On Approximate Inference of Dynamic Latent Classification Models for Oil Drilling Monitoring

Shengtong Zhong
Department of Computer and Information Science
Norwegian University of Science and Technology
7491 Trondheim, Norway
shket@idi.ntnu.no

Abstract

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. Monitoring in such a complex domain is a challenging task. Not only is such a domain typically volatile and following non-linear dynamics, but sensor input to the monitoring system can also often be high dimensional, making it difficult to model and classify the domain's states. Dynamic latent classification models are dynamic Bayesian networks capable of effective and efficient modeling and classification. An approximate inference algorithm utilizing Gaussian collapse has been tailor-made for this family of models, but the approximation's properties have not been fully explored. In this paper we compare alternatives approximate inference methods for the dynamic latent classification model, in particular focusing on traditional sampling techniques. We show that the approximate scheme based on Gaussian collapse is computationally more efficient than sampling, while offering comparable accuracy results.

1 Introduction

In the oil drilling, monitor the complex process and identify the current system state is actually very difficult. Monitoring the complex process often involves keeping an eye on hundreds or thousands of sensors to determine whether or not the process is stable. We report results on an oil production facility in the North sea, where unstable situations should be identified as soon as possible [12]. The oil drilling data that we are considering, consisting of some sixty 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 (*activity recognition*) via avoiding a number of either dangerous or costly situations (*event detection*), to optimization of the drilling operation. The variables that are collected cover both topside measurements (like flow rates) and down-hole measurements (like, for instance, gamma rate). For the discussions to be concrete, we will tie the development to the task of *activity recognition* in this paper. The drilling of a well is a complex process, which consists of activities that are performed iteratively as the length of the well increases, and knowing which activity is performed at any time is important for the further *event detection*.

Motivated by this problem setting, a generative model called dynamic latent classification models (dLCM) [12] for dynamic classification in continuous domains is proposed to help the drilling engineers by automatically analyzing the data stream and classify the situation accordingly. Dynamic latent classification models are Bayesian networks which could model the complex system process and identify its system state.

A dynamic latent classification model can be seen as combining a naïve Bayes model with a mixture of factor analyzers at each time point. The latent variables of the factor analyzers are used to capture the state-specific dynamics of the process as well as modeling dependencies between attributes. As exact inference for the model is intractable, an approximate inference scheme based on Gaussian collapse is proposed in our previous study [12]. Although the previous experiments demonstrated that the proposed approximate inference is functioned well the learning of dynamic latent classification models as well as the classification work, we further investigate the approximation's properties by introducing alternative sampling techniques.

The remaining of the paper is organized as follows. In Section 2, we introduce the detail of dLCM. The importance of Gaussian collapse in the inference of dLCM is discussed in Section 3. Next, alternative sampling techniques are proposed for dLCM in Section 4. After the experiment results are illustrated and discussed in Section 5, the conclusion of the paper is presented in Section 6.

2 Dynamic Latent Classification Models

Dynamic Latent classification models [12] are dynamic Bayesian networks, which can model the complex system process and identify its system state. The complex system process is highly dynamical and complex, which makes it difficult to model and idetentify with the static models and standard dynamic model. The framework of dLCM is specified incrementally by examining its expressivity relative to the oil drilling data.

The dLCM is established from naïve Bayes model (NB), which is one of the simplest static models. In the first step, temporal dynamics of the class variables (a first order Markov chain) is added as considerable correlation between the class variable of consecutive time slices are evidenced from the oil drilling data [12]. This results in a dynamic version of naïve Bayes, which is also equivalent to a standard first order hidden Markov model (HMM) shown in Figure 1, where C^t denotes the class variable at time t and Y_i^t denotes the i-th attribute at time t. This model type has a long history of usage in monitoring, see e.g. [10].

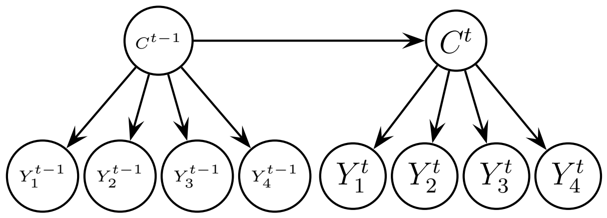

Figure 1: An example of dynamic version of naïve Bayes with 4 attributes using 2-time slice dynamic Bayesian networks representation (2TDBN). Attributes are assumed to be conditionally independent given the class variable.

This 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., $P(\boldsymbol{y}^t|c^t) = P(\boldsymbol{y}^{t-1}|c^{t-1})$ and $P(c^t|c^{t-1}) = P(c^{t+1}|c^t)$, for all t. For the continuous observation vector, the conditional distribution may be specified by a class-conditional multivariate Gaussian distribution with mean μ_{c^t} and covariance matrix $\boldsymbol{\Sigma}_{c^t}$, i.e., $\boldsymbol{Y}|\{C^t = c^t\} \sim N(\boldsymbol{\mu}_{c^t}, \boldsymbol{\Sigma}_{c^t})$

In a standard HMM, it assumes that the class variable and attributes at different time points are independent given the class variables at the current time, which is violated in many real world setting. In our oil drilling data, there is also a strong correlation between attributes given the class [12]. Modeling the dependence between attributes is then the next step in creating the dLM.

Following [7], we introduce latent variables to encode conditional dependence 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 Figure 2). It can be seen as combining the NB model with a factor analysis model at each time step.

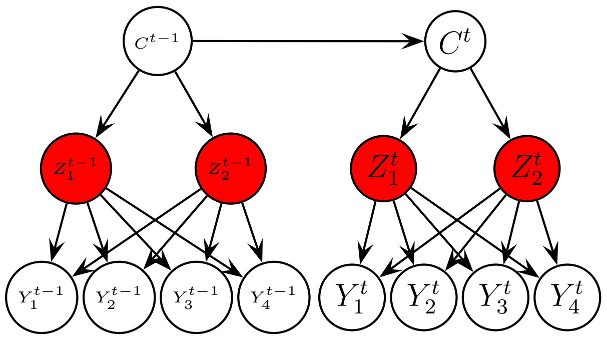

Figure 2: An example model by adding latent variables to dynamic version of naïve Bayes using 2TDBN, where the dimension of latent space is 2 and the dimension of attribute space is 4. In each time step, the conditional dependencies between the attributes are encoded by the latent variables $(Z_1^t, \ldots, Z_k^t)$.

The latent variable $\boldsymbol{Z}^t$ is assigned a multivariate Gaussian distribution conditional on the class variable and the attributes $\boldsymbol{Y}^t$ are assumed to be linear multivariate Gaussian distributions conditional on the latent variables:

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

where $\boldsymbol{\Sigma}_{c^t}$ and $\boldsymbol{\Theta}$ are diagonal covariance matrix and $\boldsymbol{L}$ is the transition matrix, $\boldsymbol{\Phi}$ is the offset from the latent space to attribute space; note that the stationarity assumption encoded in the model.

In this model, the latent variables capture the dependence 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.

At this step, the temporal dynamics of the model is assumed to be only captured at the class level. When the state specification of the class variable is coarse, then this assumption will rarely hold. This assumption does not hold in our oil drilling data, as the conditional correlation of the attribute in successive time slices is evident [12]. we address this by modeling the dynamics of the system at the level of the latent variables. The state specific dynamics is encoded by assuming that the latent variable at the current time slice follows a linear Gaussian distribution conditioned on previous time slice. Specifically, we encode the state specific dynamics by assuming that the multivariate latent variable $\boldsymbol{Z}^t$ follows a linear Gaussian distribution conditioned on $\boldsymbol{Z}^{t-1}$, and the transition dynamics between latent variable is denoted by a diagonal matrix $\boldsymbol{A}_{c^t}$:

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

A graphical representation of the model is given in Figure 3.

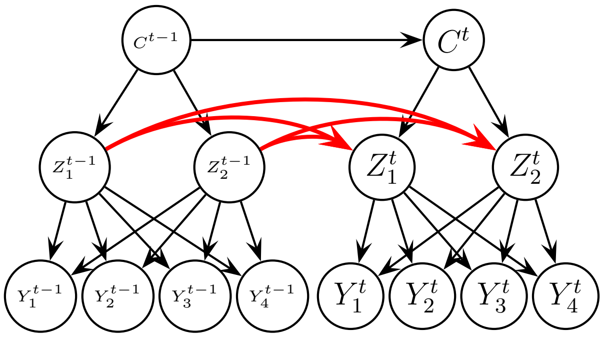

Figure 3: An example model by incrementally adding dynamics on latent variables using 2TDBN, where the dimension of latent space is 2 and the dimension of attribute space is 4. The state specific dynamics are encoded at the level of the latent variables.

A discrete mixture variable M is further introduced to the model at each time slice for the purpose of reducing the computational cost while maintaining the representational power [12]. Similar situation is done by [7] for static domains, and in the dynamic domains can be seen from [3, 6] where a probabilistic model called switching state-space model is proposed that combining discrete and continuous dynamics. 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,

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

where $1 \le m^t \le |sp\,(M)|$ ($|sp\,(M)|$ denotes the dimension of variable M space), $P(M^t=m^t|C^t=c^t) \ge 0$ and $\sum_{m^t=1}^{|sp(M)|} P(M^t=m^t|C^t=c^t)=1$ for all $1 \le c^t \le |sp\,(C)|$, $\boldsymbol{\Phi}_{m^t}$ is the offset from the latent space to attribute space.

The final model is then called dynamic latent classification model which is shown in Figure 4. The dynamic latent classification model is shown to be effective and efficient through the experiment with our oil drilling data, and the significant improvement is also demonstrated when comparing dLCM with static models (such as NB or decision tree) and HMM [12].

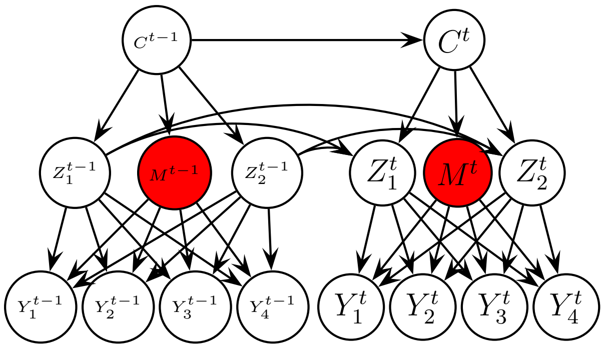

Figure 4: An example of dynamic latent classification model using 2TDBN, where the dimension of latent space is 2 and the dimension of attribute space is 4.

3 Approximate inference in dLCM

The exact inference for dLCM is intractable. To make dLCM applicable and effective in practice, approximate inference is then proposed.

3.1 Intractability of exact inference in dLCM

Seen from the dLCM in Figure 4, an equivalent probabilistic model is

$$p(\boldsymbol{y}^{1:T}, \boldsymbol{z}^{1:T}, m^{1:T}, c^{1:T}) = p(\boldsymbol{y}^1|\boldsymbol{z}^1, m^1)p(\boldsymbol{z}^1|c^1)p(m^1|c^1)p(c^1) \cdot \prod_{t=2}^{T} p(\boldsymbol{y}^t|\boldsymbol{z}^t, m^t)p(\boldsymbol{z}^t|\boldsymbol{z}^{t-1}, c^t)p(m^t|c^t)p(c^t|c^{t-1}).$$

In dLCM, exact filtered and smoothed inference is shown to be intractable (scaling exponentially with T [8]) as neither the class variables nor the mixture variables are observed: At time step 1, $p(\boldsymbol{z}^1|\boldsymbol{y}^1)$ is a mixture of $|sp\,(C)| \cdot |sp\,(M)|$ Gaussian. At time-step 2, due to the summation over the classes and mixture variables, $p(\boldsymbol{z}^2|\boldsymbol{y}^{1:2})$ will be a mixture of $|sp\,(C)|^2 \cdot |sp\,(M)|^2$ Gaussian; at time-step 3 it will be a mixture of $|sp\,(C)|^3 \cdot |sp\,(M)|^3$ Gaussian and so on until the generation of a mixture of $|sp\,(C)|^T \cdot |sp\,(M)|^T$ Gaussian at time-point T. To control this explosion in computational complexity, approximate inference techniques are adopted to the inference of dLCM.

3.2 Approximate inference: Forward pass

The structure of the proposed dLCM is similar to the linear dynamical system (LDS) [2], the standard Rauch-Tung-Striebel (RTS) smoother [9] and the expectation correction smoother [3] for LDS provide the basis for the approximate inference of dLCM. As for the RTS, the filtered estimate of dLCM $p(\boldsymbol{z}^t, m^t, c^t|\boldsymbol{y}^{1:t})$ is obtained by a forward recursion, and then following a backward recursion to calculate the smoothed estimate $p(\boldsymbol{z}^t, m^t, c^t|\boldsymbol{y}^{1:T})$. The inference of dLCM is then achieved by a single forward recursion and a single backward recursion iteratively. Gaussian collapse is incorporated into both the forward recursion and the backward recursion to form the approximate inference. The Gaussian collapse in the forward recursion is equivalent to assumed density filtering [4], and the Gaussian collapse in the backward recursion mirrors the smoothed posterior collapse from [3].

During the forward recursion of dLCM, the filtered posterior $p(\boldsymbol{z}^t, m^t, c^t|\boldsymbol{y}^{1:t})$ is computed with a recursive form. By a simple decomposition,

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

Dropping the normalization constant $p(\boldsymbol{y}^t|\boldsymbol{y}^{1:t-1})$, $p(\boldsymbol{z}^t, m^t, c^t|\boldsymbol{y}^{1:t})$ is proportional to the new joint probability $p(\boldsymbol{z}^t, m^t, c^t, \boldsymbol{y}^t|\boldsymbol{y}^{1:t-1})$, where

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

To build the forward recursion, a recursive form for each of the factors in Equation 1 is required. Given the filtered results of the previous time-step, the recursive form for each of the factors are shown to be feasible [12]. On the way to devise the recursive form, one term $p(\boldsymbol{z}^{t-1}|m^{t-1}, c^{t-1}, \boldsymbol{y}^{1:t-1})$ is required, which can be directly obtained since it is the filtered probability from the previous time step. However, the number of mixture components of $p(\boldsymbol{z}^{t-1}|\boldsymbol{y}^{t-1})$ is increasing exponentially over time as we discussed earlier, so is the case for $p(\boldsymbol{z}^{t-1}|m^{t-1}, c^{t-1}, \boldsymbol{y}^{1:t-1})$. In our Gaussian collapse implementation [12], the term $p(\boldsymbol{z}^{t-1}|m^{t-1}, c^{t-1}, \boldsymbol{y}^{1:t-1})$ is collapsed into a single Gaussian, parameterized with mean $\boldsymbol{\nu}_{m^{t-1},c^{t-1}}$ and covariance $\boldsymbol{\Gamma}_{m^{t-1},c^{t-1}}$, and then propagate this collapsed Gaussian for next time slice. With this approximation, the recursive computation of the forward pass becomes tractable.

3.3 Approximate inference: Backward pass

Similar to the forward pass, the backward pass also relies on a recursion computation of the smoothed posterior $p(\boldsymbol{z}^t, m^t, c^t|\boldsymbol{y}^{1:T})$. In detail, $p(\boldsymbol{z}^t, m^t, c^t|\boldsymbol{y}^{1:T})$ is computed from its smoothed result of the previous step $p(\boldsymbol{z}^{t+1}, m^{t+1}, c^{t+1}|\boldsymbol{y}^{1:T})$, together with some other quantities obtained from forward pass. The first smoothed posterior is $p(\boldsymbol{z}^T, m^T, c^T|\boldsymbol{y}^{1:T})$, which can be directly obtained as it is also the last filtered posterior from the forward pass. To compute $p(\boldsymbol{z}^t, m^t, c^t|\boldsymbol{y}^{1:T})$, factorize it as

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

Due to the fact that $\boldsymbol{z}^t \perp\!\!\!\perp \{\boldsymbol{y}^{t+1:T}, m^{t+1}, c^{t+1}\}|\{\boldsymbol{z}^{t+1}, m^t, c^t\}$, the term $p(\boldsymbol{z}^t|m^t, c^t, m^{t+1}, c^{t+1}, \boldsymbol{y}^{1:T})$ can be found from

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

To complete the backward recursive form, two essential assumptions are further made in the backward pass that makes the approximate inference applicable and effective. The first assumption is to approximate $p(\boldsymbol{z}^{t+1}|m^t, c^t, m^{t+1}, c^{t+1}, \boldsymbol{y}^{1:T})$ by $p(\boldsymbol{z}^{t+1}|m^{t+1}, c^{t+1}, \boldsymbol{y}^{1:T})$ [3]. This is due to that although $\{m^t, c^t\} \not\perp\!\!\!\perp \boldsymbol{z}^{t+1}|\boldsymbol{y}^{1:T}$, the influence of $\{m^t, c^t\}$ on $\boldsymbol{z}^{t+1}$ through $\boldsymbol{z}^t$ is 'weak' as $\boldsymbol{z}^t$ will be mostly influenced by $\boldsymbol{y}^{1:t}$. The benefit of this simple assumption lies in that $p(\boldsymbol{z}^{t+1}|m^{t+1}, c^{t+1}, \boldsymbol{y}^{1:T})$ can be directly obtained from the previous backward recursion. Meanwhile $p(\boldsymbol{z}^{t+1}|m^{t+1}, c^{t+1}, \boldsymbol{y}^{1:T})$ is a Gaussian mixture whose components increase exponentially in $T - t$.

The second assumption is also a Gaussian collapse process. $p(\boldsymbol{z}^{t+1}|m^{t+1}, c^{t+1}, \boldsymbol{y}^{1:T})$ is collapsed into a single Gaussian and then pass this collapsed Gaussian for the next step. This will guarantee that the back propagated term $p(\boldsymbol{z}^t|m^t, c^t, \boldsymbol{y}^{1:T})$ will be Gaussian mixture with fixed $|sp\,(C)| \cdot |sp\,(M)|$ components at next time step. With this Gaussian collapse process at each time slice, a tractable recursion in backward pass is established.

3.4 The importance of approximate inference

The exact inference is not applicable in practise as its computation cost is increasing exponentially over time. The approximate inference is then essential to dLCM. Gaussian collapse is adopted during building the recursive form for both forward and backward pass. At the same time, $p(\boldsymbol{z}^{t+1}|m^t, c^t, m^{t+1}, c^{t+1}, \boldsymbol{y}^{1:T})$ is also approximated by $p(\boldsymbol{z}^{t+1}|m^{t+1}, c^{t+1})$ in dLCM. As the approximations are made within the inference, the quality of the overall learning and inference for dLCM is rather sensitive to these approximations. Our experimental results [12] showed that the overall performance of dLCM is satisfactory, which indicate that the chosen approximations are reasonable.

Even though the proposed approximate inference in dLCM is satisfactory, is there any improvement space with alternative approximation methods? With this question in mind, we decide to investigate the approximation's properties by incorporating new approximation method. The traditional sampling techniques (e.g., [5]) are commonly used in a similar approximation situation. Next section we will briefly introduce sampling technique, and then we will explain how it is integrated in dLCM. Meanwhile the approximation of $p(\boldsymbol{z}^{t+1}|m^t, c^t, m^{t+1}, c^{t+1}, \boldsymbol{y}^{1:T})$ by $p(\boldsymbol{z}^{t+1}|m^{t+1}, c^{t+1})$ is kept unchanged.

In general, we will replace Gaussian collapse by sampling in the approximate inference of dLCM, and further investigate the effectiveness and efficiency of this proposal through a comparison experiment between original Gaussian collapse based dLCM and sampling techniques based dLCM.

4 Sampling

4.1 Background

The sampling is to select a subset of samples from within a population to estimate the characteristics of the original population. There is a commonly known tradeoff in sampling. When less samples are selected from within a population, which means the sampling process takes shorter time, the estimation of the characteristics to the original population is relatively worse. On the other hand, if more samples are selected from within the same population, which of course is much more time consuming, the characteristics of the original population is better estimated. The efficiency (time consuming) and effectiveness (characteristics estimation) are the essential concerns in the sampling techniques. In general, more samples should be selected within the tolerable time, and the better estimation of characteristics of the population can be expected.

This feature of traditional sampling techniques makes it attractive to the approximate inference of dLCM, a balance between efficiency and effectiveness is expected to be achieved according to application requirement. Meanwhile sampling can approximate any distribution as long as the sample number is sufficient. Sampling is expected to replace the Gaussian collapse for the approximation in the both forward and backward pass. We introduce particle filtering next, which will further motivate our discussion on the utilizations detail of the sampling in the inference of dLCM.

4.2 Particle Filtering

Particle filtering (PF) [1] is a technique for implementing a recursive Bayesian filter by Monte Carlo simulation, which is an efficient statistical method to estimate the system state. The Monte Carlo simulation relies on repeated random sampling techniques.

In particle filtering, let a weighted particle set $\{(\boldsymbol{s}_n^t, \boldsymbol{\pi}_n^t)\}_{n=1}^N$ at each time t denotes an approximation of required posterior probability of the system state. Each of N particles has the state s_n^t and its weight $\boldsymbol{\pi}_n^t$, the weights are normalized such that $\sum_n \boldsymbol{\pi}_n^t = 1$. The particle filtering has three operation stages: sampling (selection), prediction and observation. In the sampling stage, N particles are chosen from the prior probability according to the set $\{(\boldsymbol{s}_n^{(t-1)}, \boldsymbol{\pi}_n^{t-1})\}_{n=1}^N$. Then predict the state of the chosen particles by the dynamic model $p(\boldsymbol{s}^t|\boldsymbol{s}^{t-1})$. In observation stage, the predicted particles are weighted according to observation model $p(\boldsymbol{y}^t|\boldsymbol{s}^t)$. After obtaining the weights of particles, the state at time t can be estimated based on the weighted particle set.

4.3 Sampling in the dCLM

The sampling process that we required for the inference of dCLM is similar to the PF. In the forward pass, we know that the mixture components of $p(\boldsymbol{z}^{t-1}|\boldsymbol{y}^{t-1})$ is increasing exponentially over time in the exact inference. Instead of a recursive approximation on $p(\boldsymbol{z}^{t-1}|m^{t-1}, c^{t-1}, \boldsymbol{y}^{1:t-1})$ in the Gaussian collapse scheme, an recursive approximation on $p(\boldsymbol{z}^{t-1}|\boldsymbol{y}^{1:t-1})$ by sampling is adopted. With the obtained approximated distribution $p(\boldsymbol{z}^{t-1}|\boldsymbol{y}^{1:t-1})$ at time slice $t-1$, N weighted samples $\{(\boldsymbol{s}_n^{t-1}, \boldsymbol{\pi}_n^{t-1})\}_{n=1}^N$ are selected from this approximated distribution. These selected samples are propagated to the next time slice t with a linear transition dynamics $\boldsymbol{A}_{c^t}$. As the discrete class variable C^t has the size of $|sp\,(C)|$, then each of the selected samples will become $|sp\,(C)|$ new samples. These $|sp\,(C)| \cdot N$ propagated samples are further updated by the observation $\boldsymbol{y}^t$. The updating rule is the same as the Kalmar filter updating [11]. Due to the mixture component has size of $|sp\,(M)|$, each of these propagated samples will become $|sp\,(M)|$ new samples again. In general, the N selected samples from time slice $t-1$ will become $|sp\,(C)| \cdot |sp\,(M)| \cdot N$ samples at time slice t and its weight are updated accordingly. The weighted sample set $\{(\boldsymbol{f}_n^t, \gamma_n^t)\}_{n=1}^{|sp(C)|\cdot|sp(M)|\cdot N}$ is then the approximation to $p(\boldsymbol{z}^t|\boldsymbol{y}^{1:t})$. For next time step recursion, a new weighted sample set $\{(\boldsymbol{s}_n^t, \boldsymbol{\pi}_n^t)\}_{n=1}^N$ containing N samples will be selected from the approximated $p(\boldsymbol{z}^t|\boldsymbol{y}^{1:t})$. The recursive process is summarized in Algorithm 1.

Algorithm 1 Sampling in the forward pass

```
1: for  $t = 2 : T$ do
2:     Select $N$ samples from the previous appxoximated distribution $p(\boldsymbol{z}^{t-1}|\boldsymbol{y}^{1:t-1})$ to form a weighted sample set $\{(\boldsymbol{s}_n^{t-1}, \boldsymbol{\pi}_n^{t-1})\}_{n=1}^N$
3:     Propagate these selected samples to the next time slice $t$ by a transition dynamics $\boldsymbol{A}_{c^t}$
4:     Update the propagated samples by the observation $Y^t$.
5:     Then the updated samples form a new weighted sample set $\{(\boldsymbol{f}_n^t, \gamma_n^t)\}_{n=1}^{|sp(C)|\cdot|sp(M)|\cdot N}$, which is an approximation of $p(\boldsymbol{z}^t|\boldsymbol{y}^{1:t})$
6: end for
```

In the backward pass, with a similar sampling process as the forward pass, samples are firstly selected from the approximated distribution $p(\boldsymbol{z}^{t+1}|\boldsymbol{y}^{1:T})$. $p(\boldsymbol{z}^{t+1}|\boldsymbol{y}^{1:T})$ is approximated by a weighted sample set denoted as $\{(\boldsymbol{b}_n^{t+1}, \boldsymbol{\rho}_n^{t+1})\}_{n=1}$. Next, the selected samples are then back-propagated to the previous time slice t (which is the next step of the backward pass) with the reverse transition dynamics of $\boldsymbol{A}_c^t$. The back-propagated samples is later updated by the observation Y^{t-1} [11]. Similar to the forward pass, the approximation to $p(\boldsymbol{z}^t|\boldsymbol{y}^{1:T})$. $p(\boldsymbol{z}^t|\boldsymbol{y}^{1:T})$ is a weighted sample set $\{(\boldsymbol{g}_n^t, \boldsymbol{\tau}_n^t)\}_{n=1}^{|sp(C)|\cdot|sp(M)|\cdot N}$. For the recursive calculation of next time slice, a new weighted sample set $\{(\boldsymbol{b}_n^t, \boldsymbol{\rho}_n^t)\}_{n=1}^N$ containing N samples will be selected from this approximated distribution. The required term $p(\boldsymbol{z}^T|c^T, \boldsymbol{y}^{1:T})$ at the beginning of the backward pass is also the last time step result from the forward pass, which indicates that $\{(\boldsymbol{g}_n^T, \boldsymbol{\tau}_n^T)\}_{n=1}^{|sp(C)|\cdot|sp(M)|\cdot N}$ is the same sample set as $\{(\boldsymbol{f}_n^T, \gamma_n^T)\}_{n=1}^{|sp(C)|\cdot|sp(M)|\cdot N}$. Finally the approximate inference for dLCM is completely established with sampling technique based scheme.

The number of samples selected from the approximated distribution at each step is fixed which is dependent on the time consumption requirement and estimation quality requirement of corresponding application. There is a balance need to be addressed according to the practical application requirement. Generally, the more samples we select, the more time it costs while the estimation quality is better.

In the discussion of this section, the approximate inference of dLCM based on sampling is established by mimicking the particle filtering process both in the forward and backward pass. To investigate the effectiveness and efficiency of sampling based dLCM, a comparison experiments test will be conducted in the next section.

5 Experiment Results

In this section, the comparison experiments on simulation data and oil drilling data are conducted and their results are discussed.

5.1 Experiments on simulation data

A set of simulation data is firstly generated from dLCM, and we investigate the classification accuracy and time-consumption between Gaussian collapse scheme and sampling scheme.

5.1.1 Experiments settings on simulation data generation

The simulation data-set are generated from dLCM with parameters that is chosen by a "semi random" process. The model parameters of dLCM have two parts: model structure and model parameters with fixed model structure. For each time slice, the model structure is decided by four factors: the size of class variable C (activity state), the dimension of latent variable space $\boldsymbol{Z}$, the dimension of attribute space $\boldsymbol{Y}$ and the size of mixture components M. These values are fixed as described in Table 1. After choosing the model structure, its associated model parameters were randomly generated. The above process of choosing model structure and model parameters together is the "semi random" process. We then generate a data set with this model.

For convenience we call the model structure and its parameters as model parameters in the remaining of the paper. The generated data set and the model parameters are used as true model for the classification test purpose next. A comparison experiment between

data	$\|sp\,(C)\|$	$\|sp\,(M)\|$	$\|sp\,(Z)\|$	$\|sp\,(Y)\|$	T
*set*1	2	1	12	6	500
*set*2	2	2	18	9	1000

Table 1: The simulation data set with its model structure information.

Gaussian collapse and sampling is then conducted.

5.1.2 Results and discussion

The comparison experiment is conducted with both Gaussian collapse based and sampling based dLCM. Among sampling based scheme, there are three chosen sample sizes $40, 200, 1000$ respectively. The classification results on simulation set1 and set2 are summarized in Table 2. The classification accuracy results scheme are recorded with the average results of ten runs of each scheme, and it shows that Gaussian collapse based dLCM performs better than three sampling based dLM in both set1 and set2. Among sampling based scheme, scheme with larger sample size achieves better classification accuracy in a general sense.

scheme (samples)	set1/accuracy	set2/accuracy
Gaussian collapse	**99.60%**	**99.90%**
Sampling (40)	96.75%	95.55%
Sampling (200)	97.60%	97.95%
Sampling (1000)	97.75%	98.40%

Table 2: The average classification accuracy on simulation data set with Gaussian collapse based and sampling based (varying the number of samples) dLCM.

After investigating the effectiveness of each scheme, we continue to discuss the efficiency. The efficiency of each scheme is evaluated by the average time-cost of ten run that is required to accomplish the classification task, and the time-cost detail is shown in Table 3. In set1, the classification task is accomplished 0.47 second with Gaussian collapse, whereas the sampling scheme with 40 samples cost 2.01 second. The larger sample size in sampling scheme, the more time it costs to accomplish the classification task. Meanwhile it is clear that Gaussian collapse requires much less time to accomplish the classification task.

Compared to sampling based scheme, Gaussian collapse based scheme achieves comparable (slightly better) classification results with much less time on simulation data test.

5.2 Experiments on oil drilling data

Next the same comparison experiment is conducted with the oil drilling data from North sea.

scheme (samples)	set1/time-cost	set2/time-cost
Gaussian Collapse	**0.47(s)**	**1.91(s)**
Sampling (40)	2.01 (s)	5.97(s)
Sampling (200)	7.31(s)	17.56 (s)
Sampling (1000)	36.24 (s)	80.70 (s)

Table 3: The average time-cost (second) on simulation data set with both Gaussian collapse based and sampling based (varying the number of samples) dLCM.

5.2.1 Experiment settings on oil drilling data

As we mentioned in the introduction section, we will tie the development to the task of *activity recognition* in this paper. In total, there are 5 drilling acclivities in the dataset used for classification task. These activities are "drilling", "connection", "tripping in", "tripping out" and "other". The original oil drilling data contains more than 60 variables. Advised by oil drilling domain expert, 9 variables for the classification task here. There are two chosen data set, which contains 80000 and 50000 time slices with all 5 activities presented respectively.

For classification purpose in this paper, we combine these 5 activities into 2 activities and conduct the classification test on the combined data set. Three activities including "drilling", "connection" and "other" activities are combined as one activity, and we do the similar combination for "tripping in" and "tripping out" activities. The reason behind is that these combined actives are physically close and may have quite similar dynamics. This combination also simplify our experiments with the oil drilling data, while maintaining the comparison experiment purpose.

Before we can compare the inference of each scheme on the oil drilling data set, we learn a dLCM with the learning method proposed in [12] with the oil drilling data set containing 80000 time slices. The model structure is chosen by experience, with 2 mixture component and 16 latent variables. After learning its parameters with the chosen model structure, the dLCM for further classification experiment is then finalized. With the learnt dLM, the classification experiment will be conducted on another oil drilling data set containing 50000 time slices.

5.2.2 Results and discussion

With the fixed dLCM, the average (by ten runs) classification accuracy and average time-cost for each scheme are obtained. There are 4 scheme are presented, Gaussian collapsed based scheme and sampling techniques based scheme with $40, 200, 1000$ samples respectively. The experiments results are summarized in Table 4.

scheme (samples)	accuracy	time-cost
Gaussian Collapse	**82.9%**	**110.15(*s*)**
Sampling (40)	69.87%)	335.93(s)
Sampling (200)	76.56%	1130.85 (s)
Sampling (1000)	79.07%	4469.33 (s)

Table 4: The average classification accuracy and time-cost (second) on oil drilling data set with both Gaussian collapse based and sampling based (varying the number of samples) dLCM.

Among the sampling techniques based scheme, more samples achieves higher classification accuracy. However, with more samples in sampling techniques based scheme, the computation cot for the classification task is much more expensive. It is clearly shown in the table that sampling with 1000 samples requires more than one hour to accomplish the classification task which is around 40 times than that of Gaussian collapse, and they achieve a similar classification accuracy. In general Gaussian collapse still achieves comparable results (slightly better than Sampling), while keeping the computation cost in a rather low standard compared to sampling based scheme.

6 Conclusion

In the approximate inference of the dLCM, the Gaussian collapse is originally adopted as the core of the approximation method. In this paper, alternatively sampling technique is proposed to do the approximation. A process similar to particle filtering, utilizing sampling as the basis, is then incorporated into the approximate inference of the dLCM. We then conduct the comparison experiment results on both simulated data and real oil drilling data. The experimental results from both sets show that the approximate scheme based on Gaussian collapse is computationally more efficient than sampling, while offering comparable accuracy results.

Acknowledgements

I would like to thank Helge Langseth who helped a lot during the whole process of organizing and writing this paper.

References

[1] Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50:174–188, 2001.

[2] Yaakov Bar-Shalom and Xiao-Rong Li. *Estimation and Tracking: Principles, Techniques and software*. Artech House Publishers, 1993.

[3] David Barber. Expectation correction for smoothed inference in switching linear dynamical systems. *Journal of Machine Learning Research*, 7:2515–2540, 2006.

[4] Xavier Boyen and Daphne Koller. Approximate learning of dynamic models. In *Advances in Neural Information Processing Systems 12*, pages 396–402, 1999.

[5] Stuart Geman and Donald Geman. Stochastic relaxation, Gibbs distribution and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[6] Zoubin Ghahramani and Geoffrey E. Hinton. Variational learning for switching state-space models. *Neural Computation*, 12:963–996, 1998.

[7] Helge Langseth and Thomas D. Nielsen. Latent classification models. *Machine Learning*, 59(3):237–265, 2005.

[8] Uri Lerner. Hybrid Bayesian networks for reasoning about complex systems. *PhD thesis, Dept. of Comp. Sci. Stanford University, Stanford*, 2002.

[9] H.E. Rauch, F. Tung, and C. T. Striebel. Maximum likelihood estimates of linear dynamic systems. *AIAA Journal*, 3:1445–1450, 1965.

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

[11] Greg Welch and Gary Bishop. An introduction to the kalman filter. Technical report, University of North Carolina at Chapel Hill, 1995.

[12] Shengtong Zhong, Helge Langseth, and Thomas D. Nielsen. Bayesian networks for dynamic classification. Working Paper, `http://idi.ntnu.no/~shket/dLCM.pdf`, 2012.