}^{1:T} \setminus \boldsymbol{W}_w$.
5: Calculate the accuracy on the validation dataset $\boldsymbol{W}_w$.
6: **end for**
7: Score the parameter-pair (k, $|sp(M)|$) by the average accuracy obtained over the wrapper folds.
8: **end for**
9: Select the values of k and $|sp(M)|$ with highest accuracy.
10: **return** classifier learned with these parameters.

In data-rich situations one may alternatively choose to use a *validation-set* for model selection instead of employing the wrapper-approach. The loop starting on Line 3 in Algorithm 1 is in this case replaced by a single call to the parameter learning routine (using the full training-set $\boldsymbol{D}^{1:T}$), and the model selection part (Line 7) will be based on the accuracy obtained on the validation-set. This is the approach we will use in Section 5.

The parameters of the dLCM model are learned using the EM algorithm. At each iteration, the model parameters are obtained by maximizing the expected log likelihood function, resulting in the updating rules given in Appendix B.

5 Experiments Results

5.1 Setup

In this section we empirically analyze the performance of the proposed dynamic latent classification model. The analysis is based on the oil drilling data described in Section 2. The data consists of sensor readings from 62 sensors captured at a sampling frequency of every 5 seconds. Out of the 62 sensors, domain experts deem that only 9 of the sensors are important for recognizing the activities listed in Section 2. Consequently, we focus on this reduced

set of sensor readings in the experiments. The data was collected during approximately 31 hours, yielding time series data containing 220 000 observation vectors, where each observation vector consists of a value for the class variable (encoding the type of activity being performed) and a configuration of the nine sensor variables.

The full dataset covers a total of five oil drilling activities, namely *drilling*, *connection*, *tripping in*, *tripping out*, and *others*; *others* is an abstract activity covering all other activities besides the four mentioned previously, c.f. Section 2. In our experiments, the data was divided into training, validation, and test datasets consisting of 90 000, 80 000, and 50 000 time slices, respectively. The training data was chosen as the initial segment of the time series, whereas the validation and test set was chosen as the intermediate and end segment, respectively. We note that since the different segments represent different phases of the drilling, the data generation process may be different in the three segments (even for the same activity), and furthermore, the fraction of time spent doing the different activities may also change between the segments. As an example, tripping into the well will take more time when the length of the well increases, and correspondingly, the marginal probability for doing the *Tripping In* activity changes from 12% during the initial phase (the training set) to 21% in the last phase (the test set); see Table 1 for further details. Obviously, this complicates the classification problem further.

Table 1
Empirical distribution over the activities for training-set, validation-set and test-set

Activity	Training-set	Validation-set	Test-set
Drilling	42.7%	42.1%	42.6%
Connection	32.5%	21.4%	23.3%
Tripping in	12.2%	15.7%	21.3%
Tripping out	10.8%	20.8%	12.8%
Others	1.8%	0.0%	0.0%

For the classification of the five activities, we have followed the recommendations of the domain experts and used a two-step hierarchical classification process. At the first step we construct two abstract activities by first merging together the three activities *drilling*, *connection*, and *others* into one group, then the activities *tripping in* and *tripping out* into another. For the experiments, this corresponds to constructing a new data set, where the actual activities have been replaced by the two activities *drilling/connection* and *tripping in/out*. After having classified an activity as e.g. *tripping in/out*, we proceed with a second step and attempt to refine the classification by reclassifying the activity as either *tripping in* or *tripping out*. Consequently, we have trained three distinct classifiers (M_{top}, $M_{\text{i/o}}$, and $M_{\text{d/c}}$) corresponding to the two-step classification procedure of the activities. When conducting the ex-

periments we first deployed the M_{top} model to classify an activity at time t as either *drilling/connection* or *tripping in/out*. If the activity was classified as *tripping in/out* we used the model $M_{\text{i/o}}$ to refine the classification based on the largest consecutive sequence of data points $\boldsymbol{y}^{t':t}$ classified as *tripping in/out*. The process for reclassifying *drilling/connection* is analogous, but using $M_{\text{d/c}}$.

5.2 Learning procedure for the dLCM models

The dLCM learning framework consists of two components: learning the parameters for a given model structure and finding an appropriate model structure. In our learning setup these two activities are interleaved (see Section 4.2). For learning the model parameters, we employ the EM algorithm described in Section 4.2, where the termination condition was set to either 50 iterations or if the relative change in log-likelihood over two consecutive iterations falls below $\epsilon = 10^{-4}$. In order to find the structure of the model (i.e., the number of latent variables and the state space of the mixture variable M) we adopt a greedy search strategy. More specifically, the search strategy is characterized by *i*) a systematic approach for selecting values for $|sp(M)|$ and $|sp(\boldsymbol{Z})|$, and, given such a pair of values, *ii*) learning the parameters in the model. Each candidate model is then scored by estimating its classification accuracy using a separate validation dataset.

When learning classification models for the second hierarchical step (the $M_{\text{i/o}}$ and $M_{\text{d/c}}$ models), we first extract the relevant training data, e.g., those examples that are classified as either *tripping in* or *tripping out* are relevant for $M_{\text{i/o}}$, and subsequently use this extracted data for learning. We thereby obtain a *collection* of time series, but for the purpose of the experiments reported in the present paper, we have treated these time series as a *single* time series during learning. For future work, the learning algorithm will be adapted to allow for multiple time series.

5.3 Results

To analyze the classification performance of the dLCM classifier, we have compared the following list of classifiers: [1]

NB: The naïve Bayes classifier discussed in Section 3.1.
dNB: The naïve Bayes classifier extended with dynamics on the class variable (described in Section 3.2).

[1] Observe that all the intermediate models considered in the development of the dLCM model are included among the straw-men.

dLCM$_1$: The dLCM classifier without the mixture component, as described in Section 3.3. The name is chosen to signify that the model is identical to a dLCM with only one state for the mixture variable (i.e., $|sp(M)| = 1$).

dLCM: The full dLCM classifier, with structure learned as described above.

LGL: The local-global learning extension of the naïve Bayes classifier towards discriminative learning [38].

J48: The J48 decision tree implementation in Weka [14] using standard parameter settings.

Note that the static classifiers (NB, LGL, J48) base their activity-classification at time t only on $\boldsymbol{y}^t$, whereas the dynamic classifiers (dNB, dLCM$_1$, dLCM) use observations up until time t.

The results of the first hierarchical classification step for *drilling/connection* and *tripping in/out* are summarized in Table 2. We see that the classification accuracy increases with the expressiveness of the model: The poorest results are obtained by the naïve Bayes and LGL models. Adding dynamics at the class level (dNB) improves the results, but somewhat surprisingly, only by 0.03%. dLCM$_1$ is clearly better than dNB, hence it seems that representing the dynamics only at the class level (as the dNB does) is not sufficient to recognize the different activities; one must also capture the dynamics among the attributes to faithfully represent the important properties of the data. Introducing the ability to model non-linear systems in the full dLCM model also contributes significantly by reducing the error from 2.86% to 1.28%. Finally, it is interesting to see that J48 also fares very well at this level of classification, being able to separate the combined activities almost at the same level as dLCM$_1$.

Table 2
Summary of classification accuracy for the first hierarchical step.

Model	Classification accuracy
NB	84.77
dNB	84.80
dLCM$_1$	97.14
dLCM	98.72
LGL	84.77
J48	96.83

The detailed classification results using the dLCM classifier are shown in Fig. 7, with time on the x-axis and the class label on the y-axis. The combination *drilling/connection* corresponds to $y = 1$ and *tripping in/out* corresponds to $y = 2$. The correct classifications are shown in the topmost plot and the dLCM classifications are given in the lowermost plot. The results appear to be

quite good overall, but with some patches of observations erroneously classified as *tripping in/out*.

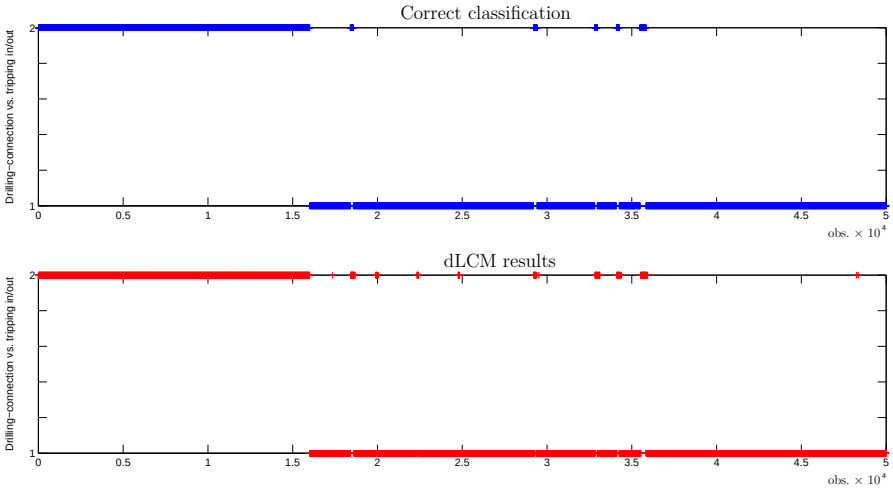

Fig. 7. The dLCM classification results for the first hierarchical step: *drilling/connection* activity (1) vs. *tripping in/out* activity (2).

Next, we trained separate classifiers for recognizing the fives activities *drilling*, *connection*, *tripping in*, *tripping out*, and *other* in an hierarchical process as outlined above. The results are given in Table 3. We can see the same tendency as we did for the aggregated activities (Table 2) apart from two important issues: Firstly, we see a clearer benefit of the dynamic model at the class level, as the dNB is now much better than the NB. Secondly, J48 does not produce good results for the full hierarchical classification procedure. Where J48 could separate the aggregated activities using combinations of attribute values inside a single time-step that were impossible (NB, dNB, LGL) or costly ($dLCM_1$, dLCM) to capture for the other models, this is clearly not sufficient for the overall classification task.

Table 3
Summary of the accuracy results for the full hierarchical classification process.

Model	Accuracy
NB	58.53
dNB	61.69
$dLCM_1$	76.29
dLCM	82.09
LGL	60.41
J48	66.75

For completeness, Table 4 shows the results of each refinement-classifier. In this table, each number gives the accuracy of that sub-classifier given that

the aggregated class is correct. For instance, 60.72 for "*NB – tripping in/out*" means that out of all examples that are either *tripping in* or *tripping out*, NB classifies 60.72% correctly. Apparent discrepancies between Table 3 and Table 4 are thus to be understood in light of misclassifications at the top-level of the hierarchical process (see also Table 2). The conclusions we can draw from Table 4 correspond well with those drawn from Table 3: Firstly, the experimental results show that the dLCM classifiers (dLCM_1, dLCM) achieve significantly better accuracy results than the static classifiers (NB, J48, and LGL). This verifies the need for dynamic classification models as outlined in this paper. Next, the results also show that the full dLCM is more effective than the two intermediate dynamic straw-men (dNB, dLCM_1), justifying the added model complexity of the dLCM classifier. Finally, we would like to note that the accuracy results for *tripping in/out* are consistently lower than those for *drilling/connection*. A potential contributing factor to this difference is the change in data characteristics (in particular, the distribution of the *tripping in/out* activities) that we observe when comparing the training, validation, and test set; see also Table 1.

Table 4
Accuracy results for the second-level classifications.

Model	tripping in/out	drilling/connection
NB	60.72	67.64
dNB	59.96	76.23
dLCM_1	75.57	76.66
dLCM	75.57	85.47
LGL	61.07	70.18
J48	57.88	69.56

Table 5 lists the models that were selected by the learning procedure. We chose the number of latent variables in the interval from $k = 3$ and up to $k = 27$ while keeping $|sp(M)| = 1$ for dLCM_1; the full dLCM models were chosen with $k \in [3, 27]$ and $|sp(M)| \in [1, 3]$.

Table 5
The complexity of the learned LCM models.

Classifier	dLCM_1	dLCM
M_{top}	$k = 24$	$k = 24$, $\lvert sp(M)\rvert = 2$
$M_{\text{i/o}}$	$k = 21$	$k = 21$, $\lvert sp(M)\rvert = 1$
$M_{\text{d/c}}$	$k = 24$	$k = 21$, $\lvert sp(M)\rvert = 3$

Finally, the detailed behavior of the dLCM classifier can be seen in Fig. 8, where again time runs along the x-axis and activity is encoded on the y-axis. For the activities, *Drilling* corresponds to a value of 1, *Connection* is given

value 2, *Tripping In* and *Tripping Out* are encoded as $y = 3$ and $y = 4$ respectively, and finally *Others* maps to a value of 5. The top-most plot shows the correct classifications, whereas the bottom-most plot shows the results of the dLCM classifier. It is apparent that many of the mistakes made by the classifier is due to an over-reliance on the *Other*-activity, which is in fact not present in the test-set at all. A reason for this behavior is that since *Other* is made up by a combination of several activities, its data pattern is not very well defined, and therefore basically used as a "default" when no activity seems to fit the data well.

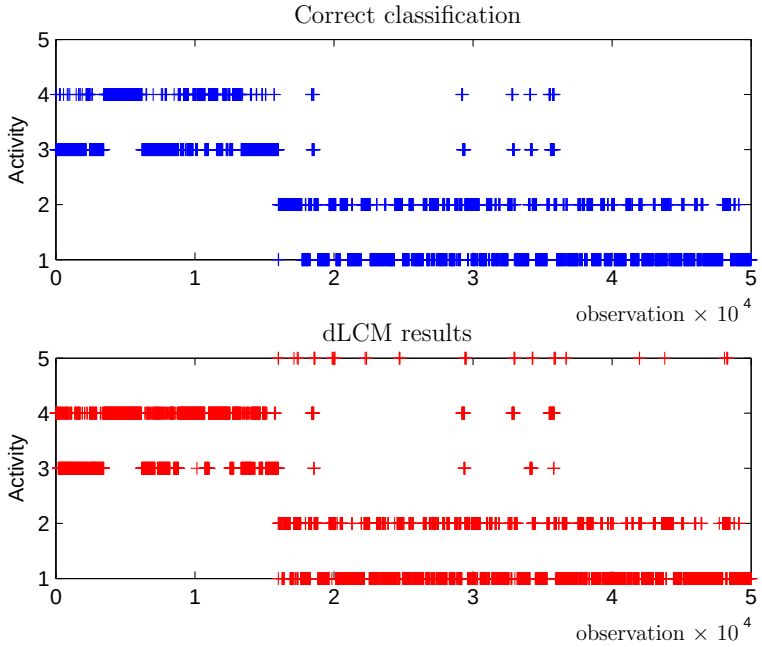

Fig. 8. The detailed accuracy results for the full hierarchical classification process.

6 Conclusions

Systems for analyzing streaming data are of great importance for reliability engineering, where an obvious application area is process monitoring; when monitoring a process the typical task is to determine the state of the process based on streaming data consisting of current and past sensor readings. In this paper we have described a new family of models specifically designed for the analysis and classification of such streaming data. We have specified data-driven learning and inference procedures for this model class and exemplified its use by looking at online activity recognition for an oil drilling facility, where we empirically showed that our classification model significantly outperforms other standard candidate straw-men classifiers.

Our dynamic latent classification model (dLCM) generalizes the latent classification model by Langseth and Nielsen [21] to dynamic domains, and is closely related to switching stat-space models [13]. The model class is sufficiently expressive to capture the underlying dynamics of the oil drilling data, but is unfortunately not amenable to exact inference. Instead we have employed an approximate inference scheme inspired by [2]. We have already initiated an investigation into the appropriateness of the approximate inference scheme by comparing our approach to traditional sampling techniques (e.g., [11]), see [37] for details, and we plan to continue this investigation as part of our future work.

The dLCM model is a general purpose classification model for dynamic data, and even though we have exemplified its use for activity detection in the oil production domain, other risk and reliability applications are also relevant. As an example, we plan to use the classification model to do *event detection* directly, i.e., to foresee – and therefore help the operators prevent – undesired situations. This can be a difficult problem if the events one tries to to detect are *rare*, as that would lead to unbalanced classification problems [16], and we thus plan to devise a semi-discriminative strategy in the spirit of [38] to examine the appropriateness of maximum-likelihood based learning in this setting. Finally, traditional probabilistic classification based on Equation (1) requires the specification of a meaningful loss-function, and in this paper we have utilized the 0/1-loss, which is identical to maximizing the classification accuracy. In the dynamic classification setting, one could also want to encode more advanced statements that take the dynamics into consideration, like for instance "*Detecting an event before a minute has past is worth $1, but detection after 90 seconds is useless*". Such statements require a richer representation than a (static) loss function, and we are planning to look into formal languages for describing them.

Acknowledgments

We would like to thank Ana M. Martínez for her participation at the offset of this work [39], and Sigve Hovda at Verdande Technology, who helped with preparing and understanding the data.

A Inference

A.1 Forward recursion: filtering

A simple decomposition of $p(\boldsymbol{z}^t, m^t, c^t|\boldsymbol{y}^{1:t})$ gives us

$$\begin{aligned} p(\boldsymbol{z}^t, m^t, c^t|\boldsymbol{y}^{1:t}) &= p(\boldsymbol{z}^t, m^t, c^t, \boldsymbol{y}^t|\boldsymbol{y}^{1:t-1})/p(\boldsymbol{y}^t|\boldsymbol{y}^{1:t-1}) \\ &\propto p(\boldsymbol{z}^t, m^t, c^t, \boldsymbol{y}^t|\boldsymbol{y}^{1:t-1}). \end{aligned}$$

Disregarding the normalisation constant $p(\boldsymbol{y}^t|\boldsymbol{y}^{1:t-1})$ for now, we examine $p(\boldsymbol{z}^t, m^t, c^t, \boldsymbol{y}^t|\boldsymbol{y}^{1:t-1})$ further. Using the law of total probability, we get

$$\begin{aligned} p(\boldsymbol{z}^t, m^t, c^t, \boldsymbol{y}^t|\boldsymbol{y}^{1:t-1}) &= \int_{\boldsymbol{z}^{t-1}} \sum_{m^{t-1}} \sum_{c^{t-1}} p(\boldsymbol{z}^{t-1:t}, m^{t-1:t}, c^{t-1:t}, \boldsymbol{y}^t|\boldsymbol{y}^{1:t-1}) \mathrm{d}\boldsymbol{z}^{t-1} \\ &= \int_{\boldsymbol{z}^{t-1}} \sum_{m^{t-1}} \sum_{c^{t-1}} p(\boldsymbol{z}^{t-1}, m^{t-1}, c^{t-1}|\boldsymbol{y}^{1:t-1}) \cdot \\ &\qquad p(\boldsymbol{y}^t, \boldsymbol{z}^t, m^t, c^t|\boldsymbol{y}^{1:t-1}, \boldsymbol{z}^{t-1}, m^{t-1}, c^{t-1}) \mathrm{d}\boldsymbol{z}^{t-1}. \end{aligned} \tag{A.1}$$

Equation (A.1) can be simplified, but first we note that the term $p(\boldsymbol{z}^{t-1}, m^{t-1}, c^{t-1}|\boldsymbol{y}^{1:t-1})$ is already available form the previous time-step, and thus needs no further re-calculation. Next,

$$\begin{aligned} &p(\boldsymbol{y}^t, \boldsymbol{z}^t, m^t, c^t|\boldsymbol{y}^{1:t-1}, \boldsymbol{z}^{t-1}, m^{t-1}, c^{t-1}) = \\ &\quad p(\boldsymbol{y}^t|\boldsymbol{z}^{t-1:t}, m^{t-1:t}, c^{t-1:t}, \boldsymbol{y}^{1:t-1}) \cdot p(\boldsymbol{z}^t|\boldsymbol{z}^{t-1}, m^{t-1:t}, c^{t-1:t}, \boldsymbol{y}^{1:t-1}) \cdot \\ &\quad p(m^t|\boldsymbol{z}^{t-1}, m^{t-1}, c^{t-1:t}, \boldsymbol{y}^{1:t-1}) \cdot p(c^t|\boldsymbol{z}^{t-1}, m^{t-1}, c^{t-1}, \boldsymbol{y}^{1:t-1}), \end{aligned}$$

and now we can utilize the conditional independence statements encoded in the model. We notice (see also Fig. 6) that

$$\boldsymbol{Y}^t \perp\!\!\!\perp \{\boldsymbol{Y}^{1:t-1}, C^{t-1:t}, Z^{t-1}, M^{t-1}\} | \{Z^t, M^t\},$$

thus $p(\boldsymbol{y}^t|\boldsymbol{z}^{t-1:t}, m^{t-1:t}, c^{t-1:t}, \boldsymbol{y}^{1:t-1}) = p(\boldsymbol{y}^t|\boldsymbol{z}^t, m^t)$, which is simply a parameter of the model, and therefore requires no further calculation. In a similar way, we can find that, $p(\boldsymbol{z}^t|\boldsymbol{z}^{t-1}, m^{t-1:t}, c^{t-1:t}, \boldsymbol{y}^{1:t-1}) = p(\boldsymbol{z}^t|\boldsymbol{z}^{t-1}, c^t)$, $p(m^t|\boldsymbol{z}^{t-1}, m^{t-1}, c^{t-1:t}, \boldsymbol{y}^{1:t-1}) = p(m^t|c^t)$ and $p(c^t|\boldsymbol{z}^{t-1}, m^{t-1}, c^{t-1}, \boldsymbol{y}^{1:t-1}) = p(c^t|c^{t-1})$. Eventually, we obtain

$$\begin{aligned} p(\boldsymbol{z}^t, m^t, c^t, |\boldsymbol{y}^{1:t}) \;\propto\; & p(\boldsymbol{y}^t|\boldsymbol{z}^t, m^t)\, p(m^t|c^t) \cdot \\ & \sum_{c^{t-1}} p(c^t|c^{t-1}) \int_{\boldsymbol{z}^{t-1}} p(\boldsymbol{z}^t|\boldsymbol{z}^{t-1}, c^t) \sum_{m^{t-1}} p(\boldsymbol{z}^{t-1}, m^{t-1}, c^{t-1}|\boldsymbol{y}^{1:t-1})\, \mathrm{d}\boldsymbol{z}^{t-1}, \end{aligned} \tag{A.2}$$

which shows how $p(\boldsymbol{z}^t, m^t, c^t, |\boldsymbol{y}^{1:t})$ can be calculated recursively in t.

However, by examining Equation (A.2) we see that exact filtering is intractable (scaling exponentially with t, see also [24,2]) because neither the class variables nor the mixture variables are observed: At time $t = 1$, $p(\boldsymbol{z}^1, m^1, c^1|\boldsymbol{y}^1)$ is built up by a single Gaussian. However, at time-step $t = 2$, due to the summation over the class c^1 and mixture variable m^1 in Equation (A.2), $p(\boldsymbol{z}^2, m^2, c^2|\boldsymbol{y}^{1:2})$ will contain a mixture of $|sp\,(C)| \cdot |sp\,(M)|$ Gaussians; the model contains a mixture of $|sp\,(C)|^2 \cdot |sp\,(M)|^2$ Gaussians at $t = 3$, and so on. To control this explosion in computational complexity, we will resort to Gaussian collapse [3,2]. The Gaussian collapse guarantees that the distribution $p(\boldsymbol{z}^t, m^t, c^t, |\boldsymbol{y}^{1:t})$ is represented by a single Gaussian at any time-step t.

A.2 Backward recursion: smoothing

To compute $p(\boldsymbol{z}^t, m^t, c^t|\boldsymbol{y}^{1:T})$, we factorize it as

$$\begin{aligned} p(\boldsymbol{z}^t,&m^t, c^t|\boldsymbol{y}^{1:T}) = \sum_{m^{t+1}} \sum_{c^{t+1}} \int_{\boldsymbol{z}^{t+1}} p\left(\boldsymbol{z}^{t:t+1}, m^{t:t+1}, c^{t:t+1}|\boldsymbol{y}^{1:T}\right) \mathrm{d}\boldsymbol{z}^{t+1} \\ &= \sum_{m^{t+1}} \sum_{c^{t+1}} p(m^{t+1}, c^{t+1}|\boldsymbol{y}^{1:T}) \cdot p(m^t, c^t|m^{t+1}, c^{t+1}, \boldsymbol{y}^{1:T}) \cdot \\ &\int_{\boldsymbol{z}^{t+1}} p(\boldsymbol{z}^t|\boldsymbol{z}^{t+1}, m^{t:t+1}, c^{t:t+1}, \boldsymbol{y}^{1:T}) \cdot p(\boldsymbol{z}^{t+1}|m^{t:t+1}, c^{t:t+1}, \boldsymbol{y}^{1:T})\, \mathrm{d}\boldsymbol{z}^{t+1}. \end{aligned}$$

We note that $\boldsymbol{Z}^{t+1} \not\perp\!\!\!\perp \{M^t, C^t\}|\{\boldsymbol{Y}^{1:T} M^{t+1}, C^{t+1}\}$, but following [2] we assume that the influence of $\{M^t, C^t\}$ on $\boldsymbol{Z}^{t+1}$ is "weak" compared to the influence from $\boldsymbol{Y}^{1:T}$, M^{t+1} and C^{t+1}, and we will thus approximate $p(\boldsymbol{z}^{t+1}|m^{t:t+1}, c^{t:t+1}, \boldsymbol{y}^{1:T})$ by $p(\boldsymbol{z}^{t+1}|m^{t+1}, c^{t+1}, \boldsymbol{y}^{1:T})$. We are left with the approximation

$$\begin{aligned} p(\boldsymbol{z}^t,&m^t, c^t|\boldsymbol{y}^{1:T}) \approx \sum_{m^{t+1}} \sum_{c^{t+1}} p(m^t, c^t|m^{t+1}, c^{t+1}, \boldsymbol{y}^{1:T}) \cdot \\ &\int_{\boldsymbol{z}^{t+1}} p(\boldsymbol{z}^{t+1}, m^{t+1}, c^{t+1}|\boldsymbol{y}^{1:T}) \cdot p(\boldsymbol{z}^t|\boldsymbol{z}^{t+1}, m^{t:t+1}, c^{t:t+1}, \boldsymbol{y}^{1:T})\, \mathrm{d}\boldsymbol{z}^{t+1}. \end{aligned} \tag{A.3}$$

Noticing that $\boldsymbol{Z}^t \perp\!\!\!\perp \{\boldsymbol{Y}^{t+1:T}, M^{t+1}\}|\{\boldsymbol{Y}^{1:t}, M^t, C^{t:t+1}, \boldsymbol{Z}^{t+1}\}$,

$$\begin{aligned} p(\boldsymbol{z}^t|\boldsymbol{z}^{t+1}, m^{t:t+1}, c^{t:t+1}, \boldsymbol{y}^{1:T}) &= p(\boldsymbol{z}^t|\boldsymbol{z}^{t+1}, m^t, c^{t:t+1}, \boldsymbol{y}^{1:t}) \\ &\propto p(\boldsymbol{z}^t, \boldsymbol{z}^{t+1}|m^t, c^{t:t+1}, \boldsymbol{y}^{1:t}) \\ &= p(\boldsymbol{z}^{t+1}|\boldsymbol{z}^t, c^{t+1}) \cdot p(\boldsymbol{z}^t|m^t, c^t, \boldsymbol{y}^{1:t}), \end{aligned}$$

where $p(\boldsymbol{z}^{t+1}|\boldsymbol{z}^t, c^{t+1})$ is a parameter in the model, and $p(\boldsymbol{z}^t|m^t, c^t, \boldsymbol{y}^{1:t})$ is calculated during the forward phase.

Since $p(\boldsymbol{z}^{t+1}, m^{t+1}, c^{t+1}|\boldsymbol{y}^{1:T})$ in Equation (A.3) is known from the previous step in the backwards recursion, the last piece of the puzzle is to determine

how to calculate

$$p(m^t,c^t|m^{t+1},c^{t+1},\boldsymbol{y}^{1:T}) \propto \int_{\boldsymbol{z}^{t+1}} p(m^t,c^t|\boldsymbol{z}^{t+1},m^{t+1},c^{t+1},\boldsymbol{y}^{1:T}) \cdot p(\boldsymbol{z}^{t+1},m^{t+1},c^{t+1}|\boldsymbol{y}^{1:T})\,\mathrm{d}\boldsymbol{z}^{t+1}.$$

Again, $p(\boldsymbol{z}^{t+1},m^{t+1},c^{t+1}|\boldsymbol{y}^{1:T})$ is known from the previous step in the backwards recursion, and

$$\begin{aligned} p(m^t,c^t|\boldsymbol{z}^{t+1},m^{t+1},c^{t+1},\boldsymbol{y}^{1:T}) &= p(m^t,c^t|\boldsymbol{z}^{t+1},m^{t+1},c^{t+1},\boldsymbol{y}^{1:t}) \\ &\propto p(\boldsymbol{z}^{t+1}|c^{t:t+1},m^{t:t+1},\boldsymbol{y}^{1:t}) \cdot p(m^t,c^t|c^{t+1},m^{t+1},\boldsymbol{y}^{1:t}) \\ &= \int_{\boldsymbol{z}^t} p(\boldsymbol{z}^{t+1}|\boldsymbol{z}^t,c^{t:t+1},m^{t:t+1},\boldsymbol{y}^{1:t}) p(\boldsymbol{z}^t|c^{t:t+1},m^{t:t+1},\boldsymbol{y}^{1:t})\,\mathrm{d}\boldsymbol{z}^t \cdot \\ &\qquad p(m^t,c^t|c^{t+1},m^{t+1},\boldsymbol{y}^{1:t}) \\ &= \int_{\boldsymbol{z}^t} p(\boldsymbol{z}^{t+1}|\boldsymbol{z}^t,c^{t+1}) p(\boldsymbol{z}^t|c^t,m^t,\boldsymbol{y}^{1:t})\,\mathrm{d}\boldsymbol{z}^t \cdot \\ &\qquad p(m^t|c^t,\boldsymbol{y}^{1:t}) \cdot p(c^t|\boldsymbol{y}^{1:t},c^{t+1}). \end{aligned} \tag{A.4}$$

Since $p(c^t|\boldsymbol{y}^{1:t},c^{t+1}) \propto p(c^{t+1}|c^t) \cdot p(c^t|\boldsymbol{y}^{1:t})$, it follows that Equation (A.4) only contains terms that are readily available from the model definition or have already been calculated during the forward phase.

The calculations above show how we achieve the recursion for the backward pass using quantities that can be computed from previous results or from the forward recursion. There are two approximations in the backward recursion: Firstly, we approximated $p(\boldsymbol{z}^{t+1}|m^{t:t+1},c^{t:t+1},\boldsymbol{y}^{1:T})$ by $p(\boldsymbol{z}^{t+1}|m^{t+1},c^{t+1},\boldsymbol{y}^{1:T})$. Secondly, we note that also $p(\boldsymbol{z}^t,m^t,c^t|\boldsymbol{y}^{1:T})$ in Equation (A.3) is a mixture of Gaussians, and this time the number of components increase exponentially in $T-t$. We employ the same solution strategy as for the forward phase, and approximate $p(\boldsymbol{z}^t,m^t,c^t|\boldsymbol{y}^{1:T})$ by a single Gaussian at each time-point t.

B Learning

B.1 The M-step

The parameters of dLCM model are $\boldsymbol{A},\boldsymbol{L},\boldsymbol{\Sigma},\boldsymbol{\Theta},\boldsymbol{\Phi},\boldsymbol{J},\boldsymbol{K}$. At each iteration, these parameters can be obtained by taking the corresponding partial derivative of the expected log likelihood. The following are the results:

$\boldsymbol{L}$, the linear dynamics from the latent space to the attribute spaces: $\boldsymbol{L}_{i,m^t}$ denotes the i'th row of this matrix when the mixture node is m^t.

$$\frac{\partial Q}{\partial \boldsymbol{L}_{i,m^t}} = \Theta_{i,m^t}^{-1} \sum_{t=1}^{T} \Big\{ P(M^t = m^t | \boldsymbol{D}^{1:T}) \mathbb{E}[\boldsymbol{Z}^t (\boldsymbol{Z}^t)^{\mathrm{T}} | M^t = m^t, \boldsymbol{D}^{1:T}] \boldsymbol{L}_{i,m^t} - (y_i^t - \Phi_{i,m^t}) P(M^t = m^t | \boldsymbol{D}^{1:T}) \mathbb{E}[\boldsymbol{Z}^t | M^t = m^t, \boldsymbol{D}^{1:T}] \Big\}$$

$$\hat{\boldsymbol{L}}_{i,m^t} = \Big(\sum_{t=1}^{T} \Big\{ P(M^t = m^t | \boldsymbol{D}^{1:T}) \mathbb{E}[\boldsymbol{Z}^t (\boldsymbol{Z}^t)^{\mathrm{T}} | M^t = m^t, \boldsymbol{D}^{1:T}] \Big\} \Big)^{-1} \cdot \sum_{t=1}^{T} \Big\{ (y_i^t - \Phi_{i,m^t}) P(M^t = m^t | \boldsymbol{D}^{1:T}) \mathbb{E}[\boldsymbol{Z}^t | M^t = m^t, \boldsymbol{D}^{1:T}] \Big\}$$

$\boldsymbol{\Phi}$, the offset from the latent space to the attribute spaces: Φ_{i,m^t} denotes the i'th row of this matrix when the mixture node is m^t.

$$\frac{\partial Q}{\partial \Phi_{i,m^t}} = \sum_{t=1}^{T} \Big\{ 2P(M^t = m^t | \boldsymbol{D}^{1:T})(y_i^t - \Phi_{i,m^t}) - 2P(M^t = m^t | \boldsymbol{D}^{1:T}) \boldsymbol{L}_{i,m^t}^{\mathrm{T}} \cdot \mathbb{E}[\boldsymbol{Z}^t | M^t = m^t, \boldsymbol{D}^{1:T}] \Big\}$$

$$\hat{\Phi}_{i,m^t} = \frac{1}{\sum_{t=1}^{T} \Big\{ P(M^t = m^t | \boldsymbol{D}^{1:T}) \Big\}} \Big(\sum_{t=1}^{T} \Big\{ P(M^t = m^t | \boldsymbol{D}^{1:T}) y_i^t \Big\} - \sum_{t=1}^{T} \Big\{ P(M^t = m^t | \boldsymbol{D}^{1:T}) \boldsymbol{L}_{i,m^t}^{\mathrm{T}} \mathbb{E}[\boldsymbol{Z}^t | M^t = m^t, \boldsymbol{D}^{1:T}] \Big\} \Big)$$

$\boldsymbol{\Theta}$, the covariance matrices of the attribute spaces: Θ_{i,m^t} denotes the i'th element on the diagonal of the matrix when the mixture node is m^t.

$$\frac{\partial Q}{\partial \Theta_{i,m^t}} = -\frac{1}{2} \Theta_{i,m^t}^{-1} \sum_{t=1}^{T} \Big\{ P(M^t = m^t | \boldsymbol{D}^{1:T}) \Big\} + \frac{1}{2} \Theta_{i,m^t}^{-2} \sum_{t=1}^{T} \Big\{ (y_i^t - \Phi_{i,m^t})^2 \cdot P(M^t = m^t | \boldsymbol{D}^{1:T}) - 2(y_i^t - \Phi_{i,m^t}) P(M^t = m^t | \boldsymbol{D}^{1:T}) \boldsymbol{L}_{i,m^t}^{\mathrm{T}} \cdot \mathbb{E}[\boldsymbol{Z}^t | M^t = m^t, \boldsymbol{D}^{1:T}] + P(M^t = m^t | \boldsymbol{D}^{1:T}) \boldsymbol{L}_{i,m^t}^{\mathrm{T}} \cdot \mathbb{E}[\boldsymbol{Z}^t (\boldsymbol{Z}^t)^{\mathrm{T}} | M^t = m^t, \boldsymbol{D}^{1:T}] \boldsymbol{L}_{i,m^t} \Big\}$$

$$\hat{\Theta}_{i,m^t} = -\frac{1}{\sum_{t=1}^{T}\Big\{P(M^t = m^t|\boldsymbol{D}^{1:T})\Big\}}\sum_{t=1}^{T}\Big\{(y_i^t - \Phi_{i,m^t})^2 P(M^t = m^t|\boldsymbol{D}^{1:T}) - 2(y_i^t - \Phi_{i,m^t})P(M^t = m^t|\boldsymbol{D}^{1:T})\boldsymbol{L}_{i,m^t}^{\mathrm{T}}\mathbb{E}[\boldsymbol{Z}^t|M^t = m^t, \boldsymbol{D}^{1:T}] + P(M^t = m^t|\boldsymbol{D}^{1:T})\boldsymbol{L}_{i,m^t}^{\mathrm{T}}\mathbb{E}[\boldsymbol{Z}^t(\boldsymbol{Z}^t)^{\mathrm{T}}|M^t = m^t, \boldsymbol{D}^{1:T}]\boldsymbol{L}_{i,m^t})\Big\}$$

$\boldsymbol{\Sigma}$, the covariance matrices of the latent spaces: $\boldsymbol{\Sigma}_c$ denotes the diagonal matrix when the class variable is c.

$$\frac{\partial Q}{\partial \boldsymbol{\Sigma}_c} = -\frac{\alpha_c}{2}\boldsymbol{\Sigma}_c^{-1}\Big(I - \frac{1}{\alpha_c}\sum_{t:c_t=c}\Big\{\mathbb{E}[\boldsymbol{Z}^t(\boldsymbol{Z}^t)^{\mathrm{T}}|\boldsymbol{D}^{1:T}]\boldsymbol{\Sigma}_c^{-1}\Big\} + 2\frac{1}{\alpha_c}\sum_{t:c_t=c}\Big\{\boldsymbol{A}_c \cdot \mathbb{E}[\boldsymbol{Z}^{t-1}(\boldsymbol{Z}^t)^{\mathrm{T}}|\boldsymbol{D}^{1:T}]\boldsymbol{\Sigma}_c^{-1}\Big\} - \frac{1}{\alpha_c}\sum_{t:c^t=c}\Big\{\boldsymbol{A}_c\mathbb{E}[\boldsymbol{Z}^{t-1}(\boldsymbol{Z}^{t-1})^{\mathrm{T}}|\boldsymbol{D}^{1:T}]\boldsymbol{A}_c^{\mathrm{T}}\boldsymbol{\Sigma}_c^{-1}\Big\}\Big)$$

$$\hat{\boldsymbol{\Sigma}}_c = \frac{1}{\alpha_c}\sum_{t:c^t=c}\Big\{\mathbb{E}[\boldsymbol{Z}^t(\boldsymbol{Z}^t)^{\mathrm{T}}|\boldsymbol{D}^{1:T}] - 2\boldsymbol{A}_c\mathbb{E}[\boldsymbol{Z}^{t-1}(\boldsymbol{Z}^t)^{\mathrm{T}}|\boldsymbol{D}^{1:T}] + \boldsymbol{A}_{c^t} \cdot \mathbb{E}[\boldsymbol{Z}^{t-1}(\boldsymbol{Z}^{t-1})^{\mathrm{T}}|\boldsymbol{D}^{1:T}]\boldsymbol{A}_{c^t}^{\mathrm{T}}\Big\}$$

$\boldsymbol{A}$, the linear dynamics within the latent space from one time slice to next time slice: $\boldsymbol{A}_c$ denotes the matrix when the class variable is c.

$$\frac{\partial Q}{\partial \boldsymbol{A}_c} = -2\sum_{t:c^t=c}\Big\{\boldsymbol{\Sigma}_c^{-1}\mathbb{E}[\boldsymbol{Z}^t(Z^{t-1})^{\mathrm{T}}|\boldsymbol{D}^{1:T}]\Big\} + 2\sum_{t:c^t=c}\Big\{\boldsymbol{\Sigma}_c^{-1}\boldsymbol{A}_c\mathbb{E}[\boldsymbol{Z}^{t-1}(\boldsymbol{Z}^{t-1})^{\mathrm{T}}|\boldsymbol{D}^{1:T}]\Big\}$$

$$\hat{\boldsymbol{A}}_c = \sum_{t:c^t=c}\Big\{\mathbb{E}[\boldsymbol{Z}^t(\boldsymbol{Z}^{t-1})^{\mathrm{T}}|\boldsymbol{D}^{1:T}]\Big\} \cdot \Big(\sum_{t:c^t=c}\Big\{\mathbb{E}[\boldsymbol{Z}^{t-1}(\boldsymbol{Z}^{t-1})^{\mathrm{T}}|\boldsymbol{D}^{1:T}]\Big\}\Big)^{-1}$$

$\boldsymbol{J}$, the transition matrix of class variable is directly obtained by using frequency estimation on $P(c^t|c^{t-1})$

$\boldsymbol{K}$, the transition matrix of the mixture node is computed based on $P(m^t|c^t = c, \boldsymbol{D}^{1:T})$:

$$P(m^t|c^t = c, \boldsymbol{D}^{1:T}) = \frac{\sum_{t:c^t=c}\Big\{P(M^t = m^t|\boldsymbol{D}^{1:T})\Big\}}{\sum_{t=1}^{T}\Big\{P(M^t = m^t|\boldsymbol{D}^{1:T})\Big\}}$$

B.2 The E-step

To complete the maximization step, the following expected terms need to be calculated given $M^t = m^t$ (this also applies when M^t is taking on other values):

- $P(M^t = m^t|\boldsymbol{D}^{1:T})$,
- $\mathbb{E}[\boldsymbol{Z}^t|M^t = m^t, \boldsymbol{D}^{1:T}]$,
- $\mathbb{E}[\boldsymbol{Z}^t(\boldsymbol{Z}^t)^{\mathrm{T}}|M^t = m^t, \boldsymbol{D}^{1:T}]$,
- $\mathbb{E}[\boldsymbol{Z}^{t-1}(\boldsymbol{Z}^{t-1})^{\mathrm{T}}|\boldsymbol{D}^{1:T}]$, and
- $\mathbb{E}[\boldsymbol{Z}^t(\boldsymbol{Z}^{t-1})^{\mathrm{T}}|\boldsymbol{D}^{1:T}]$.

$P(M^t = m^t|\boldsymbol{D}^{1:T})$ and $\mathbb{E}[\boldsymbol{Z}^t|M^t = m^t, \boldsymbol{D}^{1:T}]$ can be directly obtained from a process similar to rauch-tung-striebel smoother [30]. By a further decomposition, we can also obtain that:

$$\begin{aligned}\mathbb{E}[\boldsymbol{Z}^t(\boldsymbol{Z}^t)^{\mathrm{T}}|M^t = m^t, \boldsymbol{D}^{1:T}] = \mathrm{Cov}[\boldsymbol{Z}^t|M^t = m^t, \boldsymbol{D}^{1:T}] \\ + \mathbb{E}[\boldsymbol{Z}^t|M^t = m^t, \boldsymbol{D}^{1:T}]\Big\{\mathbb{E}[\boldsymbol{Z}^t|M^t = m^t, \boldsymbol{D}^{1:T}]\Big\}^{\mathrm{T}}\end{aligned}$$

$$\begin{aligned}\mathbb{E}[\boldsymbol{Z}^t(\boldsymbol{Z}^{t-1})^{\mathrm{T}}|\boldsymbol{D}^{1:T}] = \mathrm{Cov}[\boldsymbol{Z}^t(\boldsymbol{Z}^{t-1})^{\mathrm{T}}|\boldsymbol{D}^{1:T}] \\ + \mathbb{E}[\boldsymbol{Z}^t|\boldsymbol{D}^{1:T}]\Big\{\mathbb{E}[\boldsymbol{Z}^{t-1}|\boldsymbol{D}^{1:T}]\Big\}^{\mathrm{T}}\end{aligned}$$

where $\mathrm{Cov}[\boldsymbol{Z}^t(\boldsymbol{Z}^{t-1})^{\mathrm{T}}|\boldsymbol{D}^{1:T}]$ can be calculated with the quantities obtained from rauch-tung-striebel smoother process:

$$\begin{aligned}\mathrm{Cov}[\boldsymbol{Z}^t(\boldsymbol{Z}^{t-1})^{\mathrm{T}}|\boldsymbol{D}^{1:T}] = \mathrm{Cov}[\boldsymbol{Z}^t(\boldsymbol{Z}^{t-1})^{\mathrm{T}}|\boldsymbol{D}^{1:t}] \\ + (\mathbb{E}[\boldsymbol{Z}^t|\boldsymbol{D}^{1:T}] - \mathbb{E}[\boldsymbol{Z}^t|\boldsymbol{D}^{1:t}])/\mathbb{E}[\boldsymbol{Z}^t|\boldsymbol{D}^{1:T}] \\ \cdot \mathrm{Cov}[\boldsymbol{Z}^t(\boldsymbol{Z}^{t-1})^{\mathrm{T}}|\boldsymbol{D}^{1:t}]\end{aligned}$$

After obtaining all the expected term required from the maximization step, the EM steps for the dLCM is then complete.

C Fitting a linear dynamical system to model any time series

In the following, we will show that a linear dynamical system can model any real-world time series, given sufficient dimension of latent continuous space.

Assume we have observed the series $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_T$. Our goal is to find a linear dynamical system with transition matrix $\boldsymbol{A}$ and emission matrix $\boldsymbol{B}$ that can fit this given series. We call the latent variables at time t $\boldsymbol{H}_t$. At each time step we require that $\boldsymbol{B}\boldsymbol{H}_t = \boldsymbol{v}_t$, and utilizing the definition of the model, we have that $\boldsymbol{H}_t = \boldsymbol{A}^{t-1}\boldsymbol{h}_1$, giving the requirement that $\boldsymbol{B}\boldsymbol{A}^{t-1}\boldsymbol{h}_1 = \boldsymbol{v}_t$ for $t = 1, \ldots, T$. These requirements can naively be fulfilled by letting $\boldsymbol{H}_t$ have a number of states equal to the observation sequence (i.e., $|sp(\boldsymbol{H}_t)| = T$), define $\boldsymbol{h}_1 = [1, 0, \ldots, 0]^{\mathrm{T}}$, and let $\boldsymbol{A} = \boldsymbol{L}_1$, the lower shift matrix. In this case, $\boldsymbol{h}_t = \boldsymbol{A}^{t-1}\boldsymbol{h}_1$ is a vector of only zeros, apart from a single "1" at location t. Defining $\boldsymbol{B}$ to hold the observations, $\boldsymbol{B} = [\boldsymbol{v}_1\ \boldsymbol{v}_2\ \ldots\ \boldsymbol{v}_T]$, the constraints above are trivially fulfilled.

We note that the representation above is extremely wasteful, and therefore not useful in practice, but it still proves the point that these models can indeed represent any observation sequence if so desired.

References

[1] Y. Bar-Shalom, X.-R. Li, Estimation and Tracking: Principles, Techniques and software, Artech House Publishers, 1993.

[2] D. Barber, Expectation correction for smoothed inference in switching linear dynamical systems, Journal of Machine Learning Research 7 (2006) 2515–2540.

[3] X. Boyen, D. Koller, Approximate learning of dynamic models, in: Advances in Neural Information Processing Systems 12, 1999, pp. 396–402.

[4] A. P. Dempster, N. M. Laird, D. B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, Journal of the Royal Statistical Society, Series B 39 (1) (1977) 1–38.

[5] R. O. Duda, P. E. Hart, Pattern Classification and Scene Analysis, John Wiley & Sons, New York, 1973.

[6] J.-B. Durand, O. Gaudoin, Software reliability modelling and prediction with hidden Markov chains, Statistical Modelling 5 (1) (2005) 75–93.

[7] B. S. Everitt, An introduction to latent variable models, Chapmann & Hall, London, 1984.

[8] N. Friedman, D. Geiger, M. Goldszmidt, Bayesian network classifiers, Machine Learning 29 (2–3) (1997) 131–163.

[9] N. Friedman, M. Goldszmidt, T. J. Lee, Bayesian network classification with continuous attributes: Getting the best of both discretization and parametric fitting, in: Proceedings of the Fifteenth International Conference on Machine Learning, Morgan Kaufmann Publishers, San Francisco, CA, 1998, pp. 179–187.

[10] J. Gama, A linear-Bayes classifier, in: Proceedings of the 7th Ibero-American Conference on AI: Advances in Artificial Intelligence, vol. 1952 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, Germany, 2000, pp. 269–279.

[11] S. Geman, D. Geman, Stochastic relaxation, Gibbs distribution and the Bayesian restoration of images, IEEE Transactions on Pattern Analysis and Machine Intelligence 6 (1984) 721–741.

[12] Z. Ghahramani, Learning dynamic Bayesian networks, in: Adaptive Processing of Sequences and Data Structures, International Summer School on Neural Networks, "E.R. Caianiello"-Tutorial Lectures, Springer-Verlag, London, UK, 1998, pp. 168–197.

[13] Z. Ghahramani, G. E. Hinton, Variational learning for switching state-space models, Neural Computation 12 (1998) 963–996.

[14] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I. H. Witten, The WEKA Data Mining Software: An Update, vol. 11, SIGKDD Explorations, 2009.

[15] Y. He, A. Kundu, 2-D shape classification using hidden Markov model, IEEE Transactions on Pattern Analysis and Machine Intelligence 13 (11) (1991) 1172–1184.

[16] N. Japkowicz, S. Stephen, The class imbalance problem: A systematic study, Intelligent Data Analysis 6 (5) (2002) 429–449.

[17] F. V. Jensen, T. D. Nielsen, Bayesian Networks and Decision Graphs, 2nd ed., Springer-Verlag, 2007.

[18] N. Khakzad, F. Khan, P. Amyotte, Safety analysis in process facilities: Comparison of fault tree and Bayesian network approaches, Reliability Engineering and System Safety 96 (8) (2011) 925–932.

[19] R. Kohavi, G. H. John, Wrappers for feature subset selection, Artificial Intelligence 97 (1–2) (1997) 273–324.

[20] T. Kohda, W. Cui, Risk-based reconfiguration of safety monitoring system using dynamic Bayesian network, Reliability Engineering and System Safety 92 (12) (2007) 1716–1723.

[21] H. Langseth, T. D. Nielsen, Latent classification models, Machine Learning 59 (3) (2005) 237–265.

[22] H. Langseth, L. Portinale, Bayesian networks in reliability, Reliability Engineering and System Safety 92 (1) (2007) 92–108.

[23] G. Lavee, E. Rivlin, M. Rudzsky, Understanding video events: a survey of methods for automatic interpretation of semantic occurrences in video, IEEE Transactions on Systems, Man, and Cybernetics Part C 39 (5) (2009) 489–504.

[24] U. Lerner, Hybrid Bayesian networks for reasoning about complex systems, Ph.D. thesis, Department of Computer Science, Stanford University (2002).

[25] U. Lerner, R. Parr, Inference in hybrid networks: Theoretical limits and practical algorithms, in: Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence (UAI), 2001, pp. 310–318.

[26] M. Martínez, L. E. Sucar, Learning dynamic naive Bayesian classifiers, in: Proceedings of the Twentyfirst International Florida Artificial Intelligence Research Symposium Conference, 2008, pp. 655–659.

[27] S. Montani, L. Portinale, A. Bobbio, D. C. Raiteri, Radyban: A tool for reliability analysis of dynamic fault trees through conversion into dynamic Bayesian networks, Reliability Engineering and System Safety 93 (7) (2008) 922–932.

[28] J. Pearl, Probabilistic reasoning in intelligent systems: networks of plausible inference, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[29] L. Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition, Proceedings of the IEEE 77 (2) (1989) 257–286.

[30] H. Rauch, F. Tung, C. T. Striebel, Maximum likelihood estimates of linear dynamic systems, Journal of the American Institute of Aeronautics and Astronautics 3 (8) (1965) 1445–1450.

[31] P. Smyth, Hidden Markov models for fault detection in dynamic system, Pattern Recognition 27 (1) (1994) 149–164.

[32] R. Suzuki, M. Ogawa, S. Otake, T. Izutsu, Y. Tobimatsu, S.-I. Izumi, T. Iwaya, Analysis of activities of daily living in elderly people living alone: single-subject feasibility study., Telemedicine Journal and E-Health 10 (2) (2004) 260–276.

[33] J. G. Torres-Toledano, L. E. Sucar, Bayesian networks for reliability analysis of complex systems, in: Proceedings of the 6th Ibero-American Conference on AI: Progress in Artificial Intelligence, IBERAMIA '98, Springer-Verlag, London, UK, UK, 1998, pp. 195–206.

[34] J.-E. Vinnem, Offshore Risk Assessment – Principles, Modelling and Applications of QRA Studies, Springer Series in Reliability Engineering, 2nd ed., Springer-Verlag, 2007.

[35] G. I. Webb, J. R. Boughton, Z. Wang, Not so naïve Bayes: Aggregating one-dependence estimators, Machine Learning 58 (1) (2005) 5–24.

[36] D. Zamalieva, A. Yilmaz, T. Aldemir, A probabilistic model for online scenario labeling in dynamic event tree generation, Reliability Engineering and System Safety (2013) 1–9.

[37] S. Zhong, On approximate inference of dynamic latent classification models for oil drilling monitoring, Presented at the 9th Bayesian Modeling Applications (in conjunction with UAI-2012). http://www.abnms.org/uai2012-apps-workshop/papers/Zhong.pdf (2012).

[38] S. Zhong, H. Langseth, Local-global-learning of naive Bayesian classifier, in: Proceedings of the 2009 Fourth International Conference on Innovative Computing, Information and Control, IEEE Computer Society, Washington, DC, USA, 2009, pp. 278–281.

[39] S. Zhong, A. M. Martínez, H. Langseth, T. D. Nielsen, Towards a more expressive model for dynamic classification, in: Proceedings of the Twenty-Third International Florida Artificial Intelligence Research Society Conference, 2010, pp. 563–564.