Project no.: 619209

Project full title: Analysis of Massive Data STreams

Project Acronym: AMIDST

Deliverable no.: D2.1

Title of the deliverable: The AMIDST modelling framework – Initial draft report

Contractual Date of Delivery to the CEC:	**30.09.2014**
Actual Date of Delivery to the CEC:	**30.09.2014**
Organisation name of lead contractor for this deliverable:	**AAU**
Author(s): Hanen Borchani, Antonio Fernández, Odd Erik Gundersen, Sigve Hovda, Helge Langseth, Anders L. Madsen, Ana M. Martínez, Ramón Martínez, Andrés Masegosa, Thomas D. Nielsen, Antonio Salmerón, Frode Sørmo, Galia Weidl	
Participants(s): P01, P02, P03, P04, P05, P06, P07	
Work package contributing to the deliverable:	**WP2**
Nature:	**R**
Version:	**1.0**
Total number of pages:	**62**
Start date of project:	1st January 2014 **Duration:** 36 month

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)		
Dissemination Level		
PU	Public	**X**
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Abstract:

In this document, we firstly discuss and justify the preliminary probabilistic graphical models for the different application scenarios of the three use-case providers previously identified in Deliverable 1.2. Then, we introduce the AMIDST modelling framework as a general model class that can be applied to solve each of these application scenarios and, potentially, future similar problems in other domains.

Keyword list: AMIDST Modelling framework, application scenarios, preliminary models, data analysis, Bayesian networks, dynamic Bayesian networks, conditional linear Gaussian.

Contents

Document history

Version	Date	Author (Unit)	Description
v0.3	10/09/2014	All consortium members	The AMIDST modelling framework discussed and established
v0.6	22/09/2014	Hanen Borchani, Antonio Fernández, Odd E. Gundersen, Sigve Hovda, Helge Langseth, Anders L. Madsen, Ana M. Martínez, Ramón Martínez, Andrés Masegosa, Thomas D. Nielsen, Antonio Salmerón, Frode Sørmo, Galia Weidl	Initial draft finished
v1.0	30/09/2014	Hanen Borchani, Antonio Fernández, Odd E. Gundersen, Sigve Hovda, Helge Langseth, Anders L. Madsen, Ana M. Martínez, Ramón Martínez, Andrés Masegosa, Thomas D. Nielsen, Antonio Salmerón, Frode Sørmo, Galia Weidl	Final version of document

1 Executive summary

This document describes the AMIDST modelling framework which consists of the general AMIDST model class as well as the specification of the model classes of the three use-cases, namely, Daimler AG, Cajamar, and Verdande Technology AS.

First, the identified preliminary model classes are introduced, for each use-case, based on the requirements analysis (see Deliverable 1.2 [1]) and the preliminary data analysis (including the use of several tools such as sample correlograms, sample partial correlograms, histograms and bivariate contour plots).

Next, the general AMIDST model class is defined such that it takes all the preliminary model class characteristics into account, it is applicable to any future, potentially similar, use-cases, and it ensures as well both inference and learning from massive data streams.

It is crucial here to note that the current AMIDST modelling framework should be considered as an initial proposal that might be updated/extended in the future, in order to better meet AMIDST requirements and/or to deal with some potential unexpected events. The final AMIDST modelling framework will be presented in Deliverable 2.2.

2 Introduction

Data is being generated at a tremendous rate, e.g. with sensors providing continuous measurements of the environments in which they operate. To handle these big data streams, current systems often employ simplistic solution techniques that, e.g., only consider the most recently generated data or only store the data at a much lower frequency than with which it is generated. By doing so potentially valuable data, which could otherwise have contributed to an improved system accuracy, is being ignored.

Along this document, we analyse 3 particular domains facing extremely large datasets, namely risk prediction in credit operations, real time event detection in oil-well drilling and real time identification and interpretation of maneuvers in traffic. They will serve as a guide to design the general AMIDST model to provide a scalable framework that facilitates efficient analysis and prediction based on big streaming data. It will be designed as a probabilistic graphical model, which will entail several advantages: 1) uncertainty about predictions can be quantified; 2) missing and erroneous data records and observations can be handled using robust techniques based on well-founded principled methods; and 3) the model itself serves as a compact representation of the data, thus reducing the necessity of storing huge volumes of historical data (as the model is continuously updated with new data).

The structure of the document is as follows: Section 3 introduces the required background and notation, Section 4 describes the different domains and probabilistic graphical models proposed for each of them. Finally, Section 5 presents and discusses the proposed general AMIDST model class.

3 Background and notation

The following sections aim at describing some of the required concepts and basic structures to make it easier to interpret the AMIDST models that will be presented afterwards for the different use cases. Section 3.2 briefly introduces Bayesian networks and discusses the challenges encountered for reasoning with continuous and discrete variables, Section 3.3 describes some of the basic concepts and models related to dynamic Bayesian reasoning over time, and, finally, Section 3.4 defines the data analysis techniques used to ensure a better understanding of AMIDST models.

3.1 Graphical notation

Figure 3.1 shows in the first row the graphical representation of the different types of variables used in AMIDST model classes. A continuous variable is depicted with an ellipse with a blue background, while a discrete variable is depicted with an ellipse with a green background. If a variable could be either continuous or discrete, the background

of the ellipse is simultaneously blue and green. In addition, a hidden variable is depicted with a dashed-line ellipse, while an observed variable is depicted with a solid-line ellipse. Obviously, a continuous hidden variable, for example, would be represented with a blue dashed-line ellipse.

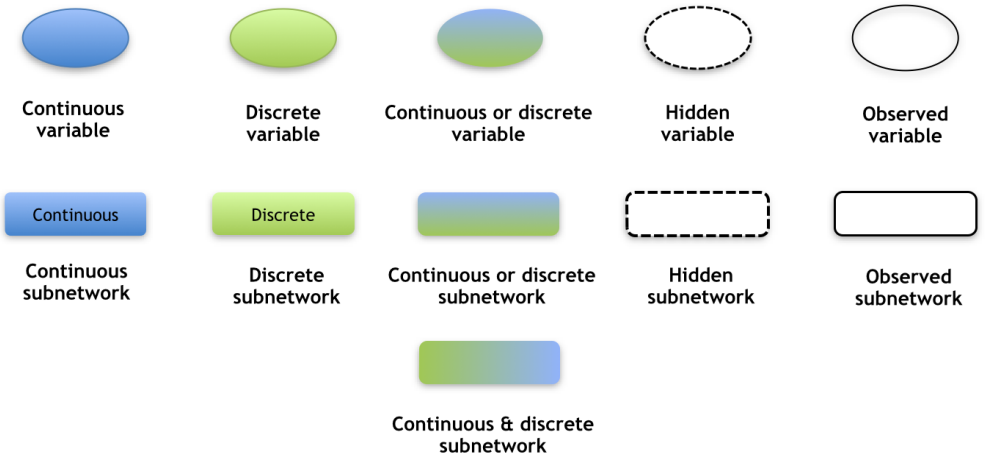


Figure 3.1: Graphical notation of the different types of variables and subnetworks.

The second row of Figure 3.1 shows the graphical representation of the different types of subnetworks. A subnetwork refers to a directed acyclic graph defined over a set of variables[1]. A continuous subnetwork (i.e., including only continuous variables) is depicted with a blue rectangle, while a discrete subnetwork (i.e., including only discrete variables) is depicted with a green rectangle.

If a subnetwork can be instantiated, at a specific moment, with either continuous or discrete variables (i.e., we have a continuous or discrete subnetwork), the rectangle background is horizontally coloured with blue and green. However, if a subnetwork is simultaneously instantiated with both continuous and discrete variables (i.e., we have a continuous & discrete subnetwork), the rectangle background is vertically coloured with blue and green. Finally, a hidden subnetwork is depicted with a dashed-line rectangle, while an observed subnetwork is depicted with a solid-line rectangle.

3.2 Bayesian networks

Bayesian networks (BNs) [2] are widely used probabilistic graphical models for reasoning under uncertainty. They graphically encode a set of conditional independence assumptions that are exploited to efficiently perform a wide variety of inference tasks such as marginal belief computation, belief updating, most probable explanation, etc.

[1] In particular, a subnetwork refers in our case to a part of a dynamic Bayesian network that will be defined later on this document.

Formally, let $\mathbf{X} = \{X_1, \ldots, X_N\}$ denote the set of stochastic random variables defining our domain problem. A BN defines a joint distribution $P(\mathbf{X})$ in the following form:

$$p(\mathbf{X}) = \prod_{i=1}^{N} p(X_i|Pa(X_i))$$

where $Pa(X_i) \subset \mathbf{X} \backslash X_i$ represents the so-called *parent variables* of X_i. Bayesian networks can be graphically represented by a directed acyclic graph (DAG). Each node, labelled X_i in the graph, is associated with a factor or conditional probability $p(X_i|Pa(X_i))$. Additionally, for each parent $X_j \in Pa(X_i)$, the graph contains one directed edge pointing from X_j to the *child* variable X_i.

Figure 3.2 shows an example of a BN model. The nodes, which correspond to variables, are coloured in green or blue to highlight their nature, i.e., discrete or continuous, respectively, according to the notation depicted in Figure 3.1. Note that the notation on the second row corresponds to how a particular subnetwork will be represented later on in this document, e.g. a subnetwork with a set of hidden discrete and continuous variables would be represented by a dotted white frame[2]. Each *subnetwork module* corresponds to a part of the DBN with common features such as all the nodes are continuous and observed. Instead of eclipsed nodes used to depict single variables in a graph, we represent here the subnetwork modules using frames. In general, dashed lines refer to hidden variables or subnetworks (i.e., those including only hidden variables), while continuous lines refer to observed variables or subnetworks (i.e., those including only observed variables). Then, depending on the background colour, variables or subnetworks can be either discrete (green), continuous (blue) or hybrid (white or green/blue).

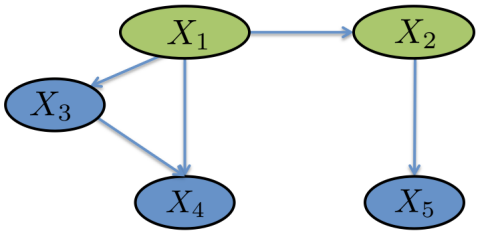


Figure 3.2: Example of a BN model with continuous and discrete variables.

Traditionally, BNs have been defined for discrete domains, where the entities of interest are modelled by discrete variables which ensures that belief updating can be performed efficiently and in closed form. However, this requirement imposes severe restrictions as many domains contain entities that are more appropriately modelled by variables with continuous state spaces; such as distance and velocity measurements which are key sensor

[2]This is indeed the only restricted subnetwork, since only links from discrete to continuous variables are allowed and not the other way around.

readings for identifying and interpreting traffic manoeuvres (see Daimler's requirement analysis [3]).

To deal with this problem, research has largely pursued three main approaches to extend BNs and support continuous variables. As a first approach, one may choose to carefully construct the model so that exact inference algorithms can still be applied. This is the case for the conditional linear Gaussian (CLG) BNs [4, 5], where the local probability distributions of continuous variables are specified as conditional linear Gaussian distributions and where discrete variables can only have discrete parents. A second approach consists of considering approximate algorithms for performing inference, allowing thereby arbitrary distributions to be associated with the BN model. Examples of this approach include the Gibbs sampler [6, 7] and variational inference [8]. Finally, the third approach consists of "transforming" the original BN model into an approximate model, for which exact inference algorithms can be applied. This can be achieved either by discretizing the continuous variables [9] or using transformations with more expressive power, such as using mixtures of truncated exponentials [10] or mixtures of truncated basis functions [11].

The models presented in this deliverable, unless otherwise stated, will be parameterized according to the conditonal linear Gaussain model [4, 5]. That means that we will have the following conditional distributions depending of the type, continuous or discrete, of the parents and the child variables:

- **Discrete** $\rightarrow$ **Discrete**: The child variable follows an independent multinomial probability distribution for each configuration or assignment of the parents variables.
- **Continuous** $\rightarrow$ **Continuous**: The child variable follows a conditional linear Gaussian distribution. I.e. the mean parameter of the Gaussian distribution of the child variable is a linear combination of the parents variables while the variance is a fixed independent parameter.
- **Discrete** $\rightarrow$ **Continuous**: In this case, the child variable is distributed as an independent Gaussian distribution for each configuration of the parents variables.
- **(Discrete, Continuous)** $\rightarrow$ **Continuous**: For each configuration of the discrete parents variables, the child variable follows an independent conditional Gaussian distribution depending of the continuous parents variables. I.e. the mean parameter of the Gaussian distribution of the child variable is expressed as a different linear combination of the continuous parents variables for each configuration of the discrete parents variables and, moreover, the variance of this Gaussian can also be different.

3.3 Probabilistic reasoning over time

Many domains, and in particular those being analysed in the AMIDST project, can be seen as having strong internal structure. This will be evident by the domains being appropriately described using an object oriented language, either due to repetitive substructures or substructures that can be naturally ordered in a superclass/subclass hierarchy. Object oriented BNs [12] (OOBNs) are defined to take advantage of such internal model structure. In dynamic models, we also find this property because the same part of the model is repeated over time (i.e., we have multiple objects of the same class under the OOBNs language). A special type of OOBNs is the dynamic BN (DBN) [13], which is used to model domains that evolve over time by representing explicitly the temporal dynamics of the system. DBNs can also be readily understood as an extension of standard BNs to the temporal domain.

Similarly to static BNs, we model our problem/system using a set of stochastic random variables, denoted $\mathbf{X}_t$, with the main difference that variables are indexed here by a discrete time index t. In this way, we explicitly model the state of the system at any given time. Moreover, we always assume that the system is described at a fixed frequency, and use $\mathbf{X}_{a:b} \equiv X_a, X_{a+1}, \ldots, X_b$ to denote the set of variables between two time points a and b.

For reasoning over time, we need to model the joint probability $p(\mathbf{X}_{1:T})$ which has the following natural cascade decomposition:

$$p(\mathbf{X}_{1:T}) = \prod_{t=1}^{T} p(\mathbf{X}_t|\mathbf{X}_{1:t-1}),$$

where $p(\mathbf{X}_t|\mathbf{X}_{1:t-1})$ is equal to $p(\mathbf{X}_1)$ for $t = 1$. As t increases, the conditional probability $p(\mathbf{X}_t|\mathbf{X}_{1:t-1})$ becomes intractable. Similarly to static BNs, *dynamic BNs* allow more compact factorization of the above joint probability. The first kind of conditional independence assumption encoded by DBNs to reduce the factorization complexity is the well-known *Markov assumption*. Under this assumption, the current state is independent from the previous one given a finite number of previous steps and the resulting models are referred to as *Markov chains*. Basically, a Markov chain can be defined on either discrete or continuous variables $\mathbf{X}_{1:T}$. It exploits the following equality:

$$p(\mathbf{X}_t|\mathbf{X}_{1:t-1}) = p(\mathbf{X}_t|\mathbf{X}_{t-V:t-1})$$

where $V \geq 1$ is the order of the Markov chain. Figure 3.3 shows two examples of DBNs corresponding to first-order (i.e., $V = 1$) and third-order (i.e., $V = 3$) Markov chains.

Among the different kinds of Markov chains, the **first-order Markov chains** are the most widely used. They assume that knowing the present makes the future conditionally independent from the past, that is, $p(\mathbf{X}_t|\mathbf{X}_{1:t-1}) = p(\mathbf{X}_t|\mathbf{X}_{t-1})$. The problem is that this

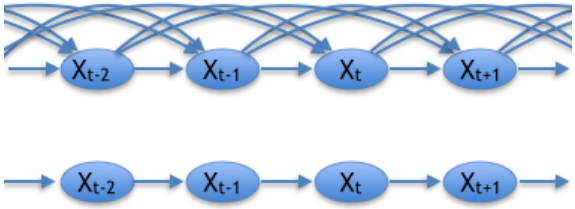


Figure 3.3: An example of DBNs assuming a third-order (above) and a first-order (below) Markov property.

could be an unrealistic assumption in some problems leading to poor approximations of the joint distribution. One could increase the Markov order to improve the approximation at the expenses of having a more complex model. Another alternative [14] would be to increase the number of variables modelling the system. This alternative is usually preferred if we have a sound understanding of the "physics" of the process being modelled (e.g. predicting whether it is going to rain or not tomorrow based on the raining evidence of the current day can be probably improve by considering the raining evidence of previous days. Alternatively, we can also improve this prediction by considering in the modelling the humidity and the pressure of the current day [14]).

An additional challenging problem is the specification of the conditional probabilities at each time step of the DBNs. To deal with this problem, we usually assume that changes in the world state are driven by a *stationary process*, that is, $p(\mathbf{X}_{t+1}|\mathbf{X}_t) = p(\mathbf{X}_t|\mathbf{X}_{t-1})\ \forall t \in \{1,\ldots,T\}$.

In the following sub-sections, we will present in more details some basic examples of DBNs, namely, *hidden Markov models* (Section 3.3.1), *Kalman filters* (Section 3.3.2), and *two-time slice DBNs* (Section 3.3.3). Let us recall here that the graphical notation employed when describing all these models is depicted in Figure 3.1.

3.3.1 Hidden Markov models

A hidden Markov model (HMM) is the simplest DBN including both hidden and observed variables, such that the latent state of the process is represented by a single discrete variable. More precisely, a HMM, shown in 3.4, defines a Markov chain over the hidden variables $X_{1:T}$. The observed variables, denoted by $\mathbf{Y}_{1:T}$, are dependent on the hidden variables under the *sensor Markov assumption*, that is, $P(\mathbf{Y}_t|X_{1:T}, \mathbf{Y}_{1:T}) = P(\mathbf{Y}_t|X_t)$, where $P(\mathbf{Y}_t|X_t)$ represents the sensor (or observation) model.

In that way, the joint probability distribution over the observed and hidden variables can be represented as:

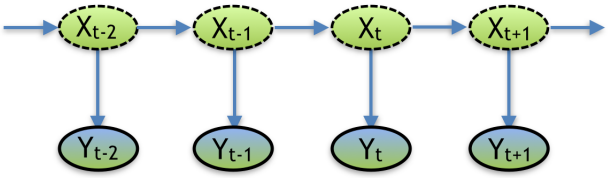


Figure 3.4: An example of a BN structure corresponding to a HMM.

$$P(\mathbf{X}_{1:T}, \mathbf{Y}_{1:T}) = \prod_{t=1}^{t} P(\mathbf{X}_t|\mathbf{X}_{t-1})P(\mathbf{Y}_t|\mathbf{X}_t). \tag{3.1}$$

Although most of our models will fit into this description of observed and hidden (state) variables, there will be cases in which the transition model takes place in the observed variables (see, e.g., the case of Cajamar), which in general simplifies the learning-inference processes of the problem.

An extension of the HMM is the so-called *input-output hidden Markov model* (IOHMM) shown in Figure 3.5. IOHMM incorporates an extra top layer of input variables $\mathbf{Y}'_{1:T}$, which can be either continuous or discrete. The existing HMM layer of observed variables, $\mathbf{Y}_{1:T}$, is referred to as the output set of variables.

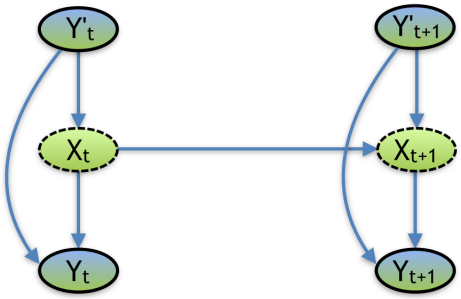


Figure 3.5: An example of a BN structure corresponding to an IO-HMM.

IOHMM is usually employed in supervised classification problems. In this case, both input and output variables are known during training, but only the former is known during testing. In fact, during testing, inference is performed to predict the output variables at each time step. In AMIDST we use this model in a different way. In our case, both set of input and output variables are always known, so that inference is only performed to predict the latent variables. The input variables $\mathbf{Y}'_{1:T}$ are introduced as a way to "relax" the stationary assumption, by explicitly introducing a dependency to some observed information at each time slice, that is, the transition probability between

$\mathbf{X}_t$ and $\mathbf{X}_{t+1}$ depends on the observed value $\mathbf{Y}_{t+1}$.

3.3.2 Kalman filters

Similar to the extension of the static BN model to hybrid domains, DBNs have likewise been extended to continuous and hybrid domains. In purely continuous domains, where the continuous variables follow linear Gaussian distributions, the DBN corresponds to (a factorized version of) a Kalman filter (KF). The structure of a KF is exactly the same as the one displayed in Figure 3.4 for the HMM, however with the restriction that all variables should be continuous. In this case, the state variables can be a combination of continuous variables with different dependences, and where the dynamics of the process are assumed to be linear.

When modelling non-linear domains, the dynamics and observational distributions are often approximated through, e.g., the *extended Kalman filter*, which models the system as *locally* linear in the mean of the current state distribution. Another type of model ensuring non-linear predictions with a more expressive representation is the *switching Kalman filter* (SKF). The type of SKF that we are going to consider here includes an extra discrete state variable that is able to use a weighted combination of the linear sub-models. That is, the discrete state variable assigns a probability to each linear term in the mixture, hence, representing the belief state as a mixture of Gaussians. In this way, it can deal, to some extent, with violations of both the assumption of linearity and Gaussian noise. Figure 3.6 depicts the graphical structure of this dynamic model.

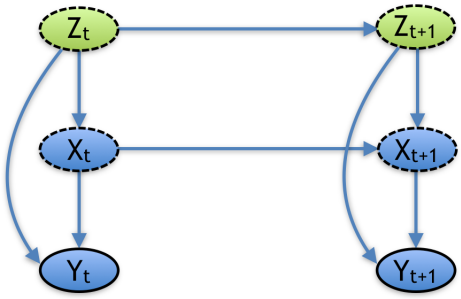


Figure 3.6: An example of a switching Kalman filter. Z_t represents the discrete state variable modelling multiple KFs running in parallel.

Similarly to HMM, these models can be extended by introducing an extra top layer of input variables to "relax" the stationary assumption, by explicitly introducing a dependency to some observed information for the transition probability of the latent variables. These models will be better explained in Section 4.3.

3.3.3 Two-time slice dynamic Bayesian networks

In general, DBNs can model arbitrary distributions over time. However, in AMIDST, we will especially focus on the so-called *two-time slice DBNs* (2T-DBNs). 2T-DBNs are characterised by an *initial model* representing the initial joint distribution of the process and a *transition model* representing a standard BN repeated over time. This kind of DBN model satisfies both the first-order Markov assumption and the stationarity assumption. Figure 3.7 shows an example of a graphical structure of a 2T-DBN model.

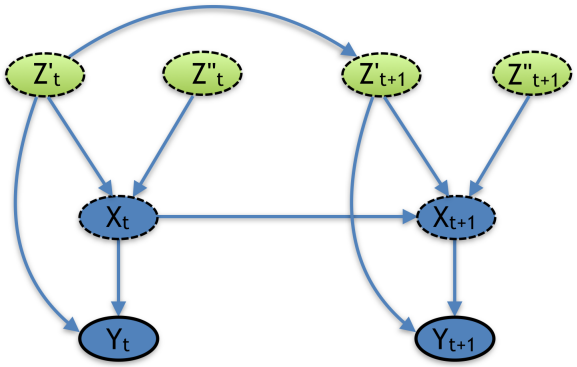


Figure 3.7: An example of a BN structure corresponding to a 2T-DBN.

In a 2T-DBN, the transition distribution is represented as follows:

$$p(\mathbf{X}_{t+1}|\mathbf{X}_t) = \prod_{X_{t+1}\in\mathbf{X}_{t+1}} p(X_{t+1}|Pa(X_{t+1})),$$

where $Pa(X_{t+1})$ refers to the set of parents of the variable X_{t+1} in the transition model, which can be variables either at the same or the previous time step.

Note that a HMM, KF or SKF are particular cases of a 2T-DBN. Equally, some 2T-DBNs can be casted to these standard models by grouping variables. For example, the 2T-DBN shown in Figure 3.7 can be considered a SKF (see Figure 3.6) if we group Z'_t and Z''_t in a single and bigger variable $Z_t \equiv Z'_t \times Z''_t$ (and likewise $Z_{t+1} \equiv Z'_{t+1} \times Z''_{t+1}$). However, it is usually preferred to factorise the transition distribution by taking the 2T-DBN graphical structure into account. This is due to the fact that we can take advantage of the sparseness of the model, especially when dealing with high-dimensional problems. For instance, in our considered example, the transition probability of the equivalent SKF model would be simply expressed as $p(Z_{t+1}|Z_t) = p(Z'_{t+1}, Z''_{t+1}|Z'_t, Z''_t)$. However, by taking the 2T-DBN graphical structure into account, this transition probability could be much more compactly expressed as $p(Z_{t+1}|Z_t) = p(Z''_{t+1})p(Z'_{t+1}|Z'_t)$. That is, instead of considering the joint probability distribution, the set of (conditional) independences encoded in the network structure is exploited.

DBNs obviously share the computational difficulties of regular BNs in inference tasks. However, in the dynamic case, we are also faced with the *entanglement problem*, i.e., after a certain time step, all variables describing the current system state will become dependent, and we therefore cannot represent the exact probability distribution over the current state (the belief state) in a compact and factorized form. To deal with this problem, approximate methods (including approximate factorizations of the joint probability distribution describing the system state) [15] as well as sampling based techniques in the form of particle filtering [16] are usually used.

3.4 Preliminary data analysis

The motivation behind using a preliminary data analysis is first to test some assumptions supporting the models elicited by the experts in the different use cases, and second, to further complement our understanding about the nature of the problem to be modelled. In the following sub-sections, we will introduce the set of tools, namely sample correlograms, sample partial correlograms, histograms, and bivariate contour plots, that allows us to get some initial insights into both structural and distributional DBN assumptions.

3.4.1 Sample correlograms and sample partial correlograms

A DBN mainly aims to model complex multivariate time series. By using sample correlograms and sample partial correlograms, we will try to test if the available data supports the temporal correlation between variables assumed by the DBN model, i.e., the temporal links between variables. However, these tools will only allow us to look at univariate time series, which may strongly limit the extent of the extracted conclusions. However, despite its limitations, this analysis will give us some interesting insights which usually cannot be elicited from experts, as we will see below for the different use cases.

- **Sample correlograms**: Let $x_1, ..., x_T$ be a univariate time series. The *sample autocorrelation coefficient* at lag v is given by

$$\hat{\rho}_v = \frac{\sum_{t=1}^{T-v}(x_t - \bar{x})(x_{t+v} - \bar{x})}{\sum_{t=1}^{T}(x_t - \bar{x})^2}$$

  where $\bar{x}$ is the sample mean and T is the total length of the considered data. The plot of $\hat{p}_v$ versus v, for $v = 1, \ldots, M$ for some maximum M is called the *sample correlogram* of the data. $\hat{p}_v$ corresponds to the Pearson correlation between the series $\{x_t\}_{t\in\{1,...,T\}}$ and $\{x_{t+v}\}_{t+v\in\{1,...,T\}}$.

  Sample correlograms can be interpreted as a way to measure the strength of the following unconditional dependences: $X_t \not\perp\!\!\!\perp X_{t+v}$ for some lag $v \geq 1$. When $\hat{\rho}_v$ is close to zero, this indicates that there exists a strong unconditional independence between X_t and X_{t+v}. However, when $\hat{\rho}_v$ is close to either 1 or -1, this indicates

that there is a strong correlation or dependency between X_t and X_{t+v}. However, once again, we should never forget that when computing these Pearson correlation coefficients, we are making a strong assumption about the normality of the data, which might not hold. Intuitively, they can be used to get an idea of the type of the "memory" encoded in the time series, differentiating between short-term and long-term memory.

Figure 3.8 shows examples of sample correlograms for two different types of data sets: Figure 3.8(a) shows a sample correlogram for a sequence of 50 i.i.d. data records sampled according to a Gaussian distribution with zero mean and unit variance $x_t \sim \mathcal{N}(0,1)$; and Figure 3.8(b) shows a sample correlogram for a sequence of 50 data samples distributed as $x_t = x_{t-1} + \epsilon$, such that $\epsilon \sim \mathcal{N}(0,1)$. As it can be seen, the correlogram for the i.i.d. data has values close to zero for all lags. However, for time series data, the correlogram clearly identifies the presence of a temporal relationship in the data. As expected, the correlation decreases with the size of the lag, and how quickly it decreases depends on the strength of the temporal relationship, or more intuitively, the "memory" of the time series. In the previous example, this "memory" inversely depends of the variance of the "white noise" ϵ value.

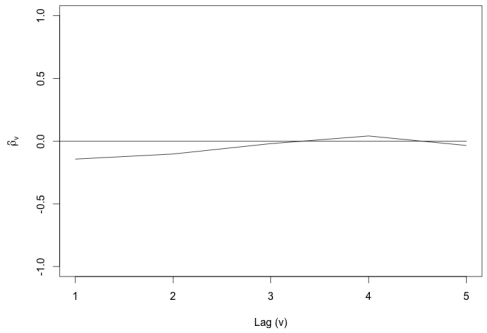
- **Sample partial correlograms**: Let X_t be a random variable associated to X taking values at time t. We can build the following regression problem:

$$X_t = a_0 + a_1 X_{t-1} + a_2 X_{t-2} + ...a_{v-1} X_{t-v+1}$$

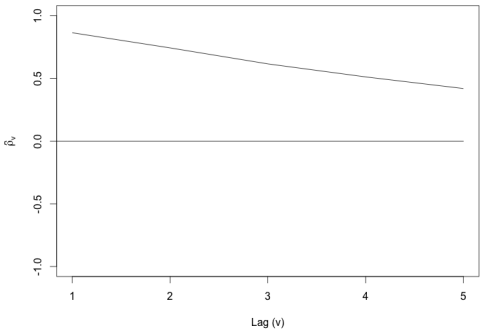
In addition, let $e_{t,v}$ denotes the residuals of this regression problem (i.e., the error when estimating X_t using a linear combination of $v-1$ previous observations). The *sample partial auto-correlation coefficient* of lag v, denoted as $\hat{\theta}_v$, is the standard sample auto-correlation between the series $\{x_{t-v}\}_{t-v\in\{1,...,T\}}$ and $\{e_{t,v}\}_{t\in\{1,...,T\}}$. Intuitively, the sample partial auto-correlation coefficient of lag v can be seen as the correlation between X_t and X_{t-v} after having removed the common *linear* effect of the data in between.

As previously, we plot in Figure 3.8 (c) and (d) the sample partial correlograms for the same two data sequences presented above. In the case of i.i.d. data, we can see again that the partial correlogram does not show any sign of partial correlation between the data sequence samples. However, for time series data, the partial correlogram takes a high value for $v = 1$ (for this lag value it is equal to the sample correlogram), but then becomes close to zero for v values higher than 1. The sample partial correlogram can be interpreted as a way to measure the strength of the following conditional dependence: $X_t \not\perp X_{t+v} | X_1, ..., X_{t+v-1}$ for some lag $v > 1$. Accordingly, the sample partial correlogram correctly identifies that we have the following conditional independencies: $X_t \perp X_{t+2} | X_{t+1}$ in the considered time series data.

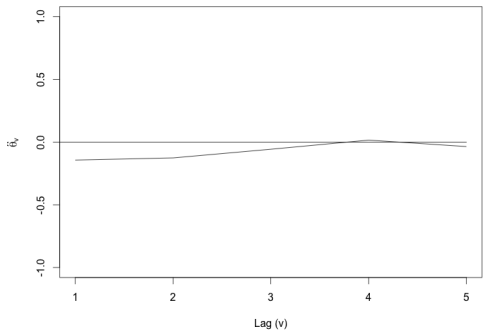
Therefore, sample partial correlograms can be seen as a tool to test the order of the Markov chain generating the time data sequence, with all the same caveats expressed for the sample correlogram.



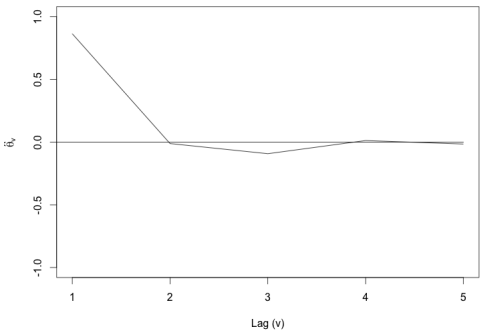
(a) Correlogram for i.i.d. data



(b) Correlogram for a time series data



(c) Partial correlogram for i.i.d. data



(d) Partial correlogram for a time series data

Figure 3.8: Examples of sample correlograms and sample partial correlograms for i.i.d. and time series data.

3.4.2 Histograms and bivariate contour plots

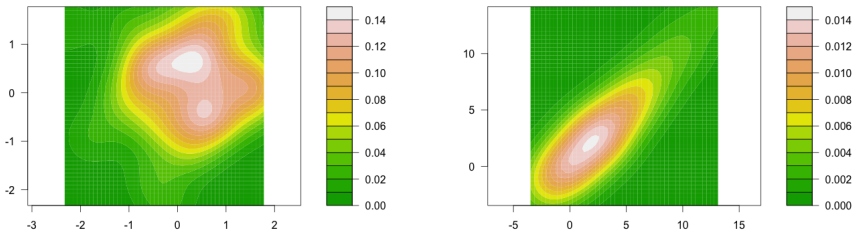
The aim here is to use histograms and bivariate contour plots in order to get insights into the probability distributions of the proposed models.

- **Histograms**: Despite the fact that this tool is quite useful in a static context, it is rather limited in dynamic models. For example, let us assume we have a time series $x_1, \ldots, x_T$ and our histogram shows that the empirical distribution of the variable when we aggregate the data samples over time looks like a mixture of Gaussian distributions. There are at least two possibilities that can give rise

to this finding: i) X_t is distributed according to a mixture of Gaussians where each Gaussian component depends on X_{t-1}; and ii) there is a discrete hidden variable Z_t that influences X_t and is the one responsible for generating the different mixture components. In any case, however, we could deduce that using Gaussian distributions would be appropriate.

Thus, despite its limitations in dynamic contexts, we will resort to the use of histograms whenever we find that they could shed some lights on the underlying sample distribution of the sample.

- **Bivariate contour plots**: The contour plots of the empirical bivariate distribution of X_t versus X_{t-1} can show many relevant information, such as the presence of linear relationships between variables or if we can assume they are normally distributed, etc. In Figure 3.9, we plot the bivariate contour plot for the i.i.d and time series data described above. As it can be seen, the bivariate contour plot for time series data shows how X_t and X_{t+1} seems to be distributed according to a bivariate normal with a covariance matrix that displays a strong degree of correlation. In the case of i.i.d. data, the bivariate contour plot does not reveal any temporal dependence between X_t and X_{t-1}.



(a) i.i.d. data (b) Time series data

Figure 3.9: Bivariate contour plots for a set of i.i.d. and time series data.

Finally, note that the usefulness of all these tools is limited due to its generative nature. That is, they do not explicitly target the prediction problem for the different use cases. After performing a proper evaluation of the considered static and dynamic models, it will be possible to re-adjust, if needed, our current assumptions.

4 Preliminary models

4.1 Daimler models

The Daimler use case is based on two application scenarios [3]: i) early recognition of a lane change manoeuvre; and ii) earlier prediction of the need for a lane change based on the relative dynamics between two vehicles driving in the same lane at different speeds.

The first scenario has previously been addressed by Daimler [17–19]. The main result of this previous work was a static object-oriented Bayesian network (OOBN) [12] able to detect a manoeuvre 0.6 seconds before execution. The goal now is to enhance the prediction horizon for manoeuvre recognition by at least 1-2 seconds before execution, in order to further improve the quality of the on-board adaptive cruise control. As we will explain later, this improvement is expected to be achieved by introducing a dynamic extension of the previously proposed static OOBN model.

We basically consider such a dynamic extension for two main reasons: First, although its limited prediction horizon, the static OOBN model has proven to be very robust for this task and it is considered as the gold-standard solution in Daimler. Second, AMIDST models are expected to be integrated in an electronic control unit (ECU) [3], and we expect that the methods developed when integrating the static OOBN in this ECU [20] can also be exploited in the integration of this dynamic version.

4.1.1 Early recognition of a lane change manoeuvre

The basic settings of this application scenario are as follows: Let us suppose we are driving our car, referred to as the EGO vehicle, on a highway. The EGO vehicle is equipped with a video camera, radar, and some on-board sensors. Using the data provided by these sensors, the challenge consists in making an early recognition of a manoeuvre either by the EGO or any other relevant car in the traffic scene referred to as Object vehicle (OBJ). In total, the system is expected to recognise the following set of manoeuvres (a visual description of them is given in Figure 4.1):

1. **OBJ-CutIn**: The OBJ vehicle is moving to the lane where the EGO vehicle is placed.
2. **OBJ-CutOut**: The OBJ vehicle, which was driving in front of the EGO vehicle, is leaving now the EGO's lane.
3. **OBJ-Follow**: There is no lane change. The EGO is driving and there is another vehicle (i.e., OBJ) in front.
4. **LANE-Follow**: There is no lane change. The EGO is driving and there is no other vehicle in front.

5. **EGO-CutIn**: The EGO vehicle is moving right-direction to a new lane already occupied by another vehicle.

6. **EGO-CutOut**: The EGO vehicle is leaving left-direction the lane where it was driving.

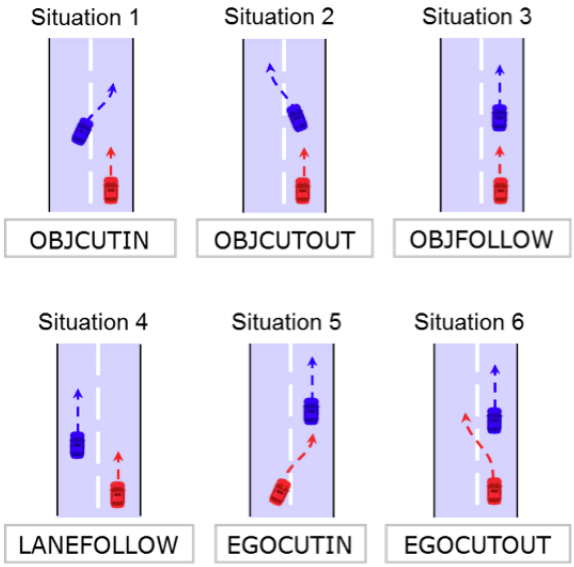


Figure 4.1: The different manoeuvres to be identified by the AMIDST system. Red blocks represent the EGO vehicle while blue ones represent the OBJ vehicle.

Instead of working directly with the raw data from the video, radar, and on-board sensors, the current manoeuvre recognition system uses the so-called "object data", which contains "high level" representations or features describing the traffic scene such as EGO's speed, distance between EGO and another vehicle in front, etc.

Figure 4.2 contains a visual description of the current data flow used to create the "object data". First, the raw data coming from the video, radar, and sensors are preprocessed. Then, the preprocessed data is merged, and the high-level or "object data" describing the traffic scene is obtained.

As commented above, using the resulting "object data", Daimler has developed a probabilistic graphical model [18] which is able to recognise an ongoing manoeuvre around 0.6 seconds before it really takes place. This model is partly described in the following section. But full details will be given in the confidential Deliverable 6.1.

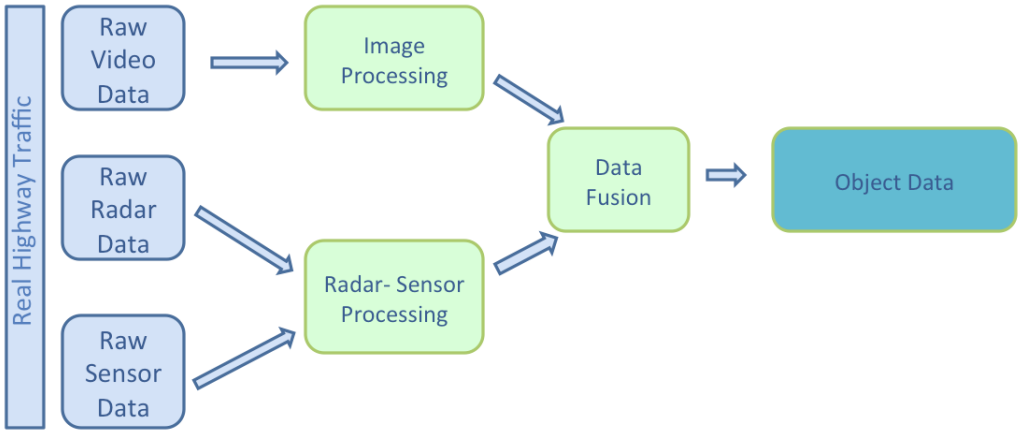


Figure 4.2: Daimler's data flow.

Static OOBN model

The model described here was presented in [18] as an object-oriented Bayesian network (OOBN) [12] for addressing the problem of early recognition of a lane change manoeuvre (application scenario 1), and will form the basis for the dynamic models developed in AMIDST for this use case.

This model works with the so-called "object data". This data mainly consists of a set of measured and/or computed signals or situation-features denoted by S (e.g., EGO speed, EGO lateral velocity, speed of car in-front, etc., see [18] for further details) describing the traffic scene. The situation features used for manoeuvre recognition are structured along three main dimensions: lateral evidence (LE), trajectory (TRAJ), and occupancy schedule grid (OCCGRID). A visual description is given in Figure 4.3. They are referred to as the three possible hypotheses of a lane change manoeuvre: 1) LE hypothesis considers the lateral offset and the lateral velocity of a car and accounts for its lateral movement; 2) TRAJ hypothesis accounts for the evidence about a car's trajectory by using the measures of the car angle and the estimated time to crossing the line; and 3) OCCGRID hypothesis allows to identify if a car neighbouring (i.e., adjacent environments) is going to be occupied by some other vehicle in the traffic scene.

The general structure of this OOBN model consists of a number of abstraction levels as detailed in Figure 4.4:

- **Class S: Sensor measurement**: This class represents objects at the lowest level of the OOBN. It models the so-called *measured data*, which consists of a set of observations, acquired from sensors and computations, characterising situations in the traffic scene. The general structure could be considered as a standard

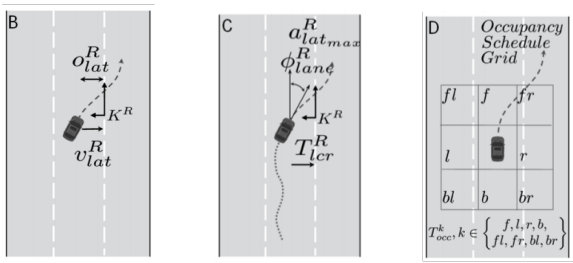


Figure 4.3: The three main dimensions of the situation features [18].

Kalman filter model (see Section 3.3.2), where S_MEAS refers to the sensor-reading value, S_REAL refers to the real value and S_SIGMA refers to the uncertainty in the measurements. As it can be seen, S_MEAS is conditionally dependent on the random changes in S_REAL as well as the sensor noise/fault (S_SIGMA). In Daimler, the observations of the measured value S_MEAS and the uncertainty of the measurement S_SIGMA are given in the object data.

- **Class H: Hypothesis**: This class pertains to a higher level and directly depends on the real values S_REAL obtained in the previous class. In fact, the real sensor values are used to evaluate the different possible hypotheses namely LE, TRAJ and OCCGRID.
- **Class E: Event**: This class is at the top level of the modelling. It allows to model a high-level Hypothesis/Event based on a set of low-level hypotheses $(Hypothesis_1, \ldots, Hypothesis_n)$ in a recursive way. This class also includes the *Event* variable representing the possible traffic manoeuvres of the EGO and OBJ vehicles.

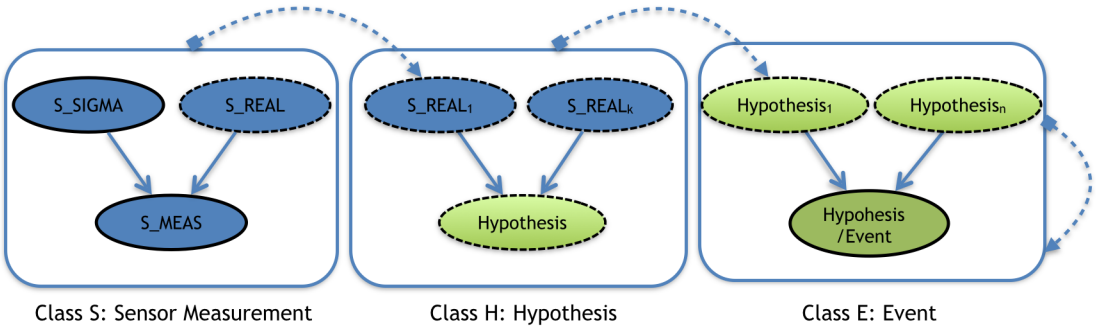


Figure 4.4: The static OOBN model used to predict an event [20].

Finally, Figure 4.5 shows a concrete fragment of the static OOBN model related to the LE hypothesis modelling. As it can be seen, LE hypothesis depends on the lateral offset to a lane marking, O_LAT_REAL, as well as on the lateral velocity, V_LAT_REAL, of the car. Both measures are estimated from their measured values, O_LAT_MEAS and V_LAT_MEAS, respectively, and from the estimated uncertainty of the measurements, O_LAT_SIGMA and V_LAT_SIGMA. This OOBN fragment is used to model the growing probability for LE to cross the lane marking, based on the vehicle coming closer to the lane marking and the increase of its lateral velocity.

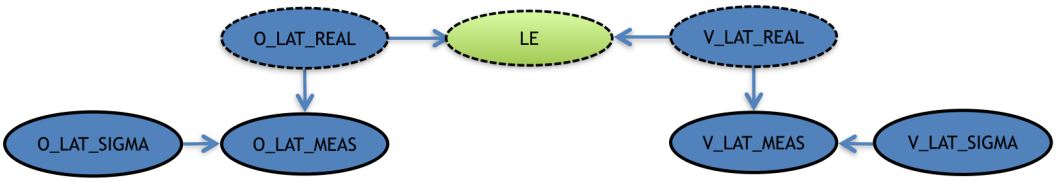


Figure 4.5: Static-OOBN fragment for the lateral evidence hypothesis.

Note that, the static OOBN model contains both continuous and discrete variables. It was parametrized using the CLG framework [2, 21] that allows continuous children with continuous parents and multinomial distributions for discrete children with discrete parents. In addition, we have discrete children with continuous parents (i.e., the low-level Hypothesis depending on $\text{S\_REAL}_1, \ldots, \text{S\_REAL}_k$ variables) that were initially parametrized using a logistic function. However, as discussed in Section 3, this modelling has a significant impact when making inference. A possible solution used in [18] is to discretize S_REAL variables. Given that S_MEAS and S_SIGMA are always observed, their potentials can be therefore collapsed/combined with the potential of the corresponding discretized S_REAL variables. Consequently, all the inferences in the resulting model can be performed using only discrete potentials [2].

The dynamic-OOBN model

The above described static OOBN is able to detect a manoeuvre 0.6 seconds before execution. As detailed in the DoW [3], the goal is to enhance the prediction of manoeuvre recognition to at least $1-2$ seconds (max. $4-5$ seconds) before the actual lane marking crossing. Moreover, this early detection of the manoeuvre should not be at the expense of the prediction accuracy. As specified in Daimler's DoW [3], the area under the ROC curve (AUC) should be greater than 0.96 for 1 second and greater than 0.9 for 2 seconds.

Figure 4.6 shows an example of the performance and limitations of the current static-OOBN model. Two randomly selected sequences, namely seq1 and seq2, where the behaviour of two cars, EGO and OBJ is considered.

In these figures, we plot the evolution of different time-steps for lateral offset (O_Lat)

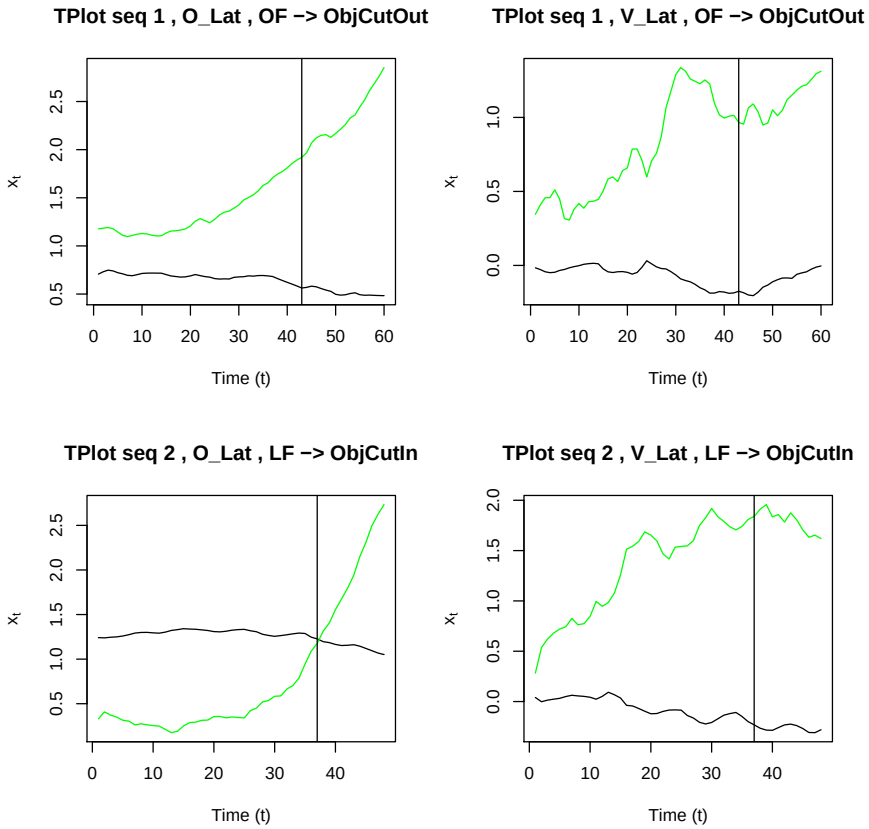


Figure 4.6: Time plots of the lateral velocity (V_Lat) and the lateral offset (O_Lat) for EGO (black curves) and OBJ (green curves) in two randomly selected sequences of an ObjCutOut (seq 1) and an ObjCutIn (seq 2). The x-axis corresponds to the different time-steps and the y-axis to the lateral offset on the left-hand side graphs and lateral velocity on the right-hand side graphs.

and lateral velocity (V_Lat) to a lane marking in ongoing OBJ-CutOut and OBJ-CutIn manoeuvres. In each figure, the vertical black bar indicates the moment in which the manoeuvre has been recognised by the current static OOBN model. The manoeuvre is finished at the end of the series, which coincides with the actual moment of changed lane. The black curve corresponds to the lateral offset and lateral velocity of the EGO car, which is just following the lane (LF), while the green curve corresponds to the values of the OBJ car performing the OBJ-CutOut/OBJ-CutIn manoeuvres. As expected for lane follow (LF), the lateral velocity of the EGO fluctuates around zero (i.e., EGO car is just driving inside its lane) and the lateral offset of the lane marking is almost constant all the time. However, when we look at the lane change behaviour of the OBJ car (green curves), we easily see a different behaviour. Firstly, we observe that the lateral velocity

is much higher, indicating a lateral movement. Similarly, we also observe how the lateral offset steadily increases, what clearly indicates that the OBJ car is leaving its current lane in both manoeuvres.

Although the manoeuvre is clearly identified before it completes in both considered sequences, it is desired to predict it *further* in advance. Looking at the evolution of lateral offset and velocity in Figure 4.6, we could argue that the detection of the manoeuvre can be performed earlier (i.e., when the lateral offset and lateral velocity starts to increase more dramatically and consistently for *several* time-steps). This is one of the basic pieces of evidence that motivates the development of the dynamic version of the static-OOBN model [20].

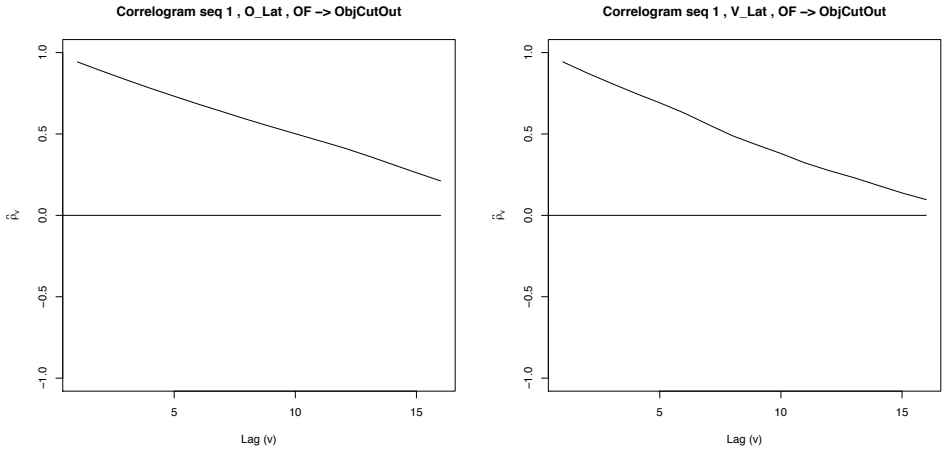


Figure 4.7: Correlograms for lateral velocity and offset. The x-axis represents the lag v or time difference, and the y-axis represents the sample autocorrelation coefficient of lag v, denoted by $\hat{\rho}_v$, i.e., the correlation between variables at times t and $t+v$ (see Section 3 for more details).

In any case, let us note, that the prediction horizon for manoeuvre recognition is going to be limited in order to avoid false positives (i.e. recognizing a lane change manoeuvre when the driver was just performing some random lateral movement). In case of a false positive, such as an erroneously OBJ-CutIn for instance, the adaptive cruise control would react with an unnecessary break, so they ought to be avoided. It is then required to find a good balance between the rate of false positives and false negatives.

An additional piece of evidence that motivates the use of dynamic models appears when we look at the sample correlograms for lateral velocity and offset (that is, the correlation of the data with lagged values of themselves) plotted in Figure 4.7. There we can see a strong temporal dependency (the correlation coefficients are high) for the two variables when we consider a *small* number of time-steps between the temporal observations. Hence, the use of a dynamic model to make predictions about the evolution of the lateral velocity and lateral offset in a near future seems to be reasonable according to this analysis. Moreover, our main assumption is that the use of dynamic Bayesian

network models (see Section 3) will allow us to build a more accurate and reliable system for manoeuvre recognition.

As explained in Section 3, the dynamic extension involves copies of the static OOBN for different number of time steps in the time window. Figure 4.8 shows an example for the LE hypothesis, where the nodes for O_LAT_REAL and V_LAT_REAL are temporal clones defining the belief state between consecutive time steps, and hence creating a first order Markov process. This model has been proposed in [20] by some of the AMIDST partners. The first order assumption is a standard assumption in this kind of models and helps to simplify the posterior inference and learning process. Fortunately, if we build a partial correlogram as the ones plotted in Figure 4.9, we can see that this assumption seems reasonable, because the influence of the lateral offset at time $t-1$ on the lateral offset at time $t+1$ given the lateral offset at time t is close to null (i.e., the value at $v=2$ for the lag is close to null in Figure 4.9). Note that the same reasoning applies also to the lateral velocity.

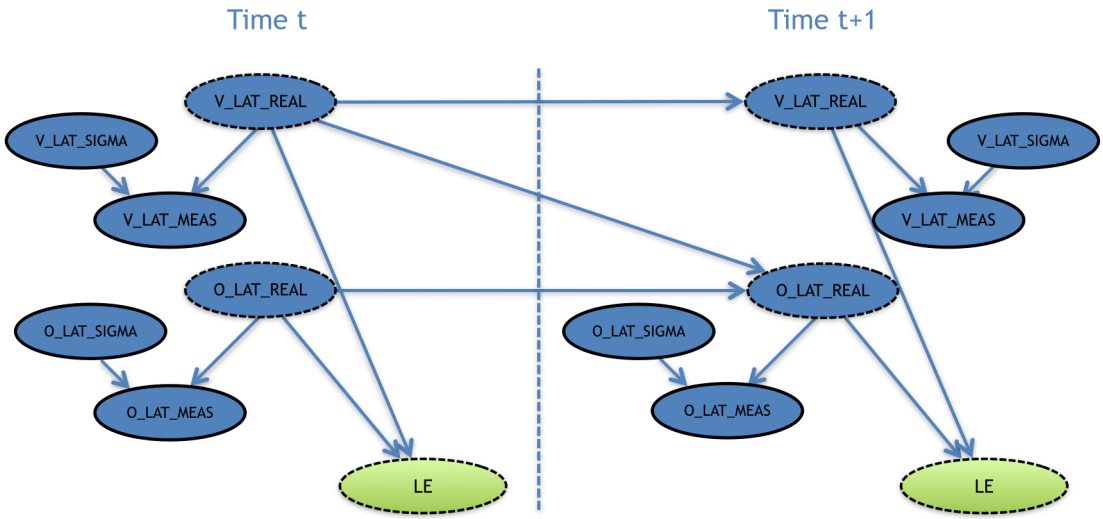


Figure 4.8: Daimler dynamic OOBN fragment for LE hypothesis.

As described in [20], the proposed DBN aims to incorporate the trend of change for the real values, where their physical relations are represented as causal dependencies between the time-steps dt. For instance, in Figure 4.8, the transition function of O_LAT_REAL at time t, denoted as $O(t)$, is modelled as a Gaussian distribution. Its mean is affected by $O(t-1)$, and by V_LAT_REAL at time $t-1$, denoted as $V(t-1)$. Therefore, we have:

$$O(t) = O(t-1) + V(t-1)dt + \epsilon \tag{4.1}$$

where ϵ denotes a white noise $\mathcal{N}(0, \sigma^2)$ which is assumed to be small. In order to cor-

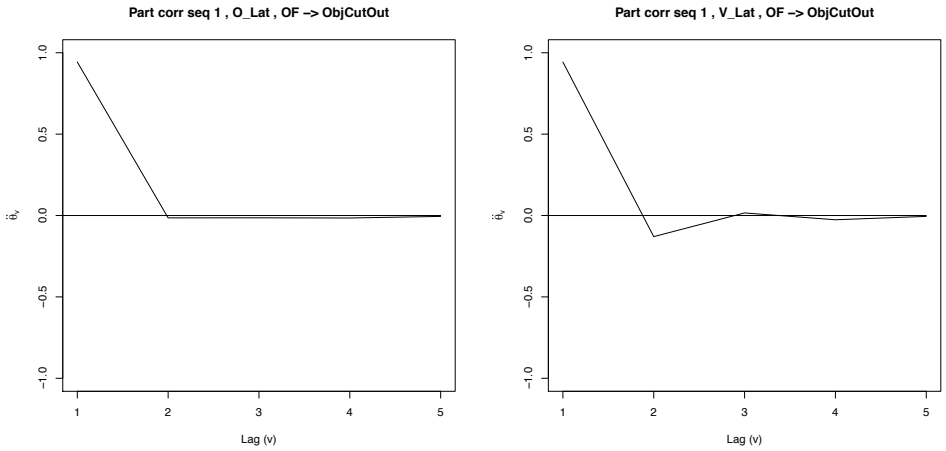


Figure 4.9: Partial correlograms for lateral velocity and offset. The x-axis represents the lag v or time difference, and the y-axis represents the partial autocorrelation coefficient of lag v, denoted by $\ddot{\theta}_v$, i.e., the correlation between variables at times t and $t+v$ after having removed the common linear effect of the data in between (see Section 3 for more details).

roborate the validity of this distributional assumption, we also analysed the hypothesis $O(t) - O(t-1) = \Delta O = V(t-1)dt + \epsilon$ on our data. Figure 4.10 shows the time and bivariate contour plots for $V(t-1)$ and ΔO for a randomly selected sequence (i.e., Seq 7) of an OBJ-CutOut manoeuvre. Both plots clearly prove that the assumption of a linear relationship with Gaussian noise might not be very far from reality.

The avid reader may have also noticed that these approaches do not take acceleration into account, i.e., they assume that velocity is constant (plus some white noise) at all times. From time plots corresponding to velocity in Figure 4.6, we can easily observe that this is an unrealistic assumption. A more detailed analysis about the possible future inclusion of acceleration in the dynamic models is included below in Section 4.1.3, where future models are discussed.

Finally, Figure 4.11 presents a rough overview of the final structure of the dynamic OOBN[3]. It illustrates how the temporal connection is only made on the top nodes (i.e., S_REAL) involving the situation-features in consecutive time steps[4]. The final event or manoeuvre prediction is then determined by the combination of all possible hypotheses ($Hypothesis_1, \ldots, Hypothesis_n$). There exist hypotheses at the top level that correspond to each of the sensor measurements, and then other different layers of hypotheses that combine the different outcomes till the final event, forming a polytree structure.

[3]Full details can not be provided for confidentiality reasons, but will be included in the confidential Deliverable 6.1.

[4]Although for the sake of clarity in the representation the sensor measurement structure is only shown for S_REAL_1, this is identical for all sensor variables.

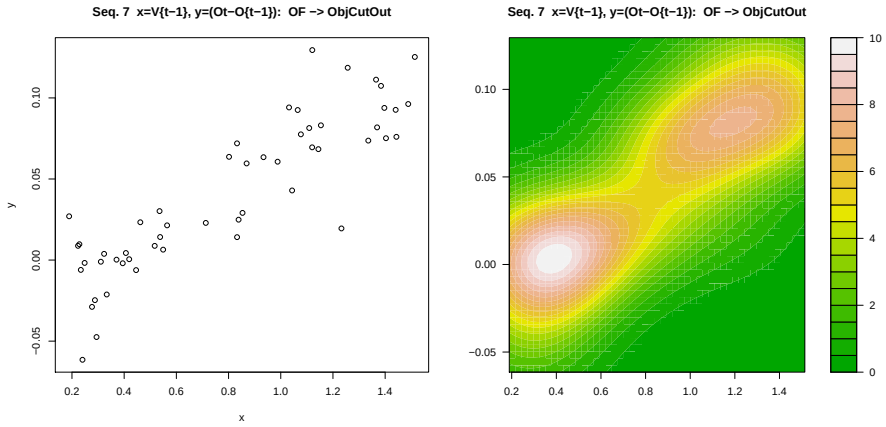


Figure 4.10: Time and contour plots attesting the linear correlation between $V(t-1)$ vs $O(t) - O(t-1)$.

Similarly to the static OOBN, if we modelled V_LAT_REAL and O_LAT_REAL as continuous variables, the conditional probability distribution of the LE hypotheses given the V_LAT_REAL and O_LAT_REAL variables would not fall in the conditional linear Gaussian family [2], because we would have discrete children with continuous parents. Once again, the pursued approach to deal with this problem is the discretization of the V_LAT_REAL and O_LAT_REAL variables. Hence, they are represented with a green color in Figure 4.11 and in all the more generic versions of it that will be included in Section 5.1.1. Since the remaining continuous variables S_MEAS and S_SIGMA are always observed, all the inferences can be implemented only over discrete potentials.

4.1.2 Earlier prediction of the need for a lane change based on relative dynamics

Earlier prediction of manoeuvre intentions could be achieved before any development of the trend for lateral evidence has been observed. A first indication of possible lane change intention could be detected through the relative dynamics between one vehicle (e.g., EGO) and the vehicles in front of it on the same lane. For instance, if a vehicle is driving slowly in front of the EGO vehicle on the same lane, this may indicate a need for a lane change. In fact, in order to continue its safe driving, the EGO vehicle should either break and reduce its speed to the speed of the vehicle in front, or alternatively, change to the adjacent lane, if the neighbour lane is free and no other vehicle is approaching with a higher speed.

A continued safe manoeuvre (of type “lane follow” or “lane change”) is modelled by estimating the time to collision (TTC) with a vehicle in front (on the same lane) or eventually with an approaching vehicle (on the neighbour lane). For a safe manoeuvre, TTC should be higher than 1 second, either when the EGO vehicle wants to change to the

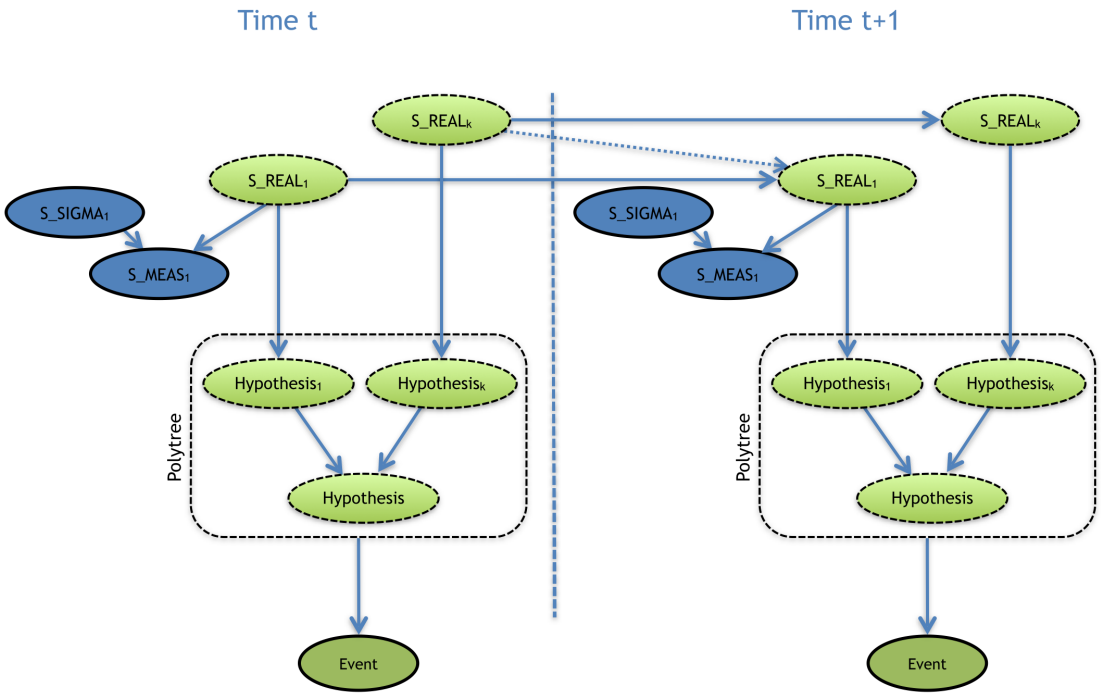


Figure 4.11: Daimler dynamic OOBN model with several hypotheses. Although for the sake of clarity in the representation the sensor measurement structure (i.e. S_MEAS and S_SIGMA) is only shown for S_REAL$_1$, this is identical for all sensor variables. S_REAL variables are plotted in green color to highlight that they are discretized, because otherwise we would have a model with continuous parents and discrete children.

neighbour lane or needs to break to ensure safe driving on the same lane. We assume that using this approach will further enhance the prediction horizon for manoeuvre recognition (up to 5 seconds).

Similarly to the models described in Figure 4.8, the dynamic OOBN model, shown in Figure 4.12, was defined with the hypothesis "relative dynamics" (REL_DYN) [22]. Due to confidentiality reasons, we can not give at this stage of the project further details about this model (full details will be given in the confidential Deliverable 6.1.).

4.1.3 Discussion and future models

The above included proposals are preliminary models designed in line with the expert knowledge facilitated by Daimler. They all try to balance a good level of expressiveness and efficiency during inference. However, in any case, they should only be seen as first proposals that might be modified/adapted in the future if they do not meet the efficiency and accuracy targets specified in the requirements [3].

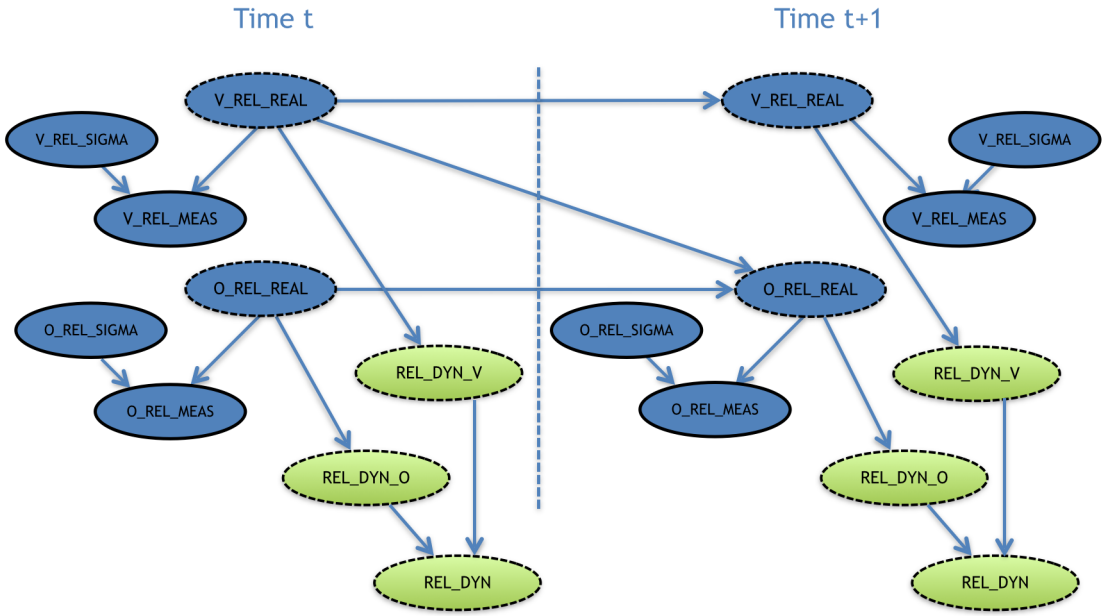


Figure 4.12: Daimler dynamic OOBN model with relative dynamics [22]. V_REL_REAL and O_REL_REAL refer to the relative velocity and relative distance between the EGO and the car in front, respectively. The other variables with _MEAS and _SIGMA suffixes refer to the measured values and the uncertainty of the measurements, respectively, as commented in the previous section for other sensor readings.

For application scenario 1 (see Section 4.1.1), we envision, for example, the possibility of considering not only the temporal dependences between consecutive time steps for the sensor measurements but also, or instead, consider the temporal links between hypotheses at consecutive time-steps (i.e., adding in Figure 4.11 temporal links between hypotheses nodes at different time-steps). It seems natural, for instance, that the probability of the EGO lateral evidence would be higher given its lateral evidence at the previous time-step. The same applies for the remaining hypotheses, such as TRAJ, OCCGRID, and the final hypothesis or event that identifies the type of manoeuvre. However, adding these type of dependences would greatly increase the complexity of inference in the resulting dynamic OOBN, because the total number of dependencies will be much higher over time. As described in D1.2 [3], models in this case are subject to very strong constraints in terms of computational efficiency, so we should be very cautions at this respect.

Another possible extension is to consider an extra hidden variable to model the acceleration in the dynamic OOBN model. We believe that assuming that lateral velocity varies according to a constant acceleration can lead to inaccuracies. Figure 4.13 (a) shows the behaviour of lateral velocity (V_Lat) for a randomly selected sequence which corresponds

to an EGO-CutIn manoeuvre. At the beginning of the sequence, acceleration is close to zero, however, approximately between time-steps 30 and 50, a higher acceleration value should be taken into account. The contour plot on Figure 4.13 (b) shows a large density of points around lower and higher values of V where the acceleration is constant, but just isolated points in between. Hence, we think that the dynamic model could benefit from an extra hidden variable to represent acceleration (A_Lat) as displayed in Figure 4.14. The following equation would be then considered for velocity at consecutive time steps:

$$V(t) = V(t-1) + A(t-1)dt + \epsilon \tag{4.2}$$

where $A(t-1)$ denotes the acceleration at time step $t-1$ and ϵ is a white noise. In this model, acceleration is assumed to be almost constant between two consecutive time steps, i.e., $A(t) = A(t-1) + \epsilon$.

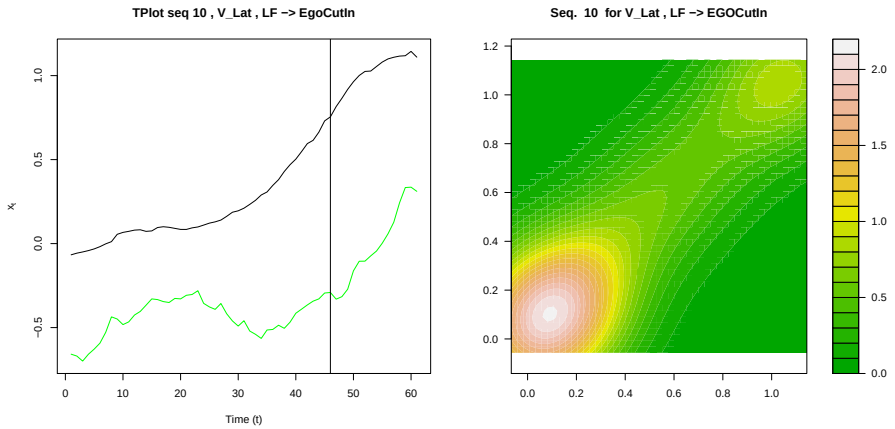


Figure 4.13: Time and contour plots for $V(t)$ versus $V(t-1)$ showing that modelling the acceleration should be considered to capture the variations in the data.

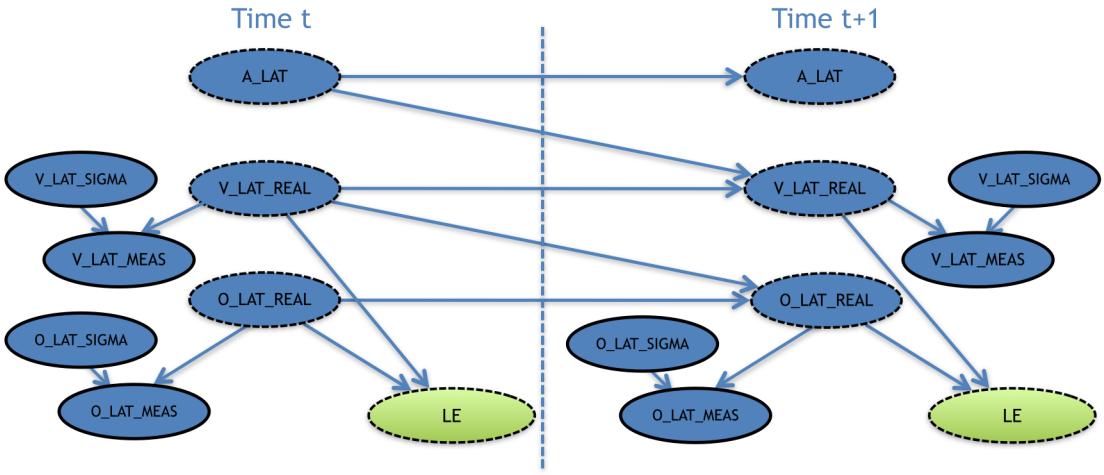


Figure 4.14: Daimler dynamic OOBN fragment for the LE hypothesis with a hidden node for acceleration (A_Lat).

4.2 Cajamar Models

There are two tasks to be solved for Cajamar's Use Case [1]. The main one is the estimation of the *probability of default*, defined as the probability that a customer will end up as defaulter within the next two years. The second task consists in obtaining good customer profiles in terms of risk, so that marketing campaigns can be specifically targeted to these low risk customers.

4.2.1 Predicting probability of default

In any commercial bank, every time a customer applies for a loan, the bank experts evaluate the customer's risk profile before making their decision.

At Cajamar, this risk evaluation protocol is addressed by evaluating whether the client is going to default or not within the following two years. When a client is labelled as defaulter, he/she will remain as defaulter in the database at least for one more year. Then, the client could become non-defaulter again when his/her debts has been appropriately paid during the last year.

The problem of predicting the risk of default is currently solved at Cajamar by an automatic supervised classification model (in particular a logistic regression model). It takes information about the recent financial activity of a customer, as well as information about recent past payment behaviour provided by other Spanish financial institutions. Finally, using this collected information, the probability that the client will default during the following two years is computed.

The methodology currently employed does not assume any dependence structure among the variables, and current predictions are made using only 27 variables out of more than 1000 variables available. Updates in risk predictions are made on a monthly basis, whereas the predictive model is only updated after several years. These low update frequencies are partly due to limitations in the available commercial software and the computing resources.

Our objective is to *daily* update the risk of default for every bank customer. This daily evaluation will be performed using two data sets: *the model training data set* and *the model evaluation data set* (see [1]). How these data sets are generated gives us some insights into the nature of this risk prediction problem.

Next, we detail how these data sets are collected (see Figure 4.15 for a better understanding).

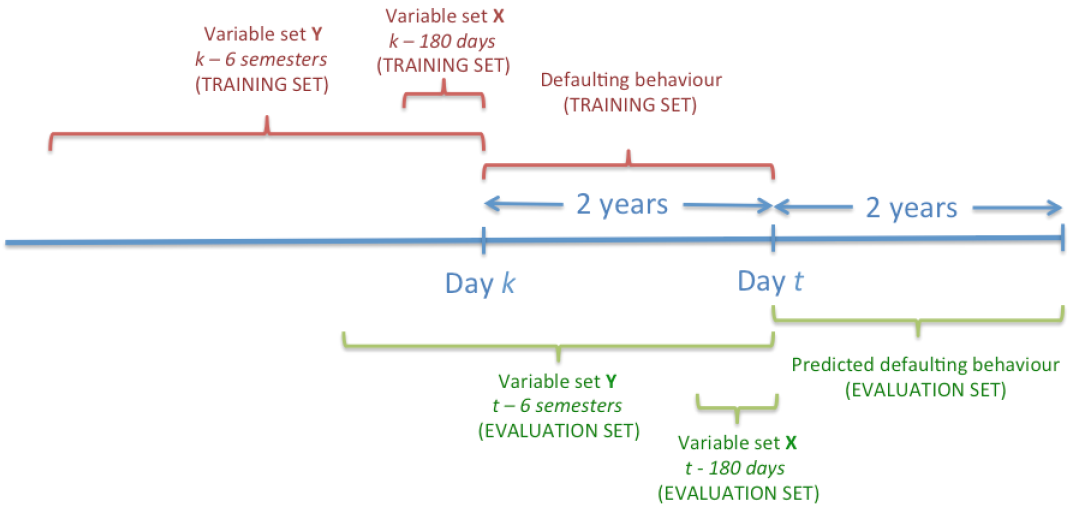


Figure 4.15: Time-line showing the generation of the evaluation (in green) and training (in red) data sets. t refers to the present time and k corresponds to time $t-2$ years. Both in the training and test data sets, there are two groups of variables, denoted as $\mathbf{X}$ and $\mathbf{Y}$, with different past information considered, 180 days back and 6 semesters back, respectively.

- **Model evaluation data set:** This data is created at time t and contains a record for every client to be evaluated. Note that information about the predicted defaulting behaviour is missing at time t and it will be obtained after performing inference on the model. Predictive variables refer here to the financial activity and payment behaviour of the customers in recent past as well as to their socio-demographic information which usually does not change over time.

  There are attributes, denoted as $\mathbf{X}$, for which information during the last 180 days

is considered. These attributes usually change daily for a customer, so they are encoded by introducing a set of variables for each attribute, one for each day back from the current time t. Hence, the financial activity of a customer is specified by a number of variables equal to 180 times the number of attributes. For others attributes, denoted as $\mathbf{Y}$, we are interested in information from the last 36 months grouped by semester. Therefore, similar to previous group of variables, 6 variables for each of these attributes will be considered. Finally, there are some other static variables, denoted as $\mathbf{Z}$, not included in Figure 4.15 as they are not indexed over time. The data set for the evaluation of customers is depicted in Table 4.1.

	Days			Semester			
Time t	$\mathbf{X}^{(t-180)}$	$\dots$	$\mathbf{X}^{(t-1)}$	$\mathbf{Y}^{(t-6)}$	$\dots$	$\mathbf{Y}^{(t-1)}$	$\mathbf{Z}$
Client_1							
$\vdots$							
Client_n							

Table 4.1: Evaluation data set at time t for all the clients. Three groups of attributes $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$ are distinguished according to the past information required. Current time is denoted as t.

Thus, the objective is to compute the probability of defaulting within the following two years of each record from the evaluation data set, and afterwards update the risk table in the system (see Table 4.2).

Time t	Risk of default
Client_1	r_1
$\vdots$	$\vdots$
Client_n	r_n

Table 4.2: Table containing the risk of defaulting for every client.

If, at some point, the probability of default of a customer rises above a predefined threshold, the bank may take preventive actions to reduce the chances of defaulting by this customer.

- **Model training data set:** This data set is also built at time t in a similar way as the evaluation data. It contains the same set of features as well as the target variable *Defaulter* but with information referred to time k instead (two years back). Note that, at time t, we have information of the *Defaulter* variable in the period of time from k to t. The data set for training/updating the model is depicted in Table 4.3.

  Table 4.3 shows the training data set where each record contains the values for all predictive variables and a class value labelled as *non-defaulter* only when there is no evidence of defaulting in the period from k to t (2 years).

	Days			Semester				
Time t	$\mathbf{X}^{(k-180)}$	...	$\mathbf{X}^{(k-1)}$	$\mathbf{Y}^{(k-6)}$	...	$\mathbf{Y}^{(k-1)}$	$\mathbf{Z}$	Defaulter$^{(k)}$
Client$_1$								
⋮								
Client$_n$								

Table 4.3: Training data set built at time t with $k = t - 2$ years. The notation for predictive variables is the same as in Table 4.1.

In what follows, we propose two modelling solutions for the risk prediction problem: static and dynamic.

Static model

In this first approach we do not explicitly consider some of the dynamics of the problem, meaning that we do not model that a customer can be non-defaulter and defaulter at different times (e.g., a customer can be creditworthy and, after some time, go bankrupt due to unemployment). Instead, we consider a static prediction model that predicts whether the client will default or not within the next 2 years based on the financial behaviour of the client over a recent past. This is in fact similar to the current Cajamar prediction system.

Figure 4.16 shows the general structure of the static model. The notation for predicting variables are the same as in previous section. Note that attributes in groups **X** and **Y** may be duplicated over the time creating the corresponding set of variables.

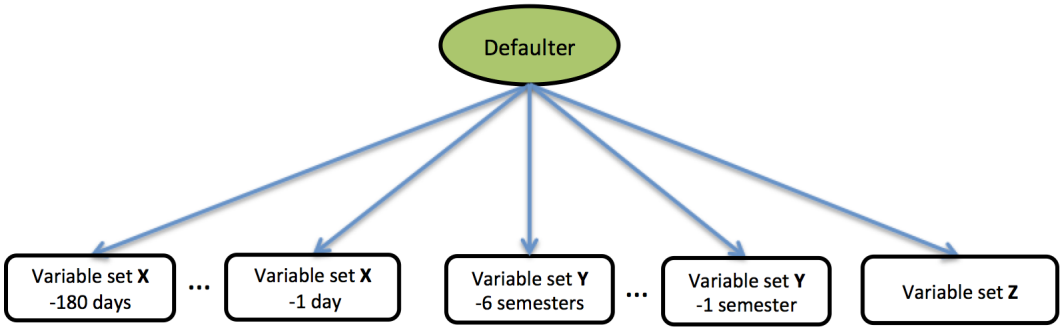


Figure 4.16: Global structure of the static model. Each white box represents a set of variables for a particular time. The green node on top is the class variable *Defaulter*, that represents the probability that a customer will default within the next two years.

According to expert knowledge from the bank, the predictive variables (mainly those within each white box) are expected to be related. Thus, in a first attempt we are assuming that only variables within each white box can be connected (e.g., according to

a tree structure).

Another issue to consider, is the possibility that one variable is a replica or contain similar information than another variable, something likely to be encountered in Cajamar data. Consequently, their joint contribution will not only make learning and inference computationally more expensive, but it may also lead to a worse performance. This justifies the need to explore and use suitable feature selection techniques as specified in Use Case 2 of Cajamar Requirement analysis [1].

It is also of major interest to analyse the type of probability distributions to use in the proposed model. Figure 4.17 shows the histograms for the values of one continuous predictive variable conditioned to defaulter (left curve) and non-defaulter (right curve) respectively. The resulting density curves represent a credible approximation using mixture of Gaussian distributions, which is the case for most of the remaining predictive variables. Note that, for defaulters, the values of this variable are overall lower compared to those corresponding to non-defaulters.

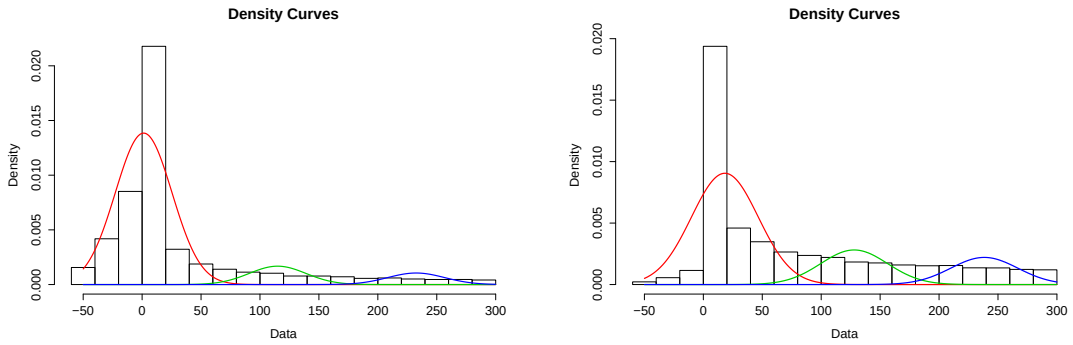


Figure 4.17: Mixture of Gaussian approximations of one predictive variable conditioned to the class variable values defaulter (left) and non-Defaulter (right).

There exist however discrete (non-ordinal) predictive attributes which impose a limitation in the structure of the conditional Gaussian network, since discrete attributes cannot have continuous parents, as pointed out in Section 3.2. However, this might not be a problem for the semi-naive Bayesian network to be considered, and it is certainly not a problem for naive Bayes. Nonetheless, if this limitation were found to be problematic, then other probability distributions families will be explored, such as Mixture of Truncated Basis Functions [11], which can cope with any generic model structure.

In summary, the process of building the static model and updating the customers' risk consists of the following steps:

1. Construct, every day, a single flat table containing the information for model training (see Table 4.3) using a set of SQL queries over the main relational data set. Note that, for instance, for the variables set $\mathbf{X}$, training data between two consec-

utive days t and $t+1$ only differ in the new data arriving at day $t+1$ and obviating the data 180 days back from day t. A similar idea is applied for the set $\mathbf{Y}$.

2. Update the existing BN classifier with the new data. If the underlying probability distributions belong to the exponential family, this step will be made incrementally by updating the sufficient statistics of the variables. Otherwise, the model must be learnt from scratch. Although relationships are more common between variables within each box, dependences among variables in different boxes could be also considered. This possibility is tackled in Section 4.2.3.

3. Construct a single flat table with the latest information about customers (evaluation data set as depicted in Table 4.1) in order to compute the latest risk profiles.

4. Update the risk in Table 4.2 for every customer by propagating each record from the evaluation data set on the static BN classifier obtained in Step 2.

Dynamic model

In this second approach, we will consider the dynamic structure of the problem, since the behaviour of the customers evolves over time (e.g., the account balance is continuously changing from one month to another, the incomes, etc.) as well as their class labels either defaulters or non-defaulters (e.g., customers can be creditworthy and, after some time, go bankrupt because they have lost their job).

Figure 4.18 represents the global idea of the proposed dynamic model. Each of the defaulter variable at different times represents whether a customer will default in the following 2 years from t.

The previous model can be compactly represented by a DBN made of components as the one displayed in Figure 4.19. Next, we detail the meaning of each node.

Defaulter_{t+1} represents the class variable at time slice $t+1$, i.e., being defaulter or non-defaulter within the next two years, and depends on class variable Defaulter_t at time t considering a first Markov order. Each feature variable in $\mathbf{X}_t$ with a clear dynamic component at time t, is linked to the same variable in $\mathbf{X}_{t+1}$ at time $t+1$. Although this is a reasonable assumption for most of the variables, this first Markov order relationship however might prove insufficient for some of the variables. Figure 4.20 shows a correlogram (left) and a partial correlogram (right) for a continuous predictive variable (see Section 3). The partial correlogram drops to almost zero for a lag equal to 2, making a first Markov order assumption reasonable. However, a higher order could also arise from data, meaning that even earlier samples still could have an influence on the current sample given the previous one.

To this end, variables in $\bar{\mathbf{X}}_{t-1}$ and $\bar{\mathbf{X}}_t$ represent memory variables at time t and $t+1$ respectively. The inclusion of this type of variables might be necessary when we have evidence that first order Markov relationships do not hold and we need to account for

information coming from the past. Memory variables representing averaged or accumulated values both during the last 180 days, could, for instance, capture these past dependences. Another reason to include these memory variables is that, even though they are computed from others, they provide information that might be disperse in the data and hardly can be captured by other variables as a whole. Moreover, the use of memory variables avoids building complex dynamic models with high Markov orders. In principle, memory variables in $\bar{\mathbf{X}}_{t-1}$ and $\bar{\mathbf{X}}_t$ are not connected over time, as their dynamic component is partially captured through the dynamic variables $\mathbf{X}_t$ and $\mathbf{X}_{t+1}$.

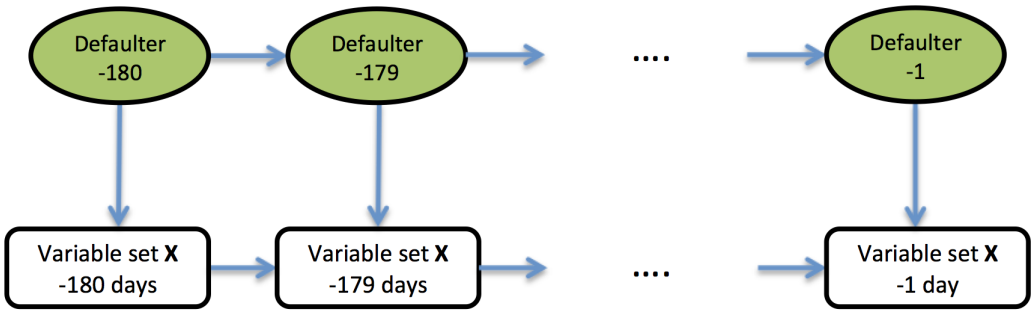


Figure 4.18: Global structure of the dynamic model. Each white box represents a set of variables measured during the same day. The variables within a box can be connected as well as variables between two consecutive days representing the same attribute. For the ease of presentation, variables sets $\mathbf{Y}$ and $\mathbf{Z}$ are omitted.

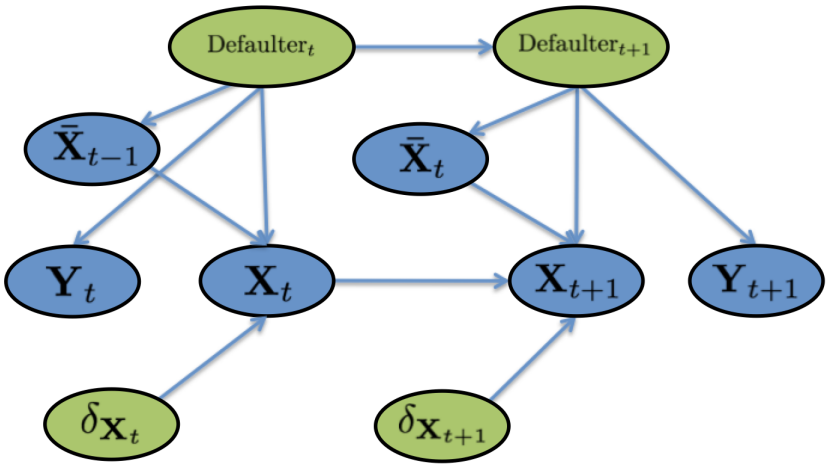


Figure 4.19: Basic components of the DBN structure.

Another node included into the network in Figure 4.19 is the indicator variable $\delta_{X_t} \in \delta_{\mathbf{X}_t}$.

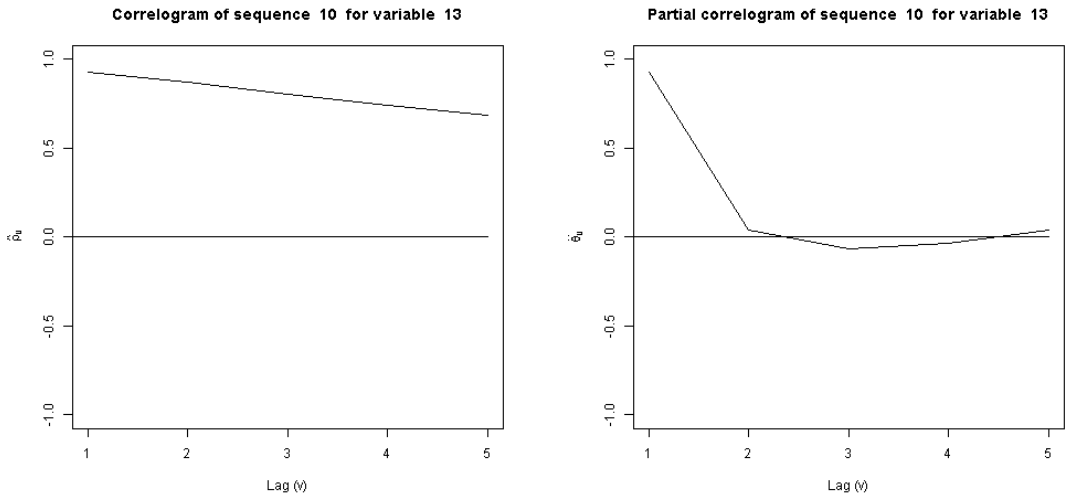


Figure 4.20: Correlogram and partial correlogram for a variable. The x-axis represents the lag v or time difference, and the y-axis represents the sample/partial autocorrelation coefficient of lag v, denoted $\hat{\rho}_v$ and $\hat{\delta}_v$, respectively (see Section 3 for more details).

This indicator variable is useful to model the situation in which a variable X_t has a large number of zeroes, which is frequently encountered in Cajamar data. This is the case for variables whose corresponding events do not occur every day. For instance, every day payments made by credit card or the historical monthly outstanding amount on the account, whose value can be equal to zero for a large number of days for most customers. This fact can seriously affect learning, because the real behavior of X_t for values different from 0 is not adequately captured by the learnt distribution. The solution we propose is to use the indicator variable to condition the value of variable X_t to $\delta_{X_t} = 0$ and $\delta_{X_t} \neq 0$. The indicator variables are not linked through consecutive time steps, as they are only used to indicate which data we model in the corresponding variable X_t. To better illustrate this, Figure 4.21 (left) displays the histogram of a variable including all values, and Figure 4.21 (right) displays the histogram of the same variable, but excluding all zero values. Note that the fitted density when zeros are discarded is extremely more representative than including zeros. This result undoubtedly justifies the use of the indicator variable δ_{X_t}.

Finally, there exist some variables for which, even though some dependences over time would be expected, no dynamic behaviour is encountered. For instance, this is the case for variables[5] whose correlograms, shown in Figure 4.22, indicate values very close to zero for all lags. These variables are hence not linked through consecutive time steps in the dynamic model, and are represented by $\mathbf{Y}_t$ and $\mathbf{Y}_{t+1}$. Moreover, static variables, denoted as $\mathbf{Z}$ in Figure 4.16, are also included in these groups with the only problem

[5]For confidentiality reasons, the names of the variables cannot be revealed.

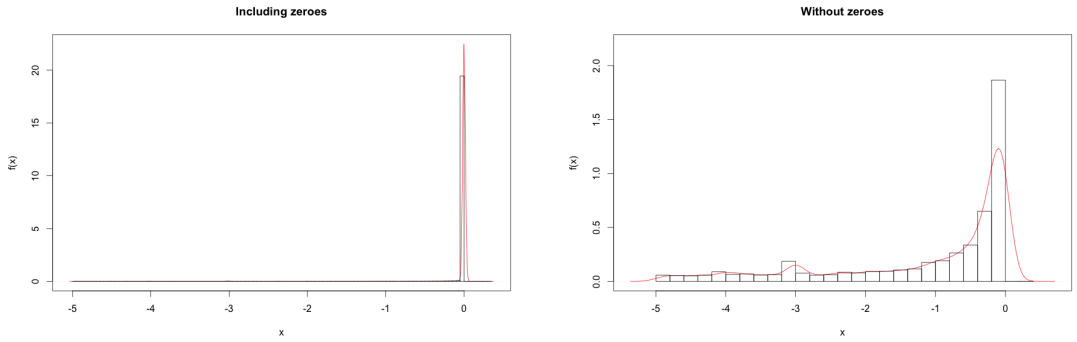


Figure 4.21: Data histograms and fitted densities for the same predictive variable including zeroes (left) and discarding them (right). In both cases, outliers have been removed for a clearer representation.

that their distributions must be replicated at times t and $t+1$.

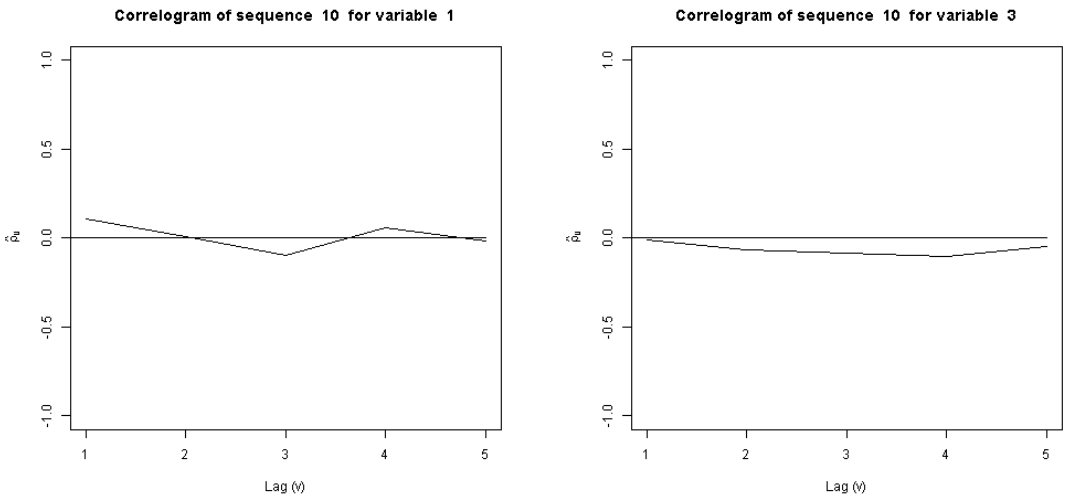


Figure 4.22: Example of correllograms for two predictive variables. The x-axis represents the lag v or time difference, and the y-axis represents the sample autocorrelation coefficient of lag v, denoted $\hat{\rho}_v$ (see Section 3 for more details).

In summary, although the dynamic model is different from the static one, the whole process of building the model from the original relational database and updating the customers risk is exactly the same as for the static case.

4.2.2 Low risk profile extraction

The marketing department at Cajamar periodically launches marketing campaigns for the recruitment of new products by existing customers (i.e., a new credit card, an insurance, etc.). The success of these campaigns depends greatly on the client group to which the campaign is targeted. It is also crucial to reduce as much as possible unnecessary expenses focused on non-potential customers.

For this purpose, the marketing group will proceed as follows. Using their own marketing models, and in collaboration with the risk department, they will provide some restrictions over a subset of variables, denoted as $\mathbf{Z}$, to be satisfied by the targeted clients. For instance, they all must be older than 60 or the annual incomes must be higher than $30,000$ euros.

Additionally, they will select a different subset of predictive variables $\mathbf{Y}$ also considered relevant when computing the final profile of the specific campaign. Mainly sociodemographic variables will take part in this set, for example, the age of a client is a relevant predictor when designing campaigns to attract customers for a death insurance.

If we denote the *Defaulter* variable as D, the objective is to compute the assignment $\mathbf{Y} = \mathbf{y}^*$ that has maximum a posteriori probability given the evidence $D = \texttt{no}$, and also given the restrictions to be satisfied by the clients, $\mathbf{Z} = \mathbf{z}$. These restrictions will be considered as soft evidence during the inference process.

Thus, the MAP operation is defined as follows:

$$\mathbf{y}^* = \underset{\mathbf{y} \in \Omega_{\mathbf{Y}}}{\arg\max}\ P(\mathbf{Y} = \mathbf{y} \mid D = \texttt{no}, \mathbf{Z} = \mathbf{z})\,. \tag{4.3}$$

Note that the MAP operation will be performed over a model containing the whole set of variables (i.e. the variables in $\mathbf{Y}$ and $\mathbf{Z}$, and the rest non-relevant variables.) There is an open issue when performing MAP inference within continuous domains, which is the definition of the intervals of the continuous variables to conform the profiles. This issue is further considered in the discussion section.

Let see a running example to clarify the process. Let us consider a simple case in which three predictors $\mathbf{Y}$ are considered relevant for a specific campaign: *Age*, *Marital Status* and *Gender* which might be relevant to identify target customers for a *death insurance* campaign. Let consider also the restriction $\mathbf{Z} = \{Age > 50\}$ coming from the marketing department, which encodes the initial group of customers that, according to the knowledge of the marketing group, are potential buyers of this product. The next step is to further refine this group of customers and identify the profile of customers which are most solvent (i.e. less likely to default). For this task we perform the computation in Eq. 4.3, this may results in the following most probable assignment: clients in the range *[60-70] years*, *single status* and *male*. Hence, this profile identifies the most solvent customers who might be interested in purchasing a *dead insurance* product. We must

say that the above procedure will be also used to get the k best assignments or in other words the best set of profiles.

As pointed out before, mainly socio-demographic variables will determine the customer profiles used in the campaigns. These variables are mostly static, as they do not change frequently over time (e.g., marital status, gender, type of job, ...). However, to enhance the analysis, a number of variables changing over time will be included. Note that this does not mean the use of a dynamic model.

The structure for the proposed static model in the profile extraction is the same as for predicting the risk of defaulting (see Fig. 4.16). However, there will be a considerable reduction in the number of features included, since the profile extraction task does not require information about all variables included in the risk prediction, i.e., only a few variables will be part of the profiles. With this reduction, we also make the problem computationally more feasible.

In this first attempt to face the profile extraction problem, only variables within each white box (see Fig. 4.16) are allowed to be connected conforming a tree structure. In Section 4.2.3 we will consider more dependences among white boxes, and also having a general Bayesian network structure more than one specifically designed for classification.

In any case, what is desirable is to avoid using a naïve Bayes structure for this task. The reason is that, once the *Defaulter* variable is evidenced to `no`, the features become independent and the analysis would be poorer (MAP in this case would correspond to the maximum probability value for each variable individually).

A possible extension of the previous static model presented above including dynamic Bayesian networks will be considered in next section.

4.2.3 Discussion and future models

The static and dynamic models presented above for the two tasks represent a first attempt to address the Cajamar use case. In this section, we will basically discuss the potential modifications and/or extensions that may occur in the subsequent phases of the AMIDST project to better meet Cajamar requirements presented in [1] or to face, for instance, other unexpected events. This discussion is also essential as we are still considering expert advice that could slightly modify some of proposals as the project moves forward.

The static model shown in Figure 4.16 for predicting the probability of default is assumed to have only dependences among variables within each group of variables (white boxes). However, according to expert knowledge, no limitations should be imposed to the model structure among the predicting variables. Figure 4.23 shows the modified structure of the static model when dependences among predicting variables are allowed, meaning that, variables within the dashed blue box could be connected as well. Note that, in this case, complexity grows considerably but, it is worth considering this approach to

capture dependences that intuitively are not expected and, therefore, the analysis will be enriched.

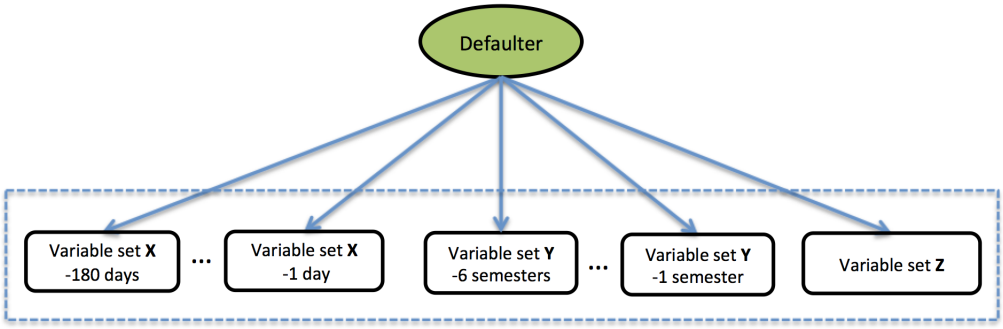


Figure 4.23: Global structure of the static model allowing dependences among any variable within the dashed blue box. Each white box represents a set of variables for a particular time. The green node on top is the class variable *Defaulter* that represents the probability that a customer will default within the next two years.

In general, the model structure in Figure 4.23 can not be modeled by CG distributions is discrete children happen to have continuous parents (see Section 3.2). If so, other probability distributions families will be explored. Mainly, they consist in translating the probability distribution into another such that learning and inference become feasible. These approaches include discretization or more complicated families based on Mixture of Truncated Basis Functions [11], which can cope with any generic model structure.

Another issue to be considered is about complexity in the profile extraction task explained in Section 4.2.2. It is well known that abductive inference over BNs and, in particular, the computation of the most probable explanations (MPEs) is an NP-hard problem [23]. Thus, the model presented in Figure 4.23, which is valid for the profile extraction as well, could be simplified if necessary by manually reducing the number of predictors in such a way that computational burden is adapted to the needs. In fact, this process is tightly connected to the expert knowledge provided by the marketing group, and this refinement should not encounter major difficulties. Moreover, we could consider having a general Bayesian network structure instead of one specifically designed for classification. In this way, dependences among the features included in the profiles might be modeled without the restrictions imposed in classifier structures.

The model for predicting the risk of default is assumed to be learnt every day. However, if required, this assumption can be relaxed and the learning process could be carried out less frequently. The results hardly will be affected as one-day data have little impact in comparison with the historical data captured in the current model so far. Note that, even if the model is not updated, the risk predictions are computed every day using the latest model available so far.

An extension of the static model proposed in Section 4.2.2 for the profile extraction would

involve not only considering those variables evolving over time, but also their tendency by using a dynamic Bayesian network. Initially, the model for the profile extraction will be the same as the one depicted in Figure 4.19 but with a considerable reduction in the number of predictors as for the static case. With this model, the MPE solution would capture the tendency of change within consecutive customer profiles.

Finally, there is an open issue when performing MAP inference within continuous domains, which is the definition of the intervals of the continuous variables to conform the profiles. A first approach could be to discretize the continuous variables and then use the intervals are used to conform the profiles.

4.3 Verdande models

As has been pointed out in Deliverable D1.2 [1], three main tasks have to be addressed for Verdande Technology use case:

1. **Automatic detection of drill string vibrations and abnormal torque states**: which aims to better diagnose the shape of the wellbore and the state of the equipment, make better decisions on how to manage the well, and thereby reduce the non-productive time. To this end, a probabilistic graphical model for erratic torque monitoring and detection will be used.

2. **Semi-automatic labelling**: Given unlabelled data streams collected over time from typical drilling conditions, semi-automatic labelling aims to compute a normality score for each considered drilling situation, then label it as either "normal" or "abnormal". As for the previous task, a probabilistic graphical model will be used, taking into account the temporal dynamics of the drilling process and continuously adapting to changes in the incoming streaming data.

3. **Automatic formation detection**: which aims to predict in real time the formation tops from the MWD (measurements while drilling) data using a probabilistic graphical model. Once again, this should be performed taking the temporal dynamics of the drilling process into account. The automatic formation detection is vital for dealing with several issues such as hole instability and vibrations, and also important for reducing the costs and the overall non-productive time.

The following three subsections detail the designed models for the different tasks.

4.3.1 Detection of drill string vibrations and abnormal torque states

The hole drill string, including the bottom hole assembly and the bit, is designed to transfer as much energy as possible from the rotary table to break rock at the bit. When this energy transfer is inefficient, the energy goes into shaking the drill string.

This involves drilling slower, a higher risk to break equipment and a lower hole quality. Automatic detection of drill string vibrations can help the drilling crew to better diagnose both the condition of the equipments and also the hole itself. Drillstring vibrations are seen as an erratic torque signal on the realtime data. We have therefore focused on detection of erratic torque.

Two time-windows of torque-readings are given in Figure 4.24; the uppermost plot traces the torque over a period of 25.000 observations, the other is zoomed in to cover only 500 observations. It is evident that the torque varies over time, and often drops to zero (in particular when the drill-bit is not rotating). Furthermore, we can see from the zoomed-in trace that the variance changes over time; this time-window commences with rapidly varying torque, before the variation is reduced. The goal is to extract passages where high variance in the observed torque cannot be explained by other drilling parameters. We will begin the following discussion by considering how to model the torque as a random variable developing over time, without thinking about vibration and the variable's relationship to other drilling parameters. Thereafter, a full model will be proposed.

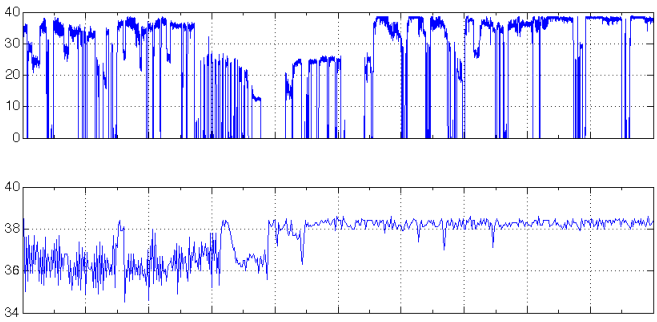


Figure 4.24: Measurements of the torque over two time-windows.

The sample correlogram and sample partial correlogram of the torque can be seen in Figure 4.25. Note that only the data from drilling (i.e., where the torque is positive and "moving naturally") have been included in these calculations. It is clear that a significant autocorrelation exists, also at high lags. The figure goes up to lag $v = 20$, where the correlation is still above 0.6. Furthermore, the partial correlogram shows very significant contributions at lags 1 and 2, and also significant effects at lags 5 –10. A model capturing the time dynamic is therefore essential.

To capture the dynamics of the torque variable, we propose a KF model (see Section 3.3.2), with a sufficiently "rich" latent variable space (that is, more than a single one-dimensional latent variable may be required to capture the partial autocorrelations in the data).

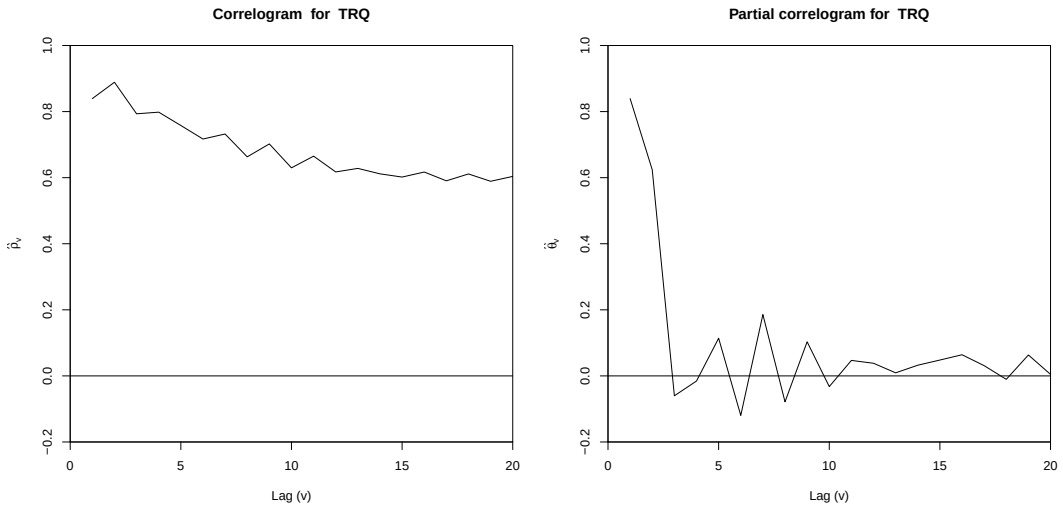


Figure 4.25: Correlogram and partial correlogram of the torque while drilling.

Next, we consider how to model a torque sequence that also includes abnormal torque measurements (those that are corrupted by vibrations). Our working assumption is that an abnormal torque signal is the result of a string vibration with frequency higher than the sampling frequency of the data. Thus, it will be observed as a white-noise signal superimposed on top of the "true" underlying torque signal. An abnormal torque regime will typically last for some unknown time significantly longer than one time step, thus we must create a model with the ability to choose a state that fits the observations best (normal or abnormal), and with a tendency to remain in a chosen state over time. This can be obtained by an SKF (see Figure 3.6), where the switching variable (connected over time) is used to show which state one is in at a given time (normal or abnormal) and the transition model for the variable is used to capture the state remaining for some time.

Preliminary results using this model class, but without learning the parameters properly, and relying on an inference scheme that does not scale to the data-sizes we have available in the AMIDST project, are shown in Figure 4.26. The same 500 data points shown in Figure 4.24 are once again used, and the positions the system detects as having "abnormal torque" are highlighted in red along the x-axis. These annotations correlate well with what can be obtained by simple inspection of the time series with the human eye.

Still, some undesired annotations remain. For instance, the two areas with small dips in the observed torque towards the end of the sequence can be explained by specific drilling activities going on, and should not be marked as abnormal. To capture this, the context of the torque observation must be included, which is what we turn to next.

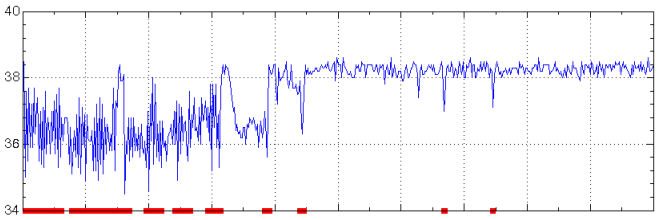


Figure 4.26: Values of torque in normal (blue line) and abnormal (red bottom marks) states resulting from an inference process performed on a SKF.

The parameters in drilling can to some extent be seen as control-response pairs. The control parameters are the flow, the rotation speed and the position of the brake handle under drilling. The position of the break handle is usually not recorded, but the response of all this is the hook load and thereby the weight on the bit (WOB), the pressure at the top of the drill string, the torque (TRQ) and the rate of penetration (ROP). In general, the control parameters are adjusted to increase ROP as much as possible. In particular, ROP and TRQ are correlated.

To analytically examine the relationship between ROP and TRQ further, we plot the joint density of (ROP_t, TRQ_t) and the sample correlogram between ROP_t and TRQ_{t+k} in Figure 4.27. We notice that even if a strong correlation is not apparent in the density plot, it can be seen as significant from the correlogram. Hence, ROP could give some information about TRQ. Our intention here is to exploit this information to improve the accuracy of the detection of "abnormal torque" readings and avoid those false positives which appeared in Figure 4.26. In terms of modelling, we update the initial SKF by introducing ROP as an "input variable" shown in in Figure 4.28, which results in a so-called *input-output SKF* (IOSKF) model which will be used to detect abnormal TRQ states that cannot be explained by the observed ROP.

In the IOSKF model, the "input" at time t is the ROP at that point in time, the "output" is the (observed) TRQ readings[6], and we have two sets of latent variables: i) a discrete switching variable, which is used to determine which TRQ regime we are currently in, and ii) a latent vector of continuous variables used to capture and model the natural behaviour of the TRQ sequence over time. During inference, the posterior probability of the discrete latent variable at time t can be used to detect whether or not the system experiences abnormal TRQ (that is, not explained by the observed ROP) over time.

Further extensions to the model, where also other predictive variables are taken into account, will be considered at a later stage. The general model structure will, however,

[6]It is generally the case that the output variables in the input-output dynamic models correspond to the target variables and are hence not observed during inference [24]. In our case, however, the target variables correspond to a discrete state/hidden variable whereas the output variables will be observed during inference. See Section 3.3.1 for further details.

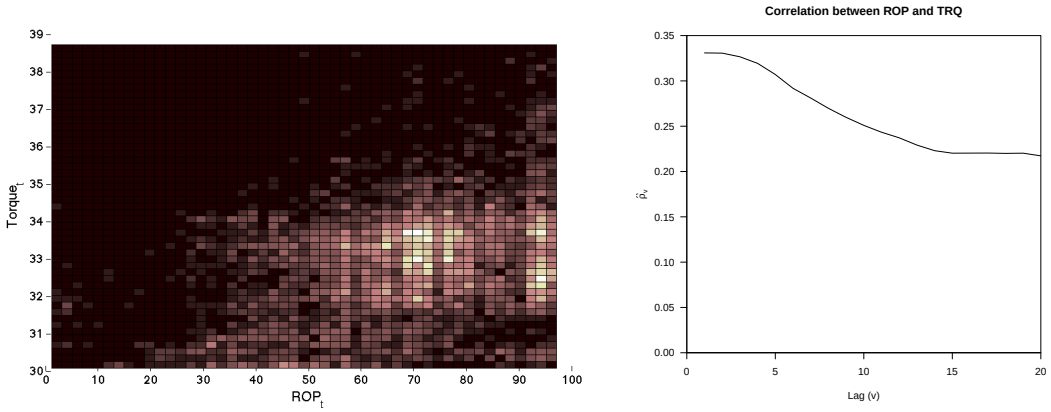


Figure 4.27: Left: joint density of (ROP_t, TRQ_t) and right: sample correlogram between ROP_t and TRQ_{t+k}.

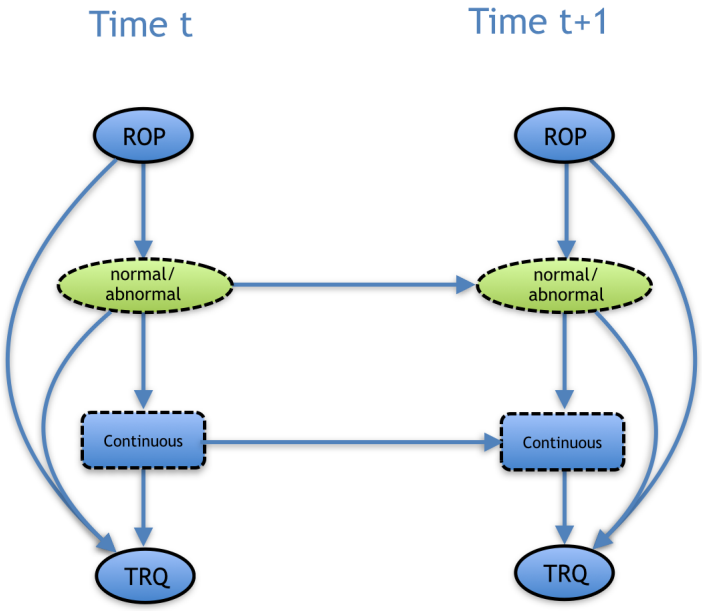


Figure 4.28: Input-output SKF structure used to detect abnormal TRQ states that cannot be explained by the observed ROP.

not be changed.

4.3.2 Semi-automatic labelling

The main contribution from DrillEdge during operation is that it enables the driller to see in real time if the current situation is "similar" to previous situations that lead to undesired circumstances. If this is the case, remedial actions can be found to avoid undesired events. At the core of the analysis are historical examples of noteworthy episodes, ranging from clearly undesired events, for instance getting stuck, to more subtle indications of evolving problems, such as abnormal readings of hook-load. Verdande Technology has already extensive datasets (made available to the project from its beginning), where such episodes are identified. However, the definitions of the episodes are not always crisp, and domain experts can at times disagree about what is happening at a certain time. This can potentially lead to inconsistencies in the dataset, complicating both the automatic learning of models as well as the comparison with previous situations. Additionally, it can confuse the driller to be presented with information that he/she does not relate to or agree with.

Manually labelling new data sequences, or adapting the existing labelling to new definitions, can be a very time-consuming activity, and Verdande Technology therefore want to start looking into techniques for semi-automatic labelling. This process is initiated by the system being fed with unlabelled data streams from typical drilling conditions. The system should adapt to this data, and recognise what is "normal", both with respect to auto-correlations of each variable over time, as well as the intricate relationships between variables. Simultaneously, sub-sequences that do not fit well with the normal data should be marked as "abnormal". The abnormal sub-sequences can thereafter be examined more closely by drilling experts. It is desired that the process should be able to take back information from this examination, and update the reasoning if the sequences marked as abnormal were not correctly labelled. The benefit of the automatic tagging is that more time can be spent on scrutinizing the "interesting" parts of the drilling logs, and less time on the "normal" parts. This implies the potential for further reduction in non-productive time. Specifically, the goal is to provide a "normality" score for historical logs to facilitate semi-automatic detection of abnormal situations. The calculations should be efficient enough to run in real time if so desired.

Drilling is performed in phases, where different activities are conducted in sequence. For instance, one can only drill the well (increase the length of the well-bore) after having transported the drill-bit to the bottom of the well (that is, performed a "tripping-in" activity). Similarly, a long drilling sequence is split up into smaller sequences, where each subsequence is initiated by a "connection" activity, where the length of the drill-string is increased. The division of the drilling log into subsequences containing only a single activity is done automatically by Verdande Technology's Drill Edge system. Therefore, the AMIDST system will be fed sequences where a single activity is conducted. The behaviour during each of these sequences is fairly consistent if the process runs smoothly, and the relationships between variables can be established (e.g., the amount of mud transported into the drill-string will, after stabilization, can be a strong indicative of the

amount of mud coming out after circulating through the well string).

Some of the variables in the data set are used to monitor quantities that are set externally by the drilling crew (examples include "weight-on-bit", "rotations-per-minute" and "mud-flow-in"). Other variables, like "torque", "rate-of-penetration" and "pressure" can be understood as (possibly delayed) responses to the input from the geology.

As said above, the real-time parameters in drilling can to some extent be divided into control and response variables, where the great unknown is the physical properties of the formation that is penetrated. The cause $\rightarrow$ response relationships are mostly unknown, but some qualitative understanding exists.

The model we propose for this task is similar to the one proposed in the previous application scenario . It is detailed in Figure 4.29 and has again an *input-ouput* Kalman filter structure with observed control variables on top and the observed response variables down. The main difference now is that the discrete variable is not temporally linked and does not try to capture any "physical" property of the process. This discrete variable is simply included to improve the modelling capacity over the observed response variables because it allows now to model mixture of Gaussians for these variables.

As commented above, this input-output SKF will be used to model the "normal behaviour" of the well. The "abnormal behaviour" will be detected using the following protocol. The normality-index for each observation at time t, denoted by $\mathbf{x}_t$, given the observations up until t, denoted by $\mathbf{x}_{1:t-1}$, will be calculated as the log conditional likelihood, $\ell_t := \log p\left(\mathbf{x}_t|\mathbf{x}_{1:t-1}\right)$, where p is the probability distribution induced by our input-output SKF. By monitoring ℓ_t over time, significant departures from its "typical behaviour" (i.e. ℓ_t will take high negative values) will be noted, and marked as potential irregularities.

4.3.3 Automatic formation detection

In many drilling operations, a precise recognition of a formation change can be vital for cost savings. For instance, it is important to cement the casing in the correct formation to manage the formation pressures, as well as correctly identifying the top and the bottom of an oil reservoir. Additionally, accurate knowledge of the formation allows the drilling crew to optimise drilling parameters and better diagnose the condition of the bit. Knowing the formation is also an important input in how to deal with symptoms such as improper hole-cleaning, hole-instability and vibration issues. A proper formation detection is therefore an important piece in the puzzle for reducing the overall non-productive time.

Before the well is drilled, a chart with the expected formation tops is available from the well plan. The chart is based on seismic data and drilling data from neighbouring wells and contains the best guesses on which depths the various formation tops are located. Since it is uncertain where the formation tops are, an uncertainty interval is associated with every formation top. This is called the lithology chart.

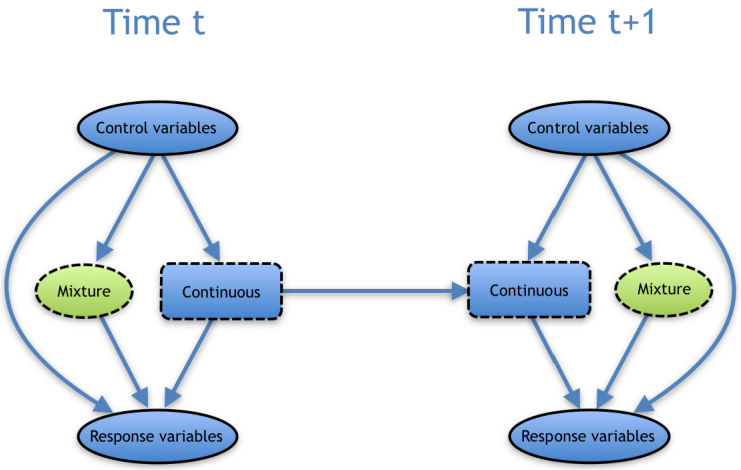


Figure 4.29: Input-output Kalman filter with Gaussian mixtures at the leaves used for semi-automatic labelling.

In practice, the prediction of the formation tops is refined by a human, who interprets *measurement while drilling* (MWD) data and aligns this with the lithology chart as the well is being drilled (see Figure 4.30).

The MWD data measures petrophysical properties of the rock such as gamma radiation, sonic speed and electrical resistivity. Often, shifts in these graphs happen when the formation is changing. We will therefore formulate the formation detection problem as change-point detection in a multivariate data stream. Indications of change-points (shifts) are to be compared with the prior belief to specifically locate the changes. The model described below is built to capture *instantaneous* changes in formation, and is not made to capture gradual changes.

The prior information, giving the expected locations (depths, in meters) of formation change, together with a degree of certainty (in terms of a standard deviation) is the point of departure for this model. Let us assume that the drilling crew, a priori, expects to see N changes in formation. This information may be erroneous, as any number of additional formation-changes (that were not expected in advance) may also occur in the ground, e.g., formation of sandstone can be a priori expected for 100 meters, later realizing that this sediment was split into two by a small layer of shale. The observation data is encoded wrt. *time*, but will be used with an encoding wrt. *depth* in this application to have it aligned with the prior information at hand. The data may not necessarily be equally spaced (e.g., two neighbouring observations may be separated by half a foot, while the two next are separated by only a tenth of a foot).

In this case, we think in terms of an IOHMM (see Section 3.3.1), in which the latent variables will have a predefined structure (see Figure 4.31):

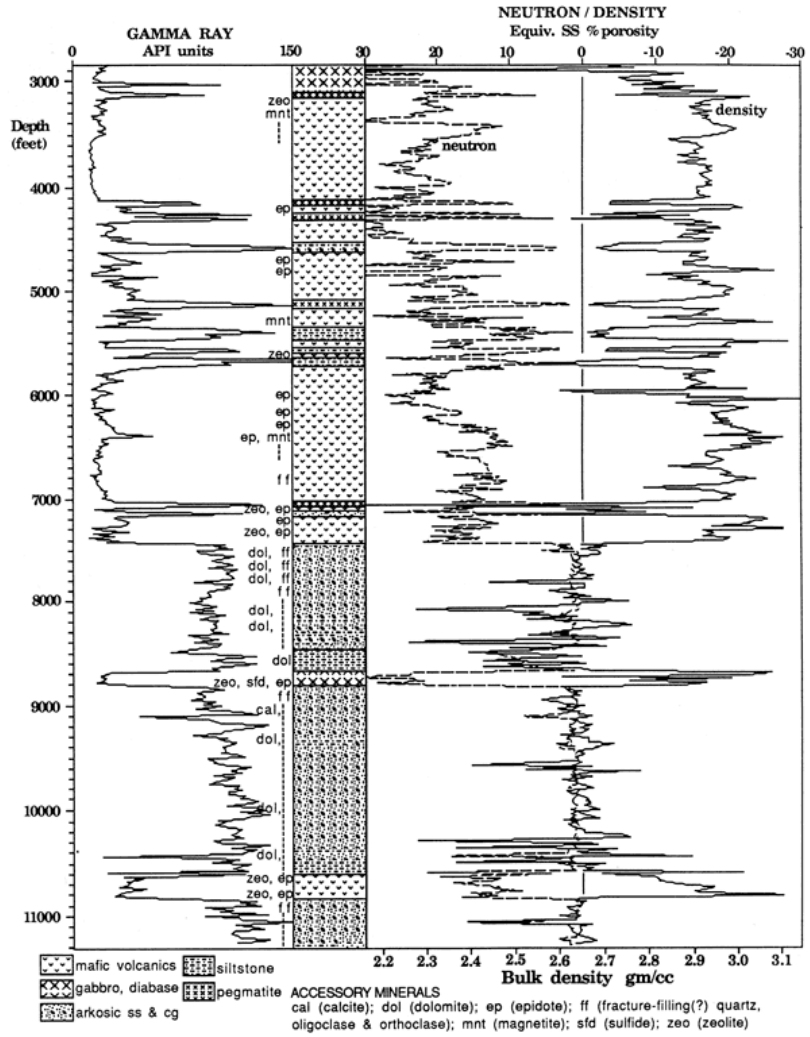


Figure 4.30: Lithology data shown together with petrophysical measurements such as gamma radiation and neutron/density.

- The input variable at time t, Depth_t, is the depth of the bit at that time.
- The next chain of variables are *counting-variables*, denoted by FormationNo_t. The variable has two parents: Depth_t and FormationNo_{t-1}. The probability that the counting variable is incremented at time t conditioned on the parent configuration is calculated by combining information from the prior information with the parents' configuration. Logistic or probit functions can be imagined, but must be compared to the actual domain knowledge to give a feasible representation. Note that the model structure, with a continuous parent having a discrete child does not entail computational difficulties in this case: The parent is always observed, and the

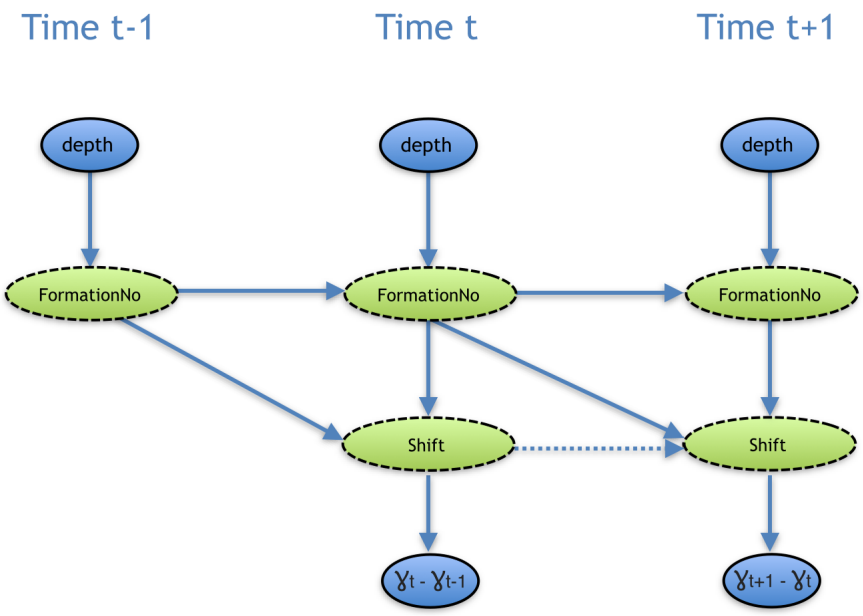


Figure 4.31: Input-output HMM used for formation detection. The output observed variables $\gamma_t - \gamma_{t-1}$ represent the differences of consecutive gamma ray observations.

calculation of the conditional probability table (encoding the transfer function) can thus be done externally. The model thus capture a non-stationary process by conditioning the transition distribution to the observed control variables.

- Shift_t generates a chain of variables used for determining when a formation-change actually occurs. It has two parents, namely FormationNo_{t-1} and FormationNo_t. The variable is discrete, and has four states: "No – As accounted for in the prior", "No – But expected in prior", "Yes – As accounted for in the prior", and "Yes – But not expected in prior". To this end, FormationNo_t only changes when expected by the prior information, while Shift_t can use the data to refine these assessments. If $\text{FormationNo}_t = \text{FormationNo}_{t-1}$, Shift_t is in the state "No – As accounted for in the prior" with probability $1-\epsilon$ and in state "Yes – But not expected in prior" with probability ϵ. If $\text{FormationNo}_t = \text{FormationNo}_{t-1} + 1$, Shift_t takes the state "Yes – As accounted for in the prior" with probability $1-\delta$ and "No – But expected in prior" with probability δ. ϵ and δ are parameters to be determined through expert knowledge or learned from data.

- Finally, the observed variables at time t only have Shift_t as parent. We will not use the actual observed sensor reading as the observed variable ("output-variable" in the input-output model). Rather, the differences are considered, e.g., GammaRay_t - GammaRay_{t-1} will be used to represent the gamma ray measurement at time t. These variables will, given that Shift_t is in the "No"-states, vary around zero.

The variable will be significantly different from zero if the parent is in any of the "Yes"-states. It is clear from Figure 4.30 that most formation changes are seen as clear shifts in the gamma ray data.

At the time of inference, we condition on the total number of switches, and calculate (using forward-backward) the posterior distribution of the locations. Similarly, we will also calculate the most probable explanation, that is, the most probable configuration of switches given the total number of events.

4.3.4 Discussion and future models

The models presented above, describe a first attempt to address the tasks involved in the Verdande problem domain. A first critical question to the model for erratic torque is that the model assumes the observations from a tach of data without erratic torque to follow a Gaussian distribution. Secondly, the assumption that erratic torque is manifested through an additive high-correlated Gaussian noise must be verified as well. A natural first step is to experiment with the model on synthetic data simulated from relevant (but non-Gaussian) distributions to quantify its robustness wrt. these assumptions. Secondly, a critical investigation into the results obtained when running on real data will be conducted, potentially leading to more expressive models having to be utilized.

The semi-automatic labelling model is so far made using a general-purpose model structure, that does not encode a priori domain knowledge. As more domain knowledge is uncovered (see the upcoming Deliverable 7.1), we will consider to make structural changes accordingly.

The model for formation detection assumes that changes in the logged data happen instantaneously. Instantaneous "jumps" are observed for, e.g., the gamma ray measurement, but may not be equally valid for other variables, like resistivity. If required, a richer model (albeit contained inside the class of input-output models) may have to be addressed, where the observations themselves (and not their time differences) are utilized. Furthermore, we have again assumed that the difference in an variable between two observations follows a Gaussian distribution with fixed variance. The assumption must be better validated with other data sets and amended if required.

5 AMIDST model class

One of the main goals of AMIDST project is the definition of a general model class with the following characteristics:

- **Feature 1:** it should be applicable to the three considered use-cases, i.e., Daimler, Cajamar, and Verdande,

- **Feature 2:** it should be general enough to be applicable to any future, potentially similar, use-cases and
- **Feature 3:** it should be scalable, supporting both inference and learning from massive data streams.

Taking these different characteristics into account and using the subnetwork graphical notation introduced in Section 3.1, the general AMIDST model class, as well as its specific instantiation to each use case, are first introduced and discussed in Section 5.1. A summary is finally included in Section 5.2.

5.1 The general AMIDST model class

Figure 5.1 shows the proposed general AMIDST model class. This model is the result of combining all the models that have been previously defined for the different use cases. As it can be seen, subnetworks have been used to group variables with similar properties, so that the commonalities between all models are taken into account.

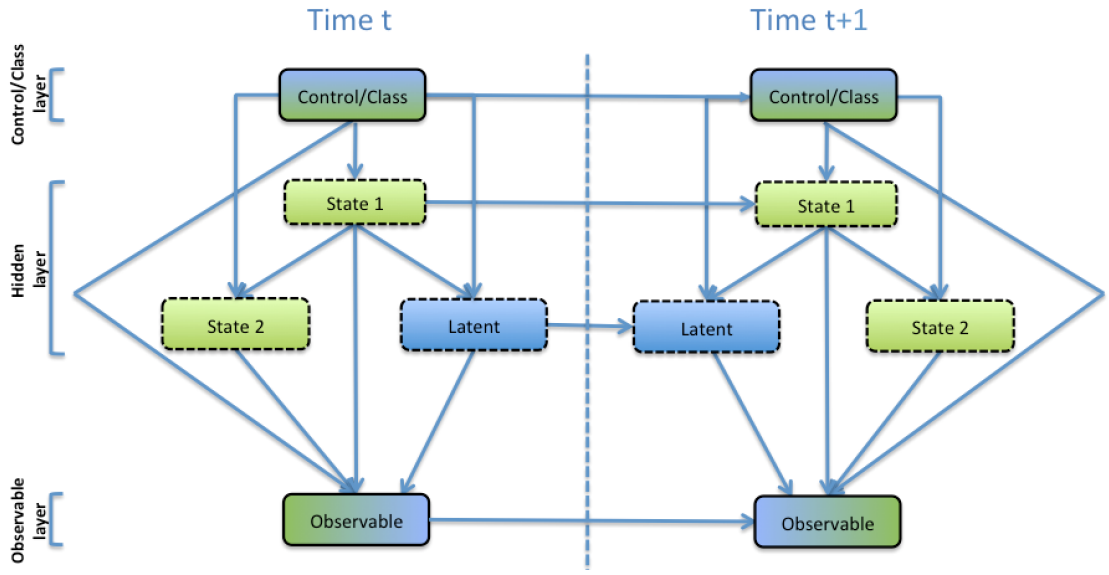


Figure 5.1: General AMIDST model class.

The general AMIDST model class can be seen as a 2T-DBN (see Section 3.3.3), i.e., it satisfies the first order Markov property and the stationary assumption. However, this model is not as general as a 2T-DBN and presents a more restricted internal structure that offers a good trade-off between expressiveness and efficiency. More precisely, the structure of this model decomposes along three main different layers which can be described, at time t, as follows:

- **Control/class layer:** The upper layer corresponds to a subnetwork including either continuous or discrete observed variables, that can act for instance as either control continuous variables or a class variable respectively. These set of variables can be connected through consecutive time steps to encode temporal dependences, e.g., the probability of a particular class label at time $t + 1$ varies with respect to its label at time t.

  The links from the continuous observed variables to the state subnetworks in the *Hidden layer* (i.e., the set of State 1 and State 2 variables) will be possibly handled with logit and probit functions.

- **Hidden layer:** The middle layer corresponds to a set of interconnected discrete and continuous hidden subnetworks, for which only links from discrete to continuous subnetworks are allowed. We will use the term *state* variables to refer to the set of discrete hidden variables in both State 1 and State 2 subnetworks, and the term *latent*[7] variables to refer to the set of continuous hidden variables in the Latent subnetwork.

  This layer can contain state variables that are not connected over time, and that could be used, for instance, to model mixture of Gaussian distributions over the observed continuous variables in the *Observable layer*. Moreover, state and latent variables that are connected over time can be used to model a process that is evolving and changing over time. Recall that, links from latent variables to state variables are strictly prohibited.

- **Observable layer:** The bottom layer corresponds to a subnetwork including both discrete and continuous observed variables. These variables can in principle be interconnected but, in our different use cases, only links from discrete to continuous nodes are required. However, in general, there is no restriction on the direction of the links between the variables in this layer. In addition, it is possible to alleviate the sensor Markov assumption by including links between observed variables at consecutive time steps.

This model class can also be parametrized in alternative ways to improve the expressibility and applicability to different problems or domains. In general, this model class falls inside the conditional linear Gaussian framework, so the conditional probability distributions are parameterized as detailed in Section 3.2. But this model class might also contain instantiations which are not covered by this general framework. More precisely, when the top layer class is instantiated to a continuous subnetwork then the conditional probabilities of "State 1" and "State 2" will be instantiated with logistic or probit distributions, because we have continuous parents with discrete children. Additionally, we also envision the possibility of introducing in the observable layer the use of probability

[7]The term *latent* for variables is generally used in statistics to refer to hidden variables as opposed to observed ones. However, we will use it here to exclusively refer to continuous hidden variables.

distributions belonging to the MoTBF family [11] to extend the modelling capacities of this model class.

In the following sections, we will show how this model class satisfies the aforementioned **Feature 1**, by describing how each particular use case fits in this restricted 2T-DBN AMIDST model. This instantiation process of the general AMIDST model to the three use-cases (all of them with different application scenarios) can also be seen as a practical example demonstrating the general applicability of the resulting AMIDST model class. This hence gives arguments in favour of **Feature 2** and shows how the AMIDST model class can be potentially instantiated to other future use cases. Finally, the remaining task is related to **Feature 3** and consists of designing inference and learning algorithms that allow the application of the AMIDST model to massive data streams.

5.1.1 Daimler model class

Daimler's model class has been previously displayed in Figure 4.11. Taking the general AMIDST model class into account, Daimler's model class can be reinterpreted, according to Figure 5.2, as follows:

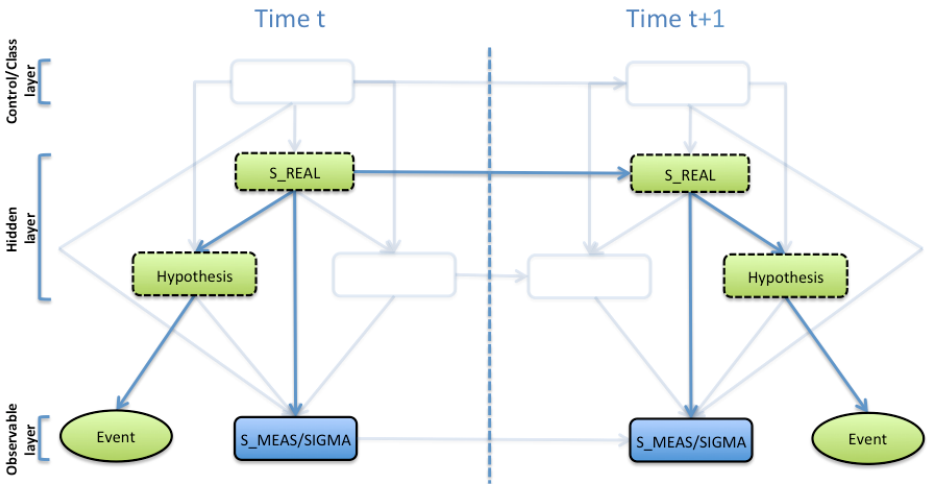


Figure 5.2: AMIDST model class - Daimler

- **Control/class layer**: is not used in this domain.
- **Hidden layer**: the two sets of state variables are required here.

  The lower set of state variables (i.e., "State 2" subnetwork in Figure 5.1) encode a polytree [2] for the hierarchy of the Hypothesis. This polytree structure, not explicitly encoded in the model, can indeed be exploited during inference.

Connection through time is only made at the top level state variables (i.e., "State 1" subnetwork in Figure 5.1), which corresponds to the signal real values (S_REAL[8]). Consequently, the future and past time slices of our 2T-DBN are conditionally independent given S_REAL variables corresponding to the present time.

In addition, inside a time slice, the observed continuous and hidden discrete subnetworks are also conditionally independent given S_REAL variables. In the case that we want to contemplate a possible extension in which the lower level Hypothesis are connected over time, then "State 1" subnetwork would contain both the S_REAL variables and the temporally linked Hypothesis. However, as commented before, this would greatly complicate the inference process.

- **Observable layer:** there are two sets of observed variables in this case. On one hand, we encounter a group of variables representing the measured data (S_MEAS), along with the variables encoding the sensor noise (S_SIGMA). The resulting structure is again a polytree, which presents some advantages for the inference process. On the other hand, we have a single discrete node for the manoeuvre Event, which will be the target variable during inference.

5.1.2 Cajamar model class

Concerning Cajamar use-case, both application scenarios share the same models with two possible versions, namely, static and dynamic (see Section 4.2). At this point, we obviate the static version, because the general AMIDST model class is primarily a dynamic model class. Therefore, the high-level description of Cajamar's dynamic model (previously displayed in Figure 4.19) can be reinterpreted according to Figure 5.3.

Following the layer-wise analysis used above, Cajamar's model is described as follows:

- **Control/class layer**: the top node "Defaulter" in this layer represents the class variable to categorize a client as defaulter or non-defaulter. There exist temporal links between consecutive time steps to model the dynamic nature of being a defaulter or non-defaulter. Note that this is indeed the target variable in the inference process.
- **Hidden layer**: is not used in this domain.
- **Observable layer:** The "Observed" continuous subnetwork represents information corresponding to the socio-demographic variables, memory variables, financial activity, and past payment behaviour of a client, which, in principle, may or may not be connected over time. Moreover, the "Indicator" discrete subnetwork includes the set of indicator discrete variables denoted as δ_{X_t}. These indicator

[8] Although these variables are inherently continuous, we notice that they are discretized to avoid the inference problems derived of having continuous parents with discrete children (see Section 4.1.1).

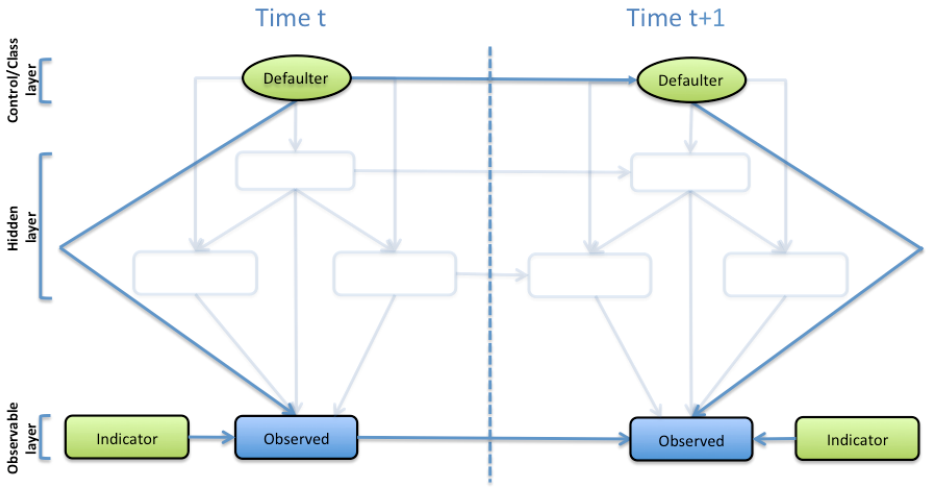


Figure 5.3: AMIDST model class - Cajamar.

variables are used for modelling situations where some variables in the "Observed" continuous subnetwork have a large number of zeros.

5.1.3 Verdande model class

Figure 5.4 shows how the general AMIDST model class is instantiated in the case of Verdande. This instantiated model encompasses the models of the three application scenarios previously discussed in Section 4.3 and depicted in Figures 4.28, 4.29 and 4.31.

The instantiated Verdande model class can be described as follows:

- **Control/class layer**: This layer includes a subnetwork modelling the set of observed Control variables. In the three application scenarios, the role of these Control variables is to condition the transition probability of the state variables, and to ensure as well the modelling of non-stationary transition probabilities.
- **Hidden layer**: This layer is directly instantiated from the general AMIDST model class. However, there are some differences when applied to each particular application scenario:
  - For the first application scenario (see Figure 4.28): "State 2" is not needed and "State 1" is instantiated to the "Normal/Abnormal" state variable.
  - For the second application scenario (see Figure 4.29): "State 1" is not needed and "State 2" is instantiated to a single multinomial variable whose role is to model mixture of Gaussians at the leaves.

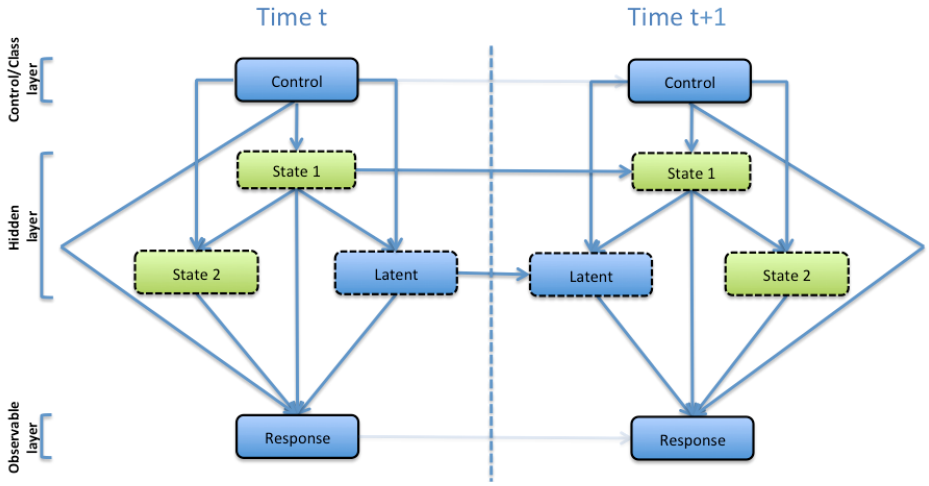


Figure 5.4: AMIDST model class - Verdande.

  - For the third application scenario (see Figure 4.31): "State 2" is not needed and "State 1" is instantiated to a subnetwork containing the "FormationNo" and "Switch" variables.

- **Observable layer:** For both application scenarios 1 and 3, this layer is instantiated as a single response variable; while for the application scenario 2, it may be instantiated as a set of (possibly interconnected) response variables.

5.2 Summary

As it has been shown in the previous sections, the proposed general AMIDST model class of Figure 5.1 encompasses the different application scenarios of the three use cases. These three use cases come from very different domains: automotive, finance and oil-drilling. In our opinion, these are strong arguments in favour of the generality and applicability of this model class. Although, in any case, a better understanding of the faced problems and/or future applications might reveal that some elements of this model class need to be refined.

A higher level view of our proposed AMIDST model is displayed in Figure 5.5. As commented before, it shows how the AMIDST model class could actually be seen as a "restricted" 2T-DBN. It is restricted in three different ways. Firstly, all the nodes are structured in three layers, each one with clear semantics while modelling: Broadly speaking, the *control/class layer* represent control variables that may affect the process homogeneity/stationarity, or represent the class variable in classification tasks; the *hidden layer* includes a sufficiently rich set of hidden variables to capture the process dynamics; and the *observable layer* encompasses sensor measures and predictive attributes.

The second restriction states that, in opposite to general 2T-DBNs, the variables in this model class can only be temporally linked to other variables in the same layer. And, finally, the last restriction states that continuous hidden variables cannot point to discrete (hidden or observed) nodes. This last constraint implies that our inference algorithms will not have to deal with the hard and open problem of computing posterior distributions or marginalize over continuous variables which are parents of discrete children nodes. So, in most of the cases, the network can be parameterized using the conditional linear Gaussian framework. And in those cases when this is not possible, because the instantiated model contains continuous parents with discrete children, we will have that the continuous nodes are always observed.

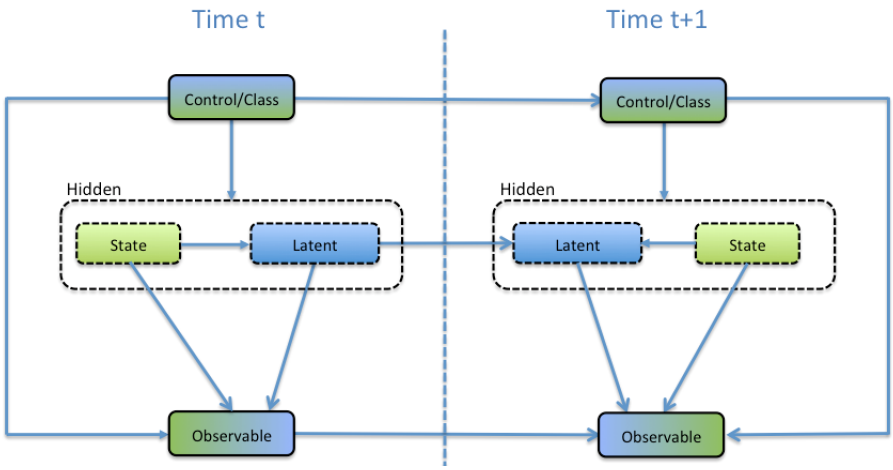


Figure 5.5: The high-level AMIDST model class

References

[1] Borchani, H., Fernández, A., Gundersen, O.E., Hovda, S., Langseth, H., Madsen, A.L., Martínez, R., Masegosa, A., Nielsen, T.D., Salmerón, A., Sørmo, F., Weidl, G.: Requirements for the automotive, oil and financial data domains. Deliverable 1.2 of the AMIDST project, available from `http://amidst.eu/`. (August 2014)

[2] Jensen, F., Nielsen, T.: Bayesian networks and decision graphs. Second edn. Springer Publishing Company, Incorporated (2007)

[3] Fernández, A., Hovda, S., Langseth, H., Madsen, A.L., Masegosa, A., Nielsen, T.D., Salmerón, A.: General methodology for requirement analysis. Deliverable 1.1 of the AMIDST project, available from `http://amidst.eu/`. (July 2014)

[4] Lauritzen, S.L.: Propagation of probabilities, means, and variances in mixed graphical association models. Journal of the American Statistical Association **87**(420) (1992) 1098–1108

[5] Lauritzen, S., Jensen, F.: Stable local computation with conditional Gaussian distributions. Statistics and Computing **11**(2) (2001) 191–203

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

[7] Hrycej, T.: Gibbs sampling in Bayesian networks. Artificial Intelligence **46**(3) (1990) 351–363

[8] Jordan, M., Ghahramani, Z., Jaakkola, T., Lawrence, S.: An introduction to variational methods for graphical models. Machine Learning **37**(2) (1999) 183–233

[9] Kozlov, A., Koller, D.: Nonuniform dynamic discretization in hybrid networks. In: Thirteenth Conference on Uncertainty in Artificial Intelligence. (1997) 314–325

[10] Moral, S., Rumí, R., Salmerón, A.: Mixtures of truncated exponentials in hybrid Bayesian networks. In Benferhat, S., Besnard, P., eds.: Sixth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU). Volume 2143 of Lecture Notes in Computer Science., Springer (2001) 156–167

[11] Langseth, H., Nielsen, T., Rumí, R., Salmerón, A.: Mixtures of truncated basis functions. International Journal of Approximate Reasoning **53**(2) (2012) 212–227

[12] Koller, D., Pfeffer, A.: Object-oriented Bayesian networks. In: Thirteenth Conference on Uncertainty in Artificial Intelligence. (1997) 302–313

[13] Dean, T., Kanazawa, K.: A model for reasoning about persistence and causation. Computational Intelligence **5**(3) (1989) 142–150

[14] Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Third edn. Prentice Hall (2009)

[15] Boyen, X., Koller, D.: Tractable inference for complex stochastic processes. In: Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI). (1998) 33–42

[16] Doucet, A., Freitas, N., Murphy, K., Russell, S.: Rao-blackwellised particle filtering for dynamic Bayesian networks. In: Sixteenth Conference on Uncertainty in Artificial Intelligence. (2000) 176–183

[17] Kasper, D., Weidl, G., Dang, T., Breuel, G., Tamke, A., Rosenstiel, W.: Object-oriented Bayesian networks for detection of lane change maneuvers. In: IEEE Intelligent Vehicles Symposium (IV), Kongresshaus Baden-Baden, Germany, June 5-9. (2011) 673–678

[18] Kasper, D., Weidl, G., Dang, T., Breuel, G., Tamke, A., Wedel, A., Rosenstiel, W.: Object-oriented Bayesian networks for detection of lane change maneuvers. IEEE Intelligent Transportation Systems Magazine **4**(3) (2012) 19–31

[19] Kasper, D.: Erkennung von Fahrmanövern mit Object-Orrientierten Bayes-Netzen in Autobahnszenarien. PhD thesis, Tübingen University, Germany (2013)

[20] Weidl, G., Madsen, A.L., Kasper, D., Breuel, G.: Optimizing Bayesian networks for recognition of driving maneuvers to meet the automotive requirements. In: IEEE Multi-Conference on Systems and Control, Nice/Antibes, France, October 8-10. (2014) To appear

[21] Lauritzen, S.: Graphical Models. Oxford University Press (1996)

[22] Tereshchenko, V.: Relative object-object dynamics for earlier recognition of maneuvers in highway traffic. Master's thesis, University of Stuttgart (to appear 2014)

[23] Shimony, S.E.: Finding MAPs for belief networks is NP-hard. Artificial Intelligence **68**(2) (1994) 399–410

[24] Barber, D.: Bayesian Reasoning and Machine Learning. Cambridge University Press (2012)