Universidad Politécnica de Madrid

Escuela Técnica Superior de Ingenieros Informáticos

Máster Universitario en Inteligencia Artificial

Trabajo Fin de Máster

Explanations for Dynamic Bayesian Networks: A Case Study in Climate Science

Autor: Enrique Valero Leal
Tutor: Pedro Larrañaga Múgica y Concha Bielza Lozoya

Madrid, Julio - 2022

Trabajo Fin de Máster
Máster Universitario en Inteligencia Artificial

Título: Explanations for Dynamic Bayesian Networks: A Case Study in Climate Science

Julio - 2022

Autor: Enrique Valero Leal
Tutor: Pedro Larrañaga Múgica y Concha Bielza Lozoya
Departamento de Inteligencia Artificial
ETSI Informáticos
Universidad Politécnica de Madrid

Resumen

Si bien la inteligencia artificial (IA) es un campo que lleva en investigación desde hace décadas, no ha sido hasta recientemente que han surgido una gran cantidad de modelos con una gran capacidad de cómputo. Sin embargo, dicha gran capacidad suele venir a costa de una opacidad en el tratamiento de los datos, lo que nos lleva a no entender el razonamiento de dicho sistema. En una sociedad donde cada vez la IA está más y más presente en nuestras vidas, esto puede dar lugar a problemas legales, éticos y de rendimiento. Como respuesta, en los últimos años está en auge la disciplina de la inteligencia artificial explicable, que busca mejorar la interpretabilidad de los sistemas de diversas maneras. Uno de los métodos consiste en seleccionar modelos que sean transparentes y mejorar su interpretabilidad, dado que transparencia no implica explicabilidad ni interpretabilidad. Un ejemplo de dichos modelos son las redes Bayesianas, un tipo de modelo gráfico probabilístico que permite una representación compacta y visual de los datos además de trazar el proceso de inferencia.

En este trabajo buscamos analizar series temporales de datos usando redes Bayesianas y mejorar la intepretabilidad de dicho análisis. Mientras que el grueso de trabajos en inteligencia artificial explicable se centran en predicción (y generalmente con cajas negras), analizar datos donde el tiempo es una variable a considerar es una tarea que ha quedado más relegada, pero que es de gran utilidad debido al número de campos de aplicación, como por ejemplo en neurociencia o en las ciencias climáticas. Las redes Bayesianas dinámicas son una extensión de estos modelos que permiten lidiar con series temporales. Sin embargo, dichos modelos suelen asumir estacionariedad en las relaciones temporales, algo que no es realista en algunos campos como en ciencias climáticas, en el que nos centraremos (si bien nuestras propuestas no se verán limitadas a dicho ámbito). Así, se buscará también mejorar la explicabilidad en modelos no estacionarios, que cuentan con una literatura menos extensa.

En primer lugar, realizamos una extensa revisión literaria sobre explicabilidad e interpretabilidad en series temporales y redes Bayesianas. También expandimos la última añadiendo una nueva categoría y completando las ya existentes, y catalogandose los tipos de exlicaciones en redes Bayesianas en las clases ya existentes de explicaciones en aprendizaje computacional. También se revisan en detalle las explicaciones que nos serán de utilidad a la hora de desarrollar nuestras propuestas, como por ejemplo las explicaciones basadas en ejemplos (prototipos y contrafactuales) y las explicaciones del modelo.

En cuanto a nuestras propuestas, en este trabajo diversificaremos nuestra investigación en lugar de centrarnos en un solo tipo de explicación, siguiendo la filosofía de “una explicación no se ajusta a todo”. Primero, mejoramos propuestas ya existentes que analizan la robustez en explicaciones en redes Bayesianas. Si bien esta propuesta no se restringe a series temporales, evaluar la robustez es, junto a la simplicidad, un factor que consideramos clave a la hora de conseguir una explicación y, por ende, será analizado en este trabajo. Volviendo al ámbito de las redes Bayesianas dinámicas, haremos uso de la capacidad generativa de estos modelos para construir explicaciones basadas en prototipos que pueden ser utilizadas en modelos estacionarios y no estacionarios, ya que el ser humano tiende a buscar ejemplos para comprender fenómenos. También se analizan los cambios producidos en modelos no estacionarios, mostrándolos gráficamente para así entender cómo las relaciones temporales han cambiado en el tiempo. En adición a estas propuestas teóricas, también ofrecemos una implementación de código abierto de aprendizaje, inferencia y visualización de redes Bayesianas dinámicas variables en el tiempo, modelos en los que las relaciones temporales cambian en cada instante. A dicha implementación se añade un módulo de explicabilidad.

Usando nuestra propia implementación, ilustramos el potencial de las propuestas con algunos experimentos, utilizando tanto conjunto de datos de referencia como la base de datos climática NCEP/DOE Reanalysis II, de extendido uso en la comunidad científica. El objetivo es mostrar que, en efecto, nuestra propuesta es aplicable computacionalmente y útil en ámbitos reales.

Este trabajo se enmarca en el proyecto de investigación concedido por el Ministerio de Educación de España con PID 2019-109247GB-I00 y en el proyecto de la fundación BBVA (convocatoria 2019) "Score-based nonstationary temporal Bayesian networks. Applications in climate and neuroscience".

Abstract

Although artificial intelligence (AI) is a field that has been subject to research for decades, recently a large number of models with high computational power have emerged. However, this often comes at the cost of obscured data processing, which leads us to not be able to understand the reasoning of such systems. In a society where AI is becoming more and more present in our lives, this can lead to legal, ethical and system performance issues. In response, the discipline of explainable artificial intelligence (XAI), which seeks to improve the explainability of systems in a variety of ways, has been on the rise in recent years. One method to accomplish that is to select models that are transparent and improve their interpretability, since transparency does not imply explainability nor interpretability. An example of such models are Bayesian networks, a type of probabilistic graphical model that allows a compact and visual representation of the data as well as tracing the inference process.

In this master thesis project we focus in analysing time series using Bayesian networks and on improving the interpretability of such analysis. While the bulk of work in XAI is focused on prediction (and usually with black boxes), analysing data where time is a variable to be considered is a task that has been subject of less study, but that is very useful due to the increasing number of application fields that incorporate a temporal dimension, such as neuroscience or climate science. Dynamic Bayesian networks are an extension of these models that allow dealing with time series. However, these models tend to assume stationarity in the temporal relationships, which is not realistic in some fields such as climate science, on which we will focus (although our proposals will not be limited to this field). Thus, we will also research how to improve the explainability of non-stationary models, which have a less extensive literature.

First, we conduct an extensive literature review on explainability in time series and Bayesian networks. We also expand the latter by adding a new category and completing the existing ones, and we categorize the different types of explanations in Bayesian networks into the already existing classifications of explanations of machine learning (ML). We also review in detail explanations that will be useful to the development of our proposals, such as example-based explanations (prototypes and counterfactuals) and model explanations.

As for our proposals, in this document we will diversify our research rather than focusing on a single type of explanation, following the philosophy of “one explanation does not fit all”. First, we improve upon existing proposals that explore the robustness of explanations in Bayesian networks. While this proposal is not restricted to time series, robustness is, together with simplicity, a key factor for us to construct good explanations and therefore, will be studied in this master thesis project. Returning to the field of dynamic Bayesian networks, we will make use of the generative capacity of these models to construct explanations based on prototypes that can be used in stationary and non-stationary models, since human beings tend to look for examples to understand phenomena. The changes produced in non-stationary models are also analysed, showing them graphically in order to understand how temporal relationships have changed over time. In addition to these theoretical proposals, we also offer an open-source implementation of learning, inference and visualization of time-varying dynamic Bayesian networks, models in which temporal relationships change at every instant. An explainability module is added to this implementation.

Using our own implementation, we illustrate the potential of our proposals with some experiments, using both simulated benchmark datasets and the NCEP/DOE Reanalysis II climate database, widely used in the scientific community. The objective is to show that our proposal is computationally applicable and useful in real environments.

This work is part of the research project granted by the Spanish Ministry of Education with code PID 2019-109247GB-I00 and by BBVA foundation project (call 2019) entitled “Score-based nonstationary temporal Bayesian networks. Applications in climate and neuroscience".

Acknowledgements

First, I want to thank and to dedicate this work to my parents and my sister. Without their support, love and patience, this work would have never been possible.

I would like to acknowledge the wise and complete guidance provided by my supervisors, Pedro Larrañaga and Concha Bielza. Their feedback and suggestions were crucial to the development of this thesis.

I'm thankful as well to all my colleagues at the CIG lab, specially to David Quesada for his help in the topic of dynamic Bayesian networks.

This research has been partially funded by the Spanish Ministry of Science and Innovation through the PID 2019-109247GB-I00 project and to the BBVA Foundation (2019 Call) through the "Score-based nonstationary temporal Bayesian networks. Applications in climate and neuroscience" project.

Contents

List of Tables

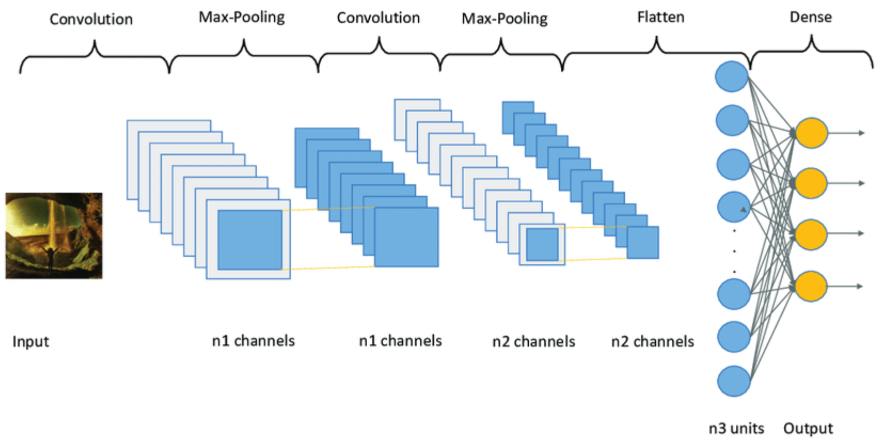
List of Figures

Chapter 1

Introduction

1.1. Motivation

Artificial intelligence (AI) and machine learning (ML) are not brand-new topics of computer science. Both of them have been in development for over fifty years, but it was not until this last decade when these terms became extremely popular: the deep learning revolution in 2012 , the Internet of Things, AlphaGo Zero and tremendous political scandals concerning illicit use of artificial intelligence and big data are just a few examples (see Figure 1.1 for an example of deep learning). This is no longer an unknown field of computer science: everybody is aware of the existence of intelligent agents, every big corporation uses it for commercial purposes and even some of them develop whole products based on it.



https://www.researchgate.net/profile/Jose-Benitez-Andrade

Figure 1.1: Alexnet (Krizhevsky et al., 2012), a convolutional neural network. This implementation was the first capable of analysing images efficiently, and thus it may be considered as the beginning of the deep learning revolution

The spread of AI and machine learning came along with a rise of complex methods that offer excellent information processing quality, which is the aforementioned deep learning revolution. However, greater complexity usually leads to a greater lack of knowledge concerning how the intelligent system works, which is not only a disadvantage but can also become a legal issue in certain critical environments, for instance, in credit loaning or disease diagnosis. These models are referred to as black-boxes (see Figure 1.2), being that it is not possible to visualize how they operate.

Input
BLACK BOX
Output

Figure 1.2: Black-box model. Given a trained black-box model and an input, it generates an output. However, the operation that leads to said output is unknown or incomprehensible

In the European General Data Protection Regulation (GDPR) we can already find a mention to the "right to explanation" (Regulation EU 2016/679), which means that an intelligent system should be able to provide an explanation of its reasoning. In addition to all kind of legal restrictions, it is manifestly clear that, in the future, AI will be the epicentro of scandals such as Cambridge Analytica or racist and misogynist autocomplete suggestion in chat applications or search engines (Lapowsky, 2018).

There is an additional category in which black-box models may fail to deliver and that is of our focal interest: in scenarios in which understanding the causal mechanisms is important, regardless of laws and without being the subject of ethics scandals, such as neuroscience and climate science. Black-box, and even transparent models, may have a good accuracy but base their predictions in wrong evidence and reasoning. This is referred to in cognitive psychology, and also most recently in machine learning, as the Clever Hans effect (Lapuschkin et al., 2019). It is necessary to know that the artificial intelligence based system used the proper information for predictive modelling and to extract valid knowledge from the model.

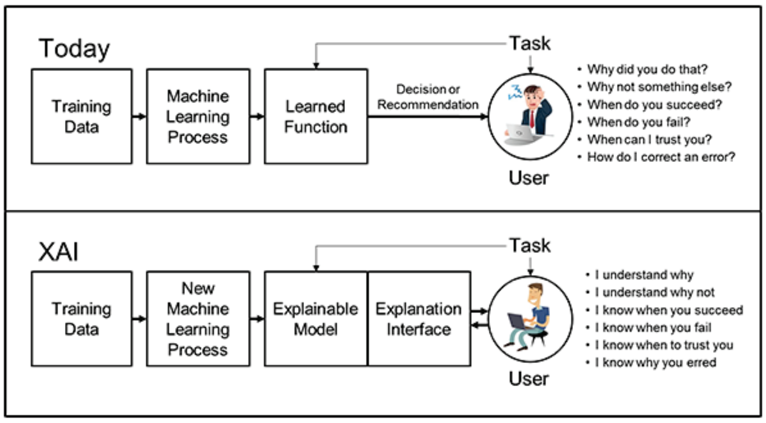
Many of these problems can be tackled at the lower level by researchers and computer scientist using explainable artificial intelligence (XAI).

1.2. Explainable AI as a solution

This discipline of AI aims to make comprehensible the reasoning of intelligent systems. Instead of providing just an output, we obtain an explanation as well (see Figure 1.3). Often, we refer to this field as interpretable AI or interpretable machine learning, although those terms may differ slightly from each other.

Explainable AI aims to build more trustworthy and transparent systems and its results are the following:

- Verify and improve the correct functioning of a machine, as it can be inspected more easily.
- Respect the legislation and a human-centred ethical framework.
- Social acceptance: We can consider this a psychological consequence of XAI. Explanations are a key factor in human learning and even in life itself. The human being knowledge is updated by finding explanations, since we tend to believe results backed by coherent explanations (Molnar, 2020).



`https://www.darpa.mil/ddm_gallery/xai-figure2-inline-graphic.png`

Figure 1.3: Explainable AI

Ultimately, the scope of XAI is much wider than just the AI (see Figure 1.4), which means that to make the system explain themselves better we have to consider also certain aspects of the cognitive and social sciences, such as understanding the characteristics of everyday explanations and the need to obscure or simplify the numbers.

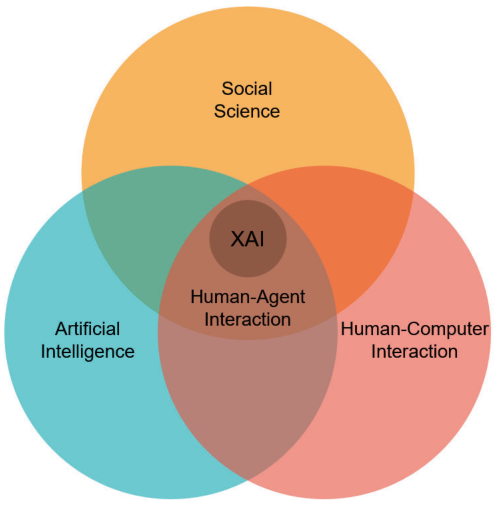
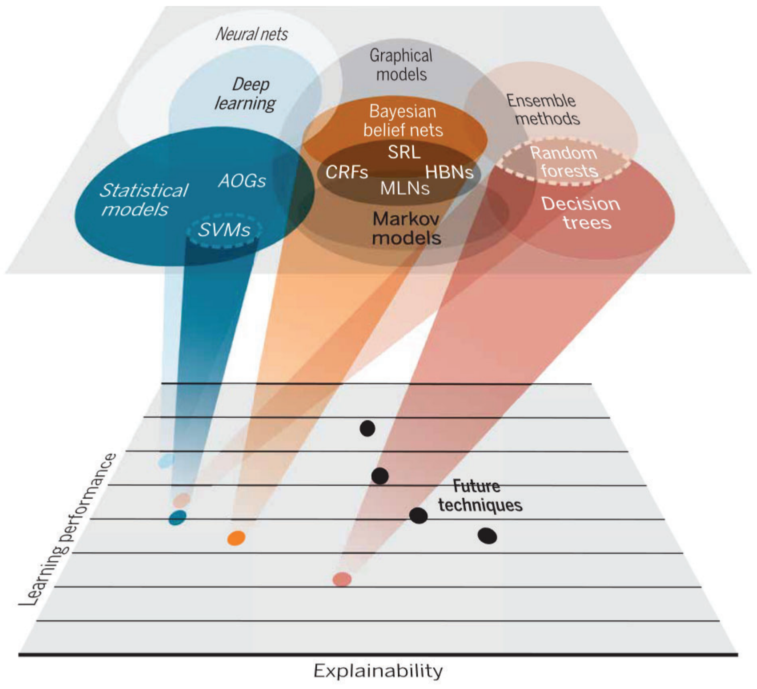


Figure 1.4: Scope of XAI, based on the illustration of Miller (2019)

Some early works started to establish a trade-off between interpretability and computational power (Casillas et al., 2003), a concept that solidified with the appearance of deep learning, as in Gunning et al. (2019). The basic idea is that the more we gain in transparency, the more we lose in predictive accuracy and vice versa (see Figure 1.5). As such, the lack of intrinsic interpretability in deep learning results in overall better results and the efforts in XAI might as well be dedicated to explain these black-box systems with various approximations.

An opposite line of research that rejects said trade-off was explored in Rudin (2019), where the author labels it as a myth and criticizes the poor quality of explanations given by neural networks, as they may lead to explanation that do not make sense or that are over-complicated for humans. Instead, it is stated that the optimal model may differ depending on the domain and that in every scenario there will be at least one interpretable and accurate model, since the set



Gunning et al. (2019)

Figure 1.5: Visualization of the accuracy-interpretability trade-off, where we can see that simple models such as decision trees offer much more interpretability at the expense of being less accurate, while in models such as neural networks, stochastic and-or grammars (AOGs) or support vector machines (SVMs), the opposite occurs. Probabilistic graphical models such as Bayesian networks, statistical relation learning (SRL), conditional random fields (CRFs), hierarchical Bayesian networks (HBNs) or Markov logic networks (MLNs) are placed in the middle of this trade-off

of possible models for solving a problem is enormous. This idea is referred to as the Rashomon sets, a set of possible models so big that at least one will work. The idea was subsequently formalised in Semenova et al. (2019).

The latter mindset is the one that we will be following throughout this work. Selecting a transparent model and then focus on how can we make it more comprehensible, since transparency does not imply interpretability. We decided to focus on Bayesian networks (Pearl, 1988), (Koller and Friedman, 2009), a type of probabilistic graphical model. There are many reason for selecting Bayesian networks over other models:

- Bayesian networks have a strong mathematical and statistical base, rather than relying on shady heuristics. They take advantage of conditional independence of variables and conditional distributions.
- This allows for representing the joint probability distribution in a modular way, making it more comprehensible.
- Such modularity also allows for both tractable and traceable inference.
- All these properties are enhanced by the fact that they are probabilistic graphical models and, as a consequence, it is easy to visualise the relations and even the reasoning process

followed.

While this work mainly focuses on explainability techniques, the actual ultimate goal is to be able to apply said techniques. The field in which we are interested is climate sciences. More specifically, we are interested in studying how different variables across the globe have evolved over the past years and study how the relations between them have changed over time. That is, our goal is to comprehend these changes through the model and being able to simplify them in order to efficiently extract knowledge from the model itself and from interventions that we can do in the model. A secondary goal will be predicting the state of many climate variables across the globe (forecasting) and being able to explain such prediction.

We will use variations of Bayesian networks that are able to process time series: dynamic Bayesian networks. Once the model is built, our proposals will help extract all the information we need in the context of time series.

1.3. Objectives

The objective of this work is to be able to interpret (dynamic) Bayesian networks and to improve upon already existing methods. This will allow to better study the far-reaching problems of climate sciences in the future. To accomplish this high level goal, we established ourselves lower level goals:

- Review the literature and identify proposals that are both incomplete or that admit improvements, or clear gaps in the literature.
- Review existing implemented tools and select one to program our experiments. If further implementation is needed, it will be done as well.
- Make the proposal and get weekly feedback by the supervisors of this work.
- Validate such proposals theoretically and/or experimentally.
- Discuss what we have (and what we have not) accomplished in this work and how should we proceed in the future.

Basically, we apply the well-known scientific method to our specific field in order to formulate a strong foundation for our work.

1.4. Contents

The contributions of this work are organized as follows:

- First, in this very chapter we have introduced the concern to the reader and explained the motivations that lead to choose this topic for this M.Sc. final thesis.
- In Chapter 2, we review the state of the art related to our proposal. In Section 2.1 we introduce time series and in Section 2.2 we explain Bayesian networks and their basic concepts, the two pillars of our work. Then, in Section 2.3 we present dynamic Bayesian networks, an adaptation of this model that let us process time series. Finally, in Section

2.4, we review more thoroughly what XAI is, as well as the most recent advances in both explainability in Bayesian networks (Section 2.4.3) and in time series analysis (Section 2.4.4).

- Next, in Chapter 3, we present our proposals. Instead of digging deep into a single research branch, we open various research lines following the philosophy of "one explanation does not fit all". In Section 3.1 we first improve a method that examines the robustness of evidence explanations given by Bayesian networks. Then, in Section 3.2, we explain how we can take advantage of the generative nature of dynamic Bayesian networks in order to construct example based-explanations. In Section 3.3, we explore how to generate explanations for changes in the structure and parameters in dynamic Bayesian networks (DBNs), an effect that occurs in non-stationary dynamic Bayesian networks (ns-DBNs) (Robinson and Hartemink, 2008) and time varying dynamic varying networks (TV-DBNs) (Song et al., 2009). Lastly, in Section 3.4, we present an implementation of a library that learns and explain TV-DBNs.

- In Chapter 4, we launch some experiments using our implementation and discuss them in order to show the potential of our proposals.

- Finally, in Chapter 5, we draw our conclusion and present our future lines of research that will expand the proposals presented in this work.

Chapter 2

State of the art

2.1. Time series

A time series is a set of observations taken sequentially in time (Box et al., 2015), such as a series of monthly temperatures and rainfalls or a set of students grades over a period of time. the analysis of time series consists of discovering dependencies between points of data, usually adjacent ones. For this purpose, dynamic and stochastic models are built.

Formally, we define a *time series* $\boldsymbol{X}^{0:T}$ as a stochastic process over a set of n variables $\boldsymbol{X} = \{X_1, X_2..., X_n\}$ during $T+1$ time instants. Time can be discrete or continuous, but in this work we will only consider discrete time. Each random variable X_i at each time instant $t \in [0, T]$ will be denoted as X_i^t and the set of variables in a given time instant t can be noted as $\boldsymbol{X}^t = \{X_1^t, X_2^t, ..., X_n^t\}$. The time series as a whole is under this definition a vector of sets of variables observed in the T different time instants and will be noted as $\boldsymbol{X}^{0:T} = (\boldsymbol{X}^0, ...\boldsymbol{X}^T)$. Additionally, we refer to the *time family* $\boldsymbol{X}_i^{0:T}$ of a variable i as the set of variables X_i^t in all the different time instants, $\boldsymbol{X}_i^{0:T} = \{X_i^0, X_i^1, ...X_i^T\}$.

The simpler scenario would be a *univariate time series*, a specific time series so that there is a single variable X^t per time instant instead of a set of variables $\boldsymbol{X}^t$ or, in other words, a time series with a single time family. Formally, $\boldsymbol{X}^{0:T} = (X^0, X^1..., X^T)$. On the other hand, we might have *multivariate series* ($n > 1$ variables), dealing with more than one variable and dependencies among them. Examples of univariate time series might be a sequence of observation of the temperature in a single place every day or the rent prices in California for 20 years, whereas their multivariate counterparts might be the temperature reading at ten nearby places ($n = 10$) every day and a set of information about California housing, such as mean rent price, family income, number of housing enterprises (n=3) over 20 years.

Box et al. (2015) identify five key application fields. It is important to note that the original authors usually refer to univariate time series, unless the contrary is explicitly stated. Even though, these five task can be extrapolated to multivariate time series:

1. Forecasting: Using the information about the variables in the known instants from 0 to t to forecast the value of variables in the future instant $t+1$. In other words, given $\boldsymbol{X}^{0:t}$, forecast the value of X^{t+1} (or $\boldsymbol{X}^{t+1}$ if dealing with multivariate series).

2. Estimation of transfer functions. This task consist in constructing a dynamic model in which we have an input time series $\boldsymbol{X}^{0:T}$ and an output to be predicted $\boldsymbol{Y}^{0:T}$.

3. Analysis of effects of unusual intervention events to a system. The objective is to study the impact of an intervention in a system, i.e. what would happen if we modify or introduce an unusual variable.

4. Analysis of multivariate time series, studying and defining the dynamic relations between the different variables

5. Discrete control systems, related with the monitoring of dynamic system to ensure a quality functioning.

In our work, we will mainly focus on task 4, analysis of multivariate time series. Given a dynamic causal model that we will present in the following sections, our goal will be to highlight the evolution of the dependencies across the time dimensions, explaining how these changes have an impact in the causal relations between variables. If we are dealing with a simple graphical model, this task can be trivial, since the study of causality differences of the models at two given instant can be reduced to visualise the differences of the arcs that establish causal relations between a set of a few variables. Although the theory tells us that using interpretable models, such as dynamic Bayesian networks, these relations changes can be understood, this ideal and transparent scenario can be hindered by three difficulties: (1) the number of variables of the multivariate series and thus the number of possible relations between them; (2) more subtle changes that are not possible to be qualitatively described, such as two variables that still have a causal relation with the past instant, but it has changed quantitatively, i.e. the conditional probability distribution given the previous changes over time; and (3) the number of time instants and relations across time instants.

Once the relations have been studied (task 4), it is of our interest to study tasks 1 (forecasting) and 3 (analysis of interventions) to gain a better insight. Understanding how the relations between $\boldsymbol{X}^{0:t}$ and $\boldsymbol{X}^{t+1}$ are set in motion to estimate the forecast, can be valuable to generalise and understand the relations themselves. Identically, we can study how an intervention can alter the progression and the relations of time series to verify how robust they are.

2.1.1. Climate time series

An example of time series data can be found in climate sciences, since in such field we make one or many observations over time. Climate data is usually presented in one of the following ways (Cano et al., 2004):

- *Climatological databases*, which consist of local observations on different points of the Earth across time. For instance, observations taken by meteorological stations.

- *Gridded databases*, in which the data is structured into an ordered 2D or 3D grid. Each time point will contain as many grids as variables of our interests (a grid for surface temperature, another for radiation ...)

- *Reanalysis databases*, which can be considered a mix of the aforementioned cases. These datasets are gridded as the previous one, but they are composed of gathered information from different sources and information may even be interpolated (using expert knowledge rather than pure mathematics).

An example of reanalysis database is the NCEP/NCAR Reanalysis I (Kalnay et al., 1996), which contains information about many variables such as temperature or humidity in the surface of the Earth and at various altitudes. In this work, we will use the database NCEP/DOE Reanalysis II (Kanamitsu et al., 2002), an improved version of the aforementioned database. A more thorough description of the dataset is provided in Section 4.2.

In the literature, we can find many works that use this database for climate analysis. One of the parts of said works that interest us the most is how to deal with the extremely high dimensionality of the dataset. Each grid has already a size of 144×73, which equals 10512 features. That quantity should be multiplied by the number of variables to observe (temperature, humidity,...) and then we have to consider that number in each time point. In short, 10512 times the number of features per time point. This enormous number of features implies another problems:

- Performance issues, as the correlation between many points in the grid will be extremely high (two adjacent points will likely have similar values), which can result in performance downgrades depending on the model selected. For instance, a naive Bayes classifier assume conditional independence of the predictor variables given the class.
- The model will not be interpretable, even if we use the simplest model possible. The number of features is too high to be understood in its totality and the features themselves are too cryptic, as they are latitude and longitude points, something that people do not commonly use when talking about location. They tend to refer to Earth regions or countries, not to exact points.

Different works deal with the problem of dimensionality in different ways. One of the simplest approaches is to subsample or to lower the resolution of the data, which results in a significant alleviation of the computational burden (Whitaker et al., 2008). For instance, reducing the resolution by only half (from 144×73 to 72×36) results in 2592, four times less features. Similarly, one could also aggregate time points.

Fountalis et al. (2014) propose a clustering algorithm that divides the Earth into connected regions, reducing then dimensionality and adding interpretability (regions instead of coordinates).

Another approach used by Runge et al. (2015) and Vejmelka et al. (2015) is to apply principal component analysis (PCA) and select the components that explain the most variance. In both works, 60 components are selected. In ours, we will select around 40 components. Bueso et al. (2020) use a non-linear version of PCA.

While in most cases PCA is regarded as a method that dooms our model to non-interpretability, here PCA allows us to detect zones in which the correlation is quite high and a lot of variance is explained. PCA (and more specifically rotated PCA) has proven in the aforementioned work to find *modes of variability*, zones of the Earth whose behaviour greatly affects the climate of the Earth. Examples of these modes of variability are the El Niño Southern Oscillation (ENSO) or the Northern Atlantic Oscillation (NAO). Thus, we would be studying the relation between the well-known (and secondary) oscillation of the Earth.

2.2. Bayesian networks

2.2.1. Brief description of Bayesian networks

A Bayesian network (Pearl, 1988; Koller and Friedman, 2009) $\mathcal{B} = (\mathcal{G}, \theta)$ is a probabilistic graphical model that encodes a joint probability distribution (JPD) $P(X_1, X_2, ..., X_n)$ over a set of variables $\boldsymbol{X} = \{X_1, X_2, ..., X_n\}$. Qualitatively, a Bayesian network is a directed acyclic graph $\mathcal{G} = (\boldsymbol{X}, A)$ that represents the conditional (in)dependencies between the variables in $\boldsymbol{X}$ and where $\boldsymbol{A}$ are the directed arcs of the graph. The local Markov property establishes that every variable X_i is independent of its non-descendants given its parents $\boldsymbol{Pa}_{X_i}$. Quantitatively, a Bayesian network factorises the JPD $P(X_1, X_2, X_3, ..., X_n)$ in the vector of parameters $\boldsymbol{\theta} = \{\boldsymbol{\theta_1}, \boldsymbol{\theta_2}, ..., \boldsymbol{\theta_n}\}$, storing only local conditional probability distributions (CPD) $\boldsymbol{\theta_i} = P(X_i|\boldsymbol{Pa}_{X_i})$ in each node X_i. The JPD factorises as:

$$P(X_1, X_2, X_3, ..., X_n) = \prod_{i=1}^{n} P(X_i|\boldsymbol{Pa}_{X_i}). \tag{2.1}$$

Trying to represent the JPD would lead to a probability table of exponential size, $2^n - 1$ in the case of binary variables (the probabilities of all the different value combinations of all variables), which in practice is intractable. Thanks to the Bayesian networks capability of compacting the JPD considering conditional independences the representation, estimation of $\boldsymbol{\theta}$ and computation of inferences are much more efficient. Furthermore, using graphs makes the representation of the domain much more intuitive and allows for better understanding the encoded knowledge.

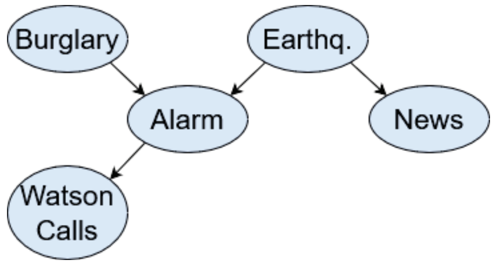
The adjective Bayesian points to the subjective nature of the a priori information, to the distinction between causal and evidential models of reasoning and to the reliance on the Bayes's rule for knowledge updating (Pearl, 2009). This rule allows us to update our belief in a hypothesis h after new evidences e were observed:

$$P(h|e) = \frac{P(e|h) \cdot P(h)}{P(e)} \tag{2.2}$$

which is the key for performing performing inference in Bayesian networks.

Take as an example the "Watson calls" Bayesian network structure, in Figure 2.1. The network describes a situation in which our protagonist, Mr. Holmes, is at the office, and he might receive a call from his neighbour, Dr. Watson. He will call Mr. Holmes when he hears the burglar alarm, that is triggered when a burglar enters in Mr. Holmes house. However, in Los Angeles there are small earthquakes, that may trigger the alarm as well and, in case the earthquake is big enough, it might be reported on the news. The joint probability distribution can be factorised as $P(A, B, E, N, W) = P(B)P(E)P(A|B, E)P(N|E)P(W|A)$ and the quantitative information about the conditional probability distributions can be visualised in Table 2.1.

In our work, we will refer with lower case $\boldsymbol{x}$ to a (joint) value assignment for a variable or set of variables $\boldsymbol{X}$, and $\Omega(\boldsymbol{X})$ will denote the set of all possible value assignments for $\boldsymbol{X}$.



$$P(A, B, E, N, W) = P(B)P(E)P(A|B, E)P(N|E)P(W|A)$$

Figure 2.1: Directed acyclic graph of the "Watson calls" network

Table 2.1: Conditional probability tables of the "Watson calls" network

		Alarm	
Burg.	Earthq.	t	f
f	f	0.05	0.95
f	t	0.40	0.60
t	f	0.90	0.10
t	t	0.99	0.01

Burglar	
t	f
0.20	0.80

Earthquake	
t	f
0.15	0.85

	News	
Earthq.	t	f
f	0.10	0.90
t	0.60	0.40

	W. calls	
Alarm	t	f
f	0.01	0.99
t	0.70	0.30

2.2.2. Conditional independence and d-separation

Formally, we can define conditional independence as follows: Given the disjoint subsets of nodes $\boldsymbol{X}_A, \boldsymbol{X}_B, \boldsymbol{X}_C \in \boldsymbol{X}$, $\boldsymbol{X}_A$ and $\boldsymbol{X}_B$ are conditionally independent given $\boldsymbol{X}_C$ iff $P(\boldsymbol{X}_A, \boldsymbol{X}_B | boldsymbol X_C) = P(\boldsymbol{X}_A|\boldsymbol{X}_C) \cdot P(\boldsymbol{X}_B|\boldsymbol{X}_C)$. This can be noted as $I_P(\boldsymbol{X}_A, \boldsymbol{X}_B|\boldsymbol{X}_C)$.

An efficient way to identify conditional independences in the graph is through the d-separation criterion. Given the disjoint subsets of nodes $\boldsymbol{X}_A, \boldsymbol{X}_B, \boldsymbol{X}_C \in \boldsymbol{X}$, $\boldsymbol{X}_A$ and $\boldsymbol{X}_B$ are d-separated by $\boldsymbol{X}_C$ if in every path between any node $X_i \in \boldsymbol{X}_A$ and any node $X_j \in \boldsymbol{X}_B$ there is an intermediate node X_k so that one of the following is true:

- X_k is a converging connection in the path, i.e., the nodes of the path adjacent to X_k point to X_k, and X_k and its descendants are not in the set $\boldsymbol{X}_C$
- X_k is not a converging connection in the path and $X_k \in \boldsymbol{X}_C$

The d-separation of $\boldsymbol{X}_A$ and $\boldsymbol{X}_B$ by $\boldsymbol{X}_C$ is denoted as $\boldsymbol{X}_A \perp_{\mathcal{G}} \boldsymbol{X}_B|\boldsymbol{X}_C$.

Conditional independences are identified using the d-separation theorem (Verma and Pearl, 1990a), which states that, given a Bayesian network with JPD P and DAG $\mathcal{G} = (\boldsymbol{X}, A)$, if we have a d-separation in the graph, $\boldsymbol{X}_A \perp_G \boldsymbol{X}_B|\boldsymbol{X}_C$, then we also have a conditional independence in the JPD, $I_P(\boldsymbol{X}_A, \boldsymbol{X}_B|\boldsymbol{X}_C)$, with $\boldsymbol{X}_A, \boldsymbol{X}_B, \boldsymbol{X}_C \in \boldsymbol{X}$ and disjoint (see an example in Figure 2.2). Formally:

$$\forall \boldsymbol{X}_A, \boldsymbol{X}_B, \boldsymbol{X}_C \in \boldsymbol{X} \text{ such that } \boldsymbol{X}_A, \boldsymbol{X}_B, \boldsymbol{X}_C \text{ are disjoint, } \boldsymbol{X}_A \perp_G \boldsymbol{X}_B | \boldsymbol{X}_C \Rightarrow I_P(\boldsymbol{X}_A, \boldsymbol{X}_B | \boldsymbol{X}_C)$$

Notice that the opposite does not necessarily hold, i.e. the DAG $\mathcal{G}$ does not encode all the conditional independences in P, but every triplet of d-separated nodes is a conditional independence. This type of graph is called *I-map*.

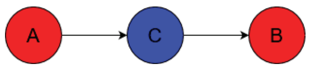


Figure 2.2: An example of the d-separation theorem. Since nodes A and B are d-separated by C, we know then that variables A and B are conditionally independent given C

As a result of the d-separation theorem, two conditions hold in Bayesian networks (Pearl, 2009; Spirtes et al., 2000):

- The *local Markov condition* or *causal Markov condition*, which has been briefly introduce in the previous subsection. It states that a node X_i is conditionally independent of its non-descendants given its parents. Formally, $\forall X_i \in \boldsymbol{X},\ I_P(X_i, \boldsymbol{X} \setminus \{\boldsymbol{Desc}_{X_i} \cup \boldsymbol{Pa}_{X_i}\} | Pa_{X_i})$, where $\boldsymbol{Desc}_{X_i}$ is the set of descendants of X_i.

- The *global Markov property* states that a node X_i is conditionally independent of any other node in the Bayesian network given its parents, children and parent's children (spouses). This last set of nodes is known as the *Markov blanket*, of X_i, $\boldsymbol{MB}_{X_i}$. Formally, $\forall X_i \in \boldsymbol{X},\ I_P(X_i, \boldsymbol{X} \setminus \boldsymbol{MB}_{X_i} | \boldsymbol{MB}_{X_i})$.

2.2.3. Causal Bayesian networks

Unlike the intuition might suggest, in the graph $\mathcal{G} = (\boldsymbol{X}, A)$ that qualitatively describes a Bayesian network, the arcs A do not necessarily describe causal relations among the set of variables $\boldsymbol{X}$, but direct probabilistic dependences that might (or might not) represent causal relations too. Take the three Bayesian networks in Figure 2.3. While at first glance it may seem that they are different networks, all of them actually represent the same set of conditional independences, which can be verified applying the Bayes' rule to their probability distribution. Verma and Pearl (1990b) refer to this as equivalence classes. Although this concept is interesting for reducing the search space while learning Bayesian networks from data, in this work we want to focus on what this graph dissonance means for causality.

Since arcs do not necessarily means causation, having this type of redundancies is perfectly acceptable. However, in causal Bayesian networks (Pearl, 2009), the DAG represents a causal network, i.e. a graph that represents causal relations and therefore only one of the three graphs in Figure 2.3 is representing correctly the causality in the process.

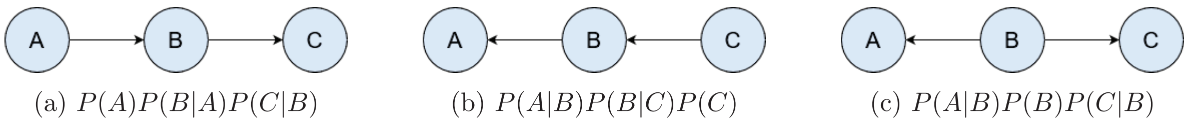


Figure 2.3: Three DAGs in the same equivalence class

Before diving deeper in causal Bayesian networks, it is important to note that there are many definitions and properties of causality. One of the first and most well-known is the Granger causality (Granger, 1969), which states that a variable X causes Y if X provides significant information of Y. This idea is particularly useful for prediction, although it might not indicate real causation between X and Y, but a temporal relation (Granger and Newbold, 1977). In causal Bayesian networks and in many modern causality works of our concern (Runge et al., 2015; Runge, 2018), the definition based on conditional independence by Pearl (2009) and Spirtes et al. (2000) is taken into account which, intuitively, considers that X causes Y if intervening the model in a way that X is modified also affects Y, but not the other way around. In causal Bayesian networks, the correct establishment of the causality relations is usually verified by intervening similarly to how it is described in the aforementioned works (Koller and Friedman, 2009).

In practice, causal Bayesian network models are an active field in research whose applications are quite far-reaching, such as in the research of epidemiological factors (Rothman and Greenland, 2005), neural connectivity (Sanchez-Romero et al., 2019), and studying relations between events in climate sciences (Runge et al., 2014, 2015, 2019), which is the main application field of this work. Our specific goal is to (1) actually use causal Bayesian networks instead of generic causal model learning with no expert and (2) specially elaborate explanations that can be given in that domain, and we leave most of the actual causality study out of the scope of this work.

2.2.4. Gaussian Bayesian networks

The variable domain of the Bayesian networks is not limited to discrete variables, instead we can find the use of continuous variables in the so-called *continuous Bayesian network* (not to be confused with *continuous-time dynamic Bayesian networks*, which are out of the scope of this work). Both continuous and discrete variables are used in *hybrid Bayesian networks*.

A widely used type of Bayesian network are the *Gaussian Bayesian networks* (GBNs) (Lauritzen and Wermuth, 1989). These networks factorise a multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{\mu};\boldsymbol{\Sigma})$ as a product of conditional linear Gaussians variables and the conditional density of each variable i given its parents $\boldsymbol{Pa}_{X_i}$ is represented as:

$$f(X_i|\boldsymbol{Pa}_{X_i}) \sim \mathcal{N}(\beta_{X_i} + \sum_{\boldsymbol{P}} \beta_{X_iP_j}(p_j - \mu_j)\ , v_{X_i}), \tag{2.3}$$

where $\boldsymbol{P}$ refers to the parents of X_i, $\boldsymbol{P} = \boldsymbol{Pa}_{X_i}$.

In plain words, the mean of normal distribution of each node is a linear combination of its parents $\boldsymbol{P}$, where the coefficient $\beta_{X_iP_j}$ represents the weight of the parent P_j in the linear combination and β_{X_i} is an intercept term. p_j is the assignment value of the parent P_j and μ_j is the mean of the normal distribution of that parent. The conditioned standard deviation v_{X_i} is not directly affected by the value assignment of the parents of X_i.

If we want to get the mean μ_{X_i} and standard deviation σ_{X_i} of a certain variable X_i of the multivariate normal, it can be computed as follows:

1. $\mu_{X_i} = \beta_{X_i} + \sum_{\boldsymbol{P}} \beta_{X_iP_j}([arg\,max_{P_j} P(P_j)] - \mu_j)$

2. $\sigma_{X_i} = \sqrt{v_{X_i}^2 + \sum_{\boldsymbol{P}} \beta_{X_i P_j}^2 \sigma_{P_j}^2}$

Note that μ_j and σ_j should be obtained recursively. At a node with no parents, $\mu_{X_i} = \beta_{X_i}$ and $\sigma_{X_i} = v_{X_i}$. Obtaining covariances is also possible through a close formula.

When intervening or making observations, inference (Section 2.2.5) shall be made to compute the mean and standard deviation. However, if only the parents of X_i are observed and X_i d-separates the descendants of X_i from its parents, we could use the rules above to obtain the probability distribution of X_i given an observation $\boldsymbol{p}$ on the parents $\boldsymbol{P}$, namely $f(X_i|\boldsymbol{p})$. μ_j will be replaced by the observation on the parent and the variance of each parent σ_j will be zero, since observing p_j equals to replace the density function by a normal $\mathcal{N}(p_j, 0)$, making $\sigma_{X_i} = v_{X_i}$. Note also that, if the observation $p_j = \mu_j$, then $\mu_{X_i} = \beta_{X_i}$.

While this condition is very restricting and even unlike to happen in a real case scenario, this premise may hold in a time-varying dynamic Bayesian network, a type of network that will be the object of study in this work and that will be reviewed in Section 2.3.2.

2.2.5. Inference

If we want to compute the probability of $\boldsymbol{X}_i \subseteq \boldsymbol{X}$ without considering the factorization of the JPD made with a Bayesian network, we would need to compute the JPD summing over all the value assignments of the variables that are not $\boldsymbol{X}_i$ (i.e., the set $\boldsymbol{X} \setminus \boldsymbol{X}_i$), an operation that is called marginalization:

$$P(\boldsymbol{X}_i) = \sum_{\boldsymbol{X} \setminus \boldsymbol{X}_i} P(\boldsymbol{X}).$$

Storing the JPD or computing it via brute force for any value assignment $\boldsymbol{x} \in \Omega(\boldsymbol{X})$ (where $\Omega(\boldsymbol{X})$ refers to the domain of $\boldsymbol{X}$) are both unfeasible, which highlights the need for Bayesian networks in inference. Since we are representing the JPD as the product of the conditional distribution of the variables given their parents, we get this expression:

$$P(\boldsymbol{X}_i) = \sum_{\boldsymbol{X} \setminus \boldsymbol{X}_i} \prod_{j=1}^{n} P(X_j|\boldsymbol{Pa}_{X_j}).$$

When having a GBN, the sum shall be replaced by an integral:

$$f(\boldsymbol{X}_i) = \int_{\boldsymbol{X} \setminus \boldsymbol{X}_i} \prod_{j=1}^{n} f(X_j|\boldsymbol{Pa}_{X_j}) d(\boldsymbol{X} \setminus \boldsymbol{X}_i).$$

If we take the example of the Watson calls network (see Figure 2.1), the probability distribution of A can be computed as follows:

$$P(A) = \sum_{B,E,N,W} P(B)P(E)P(A|B,E)P(N|E)P(W|A).$$

This method is still a brute force approach, since we need to compute the sum of the product of conditional probabilities, which is an exponential sum with the number of variables (and their possible states) in the network. There are several approaches to simplify this problem (although remaining untractable) that can be categorised into two types:

- Exact inference, which gives exact results similarly to the brute force approach.
- Approximate inference, that uses sampling and similar methods to give a good approximation of the inference.

In this work, the study on how to perform inference is left out, focusing on the type of queries rather than on how to compute them. It is still worth mentioning that the software that we will use, bnlearn (Scutari, 2010), uses probabilistic logic sampling (Henrion, 1988) and likelihood weighting (Fung and Chang, 1990; Shachter and Peot, 1990), which are approximate inference methods.

2.2.5.1. Queries in Bayesian networks

There are many type of queries that a Bayesian network can answer.

The simplest query that can be done is the likelihood of an evidence $\boldsymbol{e}$, which can be computed as follows:

$$P(\boldsymbol{e}) = \sum_{\boldsymbol{X}_u} P(\boldsymbol{e}, \boldsymbol{X}_u), \tag{2.4}$$

where the set $\boldsymbol{X}_u$ represents the variables in $\boldsymbol{X}$ that are unobserved, i.e., that are not part of the evidence set $\boldsymbol{E}$ ($\boldsymbol{X}_u = \boldsymbol{X} \setminus \boldsymbol{E}$).

Bayesian networks can help us to compute (joint) posterior probabilities, the probability of a certain variable X_i (or subset of variables $\boldsymbol{X}_i$) given an evidence. If we denote $\boldsymbol{X}_u$ as the subset of unobserved variables of $\boldsymbol{X}$ that are not in $\boldsymbol{E}$ nor in $\boldsymbol{X}_i$, i.e, $\boldsymbol{X}_{\boldsymbol{u}} = \boldsymbol{X} \setminus \{\boldsymbol{E} \cup \boldsymbol{X}_i\}$, we can formulate the posterior probability of X_i given e as follows:

$$P(\boldsymbol{X}_i|\boldsymbol{e}) = \sum_{\boldsymbol{X}_u} P(\boldsymbol{X}_i, \boldsymbol{X}_u|\boldsymbol{e}) = \sum_{\boldsymbol{X}_u} \frac{P(\boldsymbol{X}_i, \boldsymbol{X}_u, \boldsymbol{e})}{P(\boldsymbol{e})}. \tag{2.5}$$

A very interesting type of query is the abduction, which consists of finding the most plausible explanation A for an effect B. In Bayesian networks, this translates to computing the maximum a posteriori (MAP) of the unobserved variables $\boldsymbol{X}_u$ given an evidence $\boldsymbol{e}$ that is desired to be explained, i.e. finding the value assignment $\boldsymbol{x}^*{}_u$ for $\boldsymbol{X}_u$ that maximises the posterior probability given the evidence.

$$\boldsymbol{x}^*{}_u = arg\, max_{\boldsymbol{X}_u} P(\boldsymbol{X}_u|\boldsymbol{e}). \tag{2.6}$$

This type of query is more specifically referred to as most probable explanation (MPE) or total abduction, since the joint probability for all the unobserved variables ($\boldsymbol{X}_u = \boldsymbol{X} \setminus \boldsymbol{E}$) is maximised or, in other words, because we aim to explain the evidence with the whole set of remaining variables.

However, we might be interested in explaining the evidence $\boldsymbol{e}$ just with a subset of the unobserved variables, the explanation or hypothesis set $\boldsymbol{H} \subset \boldsymbol{X}_u$. This type of query is referred to as partial abduction and can be similarly computed:

$$\boldsymbol{h}^* = arg\, max_{\boldsymbol{H}} P(\boldsymbol{H}|\boldsymbol{e}). \tag{2.7}$$

We might be interested in the partial abduction in order to avoid over-specific explanations, but without specifying the explanation set $\boldsymbol{H}$. This approach is known as most relevant explanation (Yuan et al., 2011) and consists of finding the most probable explanation using subsets of $\boldsymbol{X}_u$ maximizing the Bayes generalised factor (BGF) (Fitelson, 2007).

$$\boldsymbol{h}^* = arg\, max_{\boldsymbol{H} \in \mathcal{P}^+(\boldsymbol{X}_u)} BGF(\boldsymbol{H}; \boldsymbol{e}) \tag{2.8}$$

where $\mathcal{P}^+(\boldsymbol{X}_u)$ is the powerset of $\boldsymbol{X}_u$ excluding the empty set $\{\emptyset\}$ and BGF is defined as follows:

$$BGF(\boldsymbol{h}; \boldsymbol{e}) = \frac{P(\boldsymbol{e}|\boldsymbol{h})}{P(\boldsymbol{e}|\overline{\boldsymbol{h}})} = \frac{P(\boldsymbol{h}|\boldsymbol{e}) \cdot P(\overline{\boldsymbol{h}})}{P(\overline{\boldsymbol{h}}|\boldsymbol{e}) \cdot P(\boldsymbol{h})} \tag{2.9}$$

2.2.6. Bayesian network comparison

Given two Bayesian networks, we might be interested in comparing them. This can be useful when learning a Bayesian network from data and comparing it with the expected result. In our work, we aim to use this comparison as an explanation for phenomena such as concept drift. We can distinguish between comparisons of the structure of the network and its probability distribution.

De Jongh and Druzdzel (2009) offer an excellent review on the topic of comparing the structures of causal Bayesian networks, focusing on comparing both DAGs and complete partially DAGs, structures that represent equivalence classes (see Section 2.2.3). In this work, we are only interested in comparing DAGs. This is done by measuring the distance between the two DAGs.

Colace et al. (2004) propose a structural distance referred to as "global" that measures the amount of arcs that are directed in the same direction in both network structures $\mathcal{G}_1$ and $\mathcal{G}_2$ (*correctly oriented arcs*) in relation to two kinds of arcs: the arcs present in both networks regardless of their direction (*shared arcs*) and the arcs present in just one of the networks (*diff. arcs*).

$$Global(\mathcal{G}_1, \mathcal{G}_2) = \frac{\sum correctly\ oriented\ arcs}{\sum shared\ arcs + \sum diff\ arcs} \tag{2.10}$$

Another interesting metric is the Hamming distance, which in the context of Bayesian networks measures the number of changes that have to be done to a Bayesian network structure to get the other one. The possible changes are: arc addition, arc deletion and arc reversal. As such, it can be defined as the sum of arcs that are present in both networks but oriented in opposite direction (*incorrectly oriented arcs*) and, again, the arcs present just in the first network and in the second, but not in both (De Jongh and Druzdzel, 2009):

$$Hamming(\mathcal{G}_1, \mathcal{G}_2) = \sum incorrectly\ oriented\ arcs + \sum diff\ arcs \tag{2.11}$$

Both measures are valid for comparing causal networks, since the direction of the arcs is being taken into account in both distances. Additionally, both of them are distances.

A well-known way to compare probability distributions (and Bayesian networks quantitatively, by extension) are the *f-divergences* (Rényi, 1961), functions that measure the distance between two probability distributions P and Q over the same set of random variables $\boldsymbol{X}$. A popular instantiation of f-divergence is the Hellinger distance metric:

$$H(P||Q) = \frac{1}{\sqrt{2}} \left\|\sqrt{P} - \sqrt{Q}\right\|_2, \tag{2.12}$$

where $\left\|V\right\|_2$ refer to the Euclidean norm of the vector V.

With continuous distributions, the Hellinger distance square of two density functions f and g is computed as follows:

$$H^2(P,Q) = \frac{1}{2} \int \left(\sqrt{P(\boldsymbol{x})} - \sqrt{Q(\boldsymbol{x})}\right)^2 d\boldsymbol{x} = 1 - \int \sqrt{P(\boldsymbol{x})Q(\boldsymbol{x})}\, d\boldsymbol{x}. \tag{2.13}$$

There is a closed formula for many probability distributions, including for the univariate and multivariate normal. The former, for $p \sim \mathcal{N}(\mu_P, \sigma_P)$ and $q \sim \mathcal{N}(\mu_Q, \sigma_Q)$ is a follows:

$$H^2(P,Q) = 1 - \sqrt{\frac{2\sigma_P\sigma_Q}{\sigma_P^2 + \sigma_Q^2}}\, e^{-\frac{1}{4}\frac{(\mu_P-\mu_Q)^2}{\sigma_P^2+\sigma_Q^2}}. \tag{2.14}$$

Another widely used measure in statistics that is an instantiation of the f-divergence is the Kullback-Leibler divergence (Kullback and Leibler, 1951):

$$KL(P||Q) = \sum_{\boldsymbol{x}} P(\boldsymbol{x}) log \frac{P(\boldsymbol{x})}{Q(\boldsymbol{x})} \tag{2.15}$$

For continuous distributions, the summation must be changed for an integral. Additionally, there is a closed-formula if f and g are univariate or multivariate normal distributions. The formula for the univariate case is:

$$KL(f||g) = \log \frac{\sigma_g}{\sigma_f} + \frac{\sigma_f^2 + (\mu_f - \mu_g)^2}{2\sigma_g^2} - \frac{1}{2} \tag{2.16}$$

However, the Kullback-Leibler divergence is not a real metric, since it is not symmetric and the triangular inequality does not hold. Usually, P is considered a reference “true" distribution and Q an arbitrary distribution that we want to compare. If we want to consider a metric based on the same principle as the Kullback-Leibler divergence, we can consider the Jensen-Shannon divergence (Lin, 1991):

$$JSD(P||Q) = \frac{1}{2}KL(P||M) + \frac{1}{2}KL(Q||M) \tag{2.17}$$

where M is the arithmetic mean between P and Q, $M = \frac{1}{2}(P+Q)$. For two Gaussian distributions (univariate or multivariate), the Jensen-Shannon divergence yields a closed formula if we take as the density m the geometric mean (instead of the arithmetic) of f and g, $M_G(f,g)$ (Nielsen, 2019). The probability density function of m is therefore

$$m(\boldsymbol{x}) = \frac{M_G(f(\boldsymbol{x}), g(\boldsymbol{x}))}{\int_{-\infty}^{\infty} M_G(f(\boldsymbol{x}), g(\boldsymbol{x}))d\boldsymbol{x}} = \frac{f(\boldsymbol{x})^{0.5} g(\boldsymbol{x})^{0.5}}{\int_{-\infty}^{\infty} f(\boldsymbol{x})^{0.5} g(\boldsymbol{x})^{0.5} d\boldsymbol{x}}. \tag{2.18}$$

The denominator is a normalizing factor for the product of the numerator. As the density function of M is a normalised product of two density functions of exponential families, M will be also an exponential distribution of the same family as P and Q, a Gaussian in case that both P and Q were also Gaussians. In Nielsen (2019), a formula is provided for the case of the multivariate Gaussian case and, in this work (Section 3.3.3), we derive a simple closed formula for finding M for the univariate case and derive our own closed formula for the Jensen-Shannon divergence for two univariate Gaussians from Equations (2.16) and (2.17), instead of using the complex formula for the multivariate case provided in the aforementioned paper.

2.3. Dynamic Bayesian networks

Dynamic Bayesian networks (DBNs) (Dean and Kanazawa, 1989; Murphy, 2002) are an extension of Bayesian networks that let us describe time series and work with them. They can be understood as well as generalization of Kalman filters (Kalman, 1960) and hidden Markov models (Rabiner, 1989). The time in such Bayesian networks can be discrete or continuous, but in this work the continuous-time Bayesian networks (Nodelman et al., 2002) are out of the scope. In a discrete-time dynamic Bayesian network the time dimension is discretised in slices and a time stamp is added to each variable X_i, having then X_i^t with $t \in \{0, 1, ..., T\}$. Therefore, the JPD of the process encodes a set of variables over $T+1$ different time instants $\boldsymbol{X}^{0:T} = (\boldsymbol{X}^0, \boldsymbol{X}^1, ..., \boldsymbol{X}^T)$, where $\boldsymbol{X}^t = \{X_1^t, X_2^t, ..., X_n^t\}$. Each variable X_i^t is represented by a node with the same name.

In its most generic form, the JPD defined by the dynamic Bayesian network is factorised as:

$$P(\boldsymbol{X}^0, \boldsymbol{X}^1, ..., \boldsymbol{X}^T) = P(\boldsymbol{X}^0) \prod_{t=1}^{T} P(\boldsymbol{X}^t | \boldsymbol{X}^{0:t-1}).$$

Before further introducing DBNs, we present the *Markov assumption* and the concept of *stationarity*, two important notions for the rest of the section.

1. The Markov assumption holds if, in a process over time, the instant $t+1$ is only conditioned by the instant t ($t \in \mathbb{N}^+$. In DBNs, the Markov assumption can be formalised in terms of conditional independences as follows:

   $$I_P(\boldsymbol{X}^{t+1}, \boldsymbol{X}^{0:t-1} \mid \boldsymbol{X}^t),\ t \in \mathbb{N}^+$$

   We extend this definition and present the concept of *τ-th order Markov assumption* (Ghahramani, 1997). A process over time has a Markovian order of τ if instant $t+1$ is only conditioned by the previous τ instants. Note that having a first Markovian order is equivalent to fulfill the Markov assumption. In a DBN, it can be formulated as:

   $$I_p(\boldsymbol{X}^{t+\tau+1}, \boldsymbol{X}^{0:t-1} \mid \boldsymbol{X}^{t:t+\tau}),\ t, \tau \in \mathbb{N}^+$$

   .

2. A stochastic Markovian time process is said to be stationary if the same relations hold over all the time instants. Formally, for a certain Markovian order $\tau \in \mathbb{N}^+$, the process is stationary if:

   $$P(\boldsymbol{X}^{t+\tau} \mid \boldsymbol{X}^{t:t+\tau-1}) = P(\boldsymbol{X}^{t'+\tau} \mid \boldsymbol{X}^{t':t'+\tau-1}),\ \forall t, t' \in \{0, 1, ..., T\}$$

   This concept (if holds) allows for *template-based representation.* Instead of having a representation for each specific node connecting with another specific node, we parametrise the relations of variables with their past instants over time. For instance, instead of defining the relation between instants 1 and 2, 2 and 3 and so on (for a first Markovian order), we define the relation between the parametrised instants t and $t+1$, which can be instantiated at any moment. This can potentially lead to representing a stochastic process with thousands of instants with two small Bayesian networks, as will be explained in this section.

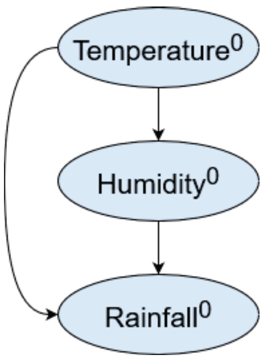
One of the simplest and most well-known type of DBN is the one that assumes *stationarity* and the *first order Markov assumption*, which can be defined by the pair $< \mathcal{B}_0, \mathcal{B}_\rightarrow >$:

- $\mathcal{B}_0$ is Bayesian network over $\boldsymbol{X}^0$, referred as prior network.
- $\mathcal{B}_\rightarrow$ is a template-based representation of a Bayesian network that defines the conditioned probability $P(X_i^{t+1} | \boldsymbol{X}^t \cup \boldsymbol{X}^{t+1} \setminus X_i^{t+1}),\ \forall i \in \{0, 1, ..., n\}$. This parameterised Bayesian network is usually referred as *transition network* and particularly, this a *2-time-slice Bayesian network* (2-TBN), which is a transition network

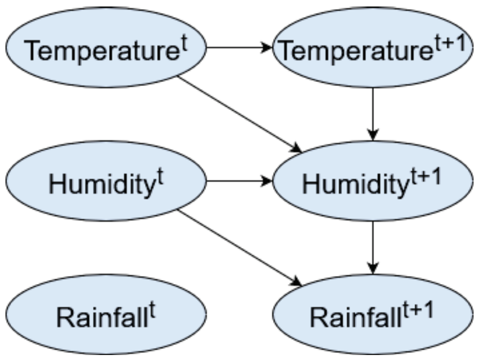
that satisfy both of the aforementioned assumptions. In a transition network, we can find both *inter-slice* and *intra-slice* edges. Inter-slice edges are the ones directed from variables in the instant t to variables in $t+1$ (the dependence of each X_i^{t+1} on $\boldsymbol{X}^t$) whereas intra-slice edges are the ones that go from any variable of $t+1$ to another one in $t+1$ (the dependence of each X_i^{t+1} on $\boldsymbol{X}^{t+1} \setminus X_i^{t+1}$).

For a grapical representation, see Figure 2.4.With $\mathcal{B}_{\rightarrow}$, we can represent a full time series in a very compact way (in total, $3n$ nodes, including the nodes in $\mathcal{B}_0$). If we want to see the full representation of the time series as a Bayesian network, we would unroll the template network over all the T time instants, giving us a network with $(T+1)n$ nodes. The JPD defined by this Bayesian network can be described as:

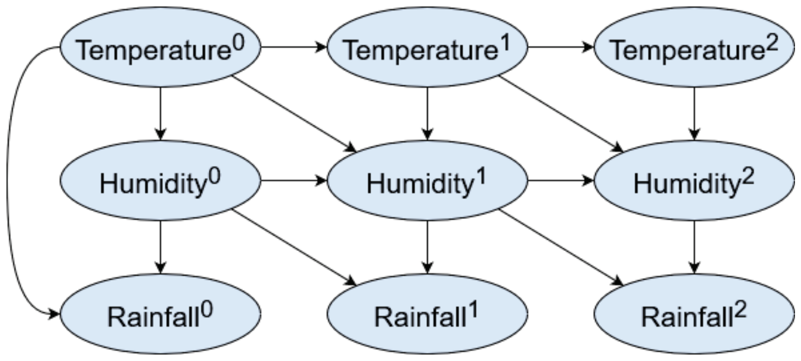
$$P(\boldsymbol{X}^0, \boldsymbol{X}^1, \boldsymbol{X}^2, ..., \boldsymbol{X}^T) = P(\boldsymbol{X}^0) \prod_{t=1}^{T} P(\boldsymbol{X}^t | \boldsymbol{X}^{t-1}).$$



(a) Prior network $\mathcal{B}_0$



(b) Transition network $\mathcal{B}_{\rightarrow}$



(c) Unrolled network in the first three instants

Figure 2.4: A stationary and first Markovian order dynamic Bayesian network structure

2.3.1. Non-stationary dynamic Bayesian networks

Many real world processes are not stationary since the relations among the variables may change both qualitatively and quantitatively, and thus *non-stationary dynamic Bayesian networks* (ns-DBNs) (Robinson and Hartemink, 2008) are required. These can be defined as DBNs in which the structure may change at a given time.

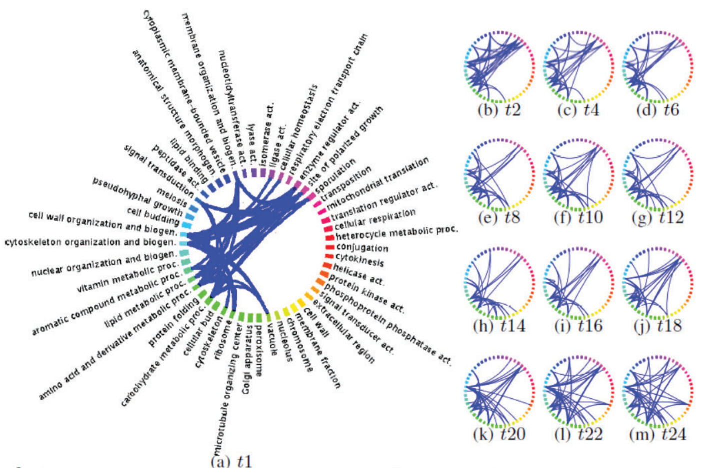
Since the process is not stationary, we need to describe more than one transition model

$P_i(\boldsymbol{X}^t|\boldsymbol{X}^{0:t-1}), i \in \{1, 2, ..., K\}$, where K is the number of transition models. The internal structure of each time slice (that is, the connection between variables of a same time stamp) may also change alongside the transition network.

The points where the transition model changes tend to be unknown and must detected a priori. In the original work by Robinson and Hartemink (2008) we have an a priori structure and sampling based on reversible jumping Markov chain Monte Carlo (RJMCMC) is used to change the networks. The sampling is also used in scenarios where the transition times or number of transitions is unknown. Jia and Huan (2010) decide to use a flexible time lag τ and, similarly, using RJMCMC find the actual time point of the change and the structure. By contrast, Grzegorczyk and Husmeier (2009) avoid using sampling for non-stationary dynamic Gaussian networks and only the parameters may change and they do it in a smooth way. The number and points of change are studied using the posterior distribution. Another interesting approach is taken by Meng et al. (2019) for data streams where, as new data arrives, the network changes progressively moving to a network in the structure neighbourhood (i.e., only an arc can be added or deleted). With this technique, they avoid abrupt changes.

2.3.2. Time-varying dynamic Bayesian networks

A critique to both stationary and non-stationary Bayesian networks points to the fact that they are not dynamic themselves, although they model a dynamic process. The structure does not change dynamically, but instead remains the same across the whole temporal process or changes in some specific scenario.



Song et al. (2009)

Figure 2.5: TV-DBN for gene regulation

In order to study the problem of gene regulation Song et al. (2009) defines the *time-varying dynamic Bayesian networks* (TV-DBN). These networks can be defined as DBNs in which the structure and parameters are learned in a way that can change at any given time, so we will have $T-1$ transition models, $P_t(\boldsymbol{X}^t|\boldsymbol{X}^{0:t-1}), t \in \{1, 2, ..., T\}$. These networks are assumed to have a first Markovian order (X_i^t can only have parents in the instant $t-1$), no intra-slice arcs are allowed and the network is also assumed to be Gaussian. In addition, arcs from X_i^{t-1} to X_i^t are forbidden, since, according to the authors, we are interested to study relations amongst different genes. All of the above conditions might be relaxed but, throughout this work, they will hold. As such, the JPD of a TV-DBN can be factorised as:

$$P(\boldsymbol{X}^0, \boldsymbol{X}^1, ..., \boldsymbol{X}^T) = P_0(\boldsymbol{X}^0) \prod_{t=1}^{T} P_t(\boldsymbol{X}^t | \boldsymbol{X}^{t-1}) = P_0(\boldsymbol{X}^0) \prod_{t=1}^{T} \prod_{i=1}^{n} P_t(X_i^t | \boldsymbol{Pa}_{X_i^t}) \tag{2.19}$$

where $\boldsymbol{Pa}_{X_i^t}$ is necessarily a subset of $\boldsymbol{X^t} \setminus X_i^t$. According to the authors, we have a Gaussian Bayesian network where each $\boldsymbol{X^t} t \in \{0, 1, ..., T\}$ can be computed as

$$\boldsymbol{X^t} = \boldsymbol{\beta_{X^t}} \cdot \boldsymbol{X^{t-1}} + \mathcal{N}(\boldsymbol{0}, \sigma^2 \boldsymbol{I}), \tag{2.20}$$

where $\boldsymbol{\beta_{X^t}}$ is a bidimensional matrix that encodes the parameters β of the connections between each node in $\boldsymbol{X^t}$ and $\boldsymbol{X^{t-1}}$ (a 0 value for a certain parameter means no connection between the corresponding variables). The variance is supposed to be the same for every single node in the network and it is not learned, hence the term $\mathcal{N}(\boldsymbol{0}, \sigma^2 \boldsymbol{I})$ representing a fixed Gaussian noise that is equal for each time slice.

A question that may arise is how we can learn the network, since we usually have a single or a small set of time series from which to learn a model that may change at any single time step. Song et al. (2009) solve the problem of data scarcity as follows:

- The problem of learning the whole structure is divided into a problem of learning the connections between each node X_i^t with their predecessors $\boldsymbol{X}^{t-1} \setminus X_i^{t-1}$, i.e. learning $P(X_i^t | \boldsymbol{X}^{t-1} \setminus X_i^{t-1})$ $\forall X_i \in \boldsymbol{X}$ $\forall t \in \{1, 2, ..., T\}$. This atomic problem is then solved using least-square optimization with $l1$-regularization to learn the connections between each X_i^t and the previous slice $\boldsymbol{X}^{t-1}$. To speed up the computation burden of fitting that many linear regression, the coefficients for X_i^t given $\boldsymbol{X}^{t-1}$ are used as the initial coefficients for the future dependencies. As we assume smoothness, this will give us a good initial point in the search space to do the optimization, speeding it up significantly.

- However, the problem of data scarcity still remains since, assuming we have a single time series from which to learn the network, we have just an instance to perform each least square optimization. To solve it, the authors decide to assume smoothness in the change of the structure of the Bayesian network. Thus, to solve the $l1$ least square problem between the target X_i^t and the previous slice $\boldsymbol{X^{t-1}}$, all the time instants of our dataset are used, thus having t training instances for each $l1$ least square problem. In addition, a weighting function that gives more weight to the training instances that are closer to the instant t is introduced (see Figure 2.6).

In addition, the authors assume that the network learned is causal, since all the arcs in the network have a certain natural causal meaning as they always flow from the past $t-1$ to the present t (as there are no intra-slice arcs).

In Wang et al. (2010) a method for online learning is introduced. The presence and absence of arcs and the changes in the parameters in the TV-DBN is understood as a temporal problem in itself and modeled as a Markov chain. The goal is to model the structure and parameters of future time instants as well as estimating absent relations in a given time instant.

In posterior works (Chu et al., 2013) (Chu et al., 2016), two new types of TV-DBN are

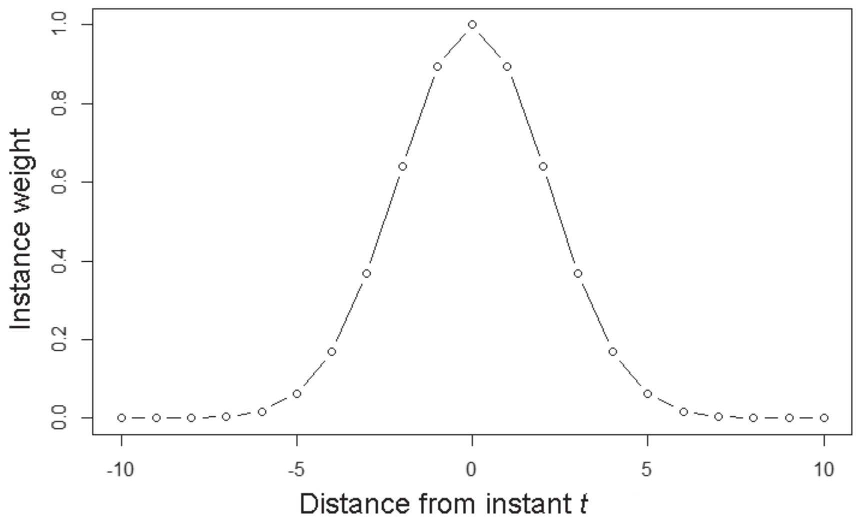


Figure 2.6: TV-DBN weighting function. Note that the function gives more weight to the instances closer to the current time instant t

introduced:

- *Relaxed TV-DBN.* These models remove the restriction of forbidding arcs between X_i^{t-1} and X_i^t, and thus now we can have arcs between the same variable in two different time steps. These networks can be understood as a generalization of the original TV-DBN, although the first Markovian order, smoothness and Gaussian data assumptions still hold.

- *Causal TV-DBN.* While it is agreed that the arcs in TV-DBN have a certain causal meaning, the structure learned via optimization does not satisfy the causal Markov condition (see Section 2.2.2), since we are learning every relation between $t-1$ and t using all the past instants. To satisfy the condition, the optimization problem between $t-1$ and t should only consider the relation between $t-2$ and $t-1$ (the past slice) and all the future relations. As such, instead of using a symmetric Gaussian kernel as weighting function (Figure 2.6), an asymmetric gamma kernel that gives 0 weight to the past instants (except to the previous one, the parents) is used (Figure 2.7).

While experiments have done with TV-DBNs, there is not an open-source implementation nor an intuitive tool to learn TV-DBN and to perform inferences over them, to the best of our knowledge. In this work, we will use the relaxed TV-DBNs, although we offer an implementation for both the original (referred to as regular in this work) TV-DBNs and causal TV-DBNs.

2.4. Explainable artificial intelligence

2.4.1. Introduction

Explainable artificial intelligence (XAI) is, at its broadest, a task aimed towards disclosing and interpreting intelligent systems with the ultimate goal of building more ethical, transparent and robust systems.

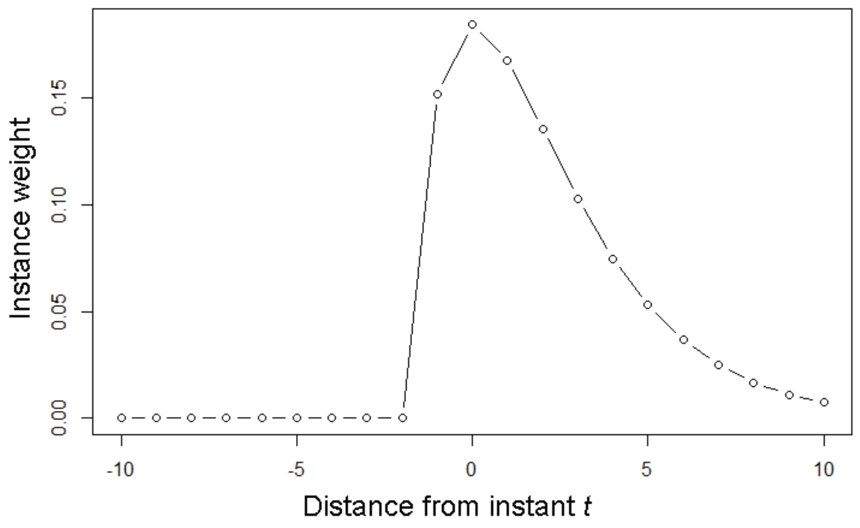


Figure 2.7: TV-DBN causal weighting function. The function uses an asymmetric gamma kernel that gives weight to the descendants and the parent instant

While there is not a concrete definition of explainability nor interpretability, we will use the one provided by Miller (2019): Interpretability is the degree to which a human can understand the cause of a decision and, by extension, interpret machine learning models. An explanation is an answer to a question of the form "What?", "How?" or "Why?" concerning the causality of a certain event that involves a cognitive process (Miller, 2019) called "abductive reasoning", which consists on deducting a plausible cause 'A' for the event 'B'. Though it is a logical fallacy (given 'B' and 'A$\rightarrow$B', 'A' is not necessarily true), in this context it can be understood as an inference to the best possible explanation (Sober, 2020) that serves our comprehension about a topic.

While some works use these terms explainability and interpretability interchangeably, (Adadi and Berrada, 2018; Tjoa and Guan, 2020), many others establish a distinction. In Montavon et al. (2018), interpretability refers to translate the abstract mechanism of the system to a comprehensible domain and explainability refers to select features from that interpretable domain to build an explanation. In the works Gilpin et al. (2018) and Barredo-Arrieta et al. (2020), "to interpret" refers to be able to understand the overall functioning of the models (by means of traceability and transparency, for instance) while "explaining" refers to summarise and denote the information with the explicit purpose of clarification. As previously pointed (Došilović et al., 2018), the previous distinctions map to the notions of model-centric and user-centric explanations presented in Edwards and Veale (2017), where interpretability can be understood as model-centric and explainability as user-centric. In a slightly different approach from the field of psychology, Broniatowski et al. (2021) claim that interpretability refers to understand the model and its predictions by means of the real world situation that is representing and explainability refers to disclose the internal mechanisms and rules that led to a decision.

In our work and as in many of the literature cited above, we refer to understand the model itself as interpretability, whereas explainability is understood as the means used to convey the information to the user (model-centric vs. user-centric). However, we ask the reader to remind that Bayesian networks are the main topic of our research and that they are fully transparent models. Therefore, the notions of model-centric and user-centric may overlap in certain situations and, as such, we allow ourselves to use interchangeably explainability and interpretability in those cases.

2.4.2. Types of explanations

We can categorise the explanations attending to various criteria. In this work, we are interested in distinguishing between model-agnostic vs. model-specific explanations (Figure 2.8) and between global and local explanations (Molnar, 2020).

Model-specific explanations refer to actually being able to interpret the model itself. This can be accomplished by means of using intrinsically interpretable models or by trying to disclose how more complex systems work. Take for instance a linear regression: a model specific explanation will try to explain the coefficients of the equation. Be aware that although transparency will always be present, the degree of interpretability is conditioned by the number of parameters of the model.

In contrast, *model-agnostic* techniques can be applied to any predictor, independently of the model. We aim to explain the results without mentioning how the actual model works, since we might not know it. This kind of explanations tend to be post-hoc: an outcome is produced and then we try to explain it with the input and training data.

In our work, we will clearly focus on model-specific explanations, as Bayesian networks are transparent models. However, some of the explanations that can be obtained with Bayesian networks may fall in a middle category: example-based explanations use other instances to explain an outcome. While at first glance the technique might seem model agnostic (as the explanations are independent from the model and they do not explain the model), in Bayesian networks we might use the model to our advantage to generate these explanations.

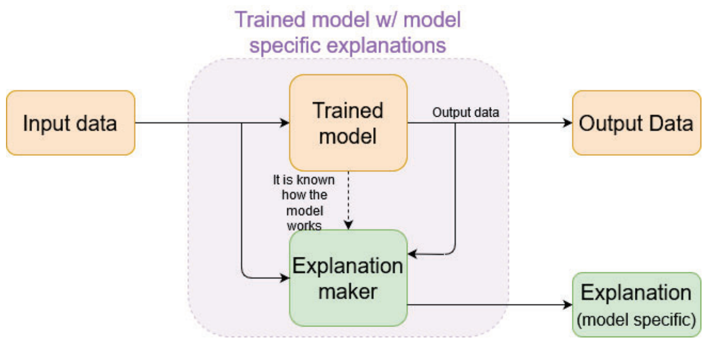
A *global explanation* is one that serves to explain the model itself, whereas a *local explanation* tries to explain why/how the model made a certain prediction or a certain related subset of predictions. In this work, we will find both global and local explanations.

2.4.3. Explainable Bayesian networks

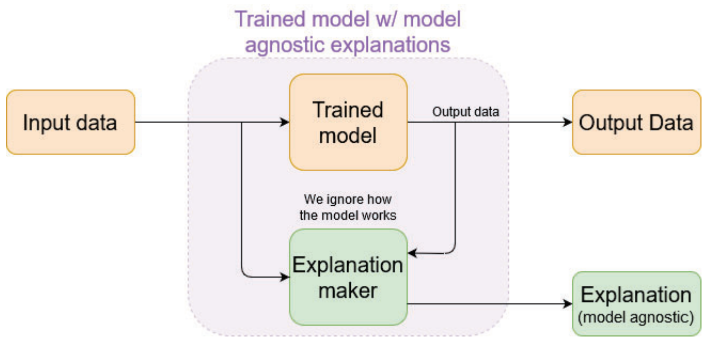
As there is no concrete definition for explainability and interpretability, in some cases it is hard to classify whether a system is explainable or not. Bayesian networks is one of them, since theoretically, everything is made explicit and can be traced, but that does not mean that the system can be easily understood (Gilpin et al., 2018; Barredo-Arrieta et al., 2020).

Accordingly, we can find works in which the transparency of Bayesian networks is used to explain other systems or decisions and others dedicated to improve and simplify explainability in Bayesian networks.

While in this work we focus on the latter case, we first provide a brief review on the applications of Bayesian networks as explainers and then move on to review the literature concerning on how to improve the inherent Bayesian network transparency.


Trained model w/ model specific explanations
Input data
Trained model
Output data
Output Data
It is known how the model works
Explanation maker
Explanation (model specific)

(a) Model-specific explanations


Trained model w/ model agnostic explanations
Input data
Trained model
Output data
Output Data
We ignore how the model works
Explanation maker
Explanation (model agnostic)

(b) Model-agnostic explanations

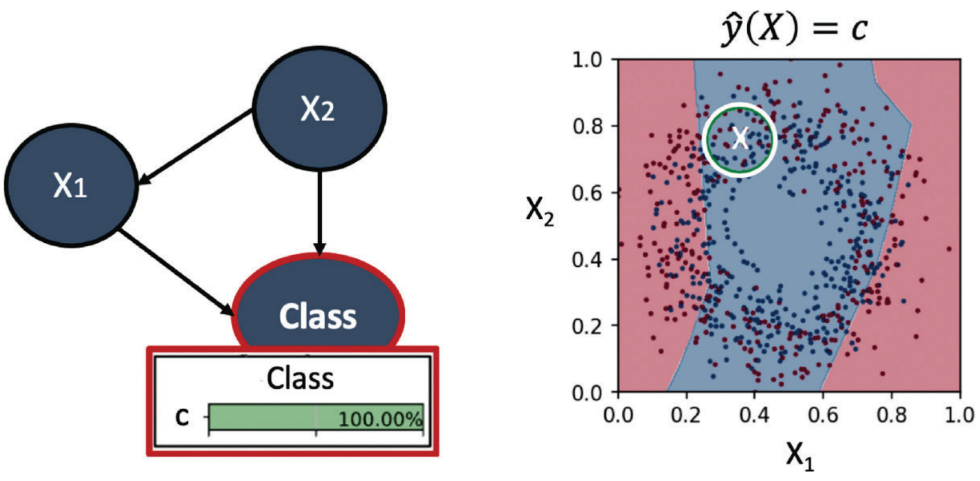
Figure 2.8: Model-specific vs. model-agnostic explanations

2.4.3.1. Bayesian networks as explainers

Since Bayesian networks are transparent, they have been used alongside black-box models, typically in clinical domains. In Ren et al. (2021), instead of using a fully connectionist approach for the problem of pneumonia detection, they opt for processing two raw inputs separately using convolutional neural networks (CNNs) and then perform classification with a Bayesian network using the two preprocessed inputs.

In other works, Bayesian networks do not fully substitute deep neural networks as classifiers. Instead, they are used as **surrogate models** (Molnar, 2020) which, in the context of XAI, refer to interpretable models trained using as input instances previously classified by a black-box model. The objective is to approximate the behaviour of a black-box model using a transparent one in order to provide explainability. Such approach was first explored with Bayesian networks in Chen et al. (2020), where although the entity-aware CNN used is perfectly capable of performing clinical diagnosis, a Bayesian network is used to transparently perform inference using the prior probabilities output by the CNN. The graph of the Bayesian network is built using expert knowledge, divided into disease and symptom nodes and three different network structures are used in parallel as an ensemble. Although in this case the Bayesian network is not learned to use the input and output of the black-box model as traditionally done in a surrogate model, it is preferred over the uninterpretable neural networks to present the results.

In contrast, the framework LINDA-BN (Moreira et al., 2021) does train a Bayesian network using the data processed by a deep neural network. In particular, it is trained with data that has been perturbed in order to observe how a change in the input distributions affects the performance. While the explanation generation is yet to be explored in future works, the article focuses on observing the Markov blanket (as it indicates the direct causes of an effect) and we can appreciate how Bayesian networks approximate the functioning of neural networks (see an example in Figure 2.9).



Moreira et al. (2021)

Figure 2.9: Simple application of LINDA-BN. An example with high confidence

Finally, other generic proposals where a white-box model is used to interpret black-box ones can be of interest if we consider Bayesian networks to be explainable enough. An example is the ProfWeight methodology (Dhurandhar et al., 2018b), where an interpretable model is retrained using the input data, but the weights of the instances that are hard to classify by a neural network are increased. This retrain increases the performance of the interpretable model while retaining the transparency.

2.4.3.2. Explaining Bayesian networks: a taxonomy

If the definition of explainability and interpretability is somewhat fuzzy regarding the overall field of artificial intelligence, Bayesian networks are not an exception, since many of the basic queries that can be performed over them can be understood as explanations themselves. For instance, the computation of a posterior probability $P(x_1, ..., x_l|\boldsymbol{e})$ can be understood as computing the likelihood of $(x_1, ..., x_l)$ as an explanation for the evidence $\boldsymbol{e}$ and, similarly, a MAP query (see Section 2.2.5.1) looks for the value assignment of set $\boldsymbol{H}$ that maximises its probability given an evidence or, in other words, we look for the most probable explanation for the evidence $\boldsymbol{e}$.

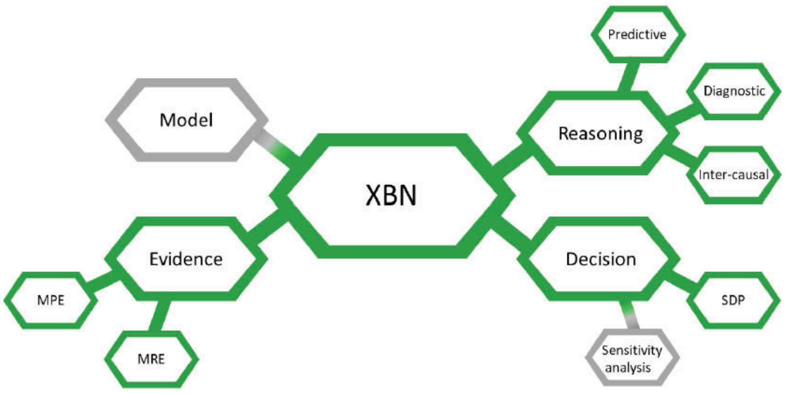
In fact, there is a taxonomy of explanation in Bayesian networks according to what they are explaining. Lacave and Díez (2002) distinguishes between three groups, and in Derks and De Waal (2021) a fourth one is added (see Figure 2.10). In this literature review, we add a fifth "group", which we will describe later. The four types of explanations according to their focus are:

- Model explanation, which refers to explaining the network structure and JPD themselves in order to assist in the model construction and provide knowledge. Part of our contributions fall within this category (Section 3.3).

- Reasoning explanations: We try to measure how likely is a conclusion given a certain evidence by means of explaining the reasoning followed. While Derks and De Waal (2021) only distinguish among three types of reasoning explanations according to the direction of the flow of information (predictive, diagnosis and inter-causal reasoning), in Lacave and Díez (2002) the subclassification is more focused on how to actually highlight the flow of information regardless of its direction. In addition, the reasoning concerning contrary outputs and hypothetical statements is also in this category according to the latter paper and so, as pointed out in Koopman and Renooij (2021), contrastive and counterfactual explanations might fall under this category. In this work, we put also ContFactual (Contrastive and counterFactual) explanations in the category of reasoning explanations.

- Evidence explanations: We try to explain a certain evidence. A MAP query (both in total and partial abduction) falls under this category and, in our work, we will dive in the robustness of these explanations (see Sections 2.4.3.6 and 3.1).

- Decision explanation refers to answering whether we have enough information to make a decision or not.



Derks and De Waal (2021)

Figure 2.10: Taxonomy of explanations in Bayesian networks. In green, the concepts discussed and classified in Derks and De Waal (2021). In grey, the concepts just classified in that paper

In this classification, model explanations are global, as they explain the model itself, whereas the other three types are local, since we are focusing on a certain instance or input. In the model agnostic vs. specific category, all of these methods fall into the latter, as the use the model is used to generate or to better convey the explanations.

As indicated before, the classification focuses on *what* is happening and distinguishes between four main types. However, we argue that *how* to convey the explanations is also important, as one cannot assume that end-users may understand a priori a full reasoning explanation or that they would be satisfied with the output of a MAP query. Additionally, users tend to prefer simple explanation, even when they are not as complete (Miller, 2019). Thus, we introduce a fifth "category" (see Figure 2.11, that is more a compilation of different techniques used to improve the explainability of the other four types of explanations rather than a full category on its own right:

- *Support methods*, an umbrella term referring to methods that do not seek to explain directly, but rather focus on how to provide simpler explanations and simplify any of the other four types of explanations. A hypothetical example would be the introduction of an automatic summary system to shorten the length of a given explanation.

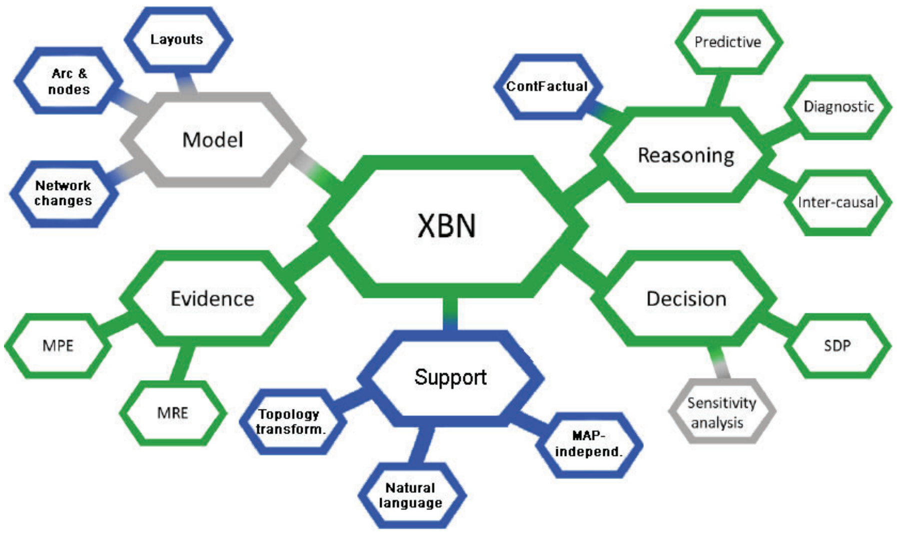


Figure 2.11: Expanded taxonomy of explanations in Bayesian networks, as modified in this literature review. In blue, the elements added to the original classification by Derks and De Waal (2021)

Once we have overviewed the explanations in Bayesian networks, we describe the most interesting approaches that have inspired our work and that have been added to the diagram made by Derks and De Waal (2021) in Figure 2.10. First, we will present and analyse the use of counterfactual and contrastive reasoning in Bayesian network reasoning due to its great addition to explainability. Next, we will discuss model explanations, that were not reviewed in Derks and De Waal (2021) and that will be of great value to one of our proposal (see Section 3.3). Then, we will enumerate some support methods to improve explainability in Bayesian networks in order to justify the inclusion of such fifth category. Finally, we will discuss the concept of MAP-independence, which can be considered a support method for evidence explanations.

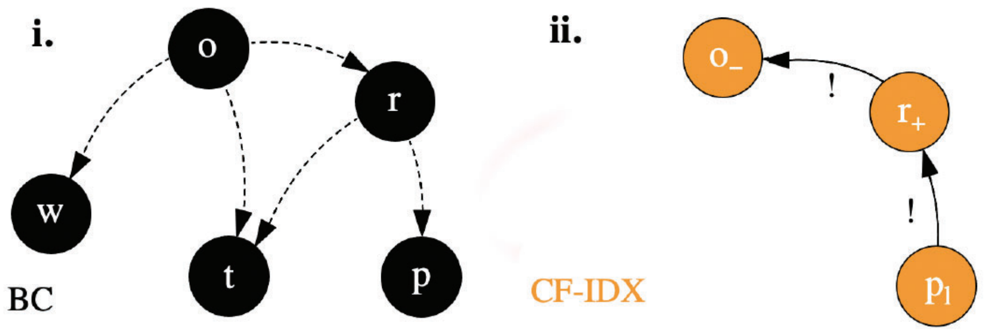
2.4.3.3. Counterfactual and contrastive explanations

A general technique to add explainability to intelligent systems is to use **counterfactuals**. According to Lewis (1973), a counterfactual is a conditional statement of the form “If it were the case that A, then it would be the case that C” where A is a hypothetical statement about a past event or, put in other words, to study the outcome if a hypothetical past alternative had occurred. Usually, we are interested in alternatives to the past that are similar to what actually happened, yet they have an impact on the present (closest-world semantics). The interest of counterfactual in XAI comes from the fact that counterfactual thinking is something that people tend to do (Miller, 2019) and, as such, they can be used as explanations that are relatively close to the human cognitive processes.

In Pearl (2009), an insight to counterfactuals is made from the perspective of probabilistic graphical models, which uses the counterfactuals to interpret the statistical data. He defines counterfactuals sentences as “The value that $\boldsymbol{Y}$ would have obtained, had $\boldsymbol{X}$ been x'” (instead of its real value x) and establishes a three-step procedure to get counterfactuals in Bayesian networks: (1) we propagate our evidence $\boldsymbol{e}$, (2) then the counterfactual nodes $\boldsymbol{X}$ are intervened with the counterfactual value $\boldsymbol{x}'$ and finally, (3) we study the value of the outcome $\boldsymbol{Y}$ (taking into account both the intervention and the evidence). The work suggests that counterfactuals can only be truly useful when we have actual expert knowledge about the relations in the network rather than just information regarding probability distributions and independencies (see Figures 1.6 and 1.7 in Pearl (2009)).

Albini et al. (2020) introduce counterfactual explanations in Bayesian network classifiers using the concept of influences. An influence is defined as a pair (X, Y) influencer-influenced, where the influenced is a target node Y of the Bayesian classifier and the influencer X is a child of the influenced, regardless of the type of node. The set of all pair of influences is represented as $\boldsymbol{I}$, whereas the influences of a single node, Y, as $\boldsymbol{I}_Y$. Influences are meant to represent the flow of information from the input to the target in a Bayesian classifier, ignoring connections between non-target nodes. Then, critical and potential influences are defined as the counterfactual explanations for a certain input $\boldsymbol{a}$ for the classifier as follows:

- A critical influence (X, Y) is one such that, for every alternative input $\boldsymbol{a'}$ in which the evaluation of the influencer X changes while the rest of other influencers $\boldsymbol{I}_Y$ of the influenced y remain the same, the evaluation of the influenced Y is modified.
- A potential influence (X, Y) is a relaxation of the first type of influence. In this case, we allow for the existence of one or more inputs $\boldsymbol{a''}$ such that the evaluation of the influencer X, the rest of influences $\boldsymbol{I}_Y$ and the influenced Y all remain unchanged.



Albini et al. (2021)

Figure 2.12: (i) The structure of a Bayesian classifier (the quantitative information is not shown). In (ii), a counterfactual explanation for the input $(W = \text{“}l\text{''}, T = \text{“}m\text{''}, P = \text{“}l\text{''})$. P is a critical influence of R, and R is a critical influence of O

Thus, we represent how a minimal change in the influences of a target node may affect the outcome of the classifier (see Figure 2.12 for an example). In Albini et al. (2021), the concept is inserted in a framework that generalises local and example-based explanations in Bayesian network classifiers. While other explanation methods proposed by the authors have linear time (the number of inferences that we have is linear with the number of value assignment of a node), for the case of counterfactuals we have an exponential growth with the number of input nodes in the network, since we have to check all possible inputs of the classifier to look for critical and potential influences. Furthermore, the inference is claimed to be quite fast, since they use relatively simple Bayesian networks classifiers whose structure is constrained and however, in our work, we assume that we are working with very large Bayesian networks.

Contrastive explanations are another possibility to add explainability to systems that differ slightly from a counterfactual: Instead of asking what would have happened to an outcome if the input were different, in a contrastive explanation we are interested in knowing why we obtained an outcome t rather than an alternative t' (McGill and Klein, 1993).

A recent work by Koopman and Renooij (2021) elaborates on how to produce contrastive explanations in Bayesian networks. They call their proposal *persuasive contrastive explanations* and is defined as follows: Given a Bayesian network and an evidence $\boldsymbol{e}$, the outcome t and another non-obtained outcome t', a persuasive contrastive explanation is a tuple $(\boldsymbol{s}, \boldsymbol{c})$, where $\boldsymbol{s}$ is a subset $\boldsymbol{s} \subset \boldsymbol{e}$ such that if any modification is made to the evidence $\boldsymbol{e}$ without changing $\boldsymbol{s}$ the output would remain t (a sufficient explanation); and where $\boldsymbol{c}$ is a subset $\boldsymbol{c} \subset \boldsymbol{e}$ such that a change in $\boldsymbol{c}$ (even if the rest of nodes in $\boldsymbol{e}$ remain unchanged) translates to a change in the output from t to t' (counterfactual explanation). In the literature, $\boldsymbol{s}$ and $\boldsymbol{c}$ are referred to as pertinent positives (PP) and pertinent negatives (PN), respectively (Dhurandhar et al., 2018a). A PP is a minimal subset of the input whose presence leads to the output, while a PN is another minimal subset of the input but whose absence would lead to another output.

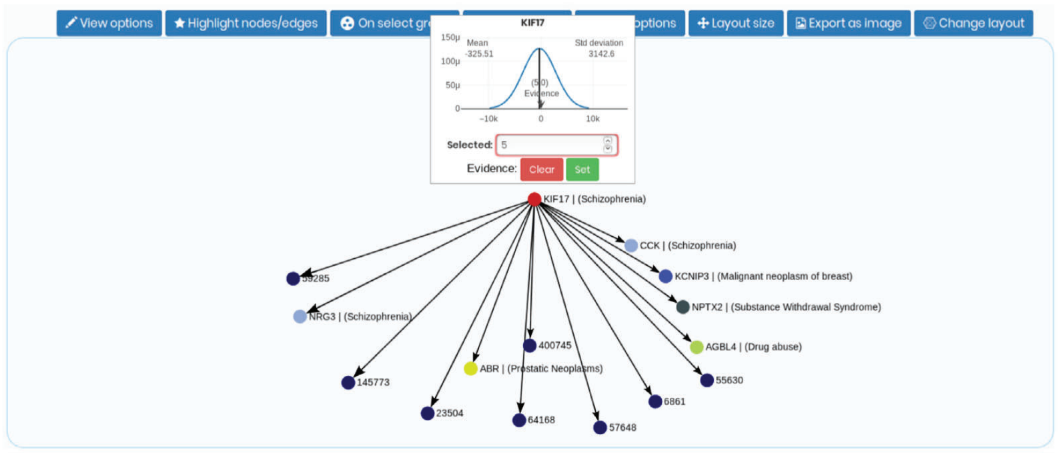
2.4.3.4. Model explanations

This kind of explanations are static (Henrion and Druzdzel, 1990) in the sense that we are not trying to explain a certain evidence or the reasoning or the process of inference, but the knowledge stored both in the graph and in theJPD of the Bayesian network. This is done through depicting the network both verbally and/or visually. The former focuses on visual descriptions rather than verbal ones that would translate the information of the nodes and arcs to natural language expressions (Henrion and Druzdzel, 1990).

The **visual exploration of the network** is the type of model explanation with which this work is more concerned about. Of course, the most straightforward method would be to just present to the end-user a graph depicting the network, so that all the variables and relations can be visualised. This bare-bones approach falters when we care about the strength of the relationships (quantitative information not shown in the graph) and in large Bayesian networks. In addition, some level of interaction is desired, usually through menus and windows that display, hide and select information as needed. Most software for Bayesian network visualization offers these capabilities, including the first ones, such as DIAVAL (Díez, 1994), Elvira (Lacave et al., 2007) or BayeSuites (Michiels et al., 2021). In Figure 2.13, an example of such software can be seen. Specially interesting is the menu approach taken by Sundararajan et al. (2013), which includes the possibility of opening many windows and focusing on many processes at the same time, allowing for simultaneous exploration of different regions of the network.

In order to present quantitative information about the arcs in the network, the Elvira software can **colour the arcs** depending on whether the relation that they represent is directly or inversely proportional (assuming that establishing a partial order in the involved variables is possible). In Zapata-Rivera et al. (1999), **colour, size and even saturation** are used to quantitatively depict the marginal probability distribution of the nodes, where a more vivid colour and thicker size correspond to a greater probability of a variable being true. If we care about explanation of reasoning, Kadaba et al. (2007) use animations to depict the causal flow of information.

Organizing the network is also useful, specially when the network grows large. The idea of presenting the network in specific **layouts** is explored in Marriott et al. (2005), while in Cossalter et al. (2011) the problems of large network are explicitly stated and some solutions



Michiels et al. (2021)

Figure 2.13: A screenshot of the software BayeSuites (Michiels et al., 2021). A node related to schizophrenia disease has been selected

are presented, such as merging CPTs in order to display less information to the end-user.

While we discussed many of the most well-known literature, a more thorough review that complements this one was made in Paniego-Blanco (2019). One of the main purposes of our review is to highlight that, to the best of our knowledge, there is no literature about explaining changes in the structure or in the conditional probability distribution of Bayesian networks. This is mainly due to the fact that the literature about changing the structure of Bayesian networks and of the JPD is also relatively scarce, where even dynamic processes are modelled using static models (Song et al., 2009). In Section 3.1 we present an approach to this novel problem.

2.4.3.5. Support methods

Amongst the newly included support methods category, we find that it is common to **transform the model** into a different type of graph (which may also be referred to as knowledge compilation) in order to make it more interpretable. In Chan and Darwiche (2002), they convert a naive Bayes classifier (a Bayesian classifier in which the inputs are considered to be conditionally independent given the class variable) into an ordered decision diagram (ODD). The diagram hides the probability distributions, which is usually not desired in reasoning explanations. In addition, the inference is traceable: we just need to follow the diagram, similarly to a decision tree. In Chubarian and Turán (2020) this idea is reviewed an expanded to other types of classifiers, such as the tree augmented naive Bayes, this time using ordered binary decision diagrams (OBDDs). This work also shows that it is necessary that the OBDD approximates the functioning of the network rather than replicate it exactly, since otherwise the knowledge compilation of the network to the OBDD would not be computationally efficient.

The approach taken by Nielsen et al. (2008) is to build causal explanation trees. While the method itself is explicitly targeted to give explanations about an evidence (specifically in MPE queries), building a causal network may give information about the reasoning followed. Each node of the tree represents a variable in the explanation set $\boldsymbol{H}$ and each branch going out of the node, a possible value assignment of the variable. The leaf nodes, in turn, represent possible value assignments for $\boldsymbol{H}$, i.e., the possible explanations for our evidence $\boldsymbol{e}$.

Another support method that we find interesting is the use of explanations **using natural language**. While there are early approaches involving natural language such as in the aforementioned work by Henrion and Druzdzel (1990) and in the first Bayesian networks softwares such as Elvira (Lacave et al., 2007), we find that this approach is today more interesting than ever thanks to the advances in natural language generation (NLG). Pereira-Fariña and Bugarín (2019) use a fuzzy syllogistic model to generate relevant natural language explanation and Keppens (2019) opts to use the support graph formalism (Timmer et al., 2017), that creates a content model from which the explanations are built using a NLG system. While here we refer to some of the most novel methods, a thorough review about NLG explanations, their importance in new software and their history are found in Hennessy et al. (2020).

2.4.3.6. MAP-independence

While in the previous points we have discussed different types of explanations, now we would like to emphasise the need for quality of explanations in Bayesian networks. An important feature of a good explanation is stability (Molnar, 2020), that means that the explanation should not vary (or should vary lightly) for two similar cases.

In MAP queries (Section 2.2.5.1), where we try to explain a certain evidence $\boldsymbol{e}$ using a set of hypothesis nodes $\boldsymbol{H}$ (Equation (2.7)), the concept of stability is related to the node relevance. Intuitively, we consider a node to be relevant if a change in it is propagated to another node of interest. According to the modelling of context-specific relevance as (in)conditional independences by Pearl and Paz (1985), we can define the set of relevant nodes for our explanation $\boldsymbol{h^*}$ as the set of intermediate nodes $\boldsymbol{R}$ (that is, neither in $\boldsymbol{E}$ or $\boldsymbol{H}$) that are not conditionally independent of $\boldsymbol{H}$ given $\boldsymbol{E}$M formally, $\mathcal{R} = \{R \in \boldsymbol{X} | R \notin \boldsymbol{H}, R \notin \boldsymbol{E}, \neg I_P(R, \boldsymbol{H}|\boldsymbol{E})\}$. Therefore, we claim that an explanation is less stable when there is a higher number of relevant intermediate nodes.

Since this definition of relevance is far too strict, Kwisthout (2021) introduces the concept of MAP-independence. An explanation $\boldsymbol{h^*} = arg\,max_{\boldsymbol{H}} P(\boldsymbol{H}|\boldsymbol{e})$ for a certain evidence $\boldsymbol{e}$ is said to be MAP-independent from a set of nodes $\boldsymbol{R}$, $MInd(\boldsymbol{h^*}, \boldsymbol{R}, \boldsymbol{e})$, if the value assignment of $\boldsymbol{H}$ that maximises the joint probability $P(\boldsymbol{H}, \boldsymbol{r}|\boldsymbol{e})$ for every value assignment $\boldsymbol{r} \in \Omega(\boldsymbol{R})$ is $\boldsymbol{h^*}$, i.e., when no observation over $\boldsymbol{R}$ alters our best explanation. Formally, it can be understood as a decision problem:

Question: Given a MAP query, is $\forall \boldsymbol{r} \in \Omega(\boldsymbol{R}),\ \boldsymbol{h^*} = arg\,max_{\boldsymbol{H}} P(\boldsymbol{H}, \boldsymbol{r}|\boldsymbol{e})$?

An example of the usefulness of MAP-independence can be seen in Figure 2.14.

Similarly, throughout this work we are also going to refer to the MAP-independence of the explanation $\boldsymbol{h^*} = arg\,max_{\boldsymbol{H}} P(\boldsymbol{H}|\boldsymbol{e})$ of the observation $\boldsymbol{r}$, which we define as follows: we consider $\boldsymbol{h^*}$ to be MAP-independent of $\boldsymbol{r}$ given an evidence $\boldsymbol{e}$ if $\boldsymbol{h^*} = arg\,max_{\boldsymbol{H}} P(\boldsymbol{H}, \boldsymbol{r}|\boldsymbol{e})$.

We also refer to MAP-dependence, which is the logical opposite of MAP-independence. Formally, we say that $\boldsymbol{h^*}$ is MAP-independent of $\boldsymbol{R}$ given an evidence $\boldsymbol{e}$ if $\exists \boldsymbol{r} \in \Omega(\boldsymbol{R}),\ \boldsymbol{h^*} \neq arg\,max_{\boldsymbol{H}} P(\boldsymbol{H}, \boldsymbol{r}|\boldsymbol{e})$

We claim that this notion is very useful to verify whether our explanation is more or less stable, depending on the number of nodes of which it is MAP-independent from. In Section 3.1,

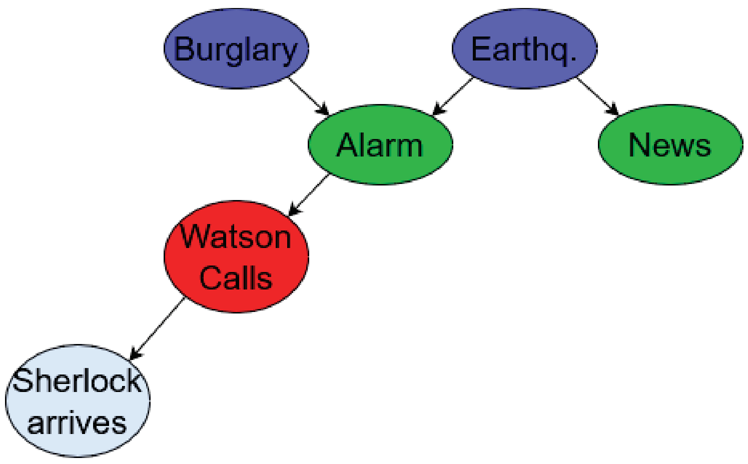


Figure 2.14: An example of MAP-query, where the node in red is the evidence $\boldsymbol{e}$ provided, the ones in blue are the explanation nodes $\boldsymbol{H}$ and in green, the relevant nodes for our explanation $\boldsymbol{h}^*$ according to the conditional independence criterion. However, let us assume that any observation on the node "News" does not lead to a change in our most probable explanation. Then, should we consider "News" to be relevant. This question is answered using MAP-independence of the explanation (Burglary and Earthquake) given that Watson Calls from News

we push this method to new paradigms.

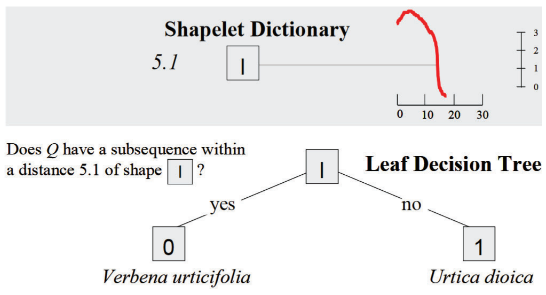
2.4.4. Explaining time series

Recently, most of the analysis of time series have been done using deep learning due to its great computational power. As such, much of the literature concerning explanation of time series has been done in the ambit of neural networks, but these works are still insightful to study what type of explanation is desirable and what elements shall be explained. Therefore, in this section of the literature review, we obscure most of the mechanisms that generate explanations in deep learning models, focusing rather on what and how they are explained.

Qin et al. (2017) present a recurrent neural network that identifies the variable that impact the most the classification of time series. Hsieh et al. (2021) argue that is not only necessary to base explanations on **important variables**, but also on **interesting time intervals**, and they design a convolutional neural network that uses its feature maps to cope with both of them for supervised classification.

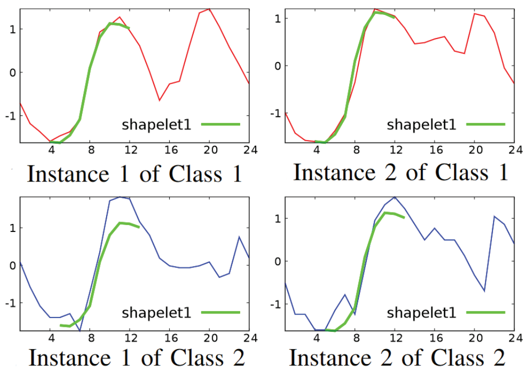
A very interesting method is the use of **shapelets** (Ye and Keogh, 2009) for classification, which are time series subsequences that are maximally representative of a class. Given a labelled dataset, we aim to look for subsequences that tend to be present in a certain subsets of classes and absent in the rest of the classes. The use of shapelets in time series classification results in interpretability in the process (Figure 2.15), since it is possible to use the subsequence as an example-based explanation. Most of the works around shapelets are aimed towards improving the efficiency of the method, since training and discovering shapelets is a very time consuming task. Different techniques involve changes in the representation, dimensionality reduction and search space pruning (Rakthanmanon and Keogh, 2013; Hou et al., 2016; Fang et al., 2018; Li et al., 2020).

A similar approach to shapelets is the use of **prototypes**. Li et al. (2018) use a neural



Ye and Keogh (2009)

(a) Interpretable model with shapelets using a decision tree



Li et al. (2020)

(b) Visualization of the classification process. The subsequences can be used as example-based explanations

Figure 2.15: Shapelets. The algorithm tries to match the subsequence with the instance and depending if it matches or not with a certain confidence, the instance is classified with a class or another

network for classification that includes an autoencoder (Hinton and Salakhutdinov, 2006), which is in itself a neural architecture that takes an input, retains the most important attributes using a latent lower-dimensional representation and then reconstructs the input with it. The autoencoder is then used to generate prototypes, representative instances of the data in order to explain it. Additionally, the prototypes generated are used in the classification process as well. In Gee et al. (2019), this proposal is proven to work in time series and a prototype diversity penalty that encourages more diverse prototypes is introduced, ensuring more consistency (Molnar, 2020) in the explanations. The difference between shapelets and prototypes is that the former is an interpretable classification method that relies on subsequences to explain the results, whereas prototypes are not a classification technique, but a post-hoc explanation (given after a black-box system performs the classification) that relies on a full example rather than on a subsequence, making it ideal for domains with short time series.

Following the line of example-based explanations, Labaien et al. (2020) adapt **contrastive explanations** to the field of time series classification (for a definition of counterfactuals and contrastive explanations, see Subsection 2.4.3.3). The paper focuses on finding pertinent negatives and, since this task is defined as an optimization problem (Dhurandhar et al., 2018a), the authors propose an autoencoder architecture based on long short-term memory networks to find the pertinent negative attributes of an input. The pertinent negatives in the context of time series classification can be viewed as groups of points whose modification would lead to a change of class in the classification. In posterior works, **counterfactual explanations** can also be found in the literature. Ates et al. (2021); Delaney et al. (2021) both build counterfactuals by modifying the instance to be explained with attributes from other instances, ensuring that the counterfactual obtained is consistent with the data probability distribution. Both works use a neural network with the ReLu activation function to solve the optimization problem presented.

Thrun et al. (2021) give explanations for time series analysis in the specific field of hydrochemistry. The framework, called XAI-DDS, first clusters the time series using projection-based clustering and then a decision tree is trained using the cluster as label. The three is meant to serve as a transparent surrogate model (for a definition of surrogate model, see Section 2.4.3.1) where the branches represent explanations and, to complement them, an expert shall interpret the statistical data obtained, usually through contrastive reasoning.

We can also find works that focus on how to evaluate the quality of explanations for time series. Schlegel et al. (2019) rework the already existing perturbation method (Zeiler and Fergus, 2014) in order to adapt it to time series. The summarise, using perturbations consist of changing the part of an input that is used for explanations and, since it is considered to be relevant for classification, we study how this affects the output (on that paper, performance should decrease massively). The adaptations made for time series can be divided into two: (1) perturbation individual time points that are considered to be relevant by permuting them and (2) perturbation of time subsequences that are relevant, either by reversing them or substituting it by the average of the values of the time points in said interval.

Chapter 3

Methods

In this chapter, we will discuss the theoretical advances proposed in this master thesis. Rather than focusing on just one method, we prefer to follow different lines of research, according to the principle that one explanation does not fit all.

This chapter follows a "cascade" structure, where we move from methods that can be applied to any type of Bayesian networks, to others that are more restrictive, specific for DBNs or even for non-stationary DBNs.

In Section 3.1, we rework the MAP-independence proposal of Kwisthout (2021). Next, in Section 3.2 we discuss how to tackle the optimization problem of finding prototypes or shapelets in DBNs. In Section 3.3, we propose how to explain the changes in the model in non-stationary and time-varying DBNs and, finally, in Section 3.4 we discuss our implementation of a library for learning TV-DBNs.

3.1. MAP-independence

3.1.1. Limitations of the original proposal

Even though MAP-independence is a useful proposal to test the robustness of abduction reasoning and it has been well analysed computationally, we still find three unsolved problems in the original MAP-independence proposal, as mentioned in Section 2.4.3.6.

First, the original proposal is far too strict, since it is necessary that all possible value assignments $\boldsymbol{r}$ for the set of nodes $\boldsymbol{R}$ hold the MAP-independence condition $arg\,max_{\boldsymbol{H}}P(\boldsymbol{H}|\boldsymbol{e}) = arg\,max_{\boldsymbol{H}}P(\boldsymbol{H},\boldsymbol{r}|\boldsymbol{e})$. In many occasions, there might be value assignments of $\boldsymbol{R}$ with a very low probability given the evidence (i.e., very unlikely) but that do not follow the condition that renders the explanation $\boldsymbol{h^*}$ and the set $\boldsymbol{R}$ MAP-independent, even when most of the value assignments with a higher probability follow the condition. To address this phenomenon, we introduce the concept of MAP-independence strength (Section 3.1.2).

Second, some properties of MAP-independence have not been explored yet. These properties, although unable to reduce the time complexity of the algorithm, can still be used to prune the search space and to reduce the computation time, notably in best case scenarios when looking for many MAP-independences. The two properties that we formulate in Section 3.1.3 are related to subsets and to conditional independence.

Lastly, the original proposal is unsuitable for networks that contain continuous variables due to two reasons:

- In MAP-independence we compare the value of $\boldsymbol{H}$ that maximises $P(\boldsymbol{H}|\boldsymbol{e})$ and $P(\boldsymbol{H}, \boldsymbol{r}|\boldsymbol{e})$, which in the case of continuous variables is the mode of the distributions, a real number or a vector of real numbers (in case we have a vectorial mode). Comparing two real numbers is usually ineffective due to the infinite number of decimals digits and as such it is unrealistic to assume that the equality of both distributions will ever hold true.

- To check the MAP-independence of an explanation $\boldsymbol{h}^*$ and a set $\boldsymbol{R}$ we need to check for every value assignment $\boldsymbol{r}$ for $\boldsymbol{R}$, which is impossible when the domain of $\boldsymbol{R}$ is infinite, as in the continuous case.

These two issues render the evaluation of MAP-independence in both continuous and hybrid networks impossible and will be addressed in Section 3.1.4.

3.1.2. MAP-independence strength

The MAP-independence strength can be summarised as the sum of the probability of all value assignments of $\boldsymbol{r}$ given the evidence $\boldsymbol{e}$ that follow the MAP-independence condition.

Formally, we first define the set of all the value assignments of $\boldsymbol{R}$ that follow the MAP-independence condition, which we will name as $\Omega^{MInd}(\boldsymbol{R}, \boldsymbol{H}, \boldsymbol{e})$:

$$\Omega^{MInd}(\boldsymbol{H}, \boldsymbol{R}, \boldsymbol{e}) = \{\boldsymbol{r} \in \Omega(\boldsymbol{R}) \mid arg\,max_{\boldsymbol{H}} P(\boldsymbol{H}|\boldsymbol{e}) = arg\,max_{\boldsymbol{H}} P(\boldsymbol{H}, \boldsymbol{r}|\boldsymbol{e}) = \boldsymbol{h}^*\}$$

With that set, we can define the MAP-independence strength $MIndStrength(\boldsymbol{h}^*, \boldsymbol{R}, \boldsymbol{e})$ as follows:

$$MIndStrength(\boldsymbol{h}^*, \boldsymbol{R}, \boldsymbol{e}) = \sum_{\boldsymbol{r} \in \Omega^{MInd}(\boldsymbol{H}, \boldsymbol{R}, \boldsymbol{e})} P(\boldsymbol{r}|\boldsymbol{e})$$

The MAP-independence strength lets us establish a threshold to discover situations in which we are close to meeting the MAP-independence condition in theory as originally defined, but that could be relaxed and considered a "MAP-independence in practice". For instance, take the case in which, under a certain evidence $\boldsymbol{e}$, we have $MIndStrength(\boldsymbol{h}^*, \boldsymbol{R}, \boldsymbol{e}) = 0.98$, which means that the sum of the probabilities $P(\boldsymbol{r}|\boldsymbol{e})$ of the observations $\boldsymbol{r}$ that will not alter the explanation $\boldsymbol{h}^*$ (follow the MAP-independence condition) is 98%, but in practice in some domains, we may consider $\boldsymbol{H}$ and $\boldsymbol{R}$ MAP-independent given $\boldsymbol{e}$. Note that in the cases in which we have a MAP-independence strictly as defined in Kwisthout (2021), the strength would be 1 (or 100%).

3.1.3. On the properties of MAP-independence

Theorem 3.1.1. *Given a Bayesian network, and three pairwise disjoint subsets of nodes of the DAG, (1) $\boldsymbol{E}$ representing the evidence and with a certain assignment $\boldsymbol{e}$, (2) the subset of* *explanation nodes* $\boldsymbol{H}$ *such that* $\boldsymbol{h}^* = arg\,max_{\boldsymbol{H}} P(\boldsymbol{H}|\boldsymbol{e})$ *and (3) the rest of nodes* $\boldsymbol{I}$ *of the DAG not belonging to* $\boldsymbol{E}$ *nor to* $\boldsymbol{H}$ *(i.e.,* $\boldsymbol{I} = \boldsymbol{X} \setminus \{\boldsymbol{E} \cup \boldsymbol{H}\}$*), and two proper subsets of* $\boldsymbol{I}$*,* $\boldsymbol{R}' \subseteq \boldsymbol{R} \subseteq \boldsymbol{I}$*; if* $\boldsymbol{h}^*$ *is MAP-independent from* $\boldsymbol{R}$*, then* $\boldsymbol{h}^*$ *is also MAP-independent from* $\boldsymbol{R}'$*. Formally:*

$$MInd(\boldsymbol{h}^*, \boldsymbol{R}, \boldsymbol{e}) \wedge (\boldsymbol{R}' \subseteq \boldsymbol{R}) \Rightarrow MInd(\boldsymbol{h}^*, \boldsymbol{R}', \boldsymbol{e})$$

Proof. Let $\boldsymbol{D} = \boldsymbol{R} \setminus \boldsymbol{R}'$ and then we reformulate MAP-independence with $\boldsymbol{R}'$ and $\boldsymbol{D}$, instead of $\boldsymbol{R}$.

$\forall (\boldsymbol{r}', \boldsymbol{d}) \in \Omega(\boldsymbol{R}') \times \Omega(\boldsymbol{D}),\ arg\,max_{\boldsymbol{H}} P(\boldsymbol{H}, \boldsymbol{r}', \boldsymbol{d}|\boldsymbol{e}) = \boldsymbol{h}^*$

We make the *arg max* explicit. The following expression just tell us what the *arg max* does implicitly: for every value assignment $\overline{\boldsymbol{h}}$ of $\boldsymbol{H}$ that is not $\boldsymbol{h}^*$, the probability $P(\boldsymbol{h}^*, \boldsymbol{r}', \boldsymbol{d}|\boldsymbol{e})$ will be always higher than $P(\overline{\boldsymbol{h}}, \boldsymbol{r}', \boldsymbol{d}|\boldsymbol{e})$. As before, this holds for every joint value assignment $(\boldsymbol{r}', \boldsymbol{d})$. Thus,

$\forall \overline{\boldsymbol{h}} \in \Omega(\boldsymbol{H}) \setminus \{\boldsymbol{h}^*\},\ \forall (\boldsymbol{r}', \boldsymbol{d}) \in \Omega(\boldsymbol{R}') \times \Omega(\boldsymbol{D}),\ P(\boldsymbol{h}^*, \boldsymbol{r}', \boldsymbol{d}|\boldsymbol{e}) > P(\overline{\boldsymbol{h}}, \boldsymbol{r}', \boldsymbol{d}|\boldsymbol{e})$

Next, we marginalise $P(\boldsymbol{h}^*, \boldsymbol{r}'|\boldsymbol{e})$ by summing up for every $\boldsymbol{D}$ in both sides of the inequality.

$\forall \overline{\boldsymbol{h}} \in \Omega(\boldsymbol{H}) \setminus \{\boldsymbol{h}^*\},\ \forall \boldsymbol{r}' \in \Omega(\boldsymbol{R}'),\ \sum_{\boldsymbol{d}} P(\boldsymbol{h}^*, \boldsymbol{r}', \boldsymbol{d}|\boldsymbol{e}) > \sum_{\boldsymbol{d}} P(\overline{\boldsymbol{h}}, \boldsymbol{r}', \boldsymbol{d}|\boldsymbol{e})$

$\forall \overline{\boldsymbol{h}} \in \Omega(\boldsymbol{H}) \setminus \{\boldsymbol{h}^*\},\ \forall \boldsymbol{r}' \in \Omega(\boldsymbol{R}'),\ P(\boldsymbol{h}^*, \boldsymbol{r}'|\boldsymbol{e}) > P(\overline{\boldsymbol{h}}, \boldsymbol{r}'|\boldsymbol{e})$

Finally, we reintroduce the *arg max* as follows. Since $P(\boldsymbol{h}^*, \boldsymbol{r}'|\boldsymbol{e})$ is higher than $P(\overline{\boldsymbol{h}}, \boldsymbol{r}'|\boldsymbol{e})$ for every value assignment $\overline{\boldsymbol{h}}$ for $\boldsymbol{H}$ (other than $\boldsymbol{h}^*$, we can safely say that $\boldsymbol{h}^*$ is the value assignment that maximises $P(\boldsymbol{H}, \boldsymbol{r}'|\boldsymbol{e})$. Again, this holds for every $\boldsymbol{r}'$.

$\forall \boldsymbol{r}' \in \Omega(\boldsymbol{R}'),\ arg\,max_{\boldsymbol{H}} P(\boldsymbol{H}, \boldsymbol{r}'|\boldsymbol{e}) = \boldsymbol{h}^*$

Therefore, $\boldsymbol{h}^*$ is MAP-independent from $\boldsymbol{R}'$. □

Theorem 3.1.2. *Under the same premises of Theorem 3.1.1 and having two subsets of* $\boldsymbol{I}$*,* $\boldsymbol{R}, \boldsymbol{S} \subseteq \boldsymbol{I}$ *(and* $\boldsymbol{R} \cap \boldsymbol{S} = \emptyset$*); if* $\boldsymbol{h}^*$ *is MAP-independent from* $\boldsymbol{R}$ *and* $\boldsymbol{S}$ *is conditionally independent of* $\boldsymbol{H}$ *given* $\boldsymbol{R}$*, then* $\boldsymbol{h}^*$ *is also MAP-independent from* $\boldsymbol{S}$*. Formally,*

$$MInd(\boldsymbol{h}^*, \boldsymbol{R}, \boldsymbol{e}) \wedge I_P(\boldsymbol{H}, \boldsymbol{S}|\boldsymbol{R}) \Rightarrow MInd(\boldsymbol{h}^*, \boldsymbol{S}, \boldsymbol{e})$$

Proof.

First, we resort to the definition of MAP-Independence, which states that $\boldsymbol{h}^*$ is MAP-Independent from $\boldsymbol{R} \subseteq \boldsymbol{I}$ if

$\forall \boldsymbol{r} \in \Omega(\boldsymbol{R}),\ \boldsymbol{h}^* = arg\,max_{\boldsymbol{H}} P(\boldsymbol{H}, \boldsymbol{r}|\boldsymbol{e})$

Since $P(\boldsymbol{H}, \boldsymbol{r}|\boldsymbol{e}) \propto P(\boldsymbol{H}|\boldsymbol{r}, \boldsymbol{e})$, we reformulate the condition as follows:

$\forall \boldsymbol{r} \in \Omega(\boldsymbol{R}),\ \boldsymbol{h}^* = arg\,max_{\boldsymbol{H}} P(\boldsymbol{H}|\boldsymbol{r}, \boldsymbol{e})$

Similarly to the previous proof, we make the *arg max* explicit.

$\forall \boldsymbol{r} \in \Omega(\boldsymbol{R}),\ \forall \overline{\boldsymbol{h}} \in \Omega(\boldsymbol{H}) \setminus \{\boldsymbol{h}^*\},\ P(\boldsymbol{h}^*|\boldsymbol{r}, \boldsymbol{e}) > P(\overline{\boldsymbol{h}}|\boldsymbol{r}, \boldsymbol{e})$

As R_j is conditionally independent of H given R_i, then $P(\boldsymbol{H}|\boldsymbol{r}, \boldsymbol{e}) = P(\boldsymbol{H}|\boldsymbol{r}, \boldsymbol{s}, \boldsymbol{e})$

$\forall \boldsymbol{s} \in \Omega(\boldsymbol{S}),\ \forall \boldsymbol{r} \in \Omega(\boldsymbol{R}),\ \forall \overline{\boldsymbol{h}} \in \Omega(\boldsymbol{H}) \setminus \{\boldsymbol{h}^*\},\ P(\boldsymbol{h}^*|\boldsymbol{r}, \boldsymbol{s}, \boldsymbol{e}) > P(\overline{\boldsymbol{h}}|\boldsymbol{r}, \boldsymbol{s}, \boldsymbol{e})$

Since $P(\boldsymbol{H}, \boldsymbol{r}, \boldsymbol{s}|\boldsymbol{e}) \propto P(\boldsymbol{H}|\boldsymbol{r}, \boldsymbol{s}, \boldsymbol{e})$

$\forall \boldsymbol{s} \in \Omega(\boldsymbol{S}),\ \forall \boldsymbol{r} \in \Omega(\boldsymbol{R}),\ \forall \overline{\boldsymbol{h}} \in \Omega(\boldsymbol{H}) \setminus \{\boldsymbol{h}^*\},\ P(\boldsymbol{h}^*, \boldsymbol{r}, \boldsymbol{s}|\boldsymbol{e}) > P(\overline{\boldsymbol{h}}, \boldsymbol{r}, \boldsymbol{s}|\boldsymbol{e})$

We marginalise $P(\boldsymbol{h}^*, \boldsymbol{s}|\boldsymbol{e})$ by summing up for every $\boldsymbol{S}$ in both sides of the inequality.

$\forall \boldsymbol{s} \in \Omega(\boldsymbol{S}),\ \forall \overline{\boldsymbol{h}} \in \Omega(\boldsymbol{H}) \setminus \{\boldsymbol{h}^*\},\ \sum_{\boldsymbol{r}} P(\boldsymbol{h}^*, \boldsymbol{r}, \boldsymbol{s}|\boldsymbol{e}) > \sum_{\boldsymbol{r}} P(\overline{\boldsymbol{h}}, \boldsymbol{r}, \boldsymbol{s}|\boldsymbol{e})$

$\forall \boldsymbol{s} \in \Omega(\boldsymbol{S}),\ \forall \overline{\boldsymbol{h}} \in \Omega(\boldsymbol{H}) \setminus \{\boldsymbol{h}^*\},\ P(\boldsymbol{h}^*, \boldsymbol{s}|\boldsymbol{e}) > P(\overline{\boldsymbol{h}}, \boldsymbol{s}|\boldsymbol{e})$

Finally, we reintroduce the *arg max* and get an expression that proves the MAP-independence of $\boldsymbol{h}^*$ and the set $\boldsymbol{S}$ given e:

$\forall \boldsymbol{s} \in \Omega(\boldsymbol{S}),\ \boldsymbol{h}^* = arg\,max_{\boldsymbol{H}} P(\boldsymbol{H}, \boldsymbol{s}|\boldsymbol{e})$

□

3.1.4. MAP-independence in continuous Bayesian networks

As explained in Section 3.1.1, in continuous and hybrid networks the evaluation is impossible due to the difficulties when comparing the argument $\boldsymbol{H}$ that maximises $f(\boldsymbol{H}, \boldsymbol{r}|\boldsymbol{e})$ and $\boldsymbol{h}^* = f(\boldsymbol{H}|\boldsymbol{e})$, since they may be real numbers (or a vector that includes real numbers) and because continuous domains are uncountable infinite sets. In this section, we propose a solution for both of these problems in continuous Bayesian networks.

Comparing the modes

First, we are going to use the conditional density $f(\boldsymbol{H}|\boldsymbol{r}, \boldsymbol{e})$ instead of the joint $f(\boldsymbol{H}, \boldsymbol{r}|\boldsymbol{e})$, in order to work with a distribution in the same probability space as $\boldsymbol{h}^* = arg\,max_{\boldsymbol{H}} f(\boldsymbol{H}|\boldsymbol{e})$. We can do this since $f(\boldsymbol{H}, \boldsymbol{r_i}|\boldsymbol{e}) \propto f(\boldsymbol{H}|\boldsymbol{r_i}, \boldsymbol{e})$.

Working with densities in the same probability space, we can relax the strict comparison between the modes of $f(\boldsymbol{H}|\boldsymbol{e})$ and $f(\boldsymbol{H}, \boldsymbol{r}|\boldsymbol{e})$ by adding a threshold of tolerance $\epsilon > 0$. Instead of comparing the equality between those two densities for every $\boldsymbol{r} \in \boldsymbol{R}$, we are going to compute its difference (absolute value) and see if it is below ϵ. In case of having a vectorial mode, this would results in a vector of differences.

Another alternative, thanks to working in the same probability space, is to eliminate the "middleman" computation of the mode *arg max* and to compare the two distributions $P(\boldsymbol{H}|\boldsymbol{e})$ and $P(\boldsymbol{H}, \boldsymbol{r}|\boldsymbol{e})$ for every $\boldsymbol{r} \in \boldsymbol{R}$ with a measure of similarity or distance. We are well aware that then we are creating a different and stronger condition that MAP-independence as formulated by Kwisthout (2021). However, we still believe that conceptually this new condition is able to capture the essence of MAP-independence: to study if adding certainty to an evidence results in a change in the explanation or, in this case, in a significant change in the posterior density function.

Both proposals can be unified into an abstract function $dist()$, that takes as parameters two density functions in the same probability space. These are some examples of $dist()$ functions that we could define:

1. $|arg\,max_{\boldsymbol{H}} f(\boldsymbol{H}|\boldsymbol{e}) - arg\,max_{\boldsymbol{H}} f(\boldsymbol{H}|\boldsymbol{r},\boldsymbol{e})|$. This measure consists of taking the modes of the posterior distributions and compare if their difference is higher than $\epsilon > 0$, as described before. Again, in case of having vectorial modes, we would have to join it into a single number (summing up, mean,...) or to compare each difference against the ϵ individually. If the variables of the network are not standardised, we may run into the problem of having a threshold ϵ too strict for some variables of $\boldsymbol{R}$ and too relaxed for others, depending on the mean of $f(\boldsymbol{R}|\boldsymbol{e})$.

2. The Jensen-Shannon divergence, $JSD(f(\boldsymbol{H}|\boldsymbol{e})||f(\boldsymbol{H}|\boldsymbol{r},\boldsymbol{e}))$

3. The overlapping coefficient (Inman and Bradley Jr, 1989) between the two densities, $OVL(f(\boldsymbol{H}|\boldsymbol{e})||f(\boldsymbol{H}|\boldsymbol{r},\boldsymbol{e}))$, that measures the hyper-volume of the intersection between the two density functions.

Iterating over the uncountable domain of $\boldsymbol{R}$

However, with this solution we still find ourselves with the issue of the infinite values of $\boldsymbol{R}$. Instead of computing all possible values of $\boldsymbol{R}$, we can get a sample $\mathcal{S}$ from the density function $f(\boldsymbol{R}|\boldsymbol{e})$ and take that as an approximation of the domain of $\boldsymbol{R}$, $\Omega(\boldsymbol{R})$. As with any simulation method, the more samples we generate, the more precise our estimation of the domain of $\boldsymbol{R}$ will be.

Note that it is unrealistic to assume that for every single sample $s \in \mathcal{S}$, the MAP-independence condition will hold in an exact manner. Instead, what we will do is to compute the average distance of all pairs of densities when each s of the sample $\mathcal{S}$ is varied and verify that it is below the aforementioned threshold ϵ:

$$\epsilon MInd(\boldsymbol{h^*}, \boldsymbol{R}, \boldsymbol{e}, \epsilon) = \frac{\sum_{s\in\mathcal{S}} dist(f(\boldsymbol{H}|\boldsymbol{e}), f(\boldsymbol{H}|\boldsymbol{R}=s,\boldsymbol{e}))}{|\mathcal{S}|} \leq \epsilon, \text{ with } \mathcal{S} \sim f(\boldsymbol{R}|\boldsymbol{e}).$$

We can also define the ϵ-MAP-independence of an explanation $\boldsymbol{h^*}$ of an observation given an evidence $\boldsymbol{e}$ as the fullfillment of the distance proximity condition with a certain threshold ϵ). Formally, $\epsilon MInd(\boldsymbol{h^*}, \boldsymbol{r}, \boldsymbol{e}, \epsilon) = dist(f(\boldsymbol{H}|\mathrm{e}), f(\boldsymbol{H}|\boldsymbol{r},\boldsymbol{e})) \leq \epsilon$.

Analogously, we can define ϵ-MAP-independence strength as the ratio of samples that follow the MAP-independence condition.:

$$\epsilon MIndStrength(\boldsymbol{h^*}, \boldsymbol{R}, \boldsymbol{e}, \epsilon) = \frac{|\{s \in \mathcal{S} | \epsilon MInd(\boldsymbol{h^*}, \boldsymbol{R}=s, \boldsymbol{e}, \epsilon)\}|}{|\mathcal{S}|}, \text{ with } \mathcal{S} \sim f(\boldsymbol{R}|\boldsymbol{e}).$$

3.1.5. The case of Gaussian Bayesian networks

Although we develop a method that can be applied to any continuous Bayesian network, here we describe a particular case of this method that simplifies the process of sampling and comparing distributions and that is specifically designed for linear Gaussian Bayesian networks (see Section 2.2.4), since the characteristic exploited is the linearity in the relations between the nodes of the network.

The foundation of our proposal lies in how the conditional probabilities are marginalised in Gaussian Bayesian networks, which is as a linear combination, that leads us to the following two lemmas. Consider two nodes of a GBN, X and Y, whose marginal densities are $f(X) \sim \mathcal{N}(\mu_X, \sigma_X)$ and $f(Y) \sim \mathcal{N}(\mu_Y, \sigma_Y)$.

Lemma 3.1.3. *If we have a value assignment x over X, then the a posteriori variance of Y, $\sigma_{Y|X}$, is reduced by a certain factor, regardless of the value of X. Formally $\sigma_{Y|X} \leq \sigma_Y$.*

We will refer to the a posteriori variance as $\sigma_{Y|X}$ instead of $\sigma_{Y|x}$, as it is not affected by the concrete value x. This holds true because when we make an observation x, the uncertainty (variance) in the rest of the nodes is reduced and said reduction is independent of the value of x since the conditional variance of the variables is independent from its parents, see Equation (2.3).

Lemma 3.1.4. *If we have a value assignment x over X such that $x = arg\,max_X f(X))$, then the mode of Y is not altered. Formally: $arg\,max_Y f(Y) = arg\,max_Y f(Y|x)$ (or $\mu_Y = \mu_{Y|x}$) when $x = arg\,max_X f(X) = \mu_X$.*

This is true because the mode of the marginal distribution of variables of the network is defined as a linear combination of other nodes. If no observation is made, then the mode of $f(Y)$ (μ_Y) is conditioned on the mean value of X (since we haven't make an observation x). If we have an observation with value assignment $X = x = arg\,max_X f(X)$, note that Y will still be conditioned by the same value of X: its mode, which in a Gaussian is equal to the mean value. The only change in Y will be a reduction in variance as a result of Lemma 3.1.3.

Let's consider another observation x' that is not the mode of the variable X, $x = arg\,max_X f(X)$. The a posteriori mode with that observation $\mu_{Y|x'}$ will be different than the one where the mode of X is observed, $\mu_{Y|x}$ (Lemma 3.1.4 does not hold), although the variances will be the same, $\sigma_{Y|x} = \sigma_{Y|x'}$ (Lemma 3.1.3 holds). As such, we can consider $f(Y|x')$ a "displaced" version of $f(Y|x)$ and thus $dist(f(Y), f(Y|x)) < dist(f(Y), f(Y|x'))$. This holds in the aforementioned distance examples. As the relations are linear, if we have a third observation x'' that is even further from the mode than x', then we know that $dist(f(Y), f(Y|x'')) > dist(f(Y), f(Y|x'))$. These two notions are formalised in Theorem 3.1.5 and can be visualised in Figure 3.1.

Theorem 3.1.5. *In a GBN, if we have two value assignments for a node X such that $arg\,max_X f(X) = x \neq x'$ then, if we consider another node Y, the following holds: $dist(f(Y), f(Y|x)) < dist(f(Y), f(Y|x'))$. If we have third value assignment such that x'' is further from the mode than x', then $dist(f(Y), f(Y|x'')) > dist(f(Y), f(Y|x'))$.*

These can be translated to the context of MAP-independence: we want to study, for the different value assignments over $\boldsymbol{r}$, if $f(\boldsymbol{H}, \boldsymbol{r}|\boldsymbol{e})$ and $f(\boldsymbol{H}|\boldsymbol{e})$ are similar by comparing the modes

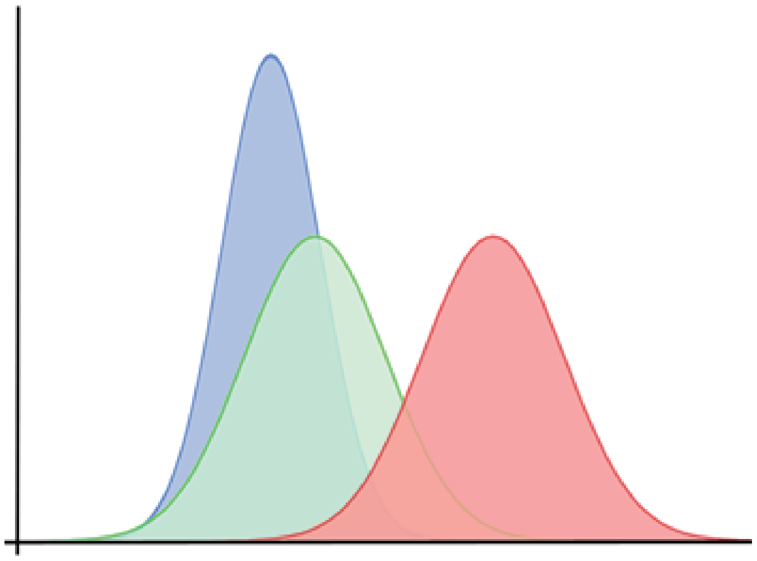


Figure 3.1: In blue, $f(Y)$; in green, $f(Y|x')$ and in red, $f(Y|x'')$, where x' is closer to the mode of $f(X)$ than x''

(or using a distance function as proposed in this work). With a single node R, we can take advantage of Theorem 3.1.5 to simplify this study as formulated in Theorem 3.1.6.

Theorem 3.1.6. *Given a MAP query and a Gaussian node R, if for an observation r at a normalised distance d from the mode of $f(R|\boldsymbol{e})$, $d = \frac{|\mu_{R|e} - r|}{\sigma_{R|E}}$, the ϵ-MAP-independence condition is met, then for any observation r' that is in range $[\mu_{R|e} - d\sigma_{R|E}, \mu_{R|e} + d\sigma_{R|E}]$ the condition will be met as well. If r does not meet the condition, some values in $[\mu_{R|e} - d\sigma_{R|E}, \mu_{R|e} + d\sigma_{R|E}]$ may not follow it either.*

Proof. Let $r' \in [\mu_{R|e} - d\sigma_{R|E}, \mu_{R|e} + d\sigma_{R|E}]$.

We know then that $dist(f(\boldsymbol{H}|\boldsymbol{e}), f(\boldsymbol{H}|r', \boldsymbol{e})) \leq dist(f(\boldsymbol{H}|\boldsymbol{e}), f(\boldsymbol{H}|r, \boldsymbol{e}))$, since r' is closer to the mean than r (Theorem 3.1.5, such that r correspond to x'' and r' corresponds to x').

If with r the ϵ-MAP-independence condition holds, then $dist(f(\boldsymbol{H}|\boldsymbol{e}), f(\boldsymbol{H}|r, \boldsymbol{e})) \leq \epsilon$. Therefore, by transitivity $dist(f(\boldsymbol{H}|\boldsymbol{e}), f(\boldsymbol{H}|r', \boldsymbol{e})) \leq \epsilon$ (i.e., the ϵ-MAP-independence holds also with r'). $\square$

Therefore, if we select an appropriate r, we can verify if the range of values $(\mu_{R|e} - d\sigma_{R|E}, \mu_{R|e} + d\sigma_{R|E})$ satisfy the ϵ-MAP-independence condition, or not. The higher the distance d, the wider the range (and probability) of R, but the stricter the condition. Although this theorem only follows when we want to verify ϵ-MAP-independence with a single node R, this is very telling on how to avoid sampling.

In case of having a set $\boldsymbol{R}$ by checking $2^{|\boldsymbol{R}|}$ values of $\boldsymbol{R}$, which correspond to all possible vectors formed with the $-d\sigma_{R|E}$ and $+d\sigma_{R|E}$ of all the nodes $\boldsymbol{R}$. However, we might be checking points that in reality are extremely unlikely to appear, as we are ignoring the variable correlation.

Another option is to check the point $\boldsymbol{b}$ of Mahalanobis distance d from the mode $\boldsymbol{\mu_{R|e}}$ that maximises $dist(f(\boldsymbol{H}|\boldsymbol{e}), f(\boldsymbol{H}, \boldsymbol{R} = \boldsymbol{b}|\boldsymbol{e}))$. If said point follows the condition, a certain probability range of $\boldsymbol{R}$ will follow it as well. It is important to note that the probability range contained within Mahalanobis distance d decreases when $|\boldsymbol{R}|$ increases. Formally:

$$\epsilon MInd(\boldsymbol{h}^*, \boldsymbol{R}, \boldsymbol{e}, \epsilon) = dist(f(\boldsymbol{H}|\boldsymbol{e}), f(\boldsymbol{H}|\boldsymbol{R} = \boldsymbol{b}, \boldsymbol{e})) \leq \epsilon$$

3.1.6. MAP-independence in the Bayesian network explanations taxonomy

The main potential of MAP-independence is to study the quality of an already existing explanation $\boldsymbol{h}^*$ with node relevance, checking if additional observations may alter it. We consider that this action might be related to verifying stability, as we study if a reduction in uncertainty of the evidence changes the explanation. Stability is a desirable property that states that the explanation should not vary when two similar instances are explained (Molnar, 2020).

As such, we claim that MAP-independence belongs to a newly introduced category that we named *support methods* (see Figure 2.10). This is more of an umbrella term that refers to methodologies aimed to improve and measure the quality of explanations in Bayesian networks. In the specific case of MAP-independence, the goal is to check if a change in the uncertainty of the model will affect the explanation of the evidence.

3.1.7. Hybrid networks

Please note that hybrid Bayesian networks (networks with both discrete and continuous variable) are out of the scope of this work, as we are sure that there are ways to formalise ϵ-MAP-independence in a more efficient way for hybrid networks. However, if we merge the original proposal and ours, we can formulate a crude approach to solve this problem in hybrid networks.

First of all, we need to distinguish between discrete and continuous nodes and therefore we divide our set $\boldsymbol{R}$ into two disjoints subsets $\boldsymbol{R}_d$ and $\boldsymbol{R}_c$, representing the discrete and continuous nodes of $\boldsymbol{R}$ respectively. This same notation can be extended to any set of nodes, for instance, to the explanation nodes, $\boldsymbol{H} = \boldsymbol{H}_d \cup \boldsymbol{H}_c$.

Second, similarly to the continuous case, we define an abstract function $dist_{hyb}()$ that measures the dissimilarity between two multivariate distributions containing both discrete and continuous elements. A possible function would be one that, given the set of variables $\boldsymbol{X}$, in which we can distinguish between discrete and continuous variables($\boldsymbol{X}_d$ and $\boldsymbol{X}_c$ respectively), checks if the discrete elements of the vectorial mode are equal and, in that case, computes the sum of the differences of the continuous components. Formally:

$$dist_{hyb}(P(\boldsymbol{X}), P(\boldsymbol{X'})) = \begin{cases} \sum |\boldsymbol{x}_c - \boldsymbol{x'}_c|, & \textit{if } \boldsymbol{x}_d = \boldsymbol{x'}_d \\ +\infty, & \textit{else} \end{cases}$$

where $(\boldsymbol{x}_d, \boldsymbol{x}_c) = arg\,max_{(\boldsymbol{X}_d, \boldsymbol{X}_c)} P(\boldsymbol{X}_d, \boldsymbol{X}_c)$ *and* $(\boldsymbol{x'_d}, \boldsymbol{x'_c}) = arg\,max_{(\boldsymbol{X'_d}, \boldsymbol{X'_c})} P(\boldsymbol{X'_d}, \boldsymbol{X'_c})$

Our proposal can be seen as a variation of the newly introduced concept of ϵ-MAP-independence in which we sample from the joint distribution containing both discrete and continuous variables $P(\boldsymbol{R}_d, \boldsymbol{R}_c|\boldsymbol{e})$.

$$\epsilon MInd_{hyb}(\boldsymbol{h^*}, \boldsymbol{R}, \boldsymbol{e}, \epsilon) = \frac{\sum_{s \in \mathcal{S}} dist_{hyb}(P(\boldsymbol{H}|\boldsymbol{e}), P(\boldsymbol{H}|\boldsymbol{R}_d = s_d, \boldsymbol{R}_c = s_c, \boldsymbol{e}))}{|\mathcal{S}|} \leq \epsilon,$$

with $\mathcal{S} = (\mathcal{S}_d, \mathcal{S}_c) \sim P(\boldsymbol{R}_d, \boldsymbol{R}_d|\boldsymbol{e})$.

3.2. Prototypes and shapelets in dynamic Bayesian networks

Both the concept of prototypes and shapelets have been discussed in Section 2.4.4. While different, they both are example-based explanations; they aim to explain an outcome by means of using clarifying examples with a similar outcome.

Another similarity between them can be found in the learning. Both problems can be formalised as an optimization problem, where we either want to find a very similar yet simple example with similar outcome in the case of prototypes, or we want to find a subsequence that maximises the information gain if we halve the time series using said subsequence as a discriminative criterion, as in shapelets. These optimization problems, specially shapelets, can potentially be hard to solve.

An advantage of Bayesian networks, and specifically DBNs, is that they are generative models, i.e., we can build a dataset (time series in the case of DBNs) using them with approximately the same probability distribution that the original data from which we learned the model. We can use this to our advantage in order to build our own synthetic examples to explain a certain prediction or a forecast in a DBN.

The intuitive idea of our preliminary proposal is the following. Given an evidence on one or many time points, we forecast until the desired time point. We desire to get an explanation of why we get said forecast. To do so, we do backwards inference and see the most likely trajectory that the time series has to follow to arrive to our forecast, a process that is known as decoding.

Let t_0 be the time point where the evidence is observed and t_f the time point that we desire to forecast, $t_0 < t_f$. The evidence will be denoted as $\boldsymbol{e}^{t_0}$, where $\boldsymbol{E} \subseteq \boldsymbol{X}$ and the variables in t_f that interest us will be named $\boldsymbol{H}^t$, with $\boldsymbol{H} \subseteq \boldsymbol{X}$. Our proposal can be formalised in a six step procedure.

1. Perform forecasting by computing $P(\boldsymbol{H}^{t_f}|\boldsymbol{e}^{t_0})$.
2. Convert the probability forecast to a vector of values, $\boldsymbol{h}^{t_f*} = arg\ \max_{\boldsymbol{H}^{t_f}} P(\boldsymbol{H}^{t_f}|\boldsymbol{e}^{t_0})$.
3. Store the *trajectory* of the variables $\boldsymbol{E}$, $\{\boldsymbol{E}^{t_0}, \boldsymbol{E}^{t_0+1}, ..., \boldsymbol{E}^{t_f}\}$. The trajectory of a variable X_i in a certain time interval $\{t_0, t_0+1, ...t_f\}$ given a certain evidence in t_0, $\boldsymbol{e}^{t_0}$, will be defined as a list of modes $\boldsymbol{x}_i^{t_0:t_f} = \{arg\,max_{X_i^t} P(X_i^t|\boldsymbol{e}^{t_0}) | t \in \{t_0, t_0+1, ...t_f\}\}$. As such, we need to store $\boldsymbol{e}_i^{t_0:t_f}$ $\forall E_i \in \boldsymbol{E}$ given the evidence $\boldsymbol{e}^{t_0}$, i.e. store the trajectory of every evidence variable E_i.
4. Forget the evidence $\boldsymbol{e}^{t_0}$ and take as new evidence $\boldsymbol{e}^{t_f} = arg\,max_{\boldsymbol{E}^{t_f}} P(\boldsymbol{E}^{t_f}|\boldsymbol{e}^{t_0})$ (which was stored in the previous step).
5. Compute the upper and lower trajectories between t_0 and t_f given the evidence $\boldsymbol{e}^{t_f}$. Instead of considering the mode μ of each node given the evidence, we will consider $\mu + \lambda\sigma$ for

the upper trajectory and $\mu - \lambda\sigma$ for the lower trajectory, with a user-specified parameter λ. As such, the upper and lower trajectory of X_i given an evidence $\boldsymbol{e}^{t_0}$ is $\boldsymbol{x}_i^{+t_0:t_f} = \{\mu_{X_i^t|\boldsymbol{e}^{t_0}} + \lambda\sigma_{X_i^t|\boldsymbol{e}^{t_0}} | t \in \{t_0, t_0+1, ...t_f\}\}$ and $\boldsymbol{x}_i^{-t_0:t_f} = \{\mu_{X_i^t|\boldsymbol{e}^{t_0}} - \lambda\sigma_{X_i^t|\boldsymbol{e}^{t_0}} | t \in \{t_0, t_0+1, ...t_f\}\}$ respectively. Note that if $\lambda = 0$ the upper and lower trajectories become regular trajectories. These allow for flexibility in the explanation, as instead of presenting the most likely trajectory (the one obtained with the mean of each node), we compute an upper and lower bound. The most likely trajectory of each variable can be computed as well for additional information.

6. The explanation for $\boldsymbol{h}^{t_f*}$ will be the set of of trajectories $\boldsymbol{e_i}^{t_0:t_f}$ given $\boldsymbol{e}^{t_0}$ and the set of upper and lower trajectories $\boldsymbol{e_i}^{+t_0:t_f}$, $\boldsymbol{e_i}^{-t_0:t_f}$ given $\boldsymbol{e}^{t_f}$. The idea is to illustrate the ideal trajectory that the nodes $\boldsymbol{E}$ should follow in order to get our forecast $\boldsymbol{h}^{t_f*}$.

While our proposal is much closer to the concept of prototypes than to the shapelets, some features of shapelets also apply here. Instead of computing the full trajectory between t_0 and t_f, we could also consider just a subsequence in that time interval as in shapelets and as opposed to prototypes, who compute a full exemplary time series. Additionally, in a way such subsequence will be maximally representative of our forecast, which is the definition of shapelet, but adapted to the forecasting problem.

Please note that the procedure described might be overgeneralised. In most cases, the variables of interest $\boldsymbol{H}$ are going to be the evidence variables $\boldsymbol{E}$ themselves in future instants, $\boldsymbol{H}^{t_f} = \boldsymbol{E}^{t_f} \subseteq \boldsymbol{X}^{t_f}$. In addition, in many domains, given a known time instant t we will know the state of every variable $\boldsymbol{X}$ instead of a subset $\boldsymbol{E} \subseteq \boldsymbol{X}$ in instant t, and thus $\boldsymbol{H} = \boldsymbol{E} = \boldsymbol{X}$. In the latter case, the procedure just simplifies to (1) perform forecast from a known time t_0 (where all variables are observed) to an unknown time t_f (all variables unobserved), (2) then take the prediction as an evidence and perform smoothing until time t_0 and (3) compute the upper and lower trajectories.

3.3. Highlighting the changes of the ns-DBN across time

3.3.1. Introduction

In this section, we focus on model explanations for non-stationary DBNs and, specially, for TV-DBNs. Our goal is to summarise the changes in both the network structure and parameters to explain how the data changes across time and potentially explaining the concept drift in the data. Our proposal is based on the use of structure and probability distribution distances (see Section 2.2.6) and the idea is to make such distances interpretable to the end user.

In Section 3.3.2, we disclose how to explain differences in the structure. However, such explanations might be insufficient to understand quantitatively the model and, as such, in Section 3.3.3 we try to explain the change in the parameters.

3.3.2. Changes in the structure

We can measure the changes in the structure using the Global and Hamming distances discussed in Section 2.2.6. The goal here is two-fold:

1. To find the points where a bigger concept drift occurs in TV-DBNs. While this is useful also in non-stationary DBN, usually the points where there is a noticeable concept drift are

known (or estimated) before the construction of the network using various methods (see Section 2.3.1). However, in TV-DBNs, the network is constructed without identifying these points and, as such, we might as well identify them using the graph rather than the raw data, as the Bayesian networks are meant to approximate and factorise the probability distribution of the data, as explained in Section 2.2.1. This goal can be understood as explainability at the macro-level, where we want to locate and understand *where* the changes are happening.

2. To study the difference between two transition networks and, by extension, how the process modelled has changed over time. On the one hand, focusing just in the structure results in losing quantitative information but on the other hand, the explanation is much simpler and visual. In contrast with the previous goal, this one can be seen as explainability at the micro level, since we are interested in studying *how* are these changes.

In order to achieve both of these goals, we will resort to the global (Equation (2.10)) and Hamming (equation (2.11)) distances and make them explainable.

For the first goal, we will compute the distance between the first transition network and the rest of them and plot the distances. This will allow us to see the points where the distribution differs more from the original transition network. The distance between every two consecutive transition networks will also be computed for visualizing the points where the changes are more abrupt and to locate time intervals in which the variance in the transition network changes might be higher than expected.

The second goal is, in our opinion, the one that gives more insight about the actual changes in the network. While for the previous one we compute the distances and use the results to explain the network, now we are interested in understanding (semantically speaking) what the global and Hamming distances actually do and how to translate its "reasoning" into a Bayesian network plot.

The global distance counts the number of correct arcs (oriented in the same direction in both networks), in relation to the total number of arcs. It is a "positive metric", in the sense that a higher value for the global distance means that networks are more similar. To visualise this idea to a plot, we propose to plot an output transition network in which only plot the edges that are oriented equally in the two transition networks (see an example in Figure 3.2). This way, we will visually locate the arcs that are *persistent* in the DBN. We will refer to the DAG obtained as *global graph* $\mathcal{G}_G$, obtained by applying the logical AND over the set of arcs of two input graphs with the same set of vertexes:

$$\mathcal{G}_G = \mathcal{G}_1 \wedge \mathcal{G}_2. \tag{3.1}$$

Note that the result of performing the logical AND between two arcs directed in opposite directions for the same two vertexes is having no arc between those two vertexes in the global graph $\mathcal{G}_G$. This situation will never occur if there are only inter-slice arcs.

The Hamming distance, intuitively speaking, counts the number of errors, i.e., arcs missing in any of the two networks in relation with the other and arcs that are present, but incorrectly oriented. In contrast to the global distance, the Hamming metric is "negative", because a higher

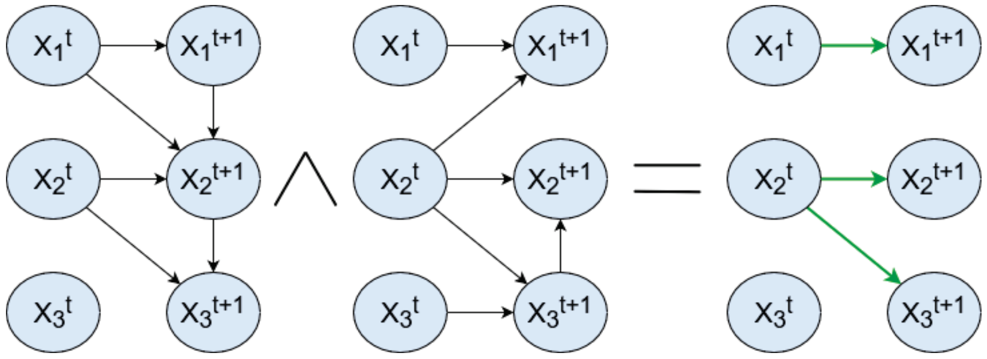


Figure 3.2: DBN global graph. We apply the AND operator over the set of arcs of the input transition network and the result shows the arcs that are persistent

value indicates more dissimilarity between the two input networks to be compared. Similarly, we can plot an output transition network with just the arcs that are wrong when comparing both networks, visualizing then the *contingent* arcs in the transition networks (see an example in Figure 3.3). The graph obtained is referred to as *Hamming graph* $\mathcal{G}_H$, and can be formally obtained by applying the XOR operator to the set of arcs of a pair of input graphs with the same set of vertexes:

$$\mathcal{G}_H = \mathcal{G}_1 \oplus \mathcal{G}_2. \tag{3.2}$$

It is noteworthy that there will be no incorrectly oriented arcs when dealing with a DBN with no intra-slice arcs, since all the arcs will be oriented from the past to the future and never the other way around. This ensures that $\mathcal{G}_H$ will be a DAG, as explained below. But if there are intra-slice arcs, how do we compute the XOR of two arcs between the same vertices that are oriented incorrectly? There are two options:

1. Leave both directed arcs in $\mathcal{G}_H$. As a result, the graph obtained will no longer be a DAG, but a graph with cycles.

2. Draw an (undirected) edge between the two vertex in $\mathcal{G}_H$. The graph obtained will be then a partially directed acyclic graph.

Both proposals can actually be used simultaneously if we plot the persistent arcs (global distance) in green and the contingent arcs (Hamming distance) in red. If we want a deeper level of information, we could use different tones of red (or outright different colours) to represent the arcs that are present in just the first network and the arcs that are just in the second one. In a single plot we can see then the changes and similarities of both networks.

This solution can easily be extended in order to compare more than two transition networks, allowing us to compare tendencies over time instead of just two given time points. It is specially interesting to see which edges persist for a long time in the network. The multiple input adaptation for both the global and Hamming distance would be, for k networks.

$$\mathcal{G}_G = global\_graph(\mathcal{G}_1, ..., \mathcal{G}_k) = global\_graph(\mathcal{G}_1, global\_graph(\mathcal{G}_2, ..., \mathcal{G}_k)) \tag{3.3}$$

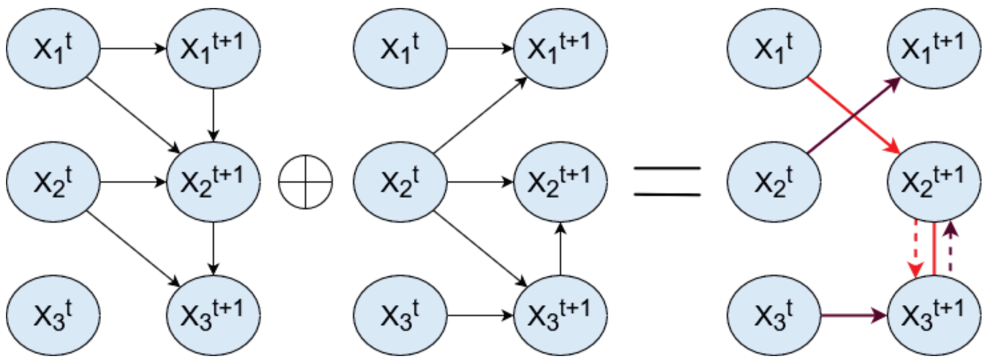


Figure 3.3: DBN Hamming graph. We apply the XOR operator over the set of arcs of the the input transition networks and the result shows the arcs that are contingent. The arcs that appear just in the first network and the ones that appear just in the second are drawn in red and purple respectively in the resulting Hamming graph. The arc oriented incorrectly between X_2^{t+1} and X_3^{t+1} can be plotted using an undirected arc (solid line) or with two undirected arcs (dashed lines)

$$\mathcal{G}_H = hamming\_graph(\mathcal{G}_1, ..., \mathcal{G}_k) = hamming\_graph(\mathcal{G}_1, \mathcal{G}_G) \vee ... \vee hamming\_graph(\mathcal{G}_k, \mathcal{G}_G) \tag{3.4}$$

For the Hamming distance, we might have to define the OR operation over two directed arcs in opposite directions between the same two verteices. As with the XOR operator, we might leave both directed arcs in the resulting graph or plot an undirected edge. Similarly, this situation will never occur in DBN with no intra-slice arcs.

Nonetheless, it might happen that if we try to find the global graph for a large set of transition networks (for example, the global graph of the whole network), we might find out that there are no persistent arcs. Take the limit situation where we input 100 transition networks and an arc is present in 99 of those networks: in the global graph, such edge will not appear. To patch this issue, we might include some kind of tolerance threshold. If set to $\delta \in [0, 1]$, it means that, if at least $100 \cdot x\%$ of the input graphs contain a certain edge, which will be considered persistent. To have a visual clue of which edges are more or less persistent we can set different thickness and/or colour saturations to the arcs. Thicker edges and/or darker tone means that the edge appears in more transition networks, whereas thinner edges and brighter colour means the opposite (the edge appears in fewer transition networks).

Another proposal would be to find the global graph of a set of consecutive transition networks of a TV-DBNs, creating a transition network with just the persistent arcs of such interval. We refer to this concept as summary network.

To end this section, we will outline that we might add some quantitative information. We could give weights to the arcs according to the β parameters. Therefore, the weight of the arc (X_i, X_j) would be $\beta_{X_j X_i}$.

3.3.3. Changes in the probability distribution

Another way to study the changes in the non-stationary DBN is to examine the changes quantitatively. The idea then is to compare the probability distribution of two transition networks $P_i(\boldsymbol{X}^t|\boldsymbol{X}^{0:t-1})$ and $P_j(\boldsymbol{X}^t|\boldsymbol{X}^{0:t-1})$, for any set of different time i and j.

We can do this using the distance measures for probability distributions presented in Section 2.2.6. Since we will be comparing conditional distributions, we might need to adapt the definitions of the distance to account for such cases. When we have an observation $\boldsymbol{x}$ from which the set of random variables $\boldsymbol{Y}$ depends, the Kullback-Leibler divergence for two probability distributions is:

$$KL(P(\boldsymbol{Y}|\boldsymbol{x})||Q(\boldsymbol{Y}|\boldsymbol{x})) = \sum_{\boldsymbol{y}} P(\boldsymbol{y}|\boldsymbol{x}) log \frac{P(\boldsymbol{y}|\boldsymbol{x})}{Q(\boldsymbol{y}|\boldsymbol{x})} \tag{3.5}$$

When $\boldsymbol{Y}$ on a random variable (or set of random variables) $\boldsymbol{X}$ rather than in a specific event $\boldsymbol{x}$, then the Kullback-Leibler divergence can be computed as the weighted sum of the divergence for every possible value $\boldsymbol{x} \in \Omega(\boldsymbol{X})$, where the weight used is the marginal probability of each event $\boldsymbol{x}$:

$$KL(P(\boldsymbol{Y}|\boldsymbol{X})||Q(\boldsymbol{Y}|\boldsymbol{X})) = \sum_{\boldsymbol{x}} P(\boldsymbol{x}) KL(P(\boldsymbol{Y}|\boldsymbol{x})||Q(\boldsymbol{Y}|\boldsymbol{x})) = \sum_{\boldsymbol{x}} P(\boldsymbol{x}) \sum_{\boldsymbol{y}} P(\boldsymbol{y}|\boldsymbol{x}) log \frac{P(\boldsymbol{y}|\boldsymbol{x})}{Q(\boldsymbol{y}|\boldsymbol{x})} = \sum_{\boldsymbol{x},\boldsymbol{y}} P(\boldsymbol{y}|\boldsymbol{x}) P(\boldsymbol{x}) log \frac{P(\boldsymbol{y}|\boldsymbol{x})}{Q(\boldsymbol{y}|\boldsymbol{x})} = \sum_{\boldsymbol{x},\boldsymbol{y}} P(\boldsymbol{x},\boldsymbol{y}) log \frac{P(\boldsymbol{y}|\boldsymbol{x})}{Q(\boldsymbol{y}|\boldsymbol{x})} \tag{3.6}$$

The conditional Jensen-Shannon divergence (see Equation (2.17)) is:

$$\begin{aligned} &JSD(P(\boldsymbol{Y}|\boldsymbol{X})||Q(\boldsymbol{Y}|\boldsymbol{X})) \\ &= \frac{1}{2} KL(P(\boldsymbol{Y}|\boldsymbol{X})||M(\boldsymbol{Y}|\boldsymbol{X})) + \frac{1}{2} KL(Q(\boldsymbol{Y}|\boldsymbol{X})||M(\boldsymbol{Y}|\boldsymbol{X})) \\ &= \frac{1}{2} \sum_{\boldsymbol{x}} \left(P(\boldsymbol{x}) \left(KL(P(\boldsymbol{Y}|\boldsymbol{x})||M(\boldsymbol{Y}|\boldsymbol{x})) + KL(Q(\boldsymbol{Y}|\boldsymbol{x})||M(\boldsymbol{Y}|\boldsymbol{x}))\right)\right) \end{aligned} \tag{3.7}$$

We are interested in using the Jensen-Shannon divergence since it is a true metric, unlike the Kullback-Leibler divergence. Furthermore, the computation of both divergences for a joint probability distributions can be factorised using the chain rule.

Lemma 3.3.1. *Given the Kullback-Leibler divergence $KL(P||Q)$ of two probabilities distributions over the same set of random variables $\boldsymbol{X} = \{X_1, ..., X_n\}$, it can be decomposed into the computation of conditional Kullback-Leibler divergences using the chain rule.*

$$\begin{aligned} &KL(P(X_1, ..., X_n)||Q(X_1, ..., X_n)) = \\ &\quad = KL(P(X_1|X_2, ..., X_n)||Q(X_1|X_2, ..., X_n)) + KL(P(X_2, ..., X_n)||Q(X_2, ..., X_n)) \end{aligned}$$

Proof.

$$\begin{aligned} &KL(P(X_1, ..., X_n)||Q(X_1, ..., X_n)) \\ &= \sum_{X_1,...,X_n} P(X_1, ..., X_n) \cdot log \left(\frac{P(X_1, ..., X_n)}{Q(X_1, ..., X_n)}\right) \end{aligned}$$

$$
\begin{aligned}
&= \sum_{X_1,...,X_n} P(X_1,...,X_n) \cdot log\left(\frac{P(X_1|X_2...,X_n)P(X_2,...,X_n)}{Q(X_1|X_2...,X_n)Q(X_2,...,X_n)}\right)\\
&= \sum_{X_1,...,X_n} P(X_1,...,X_n) \cdot \left(log\left(\frac{P(X_1|X_2...,X_n)}{Q(X_1|X_2...,X_n)}\right) + log\left(\frac{P(X_2,...,X_n)}{Q(X_2,...,X_n)}\right)\right)\\
&= \sum_{X_1,...,X_n} P(X_1,...,X_n) \cdot log\left(\frac{P(X_1|X_2...,X_n)}{Q(X_1|X_2...,X_n)}\right) + \sum_{X_1,...,X_n} P(X_1,...,X_n) \cdot log\left(\frac{P(X_2,...,X_n)}{Q(X_2,...,X_n)}\right)\\
&= KL(P(X_1|X_2,...,X_n)||Q(X_1|X_2,...,X_n)) + \sum_{X_1,...,X_n} P(X_1,...,X_n) \cdot log\left(\frac{P(X_2,...,X_n)}{Q(X_2,...,X_n)}\right)\\
&= KL(P(X_1|X_2,...,X_n)||Q(X_1|X_2,...,X_n)) + \sum_{X_1,...,X_n} P(X_1|X_2,...,X_n)P(X_2,...,X_n) \cdot log\left(\frac{P(X_2,...,X_n)}{Q(X_2,...,X_n)}\right)\\
&= KL(P(X_1|X_2,...,X_n)||Q(X_1|X_2,...,X_n)) + \sum_{X_2,...,X_n} P(X_2,...,X_n) \cdot log\left(\frac{P(X_2,...,X_n)}{Q(X_2,...,X_n)}\right) \cdot \sum_{X_1} P(X_1|X_2,...,X_n)\\
&= KL(P(X_1|X_2,...,X_n)||Q(X_1|X_2,...,X_n)) + \sum_{X_2,...,X_n} P(X_2,...,X_n) \cdot log\left(\frac{P(X_2,...,X_n)}{Q(X_2,...,X_n)}\right)\\
&= KL(P(X_1|X_2,...,X_n)||Q(X_1|X_2,...,X_n)) + KL(P(X_2,...,X_n)||Q(X_2,...,X_n))
\end{aligned}
$$

□

Since the Kullback-Leibler divergence can be decomposed using the rule chain, so can the Jensen-Shannon divergence.

Lemma 3.3.2. *Given the Jensen-Shannon divergence $JSD(P||Q)$ of two probability distributions over the same set of random variables $\boldsymbol{X} = \{X_1,...,X_n\}$, it can be decomposed into the computation of conditional Jensen-Shannon divergences using the chain rule.*

$$
\begin{aligned}
&JSD(P(X_1,...,X_n)||Q(X_1,...,X_n)) =\\
&\qquad = JSD(P(X_1|X_2,...,X_n)||Q(X_1|X_2,...,X_n)) + JSD(P(X_2,...,X_n)||Q(X_2,...,X_n))
\end{aligned}
$$

Proof. In order to proof the previous equality, we resort to the definition of the Jensen-Shannon divergence (Equation (2.17)) and then apply the chain rule for the two Kullback-Leibler divergences obtained (Theorem (3.3.1)).

$$
\begin{aligned}
&JSD(P(X_1,...,X_n)||Q(X_1,...,X_n))\\
&= \frac{1}{2}KL(P(X_1,...,X_n)||M(X_1,...,X_n)) + \frac{1}{2}KL(Q(X_1,...,X_n)||M(X_1,...,X_n))\\
&= \frac{1}{2}\left(KL(P(X_1|X_2,...,X_n)||M(X_1|X_2,...,X_n)) + KL(P(X_2,...,X_n)||M(X_2,...,X_n))\right)\\
&\qquad + \frac{1}{2}\left(KL(Q(X_1|X_2,...,X_n)||M(X_1|X_2,...,X_n)) + KL(Q(X_2,...,X_n)||M(X_2,...,X_n))\right)\\
&= \frac{1}{2}KL(P(X_1|X_2,...,X_n)||M(X_1|X_2,...,X_n)) + \frac{1}{2}KL(Q(X_1|X_2,...,X_n)||M(X_1|X_2,...,X_n))\\
&\qquad + \frac{1}{2}KL(P(X_2,...,X_n)||M(X_2,...,X_n)) + \frac{1}{2}KL(Q(X_2,...,X_n)||M(X_2,...,X_n))\\
&= JSD(P(X_1|X_2,...,X_n)||Q(X_1|X_2,...,X_n)) + JSD(P(X_2,...,X_n)||Q(X_2,...,X_n))
\end{aligned}
$$

□

We can use the Jensen-Shannon divergence to compare the probability distribution of two transition networks with probability distributions P and Q and, thanks to the chain rule, its computation equals to the sum of the Jensen-Shannon divergences between every pair of nodes (one from each network) given the union of its parents in both networks. Formally:

$$JSD(P(\boldsymbol{X})||Q(\boldsymbol{X})) = JSD(P(X_1|Pa_{X_{1PQ}})||Q(X_1|Pa_{X_{1PQ}})+...+JSD(P(X_1|Pa_{X_{nPQ}})||Q(X_1|Pa_{X_{nPQ}}), \tag{3.8}$$

where $Pa_{X_{npq}}$ is the union of the set of parents of the variable X_n in the first and the second network.

However, in some cases we might be unable to use the chain rule, as it can be impossible to find a common ancestral ordering for two transition networks (see Figure 3.4). Nonetheless, such common ordering can be found when we are trying to compare two transition networks with no intra-slice arcs (or Bayesian networks with a k-partite graph structure where the flow of the arcs between sets is restricted). This is true because any node ordering where we put first the nodes in the older time instant (regardless of the ordering inside the same time instant) is going to be a common ancestral ordering for both networks. As such, we formulate the theorem 3.3.4. First, we define the concept of *sequential k-partite DAG* in the definition 3.3.3.

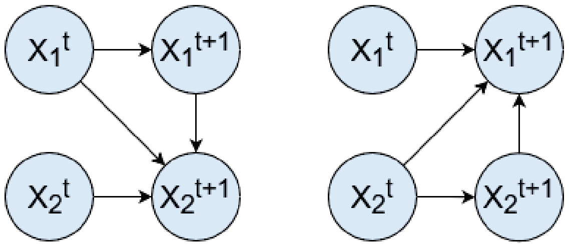


Figure 3.4: Pair of transition network with no common ancestral node ordering

Definition 3.3.3. *A sequential k-partite DAG is a directed k-partite graph in which the independent sets of nodes are numbered from* 1 *to* k *and no arc is allowed from any node* X_i *in the independent set* i *to any node* X_j *in the independent set* j *iff* $i > j$.

Such restrictions avoids any cycle in the graph. An example of sequential k-partite graph can be seen in Figure 3.5. Such graph is a generalization of a transition network of Markovian order τ with only inter-slices arcs, since a parallel can be drawn between the numbered sets of independent nodes and the time slices with no intra-slices arcs.

Theorem 3.3.4. *The Jensen-Shannon divergence between two joint probability distributions over the same set of variables* $\boldsymbol{X} = \{X_1, ..., X_n\}$ *factorised in two Bayesian networks can be factorised in the sum of the* JSD *of the nodes given their parents in both networks to compare, if the structure of both networks is a sequential k-partite DAG with the same independent sets.*

Proof. If $\mathcal{G}_\mathcal{P}$ and $\mathcal{G}_\mathcal{Q}$ are the two k-partite DAGs with the same set of nodes and sets of independent nodes (from 1 to k) corresponding to the distributions P and Q, we just need to prove that there can always be a common ancestral ordering for the nodes.

Let $\boldsymbol{X}^t$ denote the set of nodes of the t-th independent set and let X_i^t denote the i-th node of the t-th set of independent nodes. If we select any ancestral ordering where the nodes in the first independent set are the firsts in the ordering and so on, the order will be valid for both networks, since the ordering between nodes within the same independent set is irrelevant (as there are no intra-slice arcs).

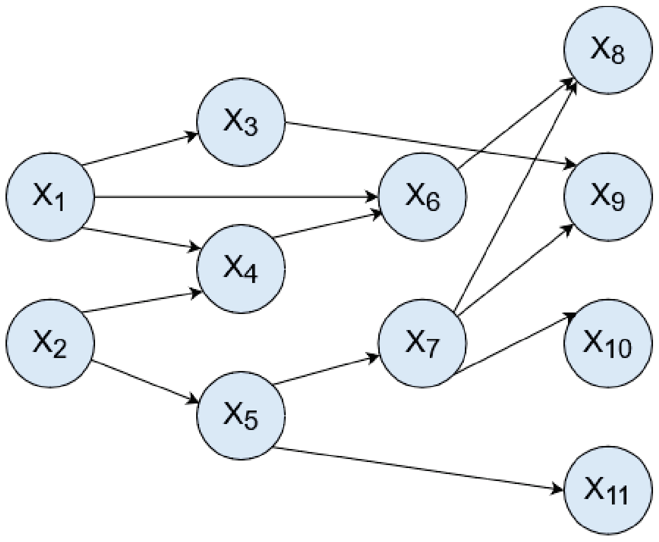


Figure 3.5: Example of a sequential k-partite graph with four independent sets, $\boldsymbol{S}_1 = \{X_1, X_2\}$, $\boldsymbol{S}_2 = \{X_3 X_4, X_5\}$, $\boldsymbol{S}_3 = \{X_6, X_7\}$ and $\boldsymbol{S}_4 = \{X_8, X_9, X_{10}, X_{11}\}$. Arcs can only be found from nodes of a independent sets to nodes of a different independent set with a higher subindex

Consider the t-th independent set that contains n nodes. The conditional probability of the nodes in the set for an order between those nodes could be $P(X_1^t|\boldsymbol{X}^{0:t-1})$, $P(X_2^t|X_1^t, \boldsymbol{X}^{0:t-1}), ..., P(X_n^t|X_1^t, ..., X_{n-1}^t \boldsymbol{X}^{0:t-1})$. However, since the nodes of $\boldsymbol{X}^t$ are conditionally independent from one another given their parents, all those conditional probabilities are equal to $P(X_i^t|\boldsymbol{Pa}_{X_i^t})$, regardless of the ordering of the nodes in $\boldsymbol{X}^t$. $\square$

The possibility of decomposing the joint Jensen-Shannon divergence into atomic divergences offers two advantages:

1. It alleviates the computational burden of the divergence.

2. It allows us to see how the distribution of each independent node has drifted in addition to how the overall probability of the transition network has changed.

Now, all what we have to do is to compute the Jensen-Shannon divergence between $P(\boldsymbol{X}^t|\boldsymbol{X}^{0:t-1})$ and $Q(\boldsymbol{X}^t|\boldsymbol{X}^{t-1})$ (assuming a first Markovian order). Since $P(\boldsymbol{X}^t|\boldsymbol{X}^{t-1}) \propto P(\boldsymbol{X}^t, \boldsymbol{X}^{t-1})$, we might as well compute the joint probability, as we have demonstrated that can be decomposed.

However, a final problem arises: if we are working with continuous variables, we might to rely heavily in Monte Carlo methods to obtain some results, for instance:

- Computing the geometric mean of two univariate continuous densities, which is needed for computing the Jensen-Shannon divergence (see Equation (2.18)).

- In the conditional Jensen-Shannon divergence, we need to sum for all values of X (see Equation (3.6)), which has to be approximated with sampling in continuous domains for the corresponding integral.

While the latter problem will remain unsolved by exact means, in this work we provide a simple formula in order to compute the geometric mean of the density function of two univariate Gaussian variables.

Lemma 3.3.5. *Given a Gaussian density $\mathcal{N}(\mu, \sigma)$ with function $f(x)$, the result of computing the n-th power of such density function represents a normal distribution $\mathcal{N}(\mu, \frac{\sigma}{\sqrt{n}})$ with an escalated density function.*

Proof.

$$\big(f(x)\big)^n = \left(\frac{1}{\sqrt{2\pi}\sigma} e^{\frac{(x-\mu)^2}{2\sigma^2}}\right)^n = \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n \left(e^{\frac{(x-\mu)^2}{2\sigma^2}}\right)^n$$

First, we focus on the right part of the product, which can be rewritten as:

$$\left(e^{\frac{(x-\mu)^2}{2\sigma^2}}\right)^n = e^{n\frac{(x-\mu)^2}{2\sigma^2}} = e^{\frac{(x-\mu)^2}{2\frac{\sigma^2}{n}}} = e^{\frac{(x-\mu)^2}{2\left(\frac{\sigma}{\sqrt{(n)}}\right)^2}}$$

This already proves that $(f(x))^n$ is a scaled Gaussian density function that represents the distribution $\mathcal{N}(\mu, \frac{\sigma}{\sqrt{n}})$.

We can obtain the scaling factor if we operate in the left part of the product. Our objective will be to swap the σ in the denominator by $\frac{\sigma^2}{n}$ and see the factor that remains multiplying the left part.

First, we have that:

$$\left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n = \frac{1}{\sqrt{2\pi}^n \sigma^n} = \frac{1}{\sqrt{2\pi}\sqrt{2\pi}^{n-1}\sigma^n} = \frac{\sqrt{2\pi}^{-(n-1)}\sigma^{-n}}{\sqrt{2\pi}}$$

Next, we multiply and divide by $\frac{\sigma}{\sqrt{n}}$:

$$\left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n = \frac{\sqrt{2\pi}^{-(n-1)}\sigma^{-n}\frac{\sigma}{\sqrt{n}}}{\sqrt{2\pi}\frac{\sigma}{\sqrt{n}}} = S\frac{1}{\sqrt{2\pi}\frac{\sigma}{\sqrt{n}}}$$

Thus, the scaling factor will be:

$$S = \sqrt{2\pi}^{-(n-1)}\sigma^{-n}\frac{\sigma}{\sqrt{n}}$$

□

Lemma 3.3.6. *Given two Gaussian densities $\mathcal{N}(\mu_p, \sigma_q)$ and $\mathcal{N}(\mu_p, \sigma_q)$ referred as $f(x)$ and $g(x)$, the product of the density functions represents a normal distribution $\mathcal{N}(\mu_{fg}, \sigma_{fg})$, where*

$$\mu_{fg} = \frac{\mu_f \sigma_g^2 + \mu_g \sigma_f^2}{\sigma_f^2 + \sigma_g^2} \qquad \textit{and} \qquad \sigma_{fg} = \sqrt{\frac{\sigma_f^2 \sigma_g^2}{\sigma_f^2 + \sigma_g^2}}$$

Proof. A full proof for this lemma is provided in Bromiley (2003). □

Theorem 3.3.7. *Given two Gaussian densities $\mathcal{N}(\mu_f, \sigma_g)$ and $\mathcal{N}(\mu_f, \sigma_q)$ referred as $P(x)$ and $Q(x)$, the geometric mean of such density functions represents a normal distribution $\mathcal{N}(\mu_m, \sigma_m)$, where*

$$\mu_m = \frac{\mu_f \sigma_g^2 + \mu_g \sigma_f^2}{\sigma_f^2 + \sigma_g^2} \qquad \textit{and} \qquad \sigma_m = \sqrt{\frac{2\sigma_f^2 \sigma_g^2}{\sigma_f^2 + \sigma_g^2}}$$

Proof. These formulas can be obtained by first applying to each distribution the equation in Lemma 3.3.5 and then multiplying the density functions using Lemma 3.3.6, according to the definition of the geometric mean for continuous distributions (Equation (2.18)). The full proof can be found in Appendix A. □

This simplification allows for an exact and faster computing of the Jensen-Shannon divergence for univariate Gaussian densities. In TV-DBNs, the simplification allows for computing multivariate Gaussian, as it can be decompose in a summation of univariate Gaussian densities.

3.4. A library for learning time-varying dynamic Bayesian networks

3.4.1. Overview

As mentioned in Section 2.3.2, there is not an available library for learning TV-DBNs. As such, we have no ground to test the proposals of this work. For that reason, we implement ourselves TV-DBNs in a library called `tvdbn`[1].

We decided to make the implementation in the programming language R (R Core Team, 2020). The main reason is to build on top of the already existing library `bnlearn` (Scutari, 2010), that let us learn discrete and Gaussian Bayesian networks and whose utilities will serve us to program our own library. Another worth-mentioning option as a baseline tool is the pomegranate library (Schreiber, 2017) for Python.

We decided to not use the library `dbnR`[2], developed by another member of the Computation Intelligence Group at the Technical University of Madrid. The aforementioned library is able to learn stationary dynamic Gaussian Bayesian networks with Markovian order τ. This decision is motivated by the fact that TV-DBNs does not allowed template-based representation (as every slice is different), which is the most interesting feature of `dbnR`.

[1]Available in GitHub: `https://github.com/Enrique-Val/tvdbn`
[2]Latest version available in GitHub: `https://github.com/dkesada/dbnR`

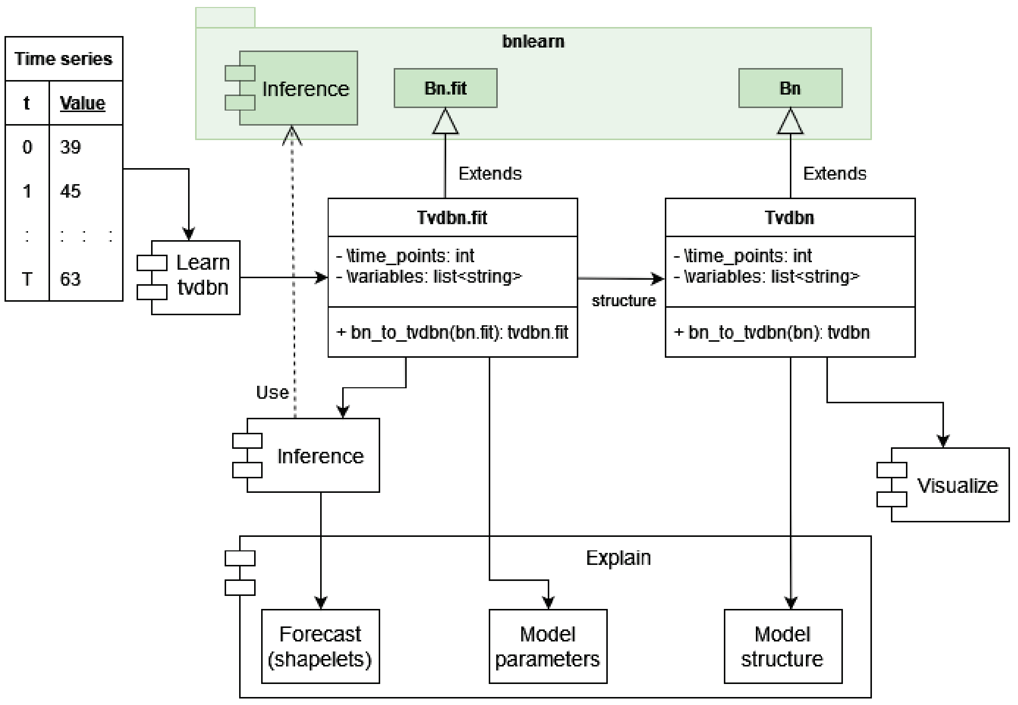


Figure 3.6: tvdbn library class diagram

While R is not an object-oriented programming language, we offer a UML-like class diagram (Figure 3.6) to illustrate the organization of our library. We also keep the notation somewhat in line with the paradigm of object oriented programming, although the notation differs in the implementation. In short, we defined two classes and a set of modules in which we organised the different functionality of our library.

3.4.2. Classes

The two classes defined are:

1. Tvdbn.fit, which is contained in the R file tvdbn_class. This class represents a time-varying dynamic Bayesian network that is fitted, and so the class represents both the structure and the parameters $P(X_i^t|\boldsymbol{Pa}_{X_i^t}), \forall X_i \in \boldsymbol{X}\ \forall t \in \{0, 1, ..., T\}$. Tvdbn.fit extends the class Bn.fit of the `bnlearn` package, adding two derived attributes:

   - time_points, a numerical attribute that represents the total number of time instants $T + 1$ in the TV-DBN.

   - variables, a list (string) that represents the set of variables $\boldsymbol{X}$ in the network.

2. Tvdbn, which is also contained in the R file tvdbn_class. This class represents the structure of a TV-DBN, but not the parameters. When training a TV-DBN both the structure and parameters are learned at the same time and, as such, Tvdbn is a by-product of the learning process that can be obtained with a “get” function that gives the user just the structure of the TV-DBN. It has the same derived attributes as Tvdbn.fit and, similarly, this class inherits from Bn.

3.4.3. Modules

Next, we are going to define the modules implemented:

- Learning module, which correspond to an R file named tvdbn_learn. The learning process is implemented as described in Song et al. (2009), i.e. we train a linear regression for every X_i^t, finding the best linear combination of the previous time slice $\boldsymbol{X}^{t-1}$ that explains it and using a Gaussian kernel as weighting function. Additionally, relaxed TV-DBNs and the causal gamma kernel function for causal TV-DBNs (Chu et al., 2013) are implemented. We added a fourth implementation that forces arcs between X_i^{t-1} and X_i^t, $\forall X_i \in \boldsymbol{X}$, $\forall t \in \{0, 1, ..., T\}$, which we called “autoregressive”.

  For implementing the linear regression, we use the R package `glmnet` (Friedman et al., 2010), as recommended by one of the original authors of TV-DBNs (Kolar, 2021). The only problem that we find with such implementation is that it does not allow for user-selected initialisation of the $\boldsymbol{\beta}$ parameters, a key factor of the learning process in TV-DBN. A re-implementation of `glmnet` that allowed for user-provided initialisation has been considered[3], but it was discarded for two reasons: (1) `glmnet` offers many more guarantees as it is implemented by an expert team of researchers and (2) the improvement in speed-up in comparison with the original `glmnet` was not significant.

- Visualisation module, implemented into the R file tvdbn_visualization. This module allows for the visualisation of the whole network, part of it or a concrete transition network.

- Inference module, which corresponds to an R file named tvdbn_inference. This module is partially powered by the inference of `bnlearn`. However, the forecasting problem in TV-DBNs can be solved much more optimally using exact inference rather than probabilistic logical sampling (as done with `bnlearn`), when a certain scenario is given. An entire time slice $\boldsymbol{X}^t$ must be observed. This is because we are propagating information in the direction of the arcs and the nodes in each time slice are independent of one another, since all the parents are observed and there are no intra-slice arcs. As such, we could just do the linear combination of the parents for each node in the network time step by time step and apply a Gaussian noise equivalent to the standard deviation of said node. This scenario is implemented in the inference module.

- Explainability module, which correspond to an R file named tvdbn_explain. At this moment, this module offers an implementation for computation of the Hamming distance between two transition networks, as well as the computation of the global (with a tolerance threshold) and Hamming graphs, a summary network (the global graphs of different intervals of the network) and allows for trimming a network to focus only on a set of variables and, optionally, on their influences (parents and children).

3.4.4. Learning from spatial data

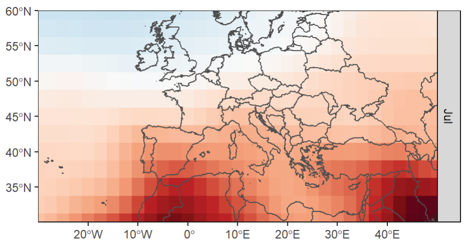
In the library, we also introduce an option to learn from data that is spatially structured, since the ultimate goal of the technical advances of this work is to be applied in climate sciences, where data structure is usually not only modelled in time but also in space, as it is the case of the NCEP/DOE RII database.

[3] Available in GitHub: `https://github.com/junyangq/glmnetPlus`

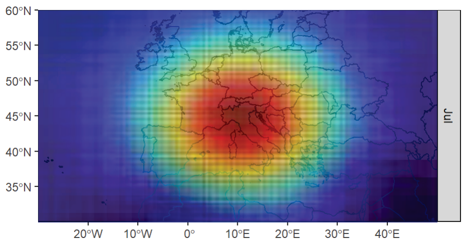
Outside the library, we created a method that converts an instance of a grid time series -which is a three-dimensional array (or an array of matrix) where the dimensions are the time, latitudes and longitudes- to a standard time series dataset with two dimensions, time and variables. For each grid (two-dimensional data with latitude and longitude), we put all the data into a one-dimensional array. Each position of the array still represents a coordinate, although it is compressed into a single dimension. The indexes of the array will be the Cartesian product between the set of latitudes and the set of longitudes of the data.

The previous implementation allows for processing spatial data as if it were a regular multivariate time series. However, in this process, we lose information about how the data is spatially structured and, in the learning process, spurious and unlikely associations are learned (for example, that the mean temperature of Spain is directly conditioned by the mean temperature of Siberia). To solve this problem, we introduce a penalty in the learning process that will increase with the distance between two coordinates. In our work, the Manhattan distance was selected, as we are working with an ordered grid. This is similar to one of the core philosophies of the TV-DBNs, where we give more weights to the instances that are closer to the instant that we are learning.

This idea is modelled as follows. As defined in Song et al. (2009), the arcs between a node slice $\boldsymbol{X}^{t-1}$ and a variable X_i^t are learned using linear regression with $l1$-regularization. The penalty factor λ tend to be the same for every variable in the series. However, we propose to give different penalty factors to different variables. If $\boldsymbol{\mu}_{X_i}$ (a bi-dimensional-vector) denotes coordinates of variable X_i^t, the penalty factor is modelled as the inverse of the probability density function of a bivariate Gaussian with mean μ_{X_i} and user-specified co-variance matrix $\boldsymbol{\Sigma}$ (see Figure 3.7 for an example). Another option would be to assume the penalty to follow a univariate Gaussian with mean 0 and user-specified variance, modelling the (Euclidean) distance between two coordinates. The higher the value of the standard deviation, the more uniform the penalization will be. If we want none, we set the standard deviation to infinite.



(a) Map of temperatures, from NCEP/DOE Reanalysis II



(b) Penalty function

Figure 3.7: Intuitive idea of the Gaussian spatial penalty function

Chapter 4

Experimental results

4.1. Purpose of the experiments and initial hypothesis

The ultimate goal of this work is to advance in the explainability field in dynamic systems. As such, we find that the objective of these experiments is two-fold.

- The first goal is to empirically analyse our proposal using numerical results that allow us to rigorously validate it. We are interested in seeing how different parameters for the learning affect our XAI proposal (global graph, Hamming distance ...) and the fitting of the networks as well.

- The second intention is to actually demonstrate how our proposals can be applied to a field of study. This objective may seem not less rigorous as the evaluation of explanations is partially subjective, but we have to keep in mind that XAI is at the intersection of social sciences (see Figure 1.4) and, as such, a more social and subjective evaluation is equally important. Ideally, we would have a control group of people (AI practitioners) validating how useful our proposals are (Valero-Leal et al., 2020).

We assume the following set of hypotheses.

1. The Hamming distance can be used to study where the biggest changes in the network are produced and, as such, indirectly study the concept drift.

2. The global and Hamming graphs can help us identify the relations that persist across the networks and study the concept drift at the micro-level respectively. Similarly, the summary network (dividing the network into intervals and calculating the global graph of each one), can help us simplify the network in order to better visualise it and explore the concept drift.

3. In TV-DBNs, when the changes are smooth in the data, the smaller the Hamming distance is, the larger the percentage of persistent arcs will be, as we expect the consecutive transition network to be very similar to each other.

4. In TV-DBNs, a higher kernel bandwidth (more equalised weights) will result in a performance boost when the changes are smooth and in a performance lost when the

changes are more abrupt. Since a wide kernel means considering more time instants for training, in case the time instants are very different (abrupt time series), this may introduce noise in the training of each time slice. On the contrary, when the network is changing smoothly, we might improve training since we are adding more meaningful instances from which to train.

5. A larger kernel bandwidth will result in a greater execution time, as more instances must be considered. With a narrower kernel bandwidth, many instances may have weight 0 and be discarded, thus faster computation is expected.

6. In spatio-temporal network, we expect the spatial-penalty to increase the accuracy for the test data, as we are introducing meaningful aprioris in our model that may help to better generalise and avoid overfitting.

7. In spatio-temporal TV-DBNs, the less uniform is the penalty (lower values for the Gaussian variance), the smaller the Hamming distance will be and number of persistent arcs will be higher.

4.2. Datasets

We will illustrate our proposal using two datasets.

4.2.1. Motor dataset

The first one is the electric motor temperature dataset[1]. This dataset records the state of many-temperature related variables two times per second (frequency of 2Hz) after starting running an electric motor. The dataset has 1330816 time records belonging to different sessions.

In our experiments, we reduce the frequency of observation from 2Hz to 0.05 Hz (i.e. an observation every 30 seconds).

4.2.2. Climate dataset

The second dataset is obtained from the NCEP/DOE Reanalysis II database (Kanamitsu et al., 2002). First we will describe the whole database and then our selection. The entire database contains observations made every 6 hours from January 1979 to this day (the database is in constant update), although we have the option to select daily or monthly observations instead. As different variables are observed, we have different Earth grids. We can categorise them in three types:

1. Pressure level. We have 7 variables at each one of the possible 17 pressure levels, ranging from 1000 hPa to 10 hPa. In total, we have $17 \times 7 = 119$ grids.

2. Surface. Observations of 12 variables in the surface of the earth. A total of 12 grids per time point.

3. Gaussian grid. Miscellaneous information about the Earth, with over 50 variables observed.

All the grids of the Earth are at a resolution of $2.5° \times 2.5°$, which results in a grid of size

[1]Available in `https://www.kaggle.com/wkirgsn/electric-motor-temperature`

144×73, which means that we have 10512 spatial points per grid. In addition, a grid represents a single variable in a time point and if we were to dump all the variables in every coordinate into a single dataset we would have approximately 2 million features and 75000 observations across time.

In our experiments, we create two dataset parting from the database that represent a relatively small percentage of the whole database. In the first dataset:

1. We focus only on the temperature at a given atmospheric pressure (we selected 850hPa).
2. We limit the observation area to Europe, specifically in latitudes that range from 30ºN to 80ºN and longitudes that range from 30°W to 60°E.
3. We limit the observation time to twenty years, from 2001 to 2020 (both included).
4. We aggregate the data by months using the mean as aggregation function.
5. Finally, we reduce the number of variables by means of subsampling the points in the grid. The subsampling function is parametrised and, in the experiments, we use a total of 45 features, distributed in a 9×5 grid.

We will refer to this dataset as monthly climate dataset.

In the second dataset:

1. We focus on the same area (Europe), variable (atmospheric pressure at 850hPa) and time range (from 2001 to 2020).
2. However, we select a single month (in our experiments, March), and aggregate by days, meaning that we have a total of 31 time points in each one of the 20 instances of the time series.
3. We apply subsampling as well to reduce the number of nodes in the network (again, 45 features distributed in a 9×5 grid).

We will refer to this second dataset as March daily climate dataset. The reason to use a second dataset from the very same database is because we hypothesise that the monthly changes are not very smooth, and the network structure may change a lot from time point to time point (Hypothesis 3 of section 4.1). We intend to analyze data coming from the same source but with a reportedly higher smoothness (day by day changes) in order to better understand the functioning of our proposal.

4.3. Experiments

First, we will draw numerical results relative to TV-DBNs. We aim to study how the size of the kernel bandwidth (and the spatial penalty in the case of spatio-temporal networks) affects a set of metrics of the obtained networks in order to validate Hypotheses 3 to 7.

To carry out the experiments, we start selecting a kernel bandwidth that is approximately $\frac{T}{7}^2$, where T is the length of the series. This value is used in the original TV-DBN article in the experiments (Song et al., 2009). Our experiments are very different and, as such, we will select higher and lower values multiplying/dividing by 10 and then some extreme values. We first analyse the motor dataset and then the climate datasets. For the climate datasets, we will also experiment with different values for the spatial penalty.

We will use 5-fold cross-validation to study our porposal, training with 80% of the data and validating with 20% of it for every value of the kernel bandwidth and spatial penalty.

The performance measures that are subject of our study are the following:

1. Mean squared error (MSE). We will measure the MSE of the forecast starting from the first time instant to the last one (sliding window). For computing the MSE, we will consider the error of every variable at every time step and, to ensure consistency in the number, the data is normalised using z-score.

   In addition to forecast from the first instant to the last, we are interested in studying how the MSE behaves at the many intervals of the network. Thus, we also performed inference over every fifth of the network (fourth for spatio-temporal networks, to roughly divides by seasons and weeks) to verify if the quality of the inference is equally good at every part of the network.

   This can be seen in the columns of Tables 4.1, 4.2 and 4.3, where “Full” refers to the inference from start to finish and, to exemplify, 1/5-2/5 refers to the inference in the second interval on the network.

2. Time. The time that takes to construct the TV-DBN.

3. Arcs. The average number of arcs per transition network.

4. Hamming. The average Hamming distance between every pair of adjacent transition networks.

5. Average persistent arcs (Avg. pers. arcs). The average percentage of arcs that are persistent in each transition network. It is computed dividing the number of persistent arcs by the number of arcs for each transition network.

6. Total persistent arcs (Total pers. arcs). The average percentage of arcs that persistent from the set of all arcs between two transition networks of the TV-DBN.

The way of presenting the results will be the following. First, we will present the MSE and the metrics (in that order) of the three datasets with various kernel bandwidths (ignoring the spatial penalty). Then, we will proceed to focus on the two climate datasets and present both the MSE and metrics obtained with different spatial penalty values.

4.3.1. Time-varying DBNs

First, we present the MSE of our three datasets: motor, climate by months and March climate by days, in Tables 4.1, 4.2 and 4.3, respectively.

Regarding the MSE, we find that higher kernel bandwidths lead to a better performance in

Table 4.1: Motor dataset MSE

	Kernel bandwidth	Full	0/5-1/5	1/5-2/5	2/5-3/5	3/5-4/5	4/5 - 5/5
Train data	0.5	0.87 ± 0.26	0.65 ± 0.27	0.65 ± 0.25	0.64 ± 0.23	0.59 ± 0.26	0.58 ± 0.26
	50	0.86 ± 0.26	0.63 ± 0.27	0.60 ± 0.23	0.58 ± 0.22	0.56 ± 0.25	0.54 ± 0.26
	500	0.86 ± 0.25	0.64 ± 0.28	0.58 ± 0.23	0.57 ± 0.22	0.53 ± 0.25	0.53 ± 0.26
	5000	0.86 ± 0.26	0.65 ± 0.29	0.58 ± 0.23	0.56 ± 0.24	0.52 ± 0.24	0.52 ± 0.25
	5e+05	0.89 ± 0.28	0.69 ± 0.35	0.58 ± 0.25	0.57 ± 0.24	0.52 ± 0.23	0.53 ± 0.23
Test data	0.5	0.91 ± 0.3	0.68 ± 0.3	0.68 ± 0.27	0.66 ± 0.24	0.62 ± 0.28	0.61 ± 0.28
	50	0.90 ± 0.31	0.66 ± 0.3	0.64 ± 0.28	0.62 ± 0.25	0.59 ± 0.3	0.56 ± 0.28
	500	0.90 ± 0.31	0.67 ± 0.3	0.60 ± 0.26	0.61 ± 0.25	0.55 ± 0.29	0.54 ± 0.28
	5000	0.90 ± 0.31	0.66 ± 0.31	0.59 ± 0.25	0.60 ± 0.27	0.53 ± 0.27	0.53 ± 0.26
	5e+05	0.92 ± 0.32	0.70 ± 0.37	0.60 ± 0.27	0.60 ± 0.27	0.53 ± 0.24	0.53 ± 0.24

Table 4.2: Monthly climate dataset MSE

	Kernel bandwidth	Full	0/4-1/4	1/4-2/4	2/4-3/4	3/4-4/4
Train data	0.005	0.10 ± 0.02	0.11 ± 0.04	0.11 ± 0.04	0.07 ± 0.02	0.08 ± 0.03
	0.05	0.10 ± 0.02	0.11 ± 0.04	0.11 ± 0.04	0.07 ± 0.02	0.08 ± 0.02
	5	0.18 ± 0.04	0.16 ± 0.06	0.15 ± 0.07	0.14 ± 0.08	0.14 ± 0.07
	50	0.58 ± 0.14	0.19 ± 0.08	0.24 ± 0.14	0.22 ± 0.15	0.20 ± 0.11
	500	0.86 ± 0.31	0.17 ± 0.07	0.30 ± 0.2	0.25 ± 0.17	0.21 ± 0.11
	50000	0.90 ± 0.34	0.17 ± 0.07	0.31 ± 0.21	0.26 ± 0.17	0.21 ± 0.11
Test data	0.005	0.11 ± 0.03	0.13 ± 0.05	0.12 ± 0.05	0.08 ± 0.03	0.08 ± 0.03
	0.05	0.11 ± 0.03	0.13 ± 0.05	0.12 ± 0.05	0.08 ± 0.03	0.09 ± 0.03
	5	0.18 ± 0.06	0.17 ± 0.08	0.16 ± 0.09	0.16 ± 0.09	0.15 ± 0.07
	50	0.55 ± 0.17	0.21 ± 0.1	0.28 ± 0.22	0.24 ± 0.17	0.21 ± 0.12
	500	0.88 ± 0.49	0.18 ± 0.09	0.34 ± 0.27	0.27 ± 0.2	0.23 ± 0.15
	50000	0.91 ± 0.52	0.18 ± 0.09	0.35 ± 0.28	0.28 ± 0.2	0.23 ± 0.15

both the motor and March daily climate datasets, whereas the network trained with the monthly climate dataset performs better with lower kernel bandwidths. This validates hypothesis 4.

The network trained with the monthly climate dataset has, with a good kernel bandwidth, a lower MSE than the other two networks.

In all the datasets, the training error is relatively similar to the test error, which indicates that the network generalises quite well and avoids overfitting. This is due to the fact that our tool for performing linear regression to learn the networks (glmnet) performs cross-validation during the training to find the best regularization parameter and best coefficients.

In Tables 4.4, 4.5 and 4.6 we can visualise the values of the variables of study.

Contrary to what was formulated in Hypothesis 5, the learning time actually tends to decrease with larger kernel bandwidths (with the exception of extremely low values). This decrease is less apparent in the case of the monthly dataset.

As can be seen in those tables, the number of arcs tends to increase with the kernel bandwidth. This might be because we might be considering more overall training instances (the number of instances with weight close to 0 will be lower) and, and as such, more relationships across time instants are found. As expected in hypothesis 3, the Hamming distance decreases, since we are giving a more uniform weight to the instances and thus each adjacent transition network will be more similar. Nonetheless, the monthly climate network

Table 4.3: March daily climate dataset MSE

	Kernel bandwidth	Full	0/4-1/4	1/4-2/4	2/4-3/4	3/4-4/4
Train data	0.02	0.89 ± 0.19	0.75 ± 0.21	0.83 ± 0.28	0.80 ± 0.24	0.88 ± 0.27
	2	0.87 ± 0.18	0.64 ± 0.18	0.70 ± 0.22	0.68 ± 0.18	0.72 ± 0.21
	20	0.89 ± 0.19	0.67 ± 0.2	0.70 ± 0.2	0.72 ± 0.21	0.76 ± 0.22
	200	0.91 ± 0.18	0.69 ± 0.2	0.72 ± 0.18	0.73 ± 0.2	0.78 ± 0.23
	20000	0.92 ± 0.19	0.70 ± 0.21	0.73 ± 0.18	0.74 ± 0.2	0.79 ± 0.23
Test data	0.02	1.01 ± 0.33	0.88 ± 0.32	1 ± 0.43	0.92 ± 0.37	1.03 ± 0.41
	2	0.98 ± 0.32	0.78 ± 0.29	0.85 ± 0.38	0.79 ± 0.29	0.89 ± 0.34
	20	0.95 ± 0.31	0.75 ± 0.3	0.79 ± 0.33	0.76 ± 0.27	0.83 ± 0.31
	200	0.94 ± 0.3	0.71 ± 0.28	0.76 ± 0.27	0.74 ± 0.26	0.80 ± 0.3
	20000	0.94 ± 0.3	0.71 ± 0.27	0.76 ± 0.27	0.74 ± 0.26	0.80 ± 0.3

Table 4.4: Measures (columns) for the motor dataset given a set of kernel bandwidths (rows)

Kernel bandwidth	Time	Arcs	Hamming	Avg. pers. arcs (%)	Total pers. arcs (%)
0.5	4.65 ± 0.06 min.	30.78 ± 6.08	12.83 ± 6.95	47.09 ± 8.81	9.72 ± 0.49
50	4.14 ± 0.09 min.	35.11 ± 5.5	2.21 ± 1.9	58.70 ± 9.12	14.03 ± 1.34
500	3.88 ± 0.06 min.	36.52 ± 5.11	0.78 ± 0.99	66.41 ± 9.15	16.53 ± 0.58
5000	3.78 ± 0.07 min.	38.40 ± 5.47	0.33 ± 0.66	76.95 ± 10.31	20.14 ± 0.49
5e+05	3.67 ± 0.06 min.	40.45 ± 3.07	0.06 ± 0.3	98.40 ± 1.96	27.64 ± 2.38

presents some spurious behaviour regarding the Hamming distance, as for low and high values of kernel bandwidth, the Hamming distance is low, unlike in the other datasets where there is a clear correlation between greater kernel bandwidth and lower Hamming distance. Parallel to this, we can observe that the average number of persistent arcs also increases with the kernel bandwidth, for the same reason as before (and without the spurious behaviour for the monthly climate network).

In the extreme situation of having a very big kernel bandwidth (that is, to weight the instances using a Gaussian with a large standard deviation), learning a TV-DBN almost results in learning a stationary DBN, as in most pairs of adjacent transition networks the set of arcs remains the same. In fact, at least 98% of the network structure remains from the beginning until the end. Another interesting phenomenon is the fact that, although the network learned with a greater kernel bandwidth is more stationary it also has, on average, a greater number of arcs.

4.3.2. Spatio-temporal TV-DBNs

We will proceed as in the previous section. First, we will present the accuracy results in terms of the MSE and then we will comment the metrics obtained. The spatial penalty column represents the standard deviations of the bivariate Gaussian that determine the penalty for each feature beased on the distance. A standard deviation of “Infinite” (Inf) means that the penalization is uniform (which is the default case presented in the previous sections) and a low value for the spatial penalty deviation means that the spatial penalty is actually very high and pronounced.

We decided to use kernel bandwidths 50 and 250 for the monthly climate dataset and 2, 20 and 200 for the daily climate dataset. Those are not necessarily the values that yield better performance, but they will allow to see how the spatial penalty can improve (or worsen) the

Table 4.5: Monthly climate dataset measures

Kernel bandwidth	Time	Arcs	Hamming	Avg. pers. arcs (%)	Total pers. arcs (%)
0.005	14.54 ± 0.63 sec.	16.05 ± 12.07	30.38 ± 16.07	0 ± 0	0 ± 0
0.05	15.87 ± 0.63 sec.	15.18 ± 12.38	27.71 ± 16.6	0 ± 0	0 ± 0
5	1.14 ± 0.06 min.	294.68 ± 79.72	189.33 ± 45.25	2.71 ± 2.12	0.34 ± 0.09
50	49.74 ± 3.6 sec.	466.67 ± 75.73	61.44 ± 17.21	47.31 ± 6.69	10.71 ± 1.17
500	49.15 ± 3.94 sec.	525 ± 79.36	11.09 ± 4.66	91.8 ± 1.88	23.77 ± 3.78
50000	49.43 ± 4.31 sec.	531.85 ± 83.97	0.36 ± 0.93	99.59 ± 0.59	26.17 ± 4.69

Table 4.6: March daily climate dataset measures

Kernel bandwidth	Time	Arcs	Hamming	Avg. pers. arcs (%)	Total pers. arcs (%)
0.02	45.16 ± 1.04 sec.	98.71 ± 23.57	138.42 ± 23.33	0 ± 0	0 ± 0
2	2.22 ± 0.05 min.	236.14 ± 44.08	182.23 ± 42.08	18.45 ± 3.51	2.08 ± 0.19
20	2.02 ± 0.05 min.	235.04 ± 27.46	56.55 ± 15.44	36.39 ± 4.38	4.17 ± 0.14
200	1.63 ± 0.05 min.	257.81 ± 18.91	14.74 ± 4.84	57.86 ± 4.83	7.33 ± 0.37
20000	1.57 ± 0.05 min.	272.71 ± 22.33	0.36 ± 0.68	98.35 ± 0.74	13.24 ± 1.2

network performance. In Tables 4.7 and 4.8 the MSE of the networks monthly climate and March daily climate respectively, is presented.

Table 4.7: Monthly climate dataset MSE (spatial network)

	Kernel bandwidth	Spatial penalty deviation	Full	0/4-1/4	1/4-2/4	2/4-3/4	3/4-4/4
Train data	50	5	0.61 ± 0.32	0.21 ± 0.08	0.28 ± 0.21	0.27 ± 0.17	0.23 ± 0.12
		10	0.53 ± 0.15	0.19 ± 0.07	0.23 ± 0.18	0.22 ± 0.17	0.20 ± 0.10
		20	0.49 ± 0.14	0.19 ± 0.08	0.24 ± 0.16	0.22 ± 0.17	0.18 ± 0.10
		50	0.51 ± 0.15	0.20 ± 0.08	0.23 ± 0.14	0.22 ± 0.16	0.18 ± 0.10
		Inf	0.52 ± 0.16	0.20 ± 0.09	0.24 ± 0.14	0.22 ± 0.15	0.18 ± 0.10
	250	5	0.64 ± 0.31	0.19 ± 0.07	0.34 ± 0.29	0.30 ± 0.19	0.23 ± 0.11
		10	0.61 ± 0.19	0.17 ± 0.06	0.27 ± 0.21	0.25 ± 0.19	0.20 ± 0.10
		20	0.59 ± 0.20	0.17 ± 0.06	0.28 ± 0.22	0.25 ± 0.18	0.19 ± 0.10
		50	0.62 ± 0.21	0.17 ± 0.07	0.28 ± 0.20	0.25 ± 0.17	0.19 ± 0.10
		Inf	0.61 ± 0.19	0.17 ± 0.07	0.29 ± 0.19	0.25 ± 0.16	0.19 ± 0.10
Test data	50	5	0.54 ± 0.18	0.23 ± 0.11	0.31 ± 0.28	0.28 ± 0.20	0.24 ± 0.13
		10	0.57 ± 0.19	0.22 ± 0.10	0.27 ± 0.24	0.24 ± 0.21	0.20 ± 0.10
		20	0.50 ± 0.18	0.22 ± 0.11	0.27 ± 0.24	0.23 ± 0.20	0.19 ± 0.10
		50	0.50 ± 0.19	0.22 ± 0.11	0.26 ± 0.20	0.24 ± 0.19	0.18 ± 0.09
		Inf	0.50 ± 0.22	0.22 ± 0.11	0.26 ± 0.18	0.23 ± 0.18	0.18 ± 0.09
	250	5	0.61 ± 0.25	0.21 ± 0.09	0.36 ± 0.30	0.31 ± 0.23	0.22 ± 0.11
		10	0.62 ± 0.23	0.20 ± 0.09	0.31 ± 0.30	0.27 ± 0.23	0.20 ± 0.10
		20	0.58 ± 0.25	0.19 ± 0.08	0.32 ± 0.29	0.26 ± 0.22	0.19 ± 0.10
		50	0.62 ± 0.26	0.19 ± 0.09	0.32 ± 0.27	0.26 ± 0.21	0.18 ± 0.09
		Inf	0.62 ± 0.28	0.20 ± 0.09	0.31 ± 0.24	0.26 ± 0.20	0.19 ± 0.09

It can be visualised that the MSE barely changes from the uniform penalization to the spatial penalization, although some medium values for the spatial penalty (around 20 for this dataset) actually show a small percentage boost, partially confirming Hypothesis 6. In addition, we notice how the MSE is, in some cases, even lower for the test data than for the biased train data evaluation. The model is able to generalise more if we force it to attend to the closer variables. In turn, we have a model that is more consistent with the physical relations of the world (closer Earth regions are more likely to influence each other) and that maintains or slightly improve the accuracy.

In Tables 4.9 and 4.10 we can visualise the value of our interest measures.

Table 4.8: March daily climate dataset MSE (spatial network)

	Kernel bandwidth	Spatial penalty deviation	Full	0/4-1/4	1/4-2/4	2/4-3/4	3/4-4/4
Train data	2	1	0.87 ± 0.19	0.64 ± 0.19	0.69 ± 0.21	0.69 ± 0.19	0.74 ± 0.21
		5	0.88 ± 0.19	0.65 ± 0.19	0.70 ± 0.21	0.69 ± 0.19	0.72 ± 0.21
		10	0.88 ± 0.19	0.65 ± 0.19	0.70 ± 0.21	0.69 ± 0.19	0.71 ± 0.21
		20	0.88 ± 0.19	0.65 ± 0.19	0.69 ± 0.22	0.69 ± 0.19	0.71 ± 0.21
		Inf	0.87 ± 0.18	0.64 ± 0.18	0.70 ± 0.22	0.68 ± 0.18	0.72 ± 0.21
	20	1	0.89 ± 0.19	0.66 ± 0.20	0.71 ± 0.19	0.72 ± 0.21	0.77 ± 0.22
		5	0.89 ± 0.19	0.67 ± 0.21	0.71 ± 0.20	0.72 ± 0.21	0.77 ± 0.23
		10	0.90 ± 0.19	0.67 ± 0.21	0.72 ± 0.20	0.72 ± 0.21	0.76 ± 0.23
		20	0.90 ± 0.19	0.67 ± 0.21	0.71 ± 0.20	0.72 ± 0.20	0.76 ± 0.23
		Inf	0.89 ± 0.19	0.67 ± 0.20	0.70 ± 0.20	0.72 ± 0.21	0.76 ± 0.22
	200	1	0.91 ± 0.19	0.69 ± 0.21	0.72 ± 0.18	0.74 ± 0.21	0.79 ± 0.24
		5	0.91 ± 0.19	0.69 ± 0.21	0.72 ± 0.18	0.74 ± 0.21	0.79 ± 0.24
		10	0.91 ± 0.19	0.69 ± 0.21	0.73 ± 0.18	0.74 ± 0.21	0.78 ± 0.24
		20	0.92 ± 0.19	0.69 ± 0.21	0.73 ± 0.18	0.74 ± 0.21	0.78 ± 0.23
		Inf	0.91 ± 0.18	0.69 ± 0.20	0.72 ± 0.18	0.73 ± 0.20	0.78 ± 0.23
Test data	2	1	0.96 ± 0.31	0.77 ± 0.29	0.81 ± 0.33	0.80 ± 0.30	0.87 ± 0.32
		5	0.97 ± 0.31	0.78 ± 0.30	0.82 ± 0.33	0.80 ± 0.30	0.87 ± 0.33
		10	0.97 ± 0.31	0.79 ± 0.31	0.83 ± 0.34	0.80 ± 0.30	0.87 ± 0.33
		20	0.97 ± 0.31	0.79 ± 0.31	0.83 ± 0.34	0.81 ± 0.31	0.87 ± 0.33
		Inf	0.98 ± 0.32	0.78 ± 0.29	0.85 ± 0.38	0.79 ± 0.29	0.89 ± 0.34
	20	1	0.94 ± 0.30	0.74 ± 0.30	0.77 ± 0.29	0.76 ± 0.28	0.83 ± 0.31
		5	0.95 ± 0.30	0.75 ± 0.30	0.78 ± 0.30	0.77 ± 0.29	0.83 ± 0.32
		10	0.95 ± 0.30	0.75 ± 0.31	0.78 ± 0.31	0.77 ± 0.29	0.83 ± 0.32
		20	0.95 ± 0.30	0.76 ± 0.31	0.79 ± 0.31	0.77 ± 0.29	0.83 ± 0.31
		Inf	0.95 ± 0.31	0.75 ± 0.30	0.79 ± 0.33	0.76 ± 0.27	0.83 ± 0.31
	200	1	0.94 ± 0.29	0.72 ± 0.28	0.75 ± 0.26	0.75 ± 0.28	0.80 ± 0.30
		5	0.94 ± 0.29	0.72 ± 0.29	0.75 ± 0.26	0.75 ± 0.28	0.80 ± 0.30
		10	0.94 ± 0.29	0.72 ± 0.29	0.75 ± 0.26	0.75 ± 0.28	0.80 ± 0.30
		20	0.94 ± 0.29	0.72 ± 0.28	0.75 ± 0.26	0.75 ± 0.27	0.81 ± 0.30
		Inf	0.94 ± 0.30	0.71 ± 0.28	0.76 ± 0.27	0.74 ± 0.26	0.80 ± 0.30

We can see how a more pronounced spatial penalty (lower values) results in a consistently shorter learning time for both networks, learning in same cases twice as fast as the case with no spatial penalty (Inf).

The number of arcs shows a different behaviour in each network. In the monthly climate network, a more pronounced penalty usually results in more arcs (except when the penalty becomes too pronounced), whereas in the March daily climate network the number of arcs actually decreases with a higher spatial penalty (lower variance value).

The same phenomenon occurs with the Hamming distance and the percentages of persistent arcs. For the monthly network, a higher spatial penalty (low spatial penalty deviation) results in a bigger Hamming distance (even considering the increase in the number of arcs) and lower proportion of persistent arcs, whereas the opposite occurs in the March daily network. This confirms Hypothesis 7 when the changes in the system are smooth enough.

Table 4.9: Monthly climate dataset measures (spatial network)

Kernel bandwidth	Spatial penalty deviation	Time	Arcs	Hamming	Avg. pers. arcs (%)	Total pers. arcs (%)
50	5	40.38 ± 1.19 sec.	573.87 ± 68.12	77.22 ± 19.55	45.82 ± 5.03	12.84 ± 0.82
	10	46.09 ± 1.85 sec.	630 ± 83.33	90.64 ± 20.65	43.65 ± 5.45	13.39 ± 1.07
	20	50.01 ± 1.46 sec.	548.40 ± 61.89	71.89 ± 15.60	44.74 ± 5.41	11.98 ± 0.89
	50	52.83 ± 1.62 sec.	491.57 ± 59.34	63 ± 13.17	46.79 ± 6	11.23 ± 1.17
	Inf	54.28 ± 1.70 sec.	460.37 ± 55.66	56.16 ± 12.55	50.59 ± 6.69	11.35 ± 0.93
250	5	37.74 ± 1.48 sec.	621.35 ± 51.22	21.78 ± 5.31	85.16 ± 1.39	26.13 ± 2.45
	10	43.01 ± 2.29 sec.	687.72 ± 53	24.87 ± 6.69	84.81 ± 2.19	28.80 ± 2.59
	20	47.97 ± 1.53 sec.	589.97 ± 36.18	19.96 ± 5.03	86.78 ± 2.09	25.28 ± 1.88
	50	50.75 ± 1.52 sec.	540.33 ± 36.01	18.75 ± 4.63	85.72 ± 2.36	22.87 ± 1.87
	Inf	51.54 ± 2.03 sec.	503.95 ± 37.22	17.05 ± 4.99	86.03 ± 2.68	21.40 ± 1.81

Table 4.10: March daily climate dataset measures (spatial network)

Kernel bandwidth	Spatial penalty deviation	Time	Arcs	Hamming	Avg. pers. arcs (%)	Total pers. arcs (%)
2	1	55.62 ± 0.34 sec.	135.99 ± 23.43	75.42 ± 21.86	33.79 ± 6.33	2.20 ± 0.17
	5	1.58 ± 0.02 min.	180.30 ± 38.52	119.94 ± 39.24	26.09 ± 5.94	2.22 ± 0.22
	10	1.83 ± 0.07 min.	197.81 ± 40.35	139.66 ± 40.79	22.36 ± 4.92	2.09 ± 0.13
	20	2 ± 0.05 min.	212.37 ± 41.41	154.62 ± 38.47	21.36 ± 4.69	2.15 ± 0.18
	Inf	2.22 ± 0.05 min.	236.14 ± 44.08	182.23 ± 42.08	18.45 ± 3.51	2.08 ± 0.19
20	1	53.19 ± 3.1 sec.	149.95 ± 19.93	25.96 ± 11.26	54.65 ± 7.26	3.98 ± 0.18
	5	1.35 ± 0.03 min.	180.39 ± 28.02	35.56 ± 14.47	49.17 ± 7.75	4.28 ± 0.13
	10	1.69 ± 0.08 min.	192.57 ± 28.92	40.06 ± 14.73	45.13 ± 6.98	4.20 ± 0.20
	20	1.81 ± 0.17 min.	203.63 ± 32.11	43.77 ± 14.31	42.85 ± 6.69	4.21 ± 0.17
	Inf	2.02 ± 0.05 min.	235.04 ± 27.46	56.55 ± 15.44	36.39 ± 4.38	4.17 ± 0.14
200	1	57.39 ± 1.67 sec.	175.40 ± 10.84	7.73 ± 4.29	69.96 ± 3.80	6.04 ± 0.27
	5	1.30 ± 0.03 min.	201.23 ± 14.12	9.97 ± 4.56	66.78 ± 4.83	6.61 ± 0.26
	10	1.47 ± 0.06 min.	207.52 ± 13.78	10.44 ± 4.82	65.80 ± 4.57	6.72 ± 0.24
	20	1.65 ± 0.04 min.	217.20 ± 15.88	11.08 ± 4.43	63.56 ± 4.67	6.79 ± 0.25
	Inf	1.63 ± 0.05 min.	257.81 ± 18.91	14.74 ± 4.84	57.86 ± 4.83	7.33 ± 0.37

4.3.3. Practical demonstration

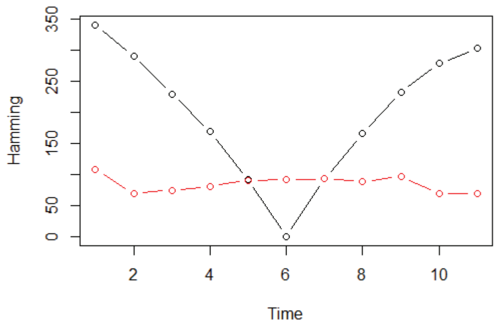
In order to prove the usefulness of our proposal, we will briefly show how it can be used to learn and understand time series.

First, we learn the networks. A wide variety of parameters can be chosen, but to keep it simple, we will go with the default options and we will select the kernel bandwidth and spatial penalty that results in an overall good network:

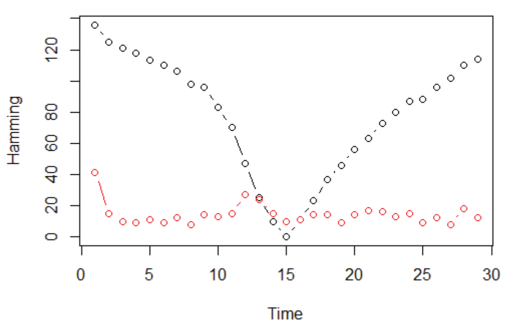
```r
tvdbn_monthly = learn_tvdbn(data_climate_monthly, kernel_bandwidth = 50, spatial_penalty = 50)
tvdbn_daily = learn_tvdbn(data_climate_days, kernel_bandwidth = 200, spatial_penalty = 10)
```

Once we learn the networks, we can study the Hamming changes between every pair of transition networks (red dots and lines) and between a given transition network and the rest (black dots and lines). The result of the following code can be visualised in Figure 4.1. The picture lets us see in the macro-level (where it has happened) the concept drift of the data and model (Hypothesis 1):

```r
plot(hamming_changes(tvdbn_monthly,time = 7)[-1], type = "b", xlab = "Time", ylab = "Hamming")
lines(hamming_changes(tvdbn_monthly), type = "b", col = "red")
###
plot(hamming_changes(tvdbn_daily,time = 16)[-1], type = "b", xlab = "Time", ylab = "
    Hamming")
lines(hamming_changes(tvdbn_daily), type = "b", col = "red")
```



(a) Hamming distances for the monthly climate network



(b) Hamming distances for the March daily climate network

Figure 4.1: Hamming distances per time. In red, the distance between two adjacent networks. In black, the distance between the 6th or 15th transition network and the rest. The "V" shaped pattern shows that the further we move from the the 6/15th instant, the more different the transition network will be

We have implemented a function to visualise networks as well. However, these networks are too big to plot and, in turn, the library allows for finding the global graph of the networks and also reducing the number of transition network by combining the ones that are adjacent. Let's plot the global graph of the daily network is plotted in Figure 4.2therefore presenting the persistent relation across the month of March. Note that the tool used (graphviz) allows for inspecting nodes and, indeed, we can see that the parents of a given node are spatially close to the children. This is especially noticeable in the daily network, as we selected a higher spatial penalty. Since those relations are in the global graph, that means that it is persistent throughout the whole network. If we decode the coordinates, we would get that central Europe (specifically Northern Germany) is influenced by itself and by some close Westward and Northern regions (the Channel and the Northern Sea) throughout the whole month.

In addition, the nodes are plotted respecting the real spatial location of the coordinates that they represent. in the future, we may add an underlying map to better visualise which node represents which region:

```
plot_spatial_tvdbn(global_graph(tvdbn_daily))
graph_weekly = summary_network(tvdbn_daily, frequency = 9)
```

Although we have studied the differences at the macro level (Figure 4.1), we can also study them at the micro level by means of the Hamming graph. Let's find the Hamming graph of the first and last transition network of the weekly graph previously obtained (Figure 4.3). The results show that further Earth regions only affect temporarily our selected nodes (we expect closer regions interaction to persist in time, whereas we expect the further ones to vanish). Specifically, the temperature in Caspian sea regions is at times influenced by the temperature in Eastern Europe, but not always.

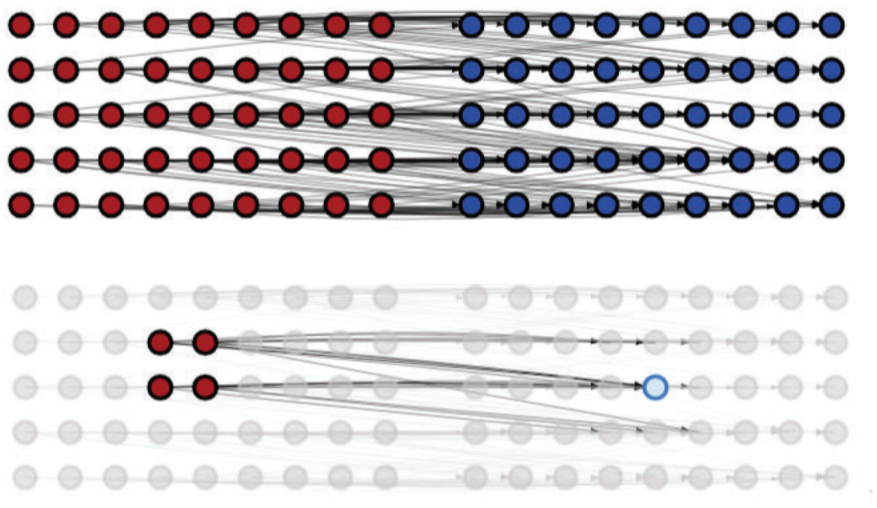


Figure 4.2: Global graph of the March daily climate network

```
plot_spatial_tvdbn(hamming_graph_2g(transition_network_graph(graph_weekly , 1),transition
    _network_graph(graph_weekly , 3)))
```

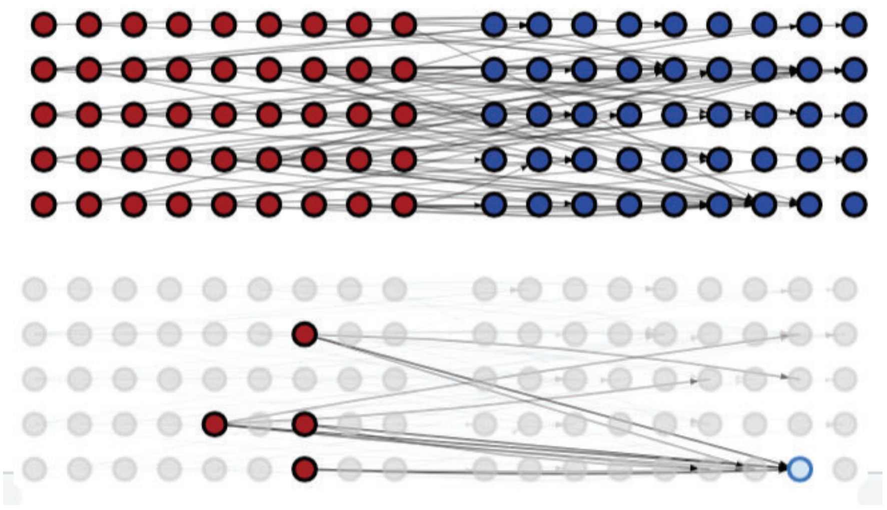


Figure 4.3: Hamming graph of the first and last transition networks of the weekly network

Being able to study how the relations hold and change in the micro-level and verifying that the observations are relatively consistent, we can confirm Hypothesis 1.

Finally, we can focus on certain variables and study their evolution just looking at them. Let's compress the monthly network in 4 time instants (4 seasons) and focus on Spain and France. According to our model (Figure 4.4), Spain only influences directly the temperature of France during winter.

```
season_graph = summary_network(tvdbn_monthly , 4)
plot_spatial_tvdbn(trim_network(season_graph , node_set = c("lat42.5N_lon2.5W","lat52.5N_
    lon7.5E")))
```

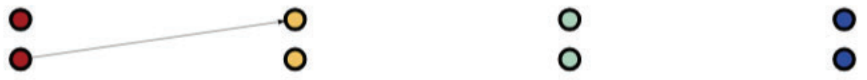


Figure 4.4: Trimmed spatial network, where the upper node represent the temperature of France and the lower node, the temperature of Spain. As can be seen, the temperature of Spain in winter will directly impact the temperature of France in spring

In addition, we can focus on a zone (or set of zones) and see how its influences change over time. Let's focus on Spain again, this time with the weekly graph (Figure 4.5):

```
plot_spatial_tvdbn(trim_network(graph_weekly, node_set = c("lat42.5N_lon2.5W"), expanded
     = TRUE))
```

In the resulting figure, we can (roughly) see the earth regions that impact the temperature of Spain.

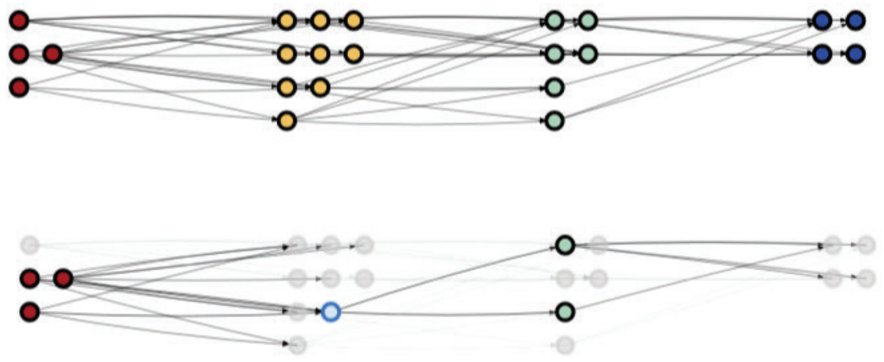


Figure 4.5: Trimmed spatial network with influences. The node of interest is Spain

4.3.4. Discussion

In our experiments, we have exposed an explainability trade-off present in the TV-DBNs: Ideally, we would want a low number of (meaningful) arcs and a network in which the changes occurs relatively smoothly, i.e. low Hamming distance and big percentage of (total) persistent arcs. However, a small kernel bandwidth yields a network with fewer arcs but with a great variance between the transition networks and, in contrast, a big kernel bandwidth results in networks with low variance in the structure but with a higher number of arcs. A feasible option is to select the parameter that yields a lower MSE, thus focusing as well on the predictive accuracy of the model.

The fact that the error tends to be lower in the climate datasets might be due to the fact that we are making inference with a much shorter horizon. The motor dataset has 174 time points, whereas the monthly climate has 13 (12 months + December of the previous year) and the daily climate has 31 instants.

The experiments also show that, in case the relations between time instants are very smooth or barely change with time (the case where we model with a large kernel bandwidth), it might be preferable to use a stationary DBN (or a ns-DBN with very few different structures) from both an explainability and performance standpoint.

Regarding the spatio-temporal networks, the results are dramatically different depending on the traits of the event observe. We believe that all comes down to the frequency with which the observations are made, which in turn define the smoothness of the changes across time. If the changes are smooth enough, as in the March daily climate network, the spatial-temporal penalty overall improves the system, as it offers a slightly boost in accuracy, faster learning, less arcs per transition network (simplicity) and a smaller Hamming distance, which means that the changes are smooth.

However, in time series with abrupt changes, the minuscule performance boost and in learning is overshadowed by a noticeable increase in the number of arcs and in a higher Hamming distance, which deems the model more complex and more difficult to interpret.

In short, the experiments showed Hypotheses 4 and 3 confirmed with great confidence and the Hypothesis 6 accepted, although we believe that more experimentation is needed to accept it with confidence. In contrast, Hypotheses 5 and 7 were refuted, since:

- For Hypothesis 5, the learning was actually faster with a higher kernel bandwidth. This might be because the tool used for optimisation in the linear regression problem is able to converge faster using more instances, as there is more information that allows for a faster convergence.
- The Hypothesis 7 regarding the Hamming change in spatio-temporal networks actually holds in the scenarios where the changes are smooth, but not in every scenario, as the monthly climate network proves.

Confirming Hypotheses 1 and 2 might be a stretch considering the lack of an actual climate scientist that validates our model. However, our practical demonstration shows that our tool is fully functional and has potential to confirm the aforementioned hypotheses.

Chapter 5

Conclusions and future work

5.1. Conclusions

In this work, we have first offered a literature review in which we explore the existing state-of-the art widely. This gives a much needed simple overview of existing explainability techniques for both Bayesian networks and time series analysis.

In addition, we have expanded the taxonomy of Bayesian network explanations by studying new methods that were missing in previous reviews, adding a fifth category that can be considered a toolbox of interesting techniques to explain Bayesian networks, and we have placed these methods into model agnostic vs. specific and global vs. local explanations categories.

Techniques that were going to be used in our work have also been revised in depth. That is the case of example-based explanations and, more specifically, counterfactuals, although we later decided to focus on prototype generation for time series. The found gaps in the literature have been exposed as well, for instance, the lack of explanations that are specific for time series or the huge restriction of only being able to use MAP-independence in discrete domains.

After revising the literature, we have proposed three different methods for improving explainability in Bayesian networks.

First, exploring and discovering new properties of MAP-independence allow us to create more efficient algorithms to check the robustness of MAP explanations. In addition, expanding the proposal to continuous (and hybrid) domains, will enable to use this concept in many more scenarios.

Second, we have also shown how to create prototypes for time series using the generative capacity of dynamic Bayesian network. In our proposal, we first perform forecasting and then smoothing to obtain the explanation. This proposal completely bypasses the optimization problem of finding prototypes or shapelets for time series.

Third, we have suggested various ways of globally explaining TV-DBNs, specifically how the structure and parameters change over time. This is done through the notions of global and Hamming graphs, (which are meaningful and logical AND and XOR operations for graphs).

The changes in parameters are studied via f-divergences, specifically using the Jensen-Shannon divergence. In this work, we greatly optimise the computation of said divergence for the transition networks of TV-DBNs.

In addition to the theoretical proposal, we offer the first implementation of TV-DBNs that is subject to further expansion and public use, as it is designed with the philosophy of usability and open-source. The library is implemented also simulating the structure of object-oriented programming, resulting in great modularity. Many of our proposals are implemented in this library, although others (MAP-independence and studying changes in the structure of TV-DBNs) have been left out due to time constraints.

5.2. Future ways

Due to the fact that our work is quite broad, many future ways are open to us.

In the field of MAP-independence, we have all the pieces necessary to formulate an algorithm that checks efficiently for MAP-independences given an evidence and a set of hypothesis nodes. In addition, we could do a better research about MAP-independence in hybrid domains, as our proposal is a simple merge of the discrete and continuous MAP-independences.

Another proposal is related to MAP-independence strength as a measure of robustness. We could add the lack of robustness as a penalty when looking for most probable explanations in MPE and MRE problems, similarly to how the number of parameters can be added as a penalty in least square optimization using l1 or l2 penalties with a penalty parameter $\lambda > 0$. For instance, in MAP queries, the formalization changes as follows:

$$\boldsymbol{h}^* = arg\,max_{\boldsymbol{H}} P(\boldsymbol{H}|\boldsymbol{e}) - \lambda MI\_strength(\boldsymbol{R}, \boldsymbol{H}, \boldsymbol{e})$$

And in MRE problems:

$$\boldsymbol{h}^* = arg\,max_{\boldsymbol{H} \in \mathcal{P}^+(\boldsymbol{X_u})} BGF(\boldsymbol{H}; \boldsymbol{e}) - \lambda MI\_strength(\boldsymbol{R}, \boldsymbol{H}, \boldsymbol{e})$$

where $\mathcal{P}^+(\boldsymbol{X_u})$ is the powerset of the set of unobserved variables $\boldsymbol{X_u}$ excluding the empty set, $\mathcal{P}^+(\boldsymbol{X_u}) = \mathcal{P}(\boldsymbol{X_u}) - \{\emptyset\}$.

The challenge would be how to select the set $\boldsymbol{R}$ and studying the computational complexity of the proposal, as computing MAP-independence is already a co-NP$^{\text{PP}}$ problem (Kwisthout, 2021).

In addition to generating prototypes/shapelets, we are also interested in other types of example-based explanations. Initially, we were interested in counterfactuals in Bayesian networks, but the proposal was removed to favour methods that are specific for DBNs. We believe that, with a formulation similar to that proposed for prototypes, counterfactuals can be brought back for DBNs in the context of time series.

The TV-DBN that has been used to test the efficacy of our model explanations has a medium size and it could be interesting to study how it scales with massive networks with hundreds of nodes per time slice. In addition, a finer data mining work will result in cleaner data that is easier to understand by the end-user.

Regarding the library, we would like to consider other implementations for performing linear regression in addition to glmnet. This tool, though powerful, does not admit user-specified initialization, which is the whole point of the efficiency of TV-DBNs. Although we have programmed our software following every single guideline from the original authors (to the point of contacting them), a cross-validation of the implementation using benchmark datasets is desirable. The cross-validation have been left out of this work since we preferred to focus on the explainability part. A baseline comparison with other XAI methods would be desirable as well, although our proposal are quite novel and it is difficult to find existing results with which to compare.

If we focus in the potential of our proposal in the framework of climate sciences, our proposal could be expanded in multiple directions. For instance, we could connect our predictions about the climate with a social model and, instead of predicting atmospheric variables, predict the value of the land over time or the probability of wildfires.

In future works, we expect to favour other programming languages such as C or C++, as they have been recently proven to be very energetically efficient, to have a fast execution time and to consume lower amounts of internal memory and energy (Pereira et al., 2017). In our opinion, this factor becomes even more important if we are developing a project in the field of climate sciences.

5.3. Scientific dissemination

Part of the MAP-independences advances of this work were presented in the Minisymposia of the workshop “New Bridges between Mathematics and Data Science" (`http://nbmds.uva.es/`) in Valladolid (Spain), 10th November 2021, in the session of “Interpretability and explainability of algorithms". A proper paper with all of the theoretical proposal of this work, some new advances and the feedback provided has been presented in the first workshop on Heterodox Methods for Interpretable and Efficient Artificial Intelligence (`https://hmieai2022.cs.umu.se/`).

The advances concerning the explainability of the changes in ns-DBNs and TV-DBNs as well as the implementation of the library `tvdbn` have been sent for publication to the 11-th Internation Conference on Probabilistic Graphical Models. The paper is currently under peer review.

5.4. Personal reflection

In addition to the scholarly conclusions drawn above, I would like to remark how this project has affected me personally.

This is the biggest project that I have completed to date and it has been an outstanding amount of work. The biggest difficulty was the fact that I was working alone, even if I had weekly meetings with my supervisors. Detecting the gaps in the literature and deciding where to explore have been real challenges, but advancing was always satisfactory.

Similarly, another difficulty that arose while developing this work is that much of the literature did not tackle similar problems from the one that I wanted to solve and I had to rely on more tangential papers. However, this allowed me to better and faster process the information and to extract from each paper little pieces of knowledge that I finally managed to compile in my literature review.

This lack of previous work also applied to the software. What I expected before entering the project was to work with TV-DBN learning software already developed by the research group or by a third party, but instead I had to program my own TV-DBN library. In order to avoid cases like mine in this field of research, I designed the library with the spirit of open-source programming and collaboration. Not everything is a disadvantage, since the lack of software allowed me to have full control over the library from the beginning.

One of the major difficulties is the fact that this project needed to be halted for three months, as I received a scholarship to do a research visit in the Tokyo Institute of Technology (Summer Exchange Research Program). Although the visit needed to be delayed and it is still to be dated (due to the pandemic), I collaborated on my own with online meetings during three months with researchers in Japan, which meant that I had to stop working in this project. This allowed me to correctly halt the project (by not leaving objectives unfinished, waiting to open new fronts) and then re-start it smoothly. The sanitary situation also had an impact in this work. The big conclusion that I find in all of this is that, more often than not, the life will change your planning and not the other way around.

To finish, this project, with all the difficulties and adversities in the road, has allowed me to move from having a good but limited knowledge about Bayesian networks to today being able to call myself an "expert" in this field. At the end, all the effort put into this piece of work has proven to be worthy.

Bibliography

Adadi, A. and Berrada, M. (2018). Peeking inside the black-box: A survey on explainable artificial intelligence (XAI). *IEEE Access*, 6:52138–52160.

Albini, E., Rago, A., Baroni, P., and Toni, F. (2020). Relation-based counterfactual explanations for Bayesian network classifiers. In *IJCAI*, pages 451–457.

Albini, E., Rago, A., Baroni, P., and Toni, F. (2021). Influence-driven explanations for Bayesian network classifiers. In *Pacific Rim International Conference on Artificial Intelligence*, pages 88–100. Springer.

Ates, E., Aksar, B., Leung, V. J., and Coskun, A. K. (2021). Counterfactual explanations for multivariate time series. In *2021 International Conference on Applied Artificial Intelligence (ICAPAI)*, pages 1–8. IEEE.

Barredo-Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R., Chatila, R., and Herrara, F. (2020). Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58:82–115.

Box, G. E., Jenkins, G. M., Reinsel, G. C., and Ljung, G. M. (2015). *Time Series Analysis: Forecasting and Control*. John Wiley & Sons.

Bromiley, P. (2003). Products and convolutions of Gaussian probability density functions. Technical report, School of Medicine, University of Manchester.

Broniatowski, D. A. et al. (2021). Psychological foundations of explainability and interpretability in artificial intelligence. Technical report, NIST Interagency/Internal Report (NISTIR), National Institute of Standards. Available at: `https://nvlpubs.nist.gov/nistpubs/ir/2021/NIST.IR.8367.pdf` (Accessed December 12, 2021).

Bueso, D., Piles, M., and Camps-Valls, G. (2020). Nonlinear PCA for spatio-temporal analysis of Earth observation data. *IEEE Transactions on Geoscience and Remote Sensing*, 58(8):5752–5763.

Cano, R., Sordo, C., and Gutiérrez, J. M. (2004). Applications of Bayesian networks in meteorology. In *Advances in Bayesian Networks*, pages 309–328. Springer.

Casillas, J., Cordón, O., Herrera, F., and Magdalena, L. (2003). Interpretability improvements to find the balance interpretability-accuracy in fuzzy modeling: An overview. In *Accuracy*

Improvements in Linguistic Fuzzy Modeling, pages 3–22. Springer.

Chan, H. and Darwiche, A. (2002). Reasoning about Bayesian network classifiers. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, pages 107–115.

Chen, J., Dai, X., Yuan, Q., Lu, C., and Huang, H. (2020). Towards interpretable clinical diagnosis with Bayesian network ensembles stacked on entity-aware CNNs. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3143–3153.

Chu, V. W., Wong, R. K., Chen, F., Fong, S., and Hung, P. C. (2016). Self-regularized causal structure discovery for trajectory-based networks. *Journal of Computer and System Sciences*, 82(4):594–609.

Chu, V. W., Wong, R. K., Liu, W., and Chen, F. (2013). Causal time-varying dynamic Bayesian networks. Technical report, Univerisity of South Wales.

Chubarian, K. and Turán, G. (2020). Interpretability of Bayesian network classifiers: OBDD approximation and polynomial threshold functions. In *International Symposium on Artificial Intelligence and Mathematics 2020*.

Colace, F., De Santo, M., Vento, M., and Foggia, P. (2004). Bayesian network structural learning from data: An algorithms comparison. In *International Conference on Enterprise Information Systems*, volume 2, pages 527–530.

Cossalter, M., Mengshoel, O., and Selker, T. (2011). Visualizing and understanding large-scale Bayesian networks. In *Workshops at the Twenty-Fifth AAAI Conference on Artificial Intelligence*, pages 12–21.

De Jongh, M. and Druzdzel, M. J. (2009). A comparison of structural distance measures for causal Bayesian network models. *Recent Advances in Intelligent Information Systems, Challenging Problems of Science, Computer Science Series*, pages 443–456. Springer.

Dean, T. and Kanazawa, K. (1989). A model for reasoning about persistence and causation. *Computational intelligence*, 5(2):142–150.

Delaney, E., Greene, D., and Keane, M. T. (2021). Instance-based counterfactual explanations for time series classification. In *International Conference on Case-Based Reasoning*, pages 32–47. Springer.

Derks, I. P. and De Waal, A. (2021). A taxonomy of xxplainable Bayesian networks. In *Southern African Conference for Artificial Intelligence Research*, pages 220–235. Springer.

Dhurandhar, A., Chen, P.-Y., Luss, R., Tu, C.-C., Ting, P., Shanmugam, K., and Das, P. (2018a). Explanations based on the missing: Towards contrastive explanations with pertinent negatives. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 590–601. Association for Computing Machinery.

Dhurandhar, A., Shanmugam, K., Luss, R., and Olsen, P. A. (2018b). Improving simple models with confidence profiles. *Advances in Neural Information Processing Systems*, 32:10317–10327.

Díez, F. J. (1994). *Sistema Experto Bayesiano para Ecocardiografía.* PhD thesis, UNED. Universidad Nacional de Educación a Distancia.

Došilović, F. K., Brčić, M., and Hlupić, N. (2018). Explainable artificial intelligence: A survey. In *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 0210–0215. IEEE.

Edwards, L. and Veale, M. (2017). Slave to the algorithm: Why a right to an explanation is probably not the remedy you are looking for. *Duke Law & Technology Review*, 16:18.

Fang, Z., Wang, P., and Wang, W. (2018). Efficient learning interpretable shapelets for accurate time series classification. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 497–508.

Fitelson, B. (2007). Likelihoodism, Bayesianism, and relational confirmation. *Synthese*, 156(3):473–489.

Fountalis, I., Bracco, A., and Dovrolis, C. (2014). Spatio-temporal network analysis for studying climate patterns. *Climate Dynamics*, 42(3-4):879–899.

Friedman, J., Hastie, T., and Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22.

Fung, R. and Chang, K.-C. (1990). Weighing and integrating evidence for stochastic simulation in Bayesian networks. In *Proceedings of the fifth Annual Conference on Uncertainty in Artificial Intelligence, 1990*, pages 209–220.

Gee, A. H., Garcia-Olano, D., Ghosh, J., and Paydarfar, D. (2019). Explaining deep classification of time-series data with learned prototypes. In *CEUR Workshop Proceedings*, volume 2429, pages 15–22. NIH Public Access.

Ghahramani, Z. (1997). Learning dynamic Bayesian networks. In *International School on Neural Networks*, pages 168–197. Springer.

Gilpin, L. H., Bau, D., Yuan, B. Z., Bajwa, A., Specter, M., and Kagal, L. (2018). Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on Data Science and Advanced Analytics*, pages 80–89.

Granger, C. W. (1969). Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: Journal of the Econometric Society*, 37(3):424–438.

Granger, C. W. J. and Newbold, P. (1977). *Forecasting Economic Time Series.* New York Academic Press.

Grzegorczyk, M. and Husmeier, D. (2009). Non-stationary continuous dynamic Bayesian networks. In *Proceedings of the 22nd International Conference on Neural Information Processing Systems*, pages 682–690. Association for Computing Machinery.

Gunning, D., Stefik, M., Choi, J., Miller, T., Stumpf, S., and Yang, G.-Z. (2019). Xai—explainable artificial intelligence. *Science Robotics*, 4(37):eaay7120.

Hennessy, C., Diz, A. B., and Reiter, E. (2020). Explaining Bayesian networks in natural language: State of the art and challenges. In *2nd Workshop on Interactive Natural Language Technology for Explainable Artificial Intelligence*, pages 28–33. Association for Computational Linguistics.

Henrion, M. (1988). Propagating uncertainty in bayesian networks by probabilistic logic sampling. In *Machine intelligence and pattern recognition*, volume 5, pages 149–163. Elsevier.

Henrion, M. and Druzdzel, M. J. (1990). Qualitative propagation and scenario-based explanation of probabilistic reasoning. In *Proceedings of the 6th Conference on Uncertainty in Artificial Intelligence*, pages 17–32. Association for Computing Machinery.

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

Hou, L., Kwok, J., and Zurada, J. (2016). Efficient learning of time series shapelets. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1).

Hsieh, T.-Y., Wang, S., Sun, Y., and Honavar, V. (2021). Explainable multivariate time series classification: A deep neural network which learns to attend to important variables as well as time intervals. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pages 607–615. Association for Computing Machinery.

Inman, H. F. and Bradley Jr, E. L. (1989). The overlapping coefficient as a measure of agreement between probability distributions and point estimation of the overlap of two normal densities. *Communications in Statistics-theory and Methods*, 18(10):3851–3874.

Jia, Y. and Huan, J. (2010). Constructing non-stationary dynamic Bayesian Networks with a flexible lag choosing mechanism. *BMC Bioinformatics*, 11(6):1–13.

Kadaba, N. R., Irani, P. P., and Leboe, J. (2007). Visualizing causal semantics using animations. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1254–1261.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems [j]. *Journal of basic Engineering*, 82(1):35–45.

Kalnay, E., Kanamitsu, M., Kistler, R., Collins, W., Deaven, D., Gandin, L., Iredell, M., and Joseph, D. (1996). The NCEP/NCAR 40-year reanalysis project. *Bulletin of the American Meteorological Society*, 77(3):437–471.

Kanamitsu, M., Ebisuzaki, W., Woollen, J., Yang, S.-K., Hnilo, J., Fiorino, M., and Potter, G. (2002). NCEP–DOE AMIP-II reanalysis (r-2). *Bulletin of the American Meteorological Society*, 83(11):1631–1644.

Keppens, J. (2019). Explainable Bayesian network query results via natural language generation systems. In *Proceedings of the 17th International Conference on Artificial Intelligence and Law*, pages 42–51. Association for Computing Machinery.

Kolar, M. (2021). Time varying DBN implementation. Personal communication.

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press.

Koopman, T. and Renooij, S. (2021). Persuasive contrastive explanations for Bayesian networks. In *Proceedings of the 16th European Conference on Symbolic and Quantitative Approaches with Uncertainty*, volume 12897, pages 229–242. Springer.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25:1106–1114.

Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86.

Kwisthout, J. (2021). Explainable AI using MAP-independence. In *Proceedings of the 16th European Conference on Symbolic and Quantitative Approaches with Uncertainty*, pages 243–254. Springer.

Labaien, J., Zugasti, E., and De Carlos, X. (2020). Contrastive explanations for a deep learning model on time-series data. In *International Conference on Big Data Analytics and Knowledge Discovery*, pages 235–244. Springer.

Lacave, C. and Díez, F. J. (2002). A review of explanation methods for Bayesian networks. *The Knowledge Engineering Review*, 17(2):107–127.

Lacave, C., Luque, M., and Diez, F. J. (2007). Explanation of Bayesian networks and influence diagrams in Elvira. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(4):952–965.

Lapowsky, I. (2018). Google autocomplete still makes vile suggestions. *Wired*. Online: `https://www.wired.com/story/google-autocomplete-vile-suggestions/` [Visited: 30/07/2021].

Lapuschkin, S., Wäldchen, S., Binder, A., Montavon, G., Samek, W., and Müller, K.-R. (2019). Unmasking clever hans predictors and assessing what machines really learn. *Nature Communications*, 10(1):1–8.

Lauritzen, S. L. and Wermuth, N. (1989). Graphical models for associations between variables, some of which are qualitative and some quantitative. *The Annals of Statistics*, 17(1):31–57.

Lewis, D. (1973). Counterfactuals and comparative possibility. *Journal of Philosophical Logic*, 2(4):418–446.

Li, G., Choi, B. K. K., Xu, J., Bhowmick, S. S., Chun, K.-P., and Wong, G. L. (2020). Efficient shapelet discovery for time series classification. *IEEE Transactions on Knowledge and Data Engineering*, 34(3):1149–1163.

Li, O., Liu, H., Chen, C., and Rudin, C. (2018). Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. In *Proceedings of 32nd the AAAI Conference on Artificial Intelligence*, volume 32, pages 3530–3537. AAAI Press.

Lin, J. (1991). Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151.

Marriott, K., Moulder, P., Hope, L., and Twardy, C. (2005). Layout of Bayesian networks. In *Proceedings of the 28th Australasian Conference on Computer Science*, volume 38, pages 97–106. Association for Computer Machinery.

McGill, A. L. and Klein, J. G. (1993). Contrastive and counterfactual reasoning in causal judgment. *Journal of Personality and Social Psychology*, 64(6):897.

Meng, Q., Wang, Y., An, J., Wang, Z., Zhang, B., and Liu, L. (2019). Learning non-stationary dynamic Bayesian network structure from data stream. In *4th International Conference on Data Science in Cyberspace (DSC)*, pages 128–134. IEEE.

Michiels, M., Larrañaga, P., and Bielza, C. (2021). BayeSuites: An open web framework for massive Bayesian networks focused on neuroscience. *Neurocomputing*, 428:166–181.

Miller, T. (2019). Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38.

Molnar, C. (2020). *Interpretable Machine Learning.* Lulu. com.

Montavon, G., Samek, W., and Müller, K.-R. (2018). Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 73:1–15.

Moreira, C., Chou, Y.-L., Velmurugan, M., Ouyang, C., Sindhgatta, R., and Bruza, P. (2021). LINDA-BN: An interpretable probabilistic approach for demystifying black-box predictive models. *Decision Support Systems*, 150:113561.

Murphy, K. P. (2002). *Dynamic bayesian networks: representation, inference and learning.* University of California, Berkeley.

Nielsen, F. (2019). On the Jensen–Shannon symmetrization of distances relying on abstract means. *Entropy*, 21(5):485.

Nielsen, U. H., Pellet, J.-P., and Elisseeff, A. (2008). Explanation trees for causal Bayesian networks. In *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence*, pages 427–434. Association for Computer Machinery.

Nodelman, U., Shelton, C. R., and Koller, D. (2002). Continuous time bayesian networks. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pages 378–387.

Paniego-Blanco, S. (2019). Visualization and interptetation in large Bayesian networks. Master's thesis, Technical University of Madrid.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann.

Pearl, J. (2009). *Causality.* Cambridge University Press.

Pearl, J. and Paz, A. (1985). Graphoids: A graph-based logic for reasoning about relevance relations. Technical report, University of California (Los Angeles). Computer Science Department.

Pereira, R., Couto, M., Ribeiro, F., Rua, R., Cunha, J., Fernandes, J. P., and Saraiva, J. (2017). Energy efficiency across programming languages: How do energy, time, and memory relate? In *Proceedings of the 10th ACM SIGPLAN International Conference on Software Language Engineering*, pages 256–267. Association for Computer Machinery.

Pereira-Fariña, M. and Bugarín, A. (2019). Content determination for natural language descriptions of predictive Bayesian Networks. In *11th Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT 2019)*, pages 784–791. Atlantis Press.

Qin, Y., Song, D., Cheng, H., Cheng, W., Jiang, G., and Cottrell, G. W. (2017). A dual-stage attention-based recurrent neural network for time series prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 2627–2633.

R Core Team (2020). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Rakthanmanon, T. and Keogh, E. (2013). Fast shapelets: A scalable algorithm for discovering time series shapelets. In *Proceedings of the 2013 SIAM International Conference on Data Mining*, pages 668–676. SIAM.

Ren, H., Wong, A. B., Lian, W., Cheng, W., Zhang, Y., He, J., Liu, Q., Yang, J., Zhang, C. J., Wu, K., and Zhang, H. (2021). Interpretable pneumonia detection by combining deep learning and explainable models with multisource data. *IEEE Access*, 9:95872–95883.

Rényi, A. (1961). On measures of entropy and information. In *Proceedings of the 4th Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*, pages 547–561. University of California Press.

Robinson, J. and Hartemink, A. (2008). Non-stationary dynamic Bayesian networks. *Advances in Neural Information Processing Systems*, 21:1369–1376.

Rothman, K. J. and Greenland, S. (2005). Causation and causal inference in epidemiology. *American Journal of Public Health*, 95(S1):S144–S150.

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215.

Runge, J. (2018). Causal network reconstruction from time series: From theoretical assumptions to practical estimation. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 28(7):075310.

Runge, J., Bathiany, S., Bollt, E., Camps-Valls, G., Coumou, D., Deyle, E., Glymour, C., Kretschmer, M., Mahecha, M. D., Muñoz-Marí, J., van Nes, E. H., Peters, J., Quax, R., Reichstein, M., Scheffer, M., Schölkopf, B., Spirtes, P., Sugihara, G., Sun, J., Zhang, K., and Zscheischler, J. (2019). Inferring causation from time series in Earth system sciences. *Nature Communications*, 10(1):1–13.

Runge, J., Petoukhov, V., Donges, J. F., Hlinka, J., Jajcay, N., Vejmelka, M., Hartman, D., Marwan, N., Paluš, M., and Kurths, J. (2015). Identifying causal gateways and mediators in complex spatio-temporal systems. *Nature Communications*, 6(1):1–10.

Runge, J., Petoukhov, V., and Kurths, J. (2014). Quantifying the strength and delay of climatic interactions: The ambiguities of cross correlation and a novel measure based on graphical models. *Journal of Climate*, 27(2):720–739.

Sanchez-Romero, R., Ramsey, J. D., Zhang, K., Glymour, M. R., Huang, B., and Glymour, C. (2019). Estimating feedforward and feedback effective connections from fMRI time series: Assessment of statistical methods. *Network Neuroscience*, 3(2):274–306.

Schlegel, U., Arnout, H., El-Assady, M., Oelke, D., and Keim, D. A. (2019). Towards a rigorous evaluation of XAI methods on time series. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 4197–4201. IEEE.

Schreiber, J. (2017). Pomegranate: Fast and flexible probabilistic modeling in Python. *The Journal of Machine Learning Research*, 18(1):5992–5997.

Scutari, M. (2010). Learning Bayesian networks with the bnlearn R package. *Journal of Statistical Software*, 35(3):1–22.

Semenova, L., Rudin, C., and Parr, R. (2019). A study in Rashomon curves and volumes: A new perspective on generalization and model simplicity in machine learning. *arXiv preprint arXiv:1908.01755*.

Shachter, R. D. and Peot, M. A. (1990). Simulation approaches to general probabilistic inference on belief networks. In *Proceedings of the Fifth Annual Conference on Uncertainty in Artificial Intelligence*, pages 221–234.

Sober, E. (2020). *Core Questions in Philosophy: A Text with Readings.* Routledge.

Song, L., Kolar, M., and Xing, E. P. (2009). Time-varying dynamic Bayesian networks. In *Proceedings of the 22nd International Conference on Neural Information Processing Systems*, pages 1732–1740.

Spirtes, P., Glymour, C. N., Scheines, R., and Heckerman, D. (2000). *Causation, Prediction, and Search.* The MIT press.

Sundarararajan, P. K., Mengshoel, O. J., and Selker, T. (2013). Multi-focus and multi-window techniques for interactive network exploration. In *Visualization and Data Analysis 2013*, volume 8654, pages 282–296. SPIE.

Thrun, M. C., Ultsch, A., and Breuer, L. (2021). Explainable AI framework for multivariate hydrochemical time series. *Machine Learning and Knowledge Extraction*, 3(1):170–205.

Timmer, S. T., Meyer, J.-J. C., Prakken, H., Renooij, S., and Verheij, B. (2017). A two-phase method for extracting explanatory arguments from Bayesian networks. *International Journal of Approximate Reasoning*, 80:475–494.

Tjoa, E. and Guan, C. (2020). A survey on explainable artificial intelligence (XAI): Toward medical XAI. *IEEE Transactions on Neural Networks and Learning Systems*, 32:4793–4813.

Valero-Leal, E., Juárez, J. M., and Campos, M. (2020). Subgroup discovery algorithms and model-agnostic explainability. Bachelor's thesis, University of Murcia.

Vejmelka, M., Pokorná, L., Hlinka, J., Hartman, D., Jajcay, N., and Paluš, M. (2015). Non-random correlation structures and dimensionality reduction in multivariate climate data. *Climate Dynamics*, 44(9):2663–2682.

Verma, T. and Pearl, J. (1990a). Causal networks: Semantics and expressiveness. In *Proceedings of the 4th Annual Conference on Uncertainty in Artificial Intelligence*, pages 69–78. Association for Computer Machinery.

Verma, T. and Pearl, J. (1990b). Equivalence and synthesis of causal models. In *Proceedings of the 6th Annual Conference on Uncertainty in Artificial Intelligence*, pages 255—270. Elsevier Science Inc.

Wang, Z., Kuruoğlu, E. E., Yang, X., Xu, Y., and Huang, T. S. (2010). Time varying dynamic Bayesian network for nonstationary events modeling and online inference. *IEEE Transactions on Signal Processing*, 59(4):1553–1568.

Whitaker, J. S., Hamill, T. M., Wei, X., Song, Y., and Toth, Z. (2008). Ensemble data assimilation with the NCEP global forecast system. *Monthly Weather Review*, 136(2):463–482.

Ye, L. and Keogh, E. (2009). Time series shapelets: A new primitive for data mining. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 947–956.

Yuan, C., Lim, H., and Lu, T.-C. (2011). Most relevant explanation in Bayesian networks. *Journal of Artificial Intelligence Research*, 42:309–352.

Zapata-Rivera, J.-D., Neufeld, E., and Greer, J. (1999). Visualization of Bayesian belief networks. In *IEEE Visualization 1999 Late Breaking Hot Topics Proceedings*, pages 85–88. IEEE.

Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *Proceedings of the 13th European Conference on Computer Vision*, number 1, pages 818–833. Springer.

Appendix A

Proof of Theorem 3.3.7

The geometric mean $m(x)$ is defined as the weighted product of $f(x) \sim \mathcal{N}(\mu_f, \sigma_f)$ and $g(x) \sim \mathcal{N}(\mu_g, \sigma_g)$ (where the weights are in the exponent and equals 0.5) divided by a scaling factor, as defined in Equation (2.18).

As such, we can decompose the computation of the geometric mean into two steps:

1. First, we find f' and g', the 0.5-th power of f and g, respectively. Applying Lemma 3.3.5, we get that $f'(x) \sim \mathcal{N}(\mu_f, \sqrt{2}\sigma_f)$ and $g'(x) \sim \mathcal{N}(\mu_g, \sqrt{2}\sigma_g)$.

2. Next, we multiply f' and g'. Applying Lemma 3.3.6:

$$\mu_m = \frac{\mu_f(\sqrt{2}\sigma_g)^2 + \mu_g(\sqrt{2}\sigma_f)^2}{(\sqrt{2}\sigma_f)^2 + (\sqrt{2}\sigma_g)^2} = \frac{2\mu_f\sigma_g^2 + 2\mu_g\sigma_f^2}{2\sigma_f^2 + 2\sigma_g^2} = \frac{\mu_f\sigma_g^2 + \mu_g\sigma_f^2}{\sigma_f^2 + \sigma_g^2}$$

$$\sigma_m = \sqrt{\frac{(\sqrt{2}\sigma_f)^2(\sqrt{2}\sigma_g)^2}{(\sqrt{2}\sigma_f)^2 + (\sqrt{2}\sigma_g)^2}} = \sqrt{\frac{4\sigma_f^2\sigma_g^2}{2(\sigma_f^2 + \sigma_g^2)}} = \sqrt{\frac{2\sigma_f^2\sigma_g^2}{\sigma_f^2 + \sigma_g^2}}$$