TÉCNICO LISBOA

Clustering multivariate time series using dynamic Bayesian networks

José Pedro de Almeida Gabriel Vieira Borges

Thesis to obtain the Master of Science Degree in

Eletrical and Computer Engeneering

Supervisor(s): Prof. Alexandra Sofia Martins de Carvalho
Prof. Susana de Almeida Mendes Vinga Martins

Examination Committee

Chairperson: Prof. Teresa Maria Sá Ferreira Vazão Vasques
Supervisor: Prof. Alexandra Sofia Martins de Carvalho
Member of the Committee: Prof. Helena Isabel Aidos Lopes Tomás

October 2020

Declaration: I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa

Acknowledgments

I wish to thank my supervisors, Alexandra Carvalho and Susana Vinga for their continuous support and for being so understanding.

I would also like to thank my parents and my sister very very much for everything they have done for me during the thesis, this absolutely would not have been possible without you and as such I am incredibly thankful.

To the giant list of close friends of course, highlighting Gonçalo and Joana for the continuous motivation.

And finally to all the members of my family who supported me with great patience during this period. With a special thanks to my cousins, my aunt and uncle and my grandparents.

Thank you all :), this thesis would not have been possible without those closest to me, and as such I want to express my deepest feelings of gratitude for everyone that is part of my life.

This work is a result of the Project PREDICT (PTDC/CCI-CIF/29877/2017), funded by Fundo Europeu de Desenvolvimento Regional (FEDER), through Programa Operacional Regional LISBOA (LISBOA2020), and by national funds, through Fundação para a Ciência e Tecnologia (FCT).

Resumo

As séries temporais multivariadas são extremamente utilizadas hoje em dia por serem uma forma conveniente de organizar e guardar grandes quantidades de informação. Nesta tese descrevemos o algoritmo CRATES, que lida especificamente com os problemas relacionados com fazer clusters de dados expressos em séries temporais multivariadas. Estes problemas são maioritariamente causados pela possível existência de valores não numéricos nas séries temporais, dificultando bastante o processo de clustering. Propomos uma alternativa a um método já existente que oferecia uma solução interessante ao usar Hidden Markov Models para modular cada série temporal, assim como a distância estatística de Kullback-Leibler para conseguir a matriz de distancias usada para fazer o clustering. No nosso caso, propomos a utilização de Redes de Bayes Dinâmicas em vez de Modelos de Markov e usamos várias outras distâncias estatísticas. Assim, melhoramos a qualidade dos clusters e removemos algumas imperfeições inerentes do algoritmo original. O CRATES foi primeiramente testado com dados sintéticos com o intuito de mostrar o seu potencial, seguido de testes em várias bases de dados reais, nos quais comparamos os resultados obtidos com outros do estado da arte usando indices de validação conhecidos para mostrar que o algoritmo proposto é competitivo, tendo um enorme potencial.

Palavras-chave: Clustering, Séries Temporais Multivariadas, Redes de Bayes Dinâmicas, Validação de Clusters

Abstract

Multivariate time series are extremely popular in today's society since they are a convenient way of organizing and storing big amounts of information. In this thesis, we describe CRATES, an algorithm that specifically addresses the problems related to clustering multivariate time series. These problems are mainly caused by the possible existence of categorical values in the time series, which makes clustering very tricky. There is a known method that offers a workaround by using Hidden Markov Models to model each Time Series as well as the Kullback-Leibler divergence to achieve the distance matrix necessary to perform partitional clustering. We propose an alternative that uses Dynamic Bayesian Networks instead, with an assortment of different statistical distances to improve cluster quality as well as overcome some obstacles for the original algorithm. We started by testing the devised method with synthetic data, showing that it is able to perform proper clusterings. Then we performed tests with several real-life datasets and compared the results with state-of-the-art methods using commonly used clustering validation indexes to prove it is a strong alternative to the few existing algorithms, showing tremendous potential.

Keywords: Clustering, Multivariate Time Series, Dynamic Bayesian Networks, Cluster Validation

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

We live in an era when access to information is easier than ever [1]. Every day there's a massive amount of new data made available [2] on all kinds of subjects such as medicine [3], economics [4] or even meteorology [5]. But a question then arises: if we have so much available data, why does it feel like we know so little?

A possible explanation is that we simply do not have the tools to process all the new information as it becomes available, making humanity fall short of its highest potential.

This gap between available information and analysis tools drew the attention of the scientific community to data mining processes as a possible solution to this obstacle. The scientific community is unison, and the investment that is being made in the area is astounding. This subject has been growing, fueled by the amount of people that invested in this blossoming area.

By using prior knowledge, we hope to achieve a deeper understanding of data mining techniques in order to be able to apply them to real life problems. Medicine is a good example of an area that could benefit greatly from the advancements made here. Thus, the main motivation behind this thesis is to be able to achieve a good clustering solution by using statistical models. Everyone is different and as such the best possible treatment for someone depends on their genome (and on the environment where it expresses itself), not only in terms of dosage but also in terms of which medicine to choose from. At the moment and in the foreseeable future, this type of precision medicine is unfeasible, but we can already try to profile people into well-defined clusters in order to maximize both the effectiveness and the efficiency of available treatments.

1.2 Objectives

This thesis aims at describing a new algorithm to successfully cluster multivariate time series. This is a challenging feat since these may contain not only numerical but also categorical values, which makes

defining a meaningful distance between each time series challenging. To avoid this problem an approach based on the one presented in [6] is suggested, as well as a direct comparison between different distances and clustering methods in order to achieve the best possible results. In short the algorithm works as follows:

- Finding a statistical model that approximates each time series.
- Find a suitable distance between each model.
- Cluster the several models according to the distances found between each generative model.

We believe this method shows promise in its way of dealing with the obstacles presented by multivariate time series based on the results presented in the paper. We pretend to add to this method experimenting with a different model as well as an assortment of different distances in order to study the effectiveness of these changes. Our hypothesis is that using dynamic Bayesian networks will not only yield similar results, if not better, but also have less constraints.

1.3 Claim of contributions

The main contributions of this thesis are:

- An implementation of a multivariate time series clustering algorithm is completely available at "https://github.com/ZeBorges/Crates".
- An in depth analysis of the developed algorithm on both simulated and real data, including some comparisons to state of the art methods.

1.4 Thesis Outline

This study is separated into several chapters. This first chapter contains the motivation as well as the main objectives of the thesis. The second chapter revolves around the theoretical background necessary to understand the method presented which is available on the third chapter, with a proper explanation of the algorithms and the train of thought behind them. The results and their interpretation is made on the fourth chapter, followed by the conclusions and future work.

Chapter 2

Background

2.1 Time Series

Time series (TS) are frequently used as a type of data to infer on, since they literally represent the evolution of a set of variables over time. A given time series Y can be represented formally as follows

$$Y = Y_1, Y_2, ..., Y_N. \tag{2.1}$$

Where N is an integer corresponding to the total number of time steps in the time series. They can be characterized as univariate or multivariate, depending on the number of features they comprise.

2.1.1 Univariate and Multivariate Time Series

Time series is a term far too vague to use when referring implementations that utilize them, so they should be separated into univariate time series (UTS) and multivariate time series (MTS), according to the number of features they contain and are measured over time. This separation is key, since there are separate methods that exist to process different types of time series.

Univariate time series are simpler in nature since they are evaluating a single feature, making them purely numeric, nominal or ordinal. They are quite common, and there is an extensive bibliography behind it, especially in forecasting analysis. Concerning clustering, an example is found in [7], in which an approach to cluster similar UTS is presented. Their strategy consists of grouping sub-trajectories instead of considering the whole sequence making a partition-and-group approach. Whilst there is certainly interest in utilizing UTS, they are not the main data that was planned to use, since when profiling something as complex as a disease for example, using MTS seemed more appropriate.

Multivariate time series have several factors which make it harder to process. Firstly, by comprising more features, methods with high computational complexity become unreliable due to time constraints. Secondly, unlike UTS, by having several features, problems with correlations between them arise, since they may depend on each other. This makes preprocessing MTS complex, specifically reducing data dimensionality, since not considering some features by assuming they are independent from the intended

ones may provide low quality results. Thirdly, the features don't have to be all of the same type, and they generally won't be. For a given patient, we have plenty of information that may be categorical (gender), or numeric (white cells count), all within the same object. This re-introduces the issues with clustering categorical variables as seen in [6] and is the main problem the proposed method tries to work around.

2.1.2 Time Series Analysis

Time Series are organized data points that allow some specific preprocessing techniques. For example, since a single TS object contains several entries, one may want to reduce their size to get a simplified approximation of the original time series. Standard methods such as sampling to more complex algorithms that preserve perceptually important points (PIP) [8], depending on the degree to which we want a good approximations of the data. Therefore, the choice of algorithm depends on the needs of the user.

By having a natural order associated, forecasting is a regular application in a wide range of scientific areas, from fields of Biology like genetics [9] to something completely different like stock market analysis [10]. Time series are also quite common in medicine, since patient records from sequential appointments form a time series. Forecasting is the attempt at predicting future events based on past information. In the case of time series, this means trying to find the generative model, or at least a similar one, in order to accurately estimate the set of desired variables in future events. An example that illustrates the importance of forecasting in medicine could be the attempt at finding people that will soon develop a disease, such as cancer, based only on their previous records. This also illustrates some of the difficulties regarding forecasting, since we may not know what features are truly important to make predictions at first, so there is a lot of trial and error. Forecasting is difficult in small time series as well, as the lack of information will heavily hinder the accuracy of any model we may attempt to fit.

As for time series classification techniques (TSC), these revolve around training a classifier on a set of labeled cases. The main difference from classifying regular objects is that in time series we have ordered data. Nevertheless, some processing of the time series is imperative, since different techniques attempt to find different discriminatory features. So in some cases, instead of comparing the several complete time series, intervals within them are defined and compared instead [7]. As mentioned before in this study we pretend to fit a generative model to each time series, some TSC techniques attempt to do the same but with the intent of classification by checking for similarity between models. Even though the modeling process is important to this study, the plan is not to classify but to cluster, which motivates the mention of time series clustering.

Clustering time series is the main goal of this study and there are several different approaches one can take [11], e.g. raw-data-based, feature-based and model-based techniques. They serve the same purpose, process the data to a point where static data clustering methods can be applied, or at least slightly altered versions of them.

Emphasis is given to model based clustering, in which, each cluster is associated to a certain model, and the data is fit to each of the models and clustered according to the cluster that offered the best possible fit. This is similar to what we intend to do, however slightly different. In our approach there is not a single model that defines a cluster, this would only happen in an ideal yet unachievable scenario.

Instead, the way we will cluster time series goes through modeling every single time series and then applying a standard clustering algorithm to the models. To do this though, statistical distances need to be defined, which are essentially distances between models.

2.2 Distances

In order to properly understand clustering the notion of distance must be properly defined, since if we are attempting to cluster objects that resemble each other, minimizing the distance between them in a well defined space should be an intuitive approach.

Euclidean Distance A brief overview of the "default" distance is necessary to define a knowledge base to compare the other more complex distances to.

In Cartesian coordinates, given two points $p = (p_1, p_2, ..., p_n)$ and $q = (q_1, q_2, ..., q_n)$, the Euclidean distance between these two points in an Euclidean n-space is given by:

$$d(p, q) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}. \tag{2.2}$$

Even though the Euclidean distance has several applications, it has limitations, and in our case those limitations are extremely relevant since it's not able to provide a meaningful value of distance between two random variables or probability distributions. This means we need another type of distance to achieve proper values. What we are looking for is a statistical distance, a value that can be interpreted as a distance between statistical objects.

Dynamic Time Warping This distance measure, although not a statistical distance, deserves to be mentioned as well, since it's the most used measure to compare two time series [12]. Its operation principle revolves around connecting time points that are not necessarily the same. This is motivated by the fact that two different time series may vary in speed. A non-linear warp in the time dimension is performed to measure the similarity between the several points of the time series in order to maximize similarity.

One can compare the use of the euclidean distance stated above with dynamic time warping (DTW), applied to time series with a brief example as shown in Figure 2.1.

As we can see, whilst performing the euclidean distance makes a relation between each time point regardless of their similarity, using the DTW allows for a better alignment between time series. DTW is used for several different applications, for example speech recognition [14].

Total Variation Distance The Total Variation Distance, TVD, is a statistical distance. A statistical distance quantifies the distance between two statistical objects as mentioned previously. This metric represents the biggest possible difference between two statistical objects.

The TVD between P and Q, two probability measures, is given by,

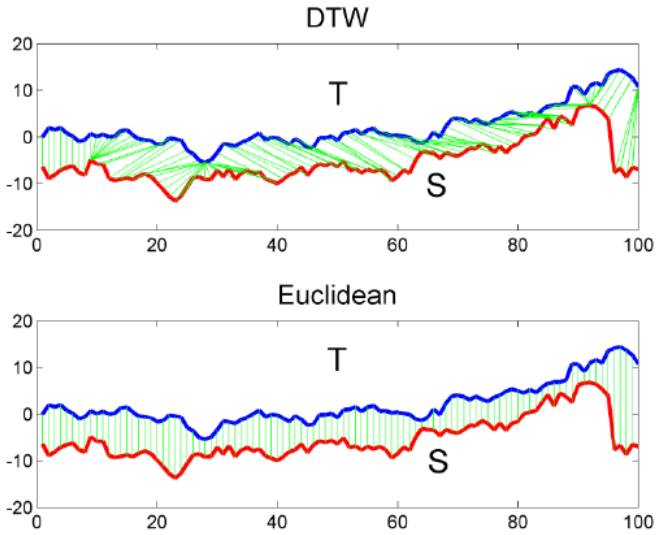

Figure 2.1: Contrast between the euclidean distance and dynamic time warping when applied to time series [13].

$$\sigma(P, Q) = \sup |P(A) - Q(A)|. \tag{2.3}$$

The TVD was chosen since it provides an intuitive although slightly pessimistic metric, that would make for a good comparison term.

Kullback-Leibler Distance This represents the distance between two statistical objects, in our case two statistical models. In [15] a fast approximation of the KLD is provided for some statistical models and in [16] we get the exact computation of KLD, providing both an intuitive idea and a formal definition.

The KLD measures how two probability distributions diverge from one another. In Ghassempour et al. [6], they resort to it to define a distance between several Hidden Markov Models by computing their likelihoods. The expression uses the fact that likelihoods can be seen as probability density functions, and as such, for two different probability density functions P and Q the KLD is defined as:

$$D(P||Q) = \int_{-\infty}^{\infty} p(x) \times log\frac{p(x)}{q(x)}dx. \tag{2.4}$$

Approaches to simplify the expression are given in [15]. Since the integral might be troublesome in a space with a high dimensionality, approximations that resort to numeric approaches are commonly used, for example the use of Monte Carlo method [6]. It is also important to note that this distance is not symmetric, so a symmetrization process is also required.

It is also related to other distance metrics, such as the aforementioned TVD by Pinkser's Inequality presented below.

Pinsker's Inequality Pinsker's Inequality gives us the relation between the TVD and KLD. It states that if P and Q are two probability distributions on a measurable space ($\mathbf{X}$, $\mathbf{\Sigma}$), then:

$$\sigma(P,Q) \leq \sqrt{1/2 D_{KL}(P||Q)} \tag{2.5}$$

where $\sigma(P,Q)$ represents the TVD, and D_{KL} represents the KLD.

Hellinger Distance In order to find the best distance metric possible, two more distances were implemented, the Hellinger Distance, HD, being one of them. The Hellinger distance is a probabilistic analog of the Euclidean distance. For two discrete distributions, P and Q, it is given by:

$$HD(P,Q) = \frac{1}{\sqrt{2}} * \sqrt{\sum_{i=1}^{k} (\sqrt{p(i)} - \sqrt{q(i)})^2}. \tag{2.6}$$

This distance metric is commonly used when using statistical models [17], so it is applicable in this study to measure efficiency in the context of clustering statistical models by comparing with results from other distances.

Bhattacharyya Distance The other implemented distance was the Bhattacharyya Distance, BD. Its expression for two discrete distributions is the following

$$BD(P,Q) = -\log(\sum_{i=1}^{k} \sqrt{p(i)q(i)}) \tag{2.7}$$

Bhattacharyya's coeficient is given by:

$$BC = \sum_{i=1}^{k} \sqrt{p(i)q(i)} \tag{2.8}$$

This distance is also heavily used in data mining, specifically for clustering purposes [18], [19], so its inclusion in this study was indispensable.

2.3 Clustering

Clustering is a process that consists in placing elements into groups according to relevant data features. These groups are called clusters, and the objects within these clusters should be as similar between themselves as possible and accordingly, they should be dissimilar from the objects in the other clusters.

This is a topic that has been heavily explored and as such a large amount of different algorithms were already devised. For instance, in [20] the different clustering algorithms are separated into four types: partitioning methods, hierarchical methods, density-based methods and grid-based methods. Ironically these are not mutually exclusive, since a method might not fall directly under one of these categories but instead be a combination of them. However, it's often hard to find relevant independent groups for information when dealing with complex objects.

Partitioning methods In general, partitioning methods revolve around creating a set of k partitions, $k \in N$, to divide the data into. A relocation of the objects between clusters is then performed to improve

the partitioning. They are distance based and as such often euclidean or manhattan distance is used but in our case, since we are interested in clustering statistical objects, a statistical distance is used instead. k-Means and k-Medoids, are commonly used clustering partitioning methods.

Partition Around Medoids An application of k-Medoids, Partition Around Medoids [21] was the partition based clustering method chosen. The algorithm itself is described in Algorithm 1.

Algorithm 1 PAM clustering process

Input : k: Number of data points that will randomly be considered medoids.

Output: $pamIndexVec$: Array with the correct cluster in each index.

```
1  Initialize k medoids
2  foreach non-medoid o do
3      mDist = 0  minDist = inf  foreach medoid m do
4          mDist = Compute distance between o and m  if mDist < minDist then
5              minDist = mDist  minMedoid = m
6          else
7              continue
8      Associate each non-medoid o to corresponding minMedoid
9  while Cost of configuration decreases do
10     foreach medoid m do
11         foreach non-medoid o do
12             Swap m and o  Cost = Sum of distances of points to their medoid
```

The algorithm displayed above starts by randomly considering points from the set of data provided as medoids, or centers of the future clusters. After having the set of medoids, each other data point is associated to the closest medoid effectively separating the data.

The second part of the algorithm consists in increasing cluster quality. This is done by minimizing a cost function that consists in the sum of all distances between each point and their corresponding medoid. Each non-medoid turns into the new medoid every iteration, in order to test all possible points to achieve the lowest possible cost.

Fuzzy Clustering Partitioning methods aim at creating hard clusters discarding any ambiguity that may lie in the grouping process. This makes every data object that is mistakenly placed into a cluster it does not belong to equally misplaced, which is not accurate. Fuzzy clustering [22] is a generalization of partitioning in which every object is given a score based on how much they belong to each cluster. This method is incredibly useful in the sheer amount of extra information it provides, allowing the user to differentiate between points that completely belong to a certain cluster and cases that present a high level

of uncertainty. Fuzzy clustering methods are also computationally demanding, having a higher complexity than methods such as PAM, which is reasonable since one has to take into account the amount of extra information that becomes available.

Since fuzzy clustering shows how much each data object belongs to each cluster in percentage, it's a powerful tool to validate results, which will be its main use in this study.

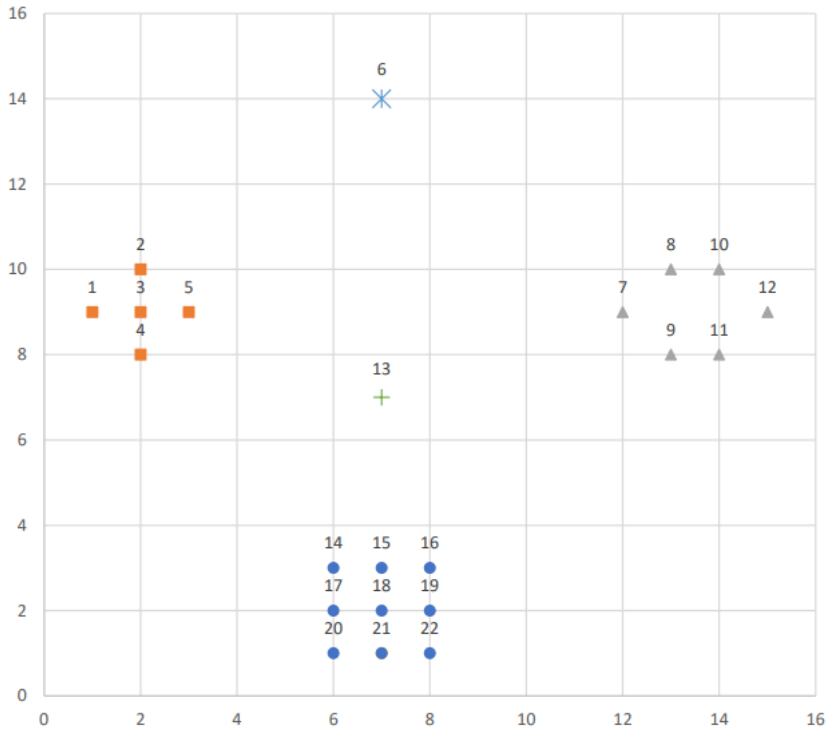

Figure 2.2: An assortment of points that attempts to depict the usefulness of fuzzy clustering. There are clearly 3 separate clusters but there are two points, 6 and 13, that are ambiguous.

For instance, in Figure 2.2, we can clearly see that there are three separate groups of points. Points 1 to 5, 7 to 12 and 14 to 22 have no problems being separated into hard clusters. Points 6 and 13 are not the case. On the surface level it's not obvious in which cluster they should be inserted and that is the motivation behind using methods such as fuzzy clustering. A fuzzy clustering algorithm would be able to provide a percentage of belonging of both points to each of the clusters and we could work with that extra information and act accordingly.

Hierarchical Clustering This is a slightly different approach to clustering than the previously presented techniques. It can be either agglomerative or divisive, depending on the starting point. If we consider a "top-down" approach we start with a cluster with every data object contained in it and divide it into smaller clusters. This is not the approach we intend to use, instead a "bottom-up", agglomerative, method in which every single data object starts off as a their own cluster. Consequently for the N data objects contained in our database we have a corresponding N clusters. An iterative process is then started, in which the N clusters are merged in pairs according to their proximity until there is only 1 big cluster that contains all of the others.

The results of Hierarchical Clustering can be graphically represented recurring to a dendogram, which is a diagram that illustrates the whole hierarchy achieved through the clustering process.

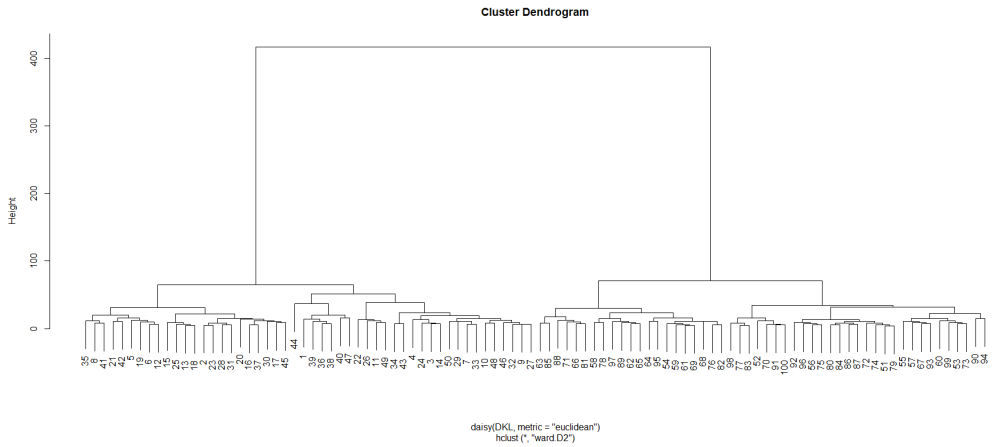

Figure 2.3: An example of a dendogram resulting from 100 simulated time-series after being processed by the proposed method.

Hierarchical Clustering is most effective when used to identify the relations between data objects at a more atomic level, since unlike partitioning methods, they establish direct correspondence between pairs of individuals. This gives a completely different insight from partitioning methods on the intrinsic relations between data objects. So for example in Figure 2.3 there are a total of 100 simulated time-series originating from 2 different dynamic bayesian networks, 50 from each, numerated accordingly (first 50 numbers were based on the first model and the last 50 on the second) . This is an example that displays the strengths in hierarchical clustering. We can clearly see the separation between time-series numbered 1-50 and 51-100. Furthermore if we want to make an in depth analysis between any subgroup of time-series, the dendogram actually hints to pairs that have a high chance of being similar.

2.3.1 Cluster Evaluation

As an input, partitioning methods like PAM require the user to input the desired number of clusters. This means that to find an optimal number, one must do some trial and error beforehand. Cluster evaluation methods allow us to compare these trials. The methods can be either Extrinsic or Intrinsic, depending on whether or not an ideal solution is available, also called *ground truth* [20]. If it is, then Extrinsic methods can be applied, making a series of direct comparisons between the ideal clusters and ours possible, particularly in terms of homogeneity and completeness. If no *ground truth* is available, which tends to be the case, Intrinsic methods have to be used instead.

Extrinsic Methods If we have the *ground truth* of a certain dataset, D, a great way to test algorithm effectiveness is to test it on D, and measure the accuracy resorting to extrinsic methods.

In [23], a selection of several methods is available and since all have their strengths and drawbacks only a few are worth noting. They are separated into three different types: measures based on counting pairs, measures based on set overlaps and measures based on mutual information.

A few notation definitions are presented for a better understanding of the following concepts, although we follow the notation used by [23], just in case anything is unclear.

Having X as a finite set with cardinality $|X| = n$, a clustering C is composed of $C_1, ..., C_k$ non-empty subsets of X and its elements collectively form X. Likewise, C' refers to a second clustering of X, composed by $C' = C'_1, ..., C'_l$.

The confusion matrix of the pair (C, C') is denoted by M, with entries $m_{ij}, i \in \{1, ..., k\}$ and $j \in \{1, ..., l\}$ is given by:

$$m_{ij} = |C_i \cap C'_j|. \tag{2.9}$$

Measures Based on Counting Pairs - This approach, like the name implies, consists of counting pairs of objects that were clustered the same way both in the clustering corresponding to the "ground truth", C, and in the new clustering provided by a clustering algorithm, C'.

Let n_{ab}, $a, b \in \{0, 1\}$, represent the various options regarding correct or incorrect clustering in both C and C'.

n_{11} and n_{00} are the pairs that are in the same cluster under C and C' and in different clusters under C and C' respectively. On the other hand n_{10} and n_{01} are pairs that have the same clustering under C and different ones in C' and vice-versa, respectively.

General Rand Index This index is fairly straight-forward and very well known [24]. It measures the fraction between the number of pairs that were equally classified in both clusterings and the total number of objects. The value is given by:

$$R(C, C') = \frac{2(n_{11} + n_{00})}{n(n-1)}, \tag{2.10}$$

where R ranges between $[0, 1]$, but it is heavily hindered by being biased by the number of clusters [25]. This is intuitive since it depends directly on n_{00}, which is a double-edged sword as a metric to measure similarity between clusterings. In the case where a lot of different possible classes exist, this number is bound to sky-rocket making this index undesirable for such conditions.

Jackard Index The Jaccard Index is similar to the Rand index. The main difference relies on disregarding n_{00}, the pairs of elements that are assigned to different clusters in both C and C'. It is defined by

$$JI(C, C') = \frac{n_{11}}{n_{11} + n_{10} + n_{01}}. \tag{2.11}$$

Fowlkes-Mallows Index The FM index [26], is a better indicator of accuracy as it can be seen as the geometric mean of precision and recall. The value is given by:

$$FM(C, C') = \frac{n_{11}}{\sqrt{(n_{11} + n_{10})(n_{11} + n_{01})}}. \tag{2.12}$$

However this measure and the previous suffer from opposite problems since the value tends to be higher with a low number of clusters. The heavy bias regarding number of clusters present in these indexes is undesirable and weakens their strength as accuracy metrics. Nevertheless, they are ways of evaluating the confusion matrix and can be interesting additions to the results evaluation.

Measures Based on Set Overlaps - As the name implies, these metrics try to match clusters that reveal some sort of overlap in their elements.

F-Measure The F-Measure tries to indicate how good a clustering C'_j describes class C_i by evaluating the precision $p_{ij} = \frac{m_{ij}}{|C'_j|}$ and recall $r_{ij} = \frac{m_{ij}}{|C_i|}$, both measure taken directly from the confusion matrix between clusterings. This measure is obtained by:

$$F(C_i, C'_j) = \frac{2r_{ij}p_{ij}}{r_{ij} + p_{ij}}, \tag{2.13}$$

followed by

$$F(C, C') = \frac{1}{n} \sum n_i \max_{j=1}^{l} \{F(C_i, C'_j)\}. \tag{2.14}$$

This is not a symmetrical measure, however this is not a problem since we intend to use it to compare a possible clustering solution with an optimal clustering, hence we only need how far from perfect our solution will be.

Maximum-Match-Measure The Maximum-Match-Measure, or MMM, can be computed by taking the largest entry of the confusion between achieved clustering C and ideal clustering C' and making the correspondence between each cluster by finding the highest element in the matrix, assigning that row and that column as the same clustering and repeating until every achieved cluster in C has a correspondent cluster in C'. The measure in then computed by summing the aforementioned matches and dividing them by total number of elements:

$$M(C, C') = \frac{1}{n} \sum_{i=1}^{\min\{k,l\}} m_{ii'}. \tag{2.15}$$

Measures Based on Mutual Information -

Normalized Mutual Information This approach, also commonly used [27] is based on information theory, which revolves around entropy, and not pairs of consistently clustered points. For a text T with alphabet α, entropy is given by

$$S(T) = -\sum_{i \in \alpha} p_i log_2(p_i), \tag{2.16}$$

where p_i is the probability of finding i in T

$$H(C) = -\sum_{i=1}^{k} P(i)log_2(P(i)). \tag{2.17}$$

The mutual information between two variables represents the mutual dependency between them. It is related to entropy since we can measure how much we can reduce the uncertainty about the clustering of a random object when comparing it to the ideal clustering. The mutual information is given by

$$I(C, C') = \sum_{i=1}^{k}\sum_{j=1}^{l} P(i,j)log_2\frac{P(i,j)}{P(i)P(j)}, \tag{2.18}$$

where

$$P(i,j) = \frac{|C_i \cap C_j|}{n}. \tag{2.19}$$

Two different approaches are presented, [28] and [29] given by:

$$NMI_1(C, C') = \frac{I(C, C')}{\sqrt{H(C)H(C')}} \tag{2.20}$$

and

$$NMI_2(C, C') = \frac{2I(C, C')}{H(C) + H(C')} \tag{2.21}$$

respectively.

Intrinsic Methods

Silhouette Considering a dataset D composed by a set of n data points, $o \in D$, separated into k clusters, $C_1, ..., C_k$ both the average distance from o to other data objects within the same cluster and the minimum average distance from o to every cluster excluding the one o belongs to must be computed in order to compute the Silhouette [30]. This is formalized in equations (2.22) and (2.23), but one can think of (2.22) as cluster compactness and of (2.23) as cluster separation.

$$a(i) = \frac{\sum_{o' \in C_i, o' \neq o} dist(o, o')}{|C_i| - 1} \tag{2.22}$$

$$b(i) = min\left\{\frac{\sum_{o' \in C_j} dist(o, o')}{|C_i|}\right\} \tag{2.23}$$

The index is computed by relating (2.22) and (2.23), as shown in (2.24).

$$S(i) = \frac{b(o) - a(o)}{max\{a(o), b(o)\}} \tag{2.24}$$

$S(i)$ gives us a value that ranges between $[-1, 1]$ for every data point in which a value close to 1 indicates that the point is well matched within its cluster. In order to evaluate the cluster, an average of the silhouette values of every data point in that cluster is common practice and it's referred as the

silhouette score of that cluster.

Davies–Bouldin Index In a cluster C_i, with centroid A_i and size T_i, one can compute the Davies-Bouldin Index or DB index for short is commonly used alongside Silhouette [31]. It is a symmetric, non-negative ratio between intra-cluster scatter (2.25) and inter-cluster separation (2.26). Given an n-dimensional feature vector X_j we can compute the intra-cluster scatter using euclidian distance by assuming $p = 2$.

$$Sc_i = \left(\frac{1}{T_i} \sum_{j=1}^{T_i} |X_j - A_i|^p \right)^p \tag{2.25}$$

The inter-cluster separation is given by computing the distance between the centers of the several clusters. The $a_{k,i}$, $a_{k,j}$ present in Equation (2.26) refer to the kth elements of centroids A_i and A_j respectively.

$$M_{i,j} = ||A_i - A_j||_p = \left(\sum_{k=1}^{n} |a_{k,i} - a_{k,j}|^p \right)^{\frac{1}{p}} \tag{2.26}$$

The DB index is then computed directly with the values obtained in equations (2.25) and (2.26) with equations (2.27), (2.28) and (2.29).

$$R_{i,j} = \frac{Sc_i + Sc_j}{M_{i,j}} \tag{2.27}$$

$$D_i \equiv \max_{j \neq i} R_{i,j} \tag{2.28}$$

$$DBI \equiv \frac{1}{N} \sum_{i=1}^{N} D_i \tag{2.29}$$

A good DB index is one that is the closest to zero as possible. This happens when the sparsity of each cluster is very small when compared to the separation in-between clusters.

2.4 Hidden Markov Models

The Hidden Markov Model (HMM) [32] is a statistical model in which we do not have access to the states, hence the name. It's characterized as being doubly stochastic, since it varies based on a probability distribution in both emission and transition.

The emission probabilities represent the likelihood that a given observation is made based on the current state, A, and the transition probabilities refer to state progression over time, B. The emission and transition distributions together with the initial state distribution, π, constitute the set of parameters commonly denoted by $\lambda(A, B, \pi)$ of the HMM.

In order to provide an example of a simple HMM, consider the one in Figure 2.4. This is the intuitive representation of an HMM, where for each time instant, t, an observation, O_t, is taken according to the current state's emission probabilities and then the state changes according to the transition probabilities. This HMM is a possible solution to the casino coin example, in which we have two coins, one is biased and one is not.

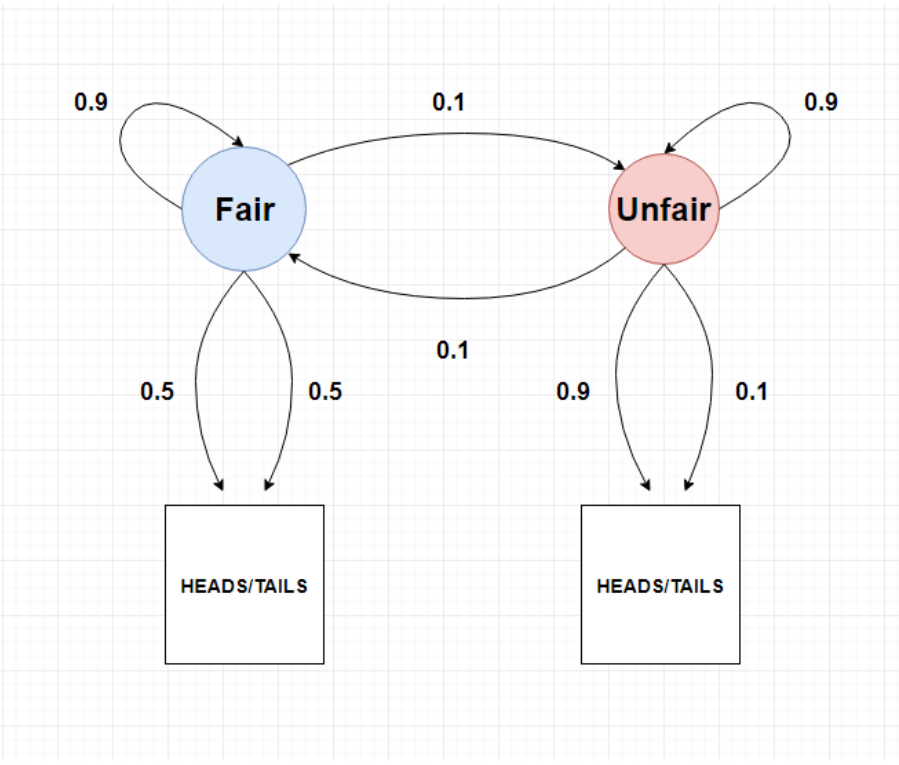

Figure 2.4: Example of an HMM with two hidden states.

The process that describes the generation of an observation sequence $O = \{O_1, O_2, ..., O_T\}$ of length T is described in Algorithm 2, which was presented in [32].

Algorithm 2 Observation sequence generation from HMM

Input : none

Output: $O = \{O_1, O_2, ..., O_T\}$

Choose an initial state, i_1, according to the initial state distribution π

Set $t = 1$

while $t < T$ **do**

  Choose O_t according to $b_{i_t}(k)$, the symbol probability distribution in state i_t

  Choose i_{t+1} according to $\{a_{i_t i_{t+1}}\}, i_{t+1} = 1, 2, ..., N$, the state transition probability distribution for state i_t

The three problems of HMM's

For the casino coin example, the presented HMM in Figure 2.4 might be the one that best describes a set of observations, but how can we know for sure? There is an infinite number of possible models for each observation sequence.

In [32] the set of problems that must be solved in order to provide useful Hidden Markov Models are described as follows:

- "Given both an observation sequence $O = O_1, O_2, ..., O_T$, and the model $\lambda = (A, B, \pi)$, how we compute $Pr(O|\lambda)$, the probability of the observation sequence."
- "Given the observation sequence $O = O_1, O_2, ..., O_T$, how we choose a state sequence $I = i_1, i_2, ..., i_T$ which is optimal in some meaningful sense."
- "How we adjust the model parameters $\lambda = (A, B, \pi)$ to maximize $Pr(O|\lambda)$."

The answers are also found in [32]. They mainly consist in three methods: the Forward-Backward algorithm to estimate the probability of an observation sequence given an HMM, the Viterbi algorithm to choose the most relevant state sequence to match a set of observations, and the Maximum Likelihood estimator to maximize the model parameters.

HMM's are an extremely versatile tool, being used extensively in both classification [33] and clustering [34], although with different approaches. If used to classify, the HMM works as a probabilistic model that describes the data. By training several HMM's and computing their likelihoods for a given observation sequence, we can then classify the observations to the HMM that returned the highest likelihood value. If the intention is clustering, then for each observation, a corresponding HMM is generated. Afterwards a distance like the KLD is defined between the several HMM produced. With a distance matrix, regular clustering algorithms may be applied. This method is applied in [6], and will be shown in Chapter 4.

2.5 Bayesian Networks

The specification of joint probability distributions is unpractical due to the absurd amounts of numbers required to do so.

For example, if we were to determine the joint distribution of

$$P(x) = P(x_n|x_1...x_n - 1)...P(x_1), \tag{2.30}$$

where we have a set of n random variables with K possible states. This would require $K^n - 1$ numbers to specify all possible K^n values. For small numbers of n, this would seem feasible, but for instance, if $n = 10$, there would already be 1024 possible values.

In an area that focuses so heavily on efficiency, this need for speed lead to the emergence of Bayesian Networks (BN) as a possible solution. A BN consists of a graphical representation of a joint probability distribution over a set of random variables [35]. It is formally defined as a triple $B = (\mathbf{X}, G, \theta)$, where:

- $\mathbf{X} = (X_1, ..., X_n)$ is a random vector where each random variable X_i takes values in a range over a finite domain D_i

- $G = (\mathbf{X}, E)$ is a directed acyclic graph (DAG) in which $\mathbf{X}$ are represented as its nodes and E as its edges. The edges represent direct dependencies between the variables.

- $\theta = \theta_{ijk_{i\epsilon\{1,...,n\},j\epsilon\{1,...,q_i\},k\epsilon\{1,...,r_i\}}}$ is a set of parameters corresponding to the local distributions of the network, in which:

$$\theta_{\mathbf{ijk}} = P_B(X_i = x_{ik}|\mathbf{pa}(X_i) = w_{ij}), \tag{2.31}$$

  where $\mathbf{pa}(\mathbf{X_i})$ refers to the set of possible parent configurations X_i, r_i is the number of values X_i can take, x_{ik} is the k-th value of X_i, and q_i is the number of parent configurations.

An example of a BN taken from [36] is depicted in Figure 2.5. It describes cash compensation and overnight accommodation to air passengers in the event of long flight delays. A flight may be delayed due to aircraft maintenance problems or severe weather (hurricane, blizzard, etc.). Whenever the delay is not caused by an external event to the airline company, a passenger may be entitled to a monetary compensation. Regardless of the cause, if the delay is long enough, the passenger might be offered an overnight accommodation. As a result of the dependences encoded by the graph, the joint probability distribution of the network can be factored as

$$P(M, S, F, O, C) = P(M)P(S)P(F|M, S)P(O|F)P(C|F, S), \tag{2.32}$$

where only the first letter of a variable name is used: M—Maintenance problems; S—Severe weather; F—Flight delay; O—Overnight accommodation; and C—Cash compensation. In this simple example, all variables are Bernoulli (ranging over `T` and `F`). Inside the callouts only the CPTs for variables taking the value `T` are given.

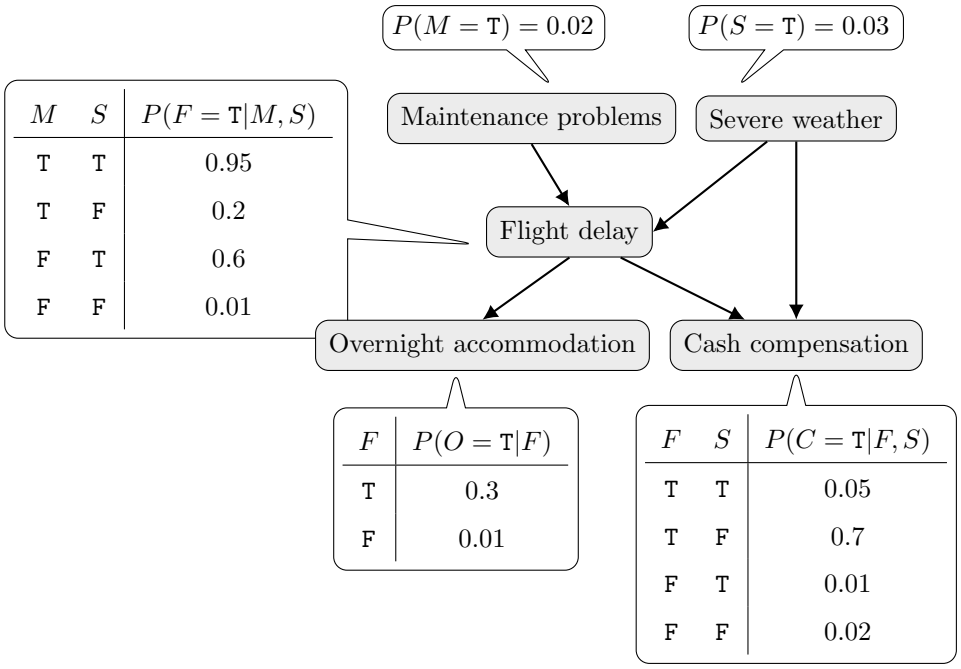

Figure 2.5: A BN example regarding airline regulations with conditional probability tables.

A Bayesian network is used to define a unique joint probability distribution over $\mathbf{X}$. It encodes independence assumptions over random variables in X, the dependencies are determined by the network structure, where an edge $Xj- > Xi$ represents direct dependency of X_i from X_j.

Conditional Independence In probability theory, two random variables A and B are considered conditionally independent given a third variable C, if knowing A does not bring any additional information to B and vice versa.

In Bayesian Networks X_i is conditionally independent of its non-descendants given its parents $\mathbf{pa}(X_i)$.

Markov Local Assumption A BN with a structure G over a set of random variables S encodes Conditional Independence between X_i and non-descendants given the set of parents of X_i

$$P_B(X_1, ..., X_N) = \prod_{i=1}^{n} P_B(X_i|\mathbf{pa}(X)) \tag{2.33}$$

2.5.1 Structure Learning

As the name implies, this refers to the process of finding a suitable graph structure G, however this is no easy feat and the main problem lies in computational complexity. In order to provide a bit of historic background about the hardness results of BN learning algorithms the following set of events is presented:

- In 1990 Cooper [37] showed that the inference of a general BN is NP-Hard, which drove the community away from exact solutions, and instead into approximate solutions.
- However in 1993 Dagum and Luby [38] showed that even finding an approximate solution was considered to be NP-hard, so the next step was to restrict the structure search space.

- Later on, Chickering [39] showed that learning the structure of a BN is NP-hard even for networks constrained to an in-degree of at most 2 and Dasgupta showed that even learning 2-polytrees is NP-hard. Due to these hardness results exact polynomial-time bounded approaches for learning BNs have been restricted to tree structures.

Thus the problem of finding the correct structure to model the data can be quite a handful. The standard approach used to learn BNs became heuristic search. The metric used to measure fitness is called a score, meaning this approach is also called score-based learning, in which a scoring functions is devised in order to measure the fitness of a model to a set of data.

Scoring Functions In order to find a suitable BN structure, a scoring function is applied.

Given a dataset $D = \{y_1, ..., y_N\}$, where $y_t = (y_{t1}, ..., y_{tN})$, for $1 \leq t \leq N$. $y_{ti} \epsilon D_i$, for all $1 \leq t \leq N$ and $1 \leq i \leq n$, and a scoring function ϕ, the problem of learning a BN turns into finding a BN B that maximizes the value of $\phi(B, D)$

Scoring functions can be separated into two categories, **Information-theoretic scoring functions** and **Bayesian scoring functions**, only the first type are explored in the context of this thesis.

Scoring functions have the following properties:

- **Score Decomposition**: Can be expressed as a sum of local terms $\phi(B, D) = \sum_i^n \phi(\mathbf{pa}(X_i), D)$, this is imperative to ensure efficiency on local search algorithms.
- **Score Equivalence**: Simply means that equivalent structures will be given the same score.

Information-theoretic scoring functions Information-theoretic scoring is based on compression. This means that the score of a given network B will value highly the amount of compression that can be achieved over the data T that was used to originate it. In order to estimate the minimum number of bits needed to encode our data T, an approach based on Shannon's entropy would be used.

The only scoring function that is explored in this thesis is the Log-Likelihood, which is presented promptly.

Log-Likelihood The LL score favors complete and consequently quite complex structures, which is often seen as a very significant hindrance, and often lessens its use. Instead an Ockham's razor approach, in which a penalization factor is applied to the original LL score is more common. Unfortunately this is not possible in our case, since the we aim to learn networks with a very limited amount of information. This means that the complex structures benefited from the Log-Likelihood score are actually what allow us to learn a relevant model, since attempts with other scoring functions like minimum description length (MDL) proved worthless.

The expression

$$LL(B|T) = \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk} log(\frac{N_{ijk}}{N_{ij}}), \tag{2.34}$$

is formalized in [40].

Learning a tBN The process of learning a Bayesian network given a dataset $D = \{x_1, ..., x_N\}$ and a scoring function ϕ, is to find the Bayesian Network B that maximizes the value of the scoring function for said dataset, $\phi(B, D)$. If we were to restrict the search space to tree networks, complexity of the algorithms would decrease significantly. An example of a staple used tree learning algorithm, is the Chow-Liu algorithm [41]. It starts by building a complete weighted undirected graph with the weights of each edge between X_i and X_j being $\phi_j(\{X_i\}, D) - \phi_j(0, D)$. Then a maximal weight spanning tree can be determined by randomly assuming of of the nodes as the root and all the edges are directed outwards of it. This is of interest to us since we plan on using tDBN structures which follow a similar process. By assuming a decomposable and score equivalent scoring function, ϕ, Heckerman [40] describes an algorithm that is a generalization of the Chow-Liu algorithm [41].

2.6 Dynamic Bayesian Networks

Dynamic Bayesian networks (DBNs) are valuable probabilistic representations that model stochastic processes, grasping the evolution of random variables through time.

Multivariate time series, usually represented by discrete-time data with T time slices, are described by transition networks, from time slice t to time slice $t + 1$, with $t \in \{0, ..., T - 1\}$. In a DBN, these transition networks are given by BNs constrained that edges between slices (inter-slice connections) must flow forward in time. These also benefits from intra-slice connections accounting for dependencies between the variables in the same time slice.

In this temporal setting, let $\mathbf{X}[t] = (X_1[t], \ldots, X_n[t])$ be a random vector that denotes a set of random variables at time t. Moreover, let $\mathbf{X}[t_1 : t_2]$ denote the set of random variables in $\mathbf{X}$ within the interval $t_1 \leq t \leq t_2$.

Using the chain rule, the joint probability over $\mathbf{X}$ is given by

$$P(\mathbf{X}[0 : t]) = P(\mathbf{X}[0]) \prod_{t=0}^{T-1} P(\mathbf{X}[t + 1] \mid \mathbf{X}[0 : t]). \tag{2.35}$$

For simplicity, two assumptions are usually considered: the Markov assumption and the stationary assumption. The first assumption is verified if and only if the right-hand side of Eq. (2.35) can be simplified as:

$$P(\mathbf{X}[t + 1] \mid \mathbf{X}[0 : t]) = P(\mathbf{X}[t + 1] \mid \mathbf{X}[t - m + 1 : t]). \tag{2.36}$$

where m is called the Markov lag of the process. In this case, predictions at time $t + 1$ depends on both the current values and the lagged ones (past period of m time slices) of the explanatory variables. On the other hand, stationarity is related with the concept of time invariance, that is, a m-th order Markov stationary process is one such that

$$P(\mathbf{X}[t + 1] \mid \mathbf{X}[t - m + 1 : t]) = P(\mathbf{X}[t + z] \mid \mathbf{X}[t + z - m + 1 : t + z]), \tag{2.37}$$

for all time slices.

$$P(X[t+1]|X[0:t]) = P(X[t+1]|X[m:t]). \tag{2.38}$$

Both the markovian lag and stationarity are important features that help simplify the concept of DBN. Meaning we can represent a probability distribution in a concise manner by having both an initial state distribution and a transition model.

Learning tDBNs In order to learn tDBNs, Dynamic Bayesian networks in which the search space is restricted to tree networks, we must recall the process of learning tree Bayesian networks mentioned in the previous section [40].

In [42] an algorithm that is capable of learning optimal stationary first-order Tree-augmented DBN structures is described. It states the following:

Algorithm 3 Optimal stationary first-order Bayesian T-DBN structure learning

Input : X: Set of network attributes

D: Dataset of MTS

ϕ: Decomposeable scoring function

Output: $tDBN$: Optimal stationary first-order Markov tree-augmented DBN structure

18 **foreach** *transition* $t \rightarrow t+1$ **do**

19 Build a complete directed graph in $X[t+1]$

20 Calculate the weight of all edges and the optimal set of parents of all nodes

21 Apply a maximum branching algorithm

22 Extract transition network of the final DBN using the maximum branching and the optimal set of parents

Chapter 3

Proposed Method

The proposed strategies are based on the one shown in [6], which is described with further detail below. Given a dataset D with N different data entries, the method can be separated into three steps:

- **Modeling -** The information contained in each data entry n_i, is used to find a DBN until we have N DBN's stored in a list. This is possible by applying an algorithm that finds optimal t-DBN structures to each of the separate data entries.
- **Obtaining the distances -** Since we have the different models corresponding to each of the objects of study and the final objective is to cluster, the intermediate step is obtaining the distance matrix M, that relates every model through the distance between them. One of the aforementioned distances is chosen and used on each of the models corresponding to each $X_i \in D$ to achieve a distance matrix M.
- **Clustering -** Lastly, we need to choose a clustering algorithm to finally group the data. The clustering algorithms presented previously will be tested with the intent of testing different approaches and reaching similar results. To achieve this we make use the distance matrix M computed in the previous step and directly apply the different clustering algorithms. In order to validate which is the optimal number of clusters for each method both Silhouette and DB index are computed and compared. To measure the accuracy on datasets in which we have labeled entries, the Rand Index, Jackard Index, Fowlkes-Mallows Index, Normalized Mutual Information, Variation of Information and Maximum-Match-Measure are computed.

The original Algorithm [6] is presented below and will be briefly described as the reasoning is similar to ours. This algorithm showed promise in its ingenious way of working around the drawbacks of working with multivariate time series, mainly since they may contain several different data types. This proves to be quite challenging for other clustering methods, since a well defined distance between categorical values is not available, so comparison is often difficult.

Firstly, to be able to model each X_i, as an independent HMM, we must firstly define the number of desired hidden states for each of them. The input trajectories T should be previously preprocessed into a list, in this case we call it $TList$. After all the necessary inputs are provided, we make use of

Depmix4's [43] functionalities to randomly initialize as many HMM's as there are trajectories, one for each. This information is stored in a list, named $mod[T]$, and the HMM's parameters λ are later computed and stored into $fmod[T]$. The distance measured between the HMM's that were trained was the KLD. In order to properly compute it we need to extract probability distributions from the HMM's, in this case the likelihoods. These are obtained using a function from R package Depmix4 yet again, and are stored in a squared matrix named LL, which is then normalized by rows and we get LLN, a squared matrix that contains all the likelihoods between each patient. After obtaining the different likelihoods, the distance between them is computed according by direct application of equation (2.4) followed by a symmetrization process. The final matrix containing the distances between every object is stored in KLD. To finish the process, PAM clustering algorithm [44] is performed with a series of different values for desired clusters as argument. This is done in an attempt to find the optimal number of clusters, since for every value, intrinsic and extrinsic cluster validation indexes are applied.

Algorithm 4 MTS clustering algorithm using HMM

Input : n: Number of hidden states for each HMM

$TList$: List of trajectories T

Output: $dummies$: Dummy variables of the features of each trajectory T

$mod[T]$: Stores the initialized HMM for trajectory T

$fmod[T]$: Stores the estimated parameters of HMM in $mod[T]$

$LLN[i,j]$: Stores the likelihood of trajectory T_i being generated by model $mod[T_j]$ with parameters $fmod[T_j]$. Normalized by rows.

$KLD_{[}n \times n]$: Squared matrix that stores symmetrized distances between all trajectories T

$pamMtx$: Matrix with the cluster information given from PAM R function

$intrVals$ and $extrVals$: Cluster validation information

```
foreach T in TList do
    dummies = Create dummy values for feature of T;
    mod[T] = Initialize HMM(n, dummies);
    fmod[T] = FitParameters(mod[T]);
foreach T_i = i-th T in TList do
    foreach T_j = j-th T in TList do
        LL[i,j]= logLik(mod[T_i],fmod[T_j]);
LLN = NormalizeRows(LL)
foreach R_i = i-th Row of LLN do
    foreach R_j = j-th Row of LLN do
        KLD=ComputeKLD(R_i,R_j)
for nrClusters = 2:10 do
    pamMtx = pam(KLD, nrClusters)
    intrVals = Intrinsic(pamMtx,DKL)
    extrVals = Extrinsic(pamMtx,DKL,realClasses)
```

With this in mind, a similar strategy arose, but instead of using HMM's and then comparing the likelihoods using KLD, we propose the use of DBN's and comparing them using an assortment of distances, in the hope of finding the one that better suits the method generally or, if we are unable to withdraw such conclusions, it would also be interesting to find which distances work best for each sets of data.

The first step is to model each data entry in a given database D. For each $X_1, ..., X_N$, a T-DBN structure is learnt using the aforementioned Algorithm 3. Usually, one would use all the database entries to train a single DBN, since such little information as the amount present in a single time series, could be insufficient to get a proper DBN structure. However, since we need a model to represent each entry X_i, there is no way around it. We cannot do any sort of preemptive grouping of the database entries since for all we know they are independent. Interestingly, the Log-Likelihood, a score function that is

often discarded and replaced by others like Minimum Description Length since it often overfits models, is precisely what we needed. In our case, since each bayesian network is based solely on a single database entry, other score functions simply don't have the intended behavior, since the information available is not enough to learn a non-trivial network. This overfit given by the Log-Likelihood may prove beneficial for a couple of reasons. Firstly we don't want all of the networks to be the same. Ideally they would fall into a few categories, so they could be cleanly clustered later on. Secondly, this method was firstly designed with the intent of modeling human beings. It would not be surprising if the network that best describes the intricacies that define a human being were to be complex. Nevertheless the information available to train a DBN is so scarce that using other scoring functions such as the MDL does not work. After having the list of DBN, in order to compute the TVD we have to generate all possible combinations for each variable present in each network according to their domain in t as well as in $t+1$. We then have to go through the list of combinations to compute the probability that the given combination will be generated by each DBN. The distance between each pair of DBN is the biggest difference in terms of probability of generating the given combination out of all possible combinations. This metric is by default pessimistic and it becomes intuitive why using an example. Given a Bayesian Network with 5 binary attributes, all the possible combinations of values for t and $t+1$ would equate to $2^{10} = 1024$ different combinations. Even if for 1023 of them the probability of generating the given combination is extremely similar, it only takes 1 extremely discrepant value for the distance to be sizeable. This is the main reason behind the smoothing process applied to the likelihoods of a DBN. Experiments in which the considered distance is the mode, median and average instead of the most discrepant were also tested. When all of the distances are successfully computed, the distance matrix is then submitted to cluster analysis.

Algorithm 5 Computing TVD using DBN

Input : X: Set of network attributes

D: Dataset of MTS

ϕ: Decomposeable scoring function

Output: $DBNList$: List of several DBN corresponding to each entry of D

M: Distance Matrix

```
foreach row of D do
    DBNList = tDBN(D)
BiggestDif = −∞
foreach entry i of DBNList do
    foreach entry j of DBNList do
        foreach possible configuration of X do
            P_i = Probability of i generating the given configuration of X  P_j = Probability of j generating
             the given configuration of X
            if |P_i - P_j| > BiggestDif then
                BiggestDif = |P_i - P_j|
            else
                continue
            M[i, j] = BiggestDif
```

Another facet of the same algorithm was devised in an attempt to achieve stronger results. Since the TVD could be the source of sub-optimal results, a similar algorithm was devised but using a set of different distances. In this case, after attaining the list with all the proper networks, either the KLD, the HD or the BD can be computed. In order to do this, a way of comparing how likely it is for 2 networks to have similar outputs had to be devised. The strategy was to compute the likelihood that each network would generate the observations used to learn the different networks. The thought process behind this was that similar networks would be likely to generate similar outputs. Having the likelihoods and the implemented formulas to compute the different distances, the distance matrix is achieved and the clustering process is performed. In the end, the algorithm was named CRATES after the words Clustering Multivariate Time Networks and it's presented in Algorithm 6.

Algorithm 6 CRATES

Input : X: Set of network attributes

D: Dataset of MTS

ϕ: Decomposeable scoring function(LL)

dist: distance option (KL, HD, BTC)

Output: $DBNList$: List of DBN corresponding to each entry of D

M: Distance Matrix

LLN: Normalized scores of each instance

```
foreach row of D do
    Add to DBNList output of Algorithm 1
foreach entry DBN in DBNList do
    LLscore = 0.0 foreach index in 1:nrChecks do
        if index = 1 then
            LLscore = LL(DBN , config[index])
        else
            LLscore *= LL(DBN , config[index])
LLN = NormalizeRows(LL)
foreach R_i = i-th Row of LLN do
    foreach R_j = j-th Row of LLN do
        M = dist(R_i, R_j)
Cluster with M
Compute validation indexes
```

Chapter 4

Results and Discussion

In this chapter, we start by making a set of experiments on synthetic data before we use real data. A controlled environment will allow us to identify the algorithms strengths and weaknesses, which will hopefully highlight the most important characteristics that the real data needs to have to use the algorithms effectively. All the experiments were performed in an Intel Core i7-6700K CPU @ 4.00Hz $\times$ 8 processor and 8 GB of RAM. All the code is available at "https://github.com/ZeBorges/Crates" alongside some datasets to test the algorithm.

4.1 Synthetic Data

Experience 1 - A Sanity Check The very first experiment consisted in separating synthetically generated data. It was obtained by artificially building dynamic Bayesian networks and generating as many different multivariate time series as we desire. The dataset for this experiment was populated by generating 50 different subjects from each of the DBNs with the transition networks represented in Figure 4.1. Each multivariate time series generated had 5 different attributes and 20 time steps.

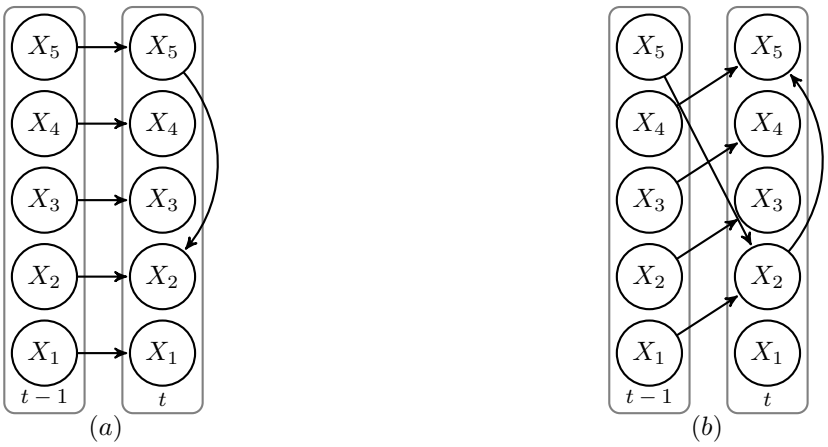

Figure 4.1: Transition networks of the two generated DBNs to perform the first experiment.

As we can see in Table 4.1 the results were excellent, being able to perfectly separate every single subject. All the chosen external validation indexes had a perfect score, as well as extremely solid internal

Table 4.1: First experiment on simulated data using the networks present in Figure 4.1. 100 total individuals, 50 from each network, each with 5 different binary/ternary attributes, 20 time steps, 2 possible classes considered for clustering.

Distance	RI	JI	FMI	NMI	MMM	DBI	AS
KLD	1.000	1.000	1.000	0.999	1.000	0.417	0.761
HD	1.000	1.000	1.000	0.999	1.000	0.616	0.596
BD	1.000	1.000	1.000	0.999	1.000	0.371	0.750

validation indexes for every single distance. From the values of the DB-index and the Average Silhouette, the Hellinger Distance seemed to produce the worst quality clusters, but not by much and since they all produced perfect clusters it's not really worth jumping to any conclusions in terms of performance.

Something extremely interesting that resulted from this experience was that after investigating, the tDBN structures that were being learnt from the algorithm were not identical to the ones that generated the data. In Figure 4.2, we can see represented as (b) one of 50 learnt structures for network (a) in Figures 4.1 and 4.2. Although it has some similarities, its different from the original. This is not at all surprising, especially since we can observe that the learnt structure is more complex than the original. This is a consequence of using the Log-Likelihood as the scoring function, as mentioned previously it favors more complex structures. Tests with another scoring function, Minimum Description Length, were fruitless, since the amount of information present in a single time series was not enough to learn any structures... It was very pleasing to observe that using a less conventional scoring function did not hinder the results at all but instead potentiated them.

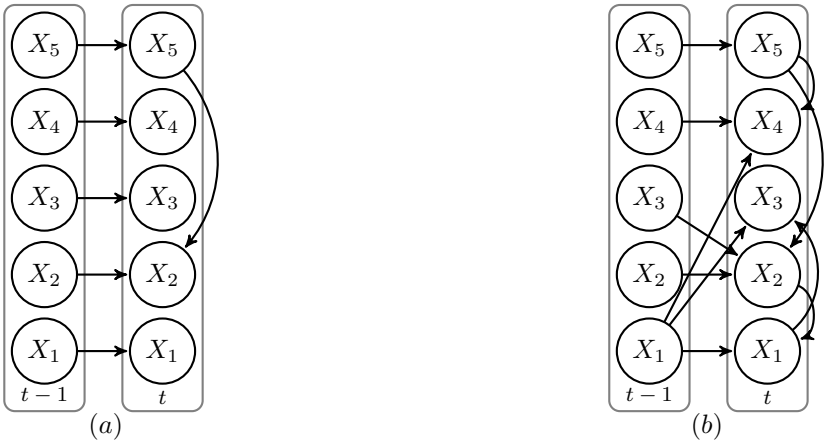

Figure 4.2: A way to depict the difference between the DBN from which the data was generated (a) and 1 out of the 50 learnt structures that closely resemble the original (b).

Experience 2 - Grasping the thresholds After an extremely successful first experience, the next step was to find in which conditions the algorithm stopped working as intended in a controlled environment. To do this properly we prepared two different scenarios. The first one was similar to Experience 1, again generating 50 subjects from each network in Figure 4.1, but now changing the properties of the time series. Instead of only inspecting what happens when the MTS have 5 attributes

Table 4.2: In depth test on simulated data using the two networks represented in Figure 4.1 to find thresholds. 100 total individuals, 50 from each network, the table on the left of the double line separation corresponds to experiments with 5 binary/ternary attributes and to the right of the double line 10 attributes. 5, 10, 15, 20 time points for each, 2 possible clusters and the values of the assortment of validation indexes for each different distance.

		5 attributes							10 attributes						
Time	Dist	RI	JI	FMI	NMI	MMM	DBI	AS	RI	JI	FMI	NMI	MMM	DBI	AS
5	KLD	0.834	0.713	0.832	0.564	0.910	1.549	0.283	0.511	0.433	0.617	0.062	0.590	1.418	0.451
	HD	0.728	0.570	0.726	0.370	0.840	1.633	0.226	0.631	0.460	0.630	0.209	0.760	2.223	0.151
	BD	0.756	0.607	0.756	0.429	0.860	1.827	0.255	0.592	0.425	0.596	0.154	0.720	2.855	0.200
10	KLD	1.000	1.000	1.000	0.999	1.000	0.543	0.637	1.000	1.000	1.000	0.999	1.000	0.677	0.643
	HD	1.000	1.000	1.000	0.999	1.000	0.686	0.535	1.000	1.000	1.000	0.999	1.000	0.621	0.585
	BD	1.000	1.000	1.000	0.999	1.000	0.543	0.633	1.000	1.000	1.000	0.999	1.000	0.535	0.651
15	KLD	1.000	1.000	1.000	0.999	1.000	0.612	0.714	1.000	1.000	1.000	0.999	1.000	0.433	0.739
	HD	1.000	1.000	1.000	0.999	1.000	0.737	0.556	1.000	1.000	1.000	0.999	1.000	0.577	0.631
	BD	1.000	1.000	1.000	0.999	1.000	0.674	0.717	1.000	1.000	1.000	0.999	1.000	0.403	0.745
20	KLD	1.000	1.000	1.000	0.999	1.000	0.417	0.761	1.000	1.000	1.000	0.999	1.000	0.428	0.775
	HD	1.000	1.000	1.000	0.999	1.000	0.616	0.596	1.000	1.000	1.000	0.999	1.000	0.602	0.615
	BD	1.000	1.000	1.000	0.999	1.000	0.371	0.750	1.000	1.000	1.000	0.999	1.000	0.400	0.785

and 20 time steps, since we already know that results in perfect clustering, in Table 4.2, we test MTS ranging from 5 to 20 time steps in intervals of 5 and with both 5 and 10 different attributes.

Another very exciting set of results since by observing Table 4.2 we realize that the time series don't have to be very long at all in order for the algorithm to produce proper results, since we achieved perfect clusters with only 10 time steps. The only section of the table that underperformed was for time series with 10 attributes described in only 5 time steps present in the top right of the table. This is perfectly reasonable. In this case the external validation indexes indicate the accuracy of around $50 - 70\%$ for a binary clustering scenario depending on the distance. Assuming that random clustering would have around 50% success, this is still a slight improvement for a situation with very little information. If we compare the left and the right tables for longer time series we see that having more attributes is actually provides better clusters albeit marginally.

After some more testing, we concluded that with 2 features or less it was simply not possible to get proper results. The reasoning is that there is just not enough information available for the algorithm to learn proper structures, meaning they are either trivial or not complex enough to be distinguishable and clustering inevitably fails. Also any less than 5 time steps is simply too little to yield proper results, which makes sense considering that the attributes probably wont range through their entire domain in such a short time period and as such the learnt models are extremely weak.

These results led us to up the ante and introduce more multivariate time series from 2 additional synthetically created networks, represented in Figure 4.3 as (c) and (d) and repeat the same experiment with the intent of testing non-binary clustering. The results of this experiment are presented in Table 4.3.

The results in Table 4.3 are worst than in Table 4.2, which is understandable given that this experiment increases the clustering complexity quite significantly, since if we were to randomly cluster the data assuming equal numbers in all 4 classes, which was the case, the accuracy should be around 25%. In this case the results were still extremely satisfactory according to the values of both intrinsic and extrinsic indexes.

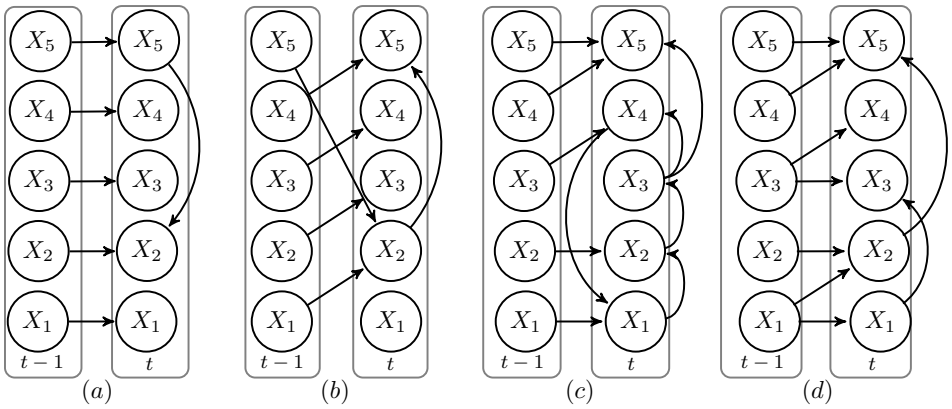

Figure 4.3: Transition networks of stationary first-order DBNs used to generate synthetic data to test the efficiency of the algorithm before using real data. All networks are represented with 5 features.

Table 4.3: In depth test on simulated data using the four networks represented in Figure 4.3 to find thresholds. 200 total individuals, 50 from each network, the table on the left of the double line separation corresponds to experiments with 5 attributes and to the right of the double line 10 attributes. 5, 10, 15, 20 time points for each, 4 possible clusters and the values of the assortment of validation indexes for each different distance.

		5 attributes							10 attributes						
Time	Dist	RI	JI	FMI	NMI	MMM	DBI	AS	RI	JI	FMI	NMI	MMM	DBI	AS
5	KLD	0.626	0.212	0.353	0.167	0.410	1.640	0.140	0.581	0.209	0.356	0.096	0.380	0.967	0.223
	HD	0.667	0.196	0.329	0.141	0.435	2.050	0.139	0.618	0.164	0.282	0.035	0.330	1.368	0.173
	BD	0.665	0.194	0.325	0.137	0.440	2.323	0.150	0.552	0.191	0.332	0.059	0.335	1.073	0.270
10	KLD	0.800	0.474	0.648	0.647	0.695	1.215	0.309	0.838	0.513	0.678	0.639	0.800	1.650	0.226
	HD	0.819	0.475	0.645	0.600	0.780	1.449	0.189	0.809	0.456	0.627	0.566	0.770	1.450	0.201
	BD	0.789	0.408	0.580	0.521	0.715	1.375	0.241	0.769	0.389	0.561	0.479	0.705	1.452	0.232
15	KLD	0.874	0.599	0.749	0.739	0.815	1.844	0.329	0.970	0.888	0.940	0.908	0.970	1.117	0.368
	HD	0.857	0.568	0.725	0.696	0.820	1.226	0.289	0.990	0.960	0.979	0.964	0.990	1.078	0.374
	BD	0.864	0.578	0.733	0.718	0.790	1.901	0.344	0.966	0.872	0.932	0.901	0.965	1.116	0.358
20	KLD	0.853	0.544	0.705	0.676	0.760	1.310	0.322	0.975	0.905	0.950	0.916	0.975	1.278	0.420
	HD	0.857	0.553	0.712	0.673	0.815	1.141	0.293	1.000	1.000	1.000	0.999	1.000	1.075	0.345
	BD	0.836	0.562	0.729	0.767	0.745	1.143	0.392	0.965	0.870	0.930	0.886	0.965	1.379	0.405

This experiment proved extremely useful to not only understand in which conditions the algorithm is capable of functioning properly but also to observe that when these conditions are met, it really is capable of clustering, even in non-trivial scenarios, such as the example depicted in Table 4.3.

For example if we compares Figures 4.4 and 4.5, we can see the tremendous difference having more time steps in our multivariate time series makes. Figure 4.4 corresponds to clustering performed with the Hellinger Distance with 5 time steps vs Figure 4.5 with 20 time steps, in which we achieve perfect clusters again.

```
  [1] 1 2 1 2 2 2 1 3 2 1 4 4 1 1 1 1 2 2 2 1 3 2 1 1 2 1 2 2 1
 [30] 1 1 2 1 2 1 1 2 2 1 1 2 1 2 1 1 1 3 1 1 3 4 4 3 3 4 2 4 4
 [59] 4 4 4 4 3 3 2 2 4 4 4 2 4 4 3 4 4 4 4 3 4 2 3 3 4 3 2 4 1
 [88] 3 4 4 4 3 3 4 2 3 4 4 4 4 3 2 3 2 2 2 2 1 4 3 2 2 2 4 4 3
[117] 3 1 2 2 1 2 1 2 3 2 4 2 2 2 4 3 1 1 1 3 4 4 1 3 3 1 4 3 2
[146] 2 1 3 4 2 3 4 1 1 3 1 3 4 3 2 3 3 3 1 1 2 1 1 1 4 4 1 1 1
[175] 4 4 1 3 1 4 3 4 2 3 2 4 1 3 2 1 2 4 4 1 4 2 4 3 1 4
```

Figure 4.4: Figure representing the clustering vector for 5 time steps and 10 attributes using the Hellinger distance, perfect case would be with 50 1's in a row, followed by 50 2's and so on.

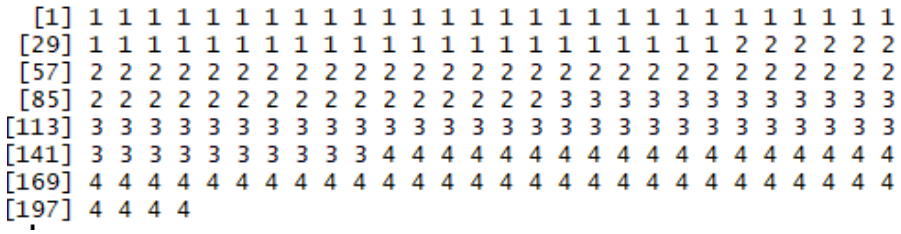

Figure 4.5: Figure representing the clustering vector for 20 time steps and 10 attributes using the Hellinger distance, perfect case would be with 50 1's in a row, followed by 50 2's and so on.

Of course one of the reasons the method works so well when using synthetic data is that the data itself is generated from Dynamic Bayesian Networks. This means that when attempting to find a generative model for the data there is in fact a possible DBN that can be learnt that describes the data perfectly. This is a luxury that will not happen when working with real data that is not only impossible to describe according to a learnt DBN but also might contain noisy values that hinder the learning process.

4.2 Real Data

Experience 3 - A Reality check A very successful set of experiments on synthetic data led us to having a deeper understanding of how our algorithm behaves. In this experiment we want to apply CRATES to real datasets. The datasets used in this experiment were mostly from Machine Learning Repository and Time Series Classification web repositories. The used datasets were the following:

- Uwave(4478 individuals, 3 attributes, 99 time steps, 8 possible classes) [45];
- Wafer(1194 individuals, 6 attributes, 99 time steps, 2 possible classes) [46];
- CT(2858 individuals, 3 attributes, 100 time steps, 20 possible classes) [47];
- Libras(360 individuals, 2 attributes, 45 time steps, 15 possible classes) [47];
- ECG(200 individuals, 2 attributes, 39 time steps, 2 possible classes) [48];
- ALS(100 individuals, 18 attributes, 6 time steps, 2 possible classes), from Neuroclinomics2 project;
- JV(640 individuals, 12 attributes, 7 time steps, 9 possible classes) [47];

None of these datasets is likely to make CRATES shine. This is because the multivariate time series are simply too small, have too many attributes for the number of time steps or have a lot of possible classes, resulting in a very hard dataset to cluster in general. Keeping this in mind, we still expect to have reasonable results in every dataset, but achieving any sort of perfect clustering seems unlikely. The GAK distance is a function present in dtwclust R package [49] that uses the Triangular Global Alignment Kernel (TGAK) [50], it is based on dtw distance and as such should produce interesting results to compare. After using GAK to obtain the distance matrix we simply cluster the distance matrix as we would with the ones resulting from CRATES, using PAM [44] and evaluating with the chosen indexes.

The reason we did not compare the results with the original presented by Ghassempour [6], was simply because we couldn't. The way the algorithm was programmed required an *a priori* knowledge

Table 4.4: Results of the experiments in several sets of real data Uwave(4478 individuals, 3 attributes, 99 time steps, 8 possible classes), Wafer(1194 individuals, 6 attributes, 99 time steps, 2 possible classes), CT(2858 individuals, 3 attributes, 100 time steps, 20 possible classes), Libras(360 individuals, 2 attributes, 45 time steps, 15 possible classes), ECG(200 individuals, 2 attributes, 39 time steps, 2 possible classes), ALS(100 individuals, 18 attributes, 6 time steps, 2 possible classes), JV(640 individuals, 12 attributes, 7 time steps, 9 possible classes)

Dataset	Distance	RI	JI	FMI	NMI	MMM	DBI	AS
Uwave	KLD	0.782	0.102	0.185	0.125	0.276	1.832	0.266
	HD	0.736	0.102	0.189	0.113	0.272	0.814	0.556
	BD	0.789	0.104	0.188	0.119	0.286	1.931	0.219
	GAK	0.886	0.405	0.578	0.627	0.690	1.110	0.440
Wafer	KLD	0.558	0.522	0.694	0.000	0.893	0.051	0.981
	HD	0.558	0.522	0.694	0.000	0.893	0.180	0.872
	BD	0.558	0.522	0.694	0.000	0.893	0.090	0.969
	GAK	0.499	0.447	0.635	0.000	0.893	1.245	0.280
ct	KLD	0.900	0.086	0.160	0.310	0.305	1.503	0.257
	HD	0.903	0.104	0.190	0.295	0.306	1.482	0.365
	BD	0.899	0.085	0.159	0.301	0.291	1.565	0.313
	GAK	0.961	0.471	0.643	0.770	0.740	1.150	0.420
Libras	KLD	0.876	0.091	0.169	0.331	0.311	1.454	0.324
	HD	0.883	0.086	0.159	0.324	0.294	0.839	0.599
	BD	0.874	0.084	0.157	0.324	0.291	1.896	0.189
	GAK	NaN	NaN	NaN	NaN	NaN	NaN	NaN
ECG	KLD	0.508	0.370	0.541	0.006	0.665	1.433	0.353
	HD	0.581	0.472	0.642	0.073	0.705	1.260	0.257
	BD	0.524	0.394	0.565	0.016	0.665	1.611	0.227
	GAK	0.590	0.500	0.670	0.079	0.715	0.614	0.784
ALS	KLD	0.531	0.414	0.586	0.017	0.673	1.051	0.580
	HD	0.521	0.400	0.571	0.010	0.673	0.600	0.575
	BD	0.536	0.429	0.601	0.018	0.673	0.493	0.665
	GAK	NaN	NaN	NaN	NaN	NaN	NaN	NaN
JV	KLD	0.814	0.166	0.286	0.275	0.337	0.964	0.574
	HD	0.827	0.173	0.295	0.313	0.400	0.954	0.459
	BD	0.819	0.162	0.280	0.273	0.334	1.107	0.591
	GAK	0.886	0.389	0.562	0.681	0.753	1.370	0.158

of how each attribute in the time series was related one another or at least some insight about it, since you had to manually insert it. Moreover the algorithm was not able to compare all kinds of subjects. If, for example, one of the multivariate time series were to never change the values of one ,or several, of its attributes or at least not range it across its domain, this MTS would not be comparable to others in which the attributes varied widely. This the main reason why we did not use it, since it would require us to remove a hefty amount of the test subjects, rendering any comparison meaningless. Fortunately the proposed approach does not suffer from such obstacles.

The results are displayed in Table 4.4

As predicted, no perfect clusters were achieved. The GAK distance behaved surprisingly well in most datasets, achieving some very impressive results, but it was unable to cluster 2 out of the 7 datasets, classifying every subject as being in the same cluster which is just plain wrong. In terms of external indexes, on the other 5 cases, it had better results than any of the 3 distances in CRATES, but not in

terms of internal indexes. This is curious since one would expect them to be concordant, if the accuracy is better why are the quality of the clusters in general worse? To fully understand this one would have to analyze the subjects present in each of the clusters and search for similarities. This was not possible, it can't be done except by specialists. But the fact that the produced clusters were of very high quality is very satisfactory, since it can just be an alternative sensible clustering. The most interesting dataset to analyze the results would be the Wafer [46] dataset. This is because it was the closest to the synthetic data tests in the real datasets. With only 3 attributes, it could be hard to learn proper networks complex enough to be distinguishable. However, with 100 time steps, these very long time series proved to be very good for CRATES. The results show something extremely odd, all the distances had the exact same behavior, having all the indexes with exactly the same values, at a relatively low accuracy percentage for what would be expected, but with almost perfect internal index values. This has to be an alternative clustering to the original classification, because its extremely rare to see such good values not only for the Silhouette but also for the DB-index, which directly correlate to cluster quality. It's hard to pinpoint a best distance to use with our algorithm. They each have their strong points and its dataset dependant. In the end, dtw based distance had very good results overall but it has to be mentioned that it is extremely slow for big datasets, taking upwards of 1 week to find the distance matrix for sets like Wafer, UWave and CT. Our algorithm is very much built for speed, since the learnt structures are heavily restricted. The next experiment features a sneak peak of the future work, and the potential this algorithm could have in the future.

Extra Experience - What the future holds An extra experiment was performed featured in Table 4.5 in which slightly different structures were learnt. In the previous experiments all learnt structures were restricted to tDBNs with maximum of 2 parents and in-degree 2, but here we compare with a simpler structure, in which the attributes can have at most 1 parent just to check if allowing for more complex structures to be learnt had a significant impact on the clustering process.

There was not a very big difference in the values obtained. However, we can see that the overwhelming majority of the index values, is better when we permit more complex structures to be learnt. Even if it is only a small percentage of subjects get properly clustered because of this, that is still relevant. It means that with a proper study, we can optimize the parameters in order to have a better clustering algorithm, of course at the cost of time.

Table 4.5: Difference between learning structures with 1 parent, left side of the table, versus 2 parents, right side of the table. The datasets are the same as in Table 4.4 in the previous experiment.

		1 parent							2 parents						
Dataset	Dist	RI	JI	FMI	NMI	MMM	DBI	AS	RI	JI	FMI	NMI	MMM	DBI	AS
Uwave	KLD	0.781	0.107	0.193	0.120	0.286	2.343	0.237	0.782	0.102	0.185	0.125	0.276	1.832	0.266
	HD	0.770	0.096	0.177	0.076	0.259	0.834	0.589	0.736	0.102	0.189	0.113	0.272	0.814	0.556
	BD	0.784	0.096	0.176	0.089	0.266	1.988	0.166	0.789	0.104	0.188	0.119	0.286	1.931	0.219
Wafer	KLD	0.558	0.522	0.694	0.000	0.893	0.051	0.979	0.558	0.522	0.694	0.000	0.893	0.051	0.981
	HD	0.558	0.522	0.694	0.000	0.893	0.255	0.824	0.558	0.522	0.694	0.000	0.893	0.180	0.872
	BD	0.558	0.522	0.694	0.000	0.893	0.090	0.970	0.558	0.522	0.694	0.000	0.893	0.090	0.969
CT	KLD	0.896	0.077	0.146	0.262	0.253	1.922	0.280	0.900	0.086	0.160	0.310	0.305	1.503	0.257
	HD	0.640	0.065	0.185	0.159	0.183	1.050	0.714	0.903	0.104	0.190	0.295	0.306	1.482	0.365
	BD	0.882	0.068	0.132	0.190	0.199	1.560	0.410	0.899	0.085	0.159	0.301	0.291	1.565	0.313
Libras	KLD	0.861	0.092	0.173	0.322	0.288	1.384	0.314	0.876	0.091	0.169	0.331	0.311	1.454	0.324
	HD	0.868	0.082	0.154	0.299	0.269	1.032	0.580	0.883	0.086	0.159	0.324	0.294	0.839	0.599
	BD	0.871	0.082	0.153	0.299	0.286	1.999	0.176	0.874	0.084	0.157	0.324	0.291	1.896	0.189
ECG	KLD	0.498	0.358	0.528	0.000	0.665	2.170	0.353	0.508	0.370	0.541	0.006	0.665	1.433	0.353
	HD	0.500	0.355	0.525	0.003	0.665	0.698	0.656	0.581	0.472	0.642	0.073	0.705	1.260	0.257
	BD	0.882	0.068	0.132	0.190	0.199	1.560	0.410	0.524	0.394	0.565	0.016	0.665	1.611	0.227
ALS	KLD	0.525	0.407	0.579	0.013	0.673	0.669	0.578	0.531	0.414	0.586	0.017	0.673	1.051	0.580
	HD	0.521	0.400	0.571	0.011	0.673	0.596	0.578	0.521	0.400	0.571	0.010	0.673	0.600	0.575
	BD	0.521	0.400	0.571	0.017	0.673	0.596	0.578	0.536	0.429	0.601	0.018	0.673	0.493	0.665
JV	KLD	0.819	0.164	0.282	0.270	0.334	1.527	0.529	0.814	0.166	0.286	0.275	0.337	0.964	0.574
	HD	0.829	0.175	0.298	0.310	0.406	1.077	0.420	0.827	0.173	0.295	0.313	0.400	0.954	0.459
	BD	0.818	0.163	0.281	0.272	0.329	0.915	0.551	0.819	0.162	0.280	0.273	0.334	1.107	0.591

Chapter 5

Conclusions

5.1 Achievements

In this thesis we propose a method to cluster multivariate time series based on the HMM model based algorithm approach presented by [6]. Our approach uses DBNs instead, as well as an assortment of statistical distances in an attempt to improve results as well as compete with other state of the art MTS clustering algorithms. We achieved very good results, validating the use of CRATES algorithm amongst other utilized clustering algorithms for MTS, especially when working with data that can be effectively modeled by a DBN structure. We also tested the algorithm using four different statistical distances in order to see if they would have a significant impact on the final clusters. Other than the Total Variation Distance which failed to have any sort of meaningful results, the other three had a similar behavior, each with their own niche applications. They were still very similar to one another, all being capable of producing satisfactory results in the algorithm. However, this method also has limitations. The structures learnt were kept relatively simple to make up for the exponential nature of DBN learning algorithms. To deal with this NP-hard problem, the DBN structures were restricted to tDBN structures with maximum of 2 parents and in-degree of 2 which results in faster results but also worse than they could be ideally. All the code is available at "https://github.com/ZeBorges/Crates" alongside some datasets to test the algorithm.

5.2 Future Work

In the future it would be interesting to investigate more state of the art algorithms to compare results with, since it was extremely challenging to find algorithms that can successfully cluster MTS. On this note, it would also be of interest to have better datasets, in which an in depth analysis of the resulting clusters could be performed by specialists of the area, as this would bring another dimension of validation to the results of the analysis. In terms of the algorithm itself, there is also a lot of possible improvements that can be investigated to optimize the performance. Firstly, an analysis on the repercussions of allowing more complex structures to be learnt would be extremely interesting. Learning structures other than tDBN,

like cDBNs or bcDBNs would prove possibly beneficial for the results but at the cost of an increase in time spent running the algorithm. More specific tests that could be of use to enhance results and learn in which situations CRATES works best would be analyzing different Markov lags, the concept of learning non-stationary networks for long time series and how hard is it to analyze attributes that have big domains versus binary/ternary discrete attributes. Something that was also tested but relatively fruitlessly was using fuzzy clustering, which was also extremely interesting but hard to manage.

Bibliography

[1] K. Cukier and V. Mayer-Schoenberger. The rise of big data: How it's changing the way we think about the world. *Foreign Aff.*, 92:28, 2013.

[2] S. Lohr. The age of big data. *New York Times*, 11(2012), 2012.

[3] W. Raghupathi and V. Raghupathi. Big data analytics in healthcare: promise and potential. *Health information science and systems*, 2(1):3, 2014.

[4] T. Ando and J. Bai. Clustering huge number of financial time series: A panel data approach with high-dimensional predictors and factor structures. *Journal of the American Statistical Association*, 112(519):1182–1198, 2017.

[5] P. Cortez and A. d. J. R. Morais. A data mining approach to predict forest fires using meteorological data. 2007.

[6] S. Ghassempour, F. Girosi, and A. Maeder. Clustering multivariate time series using hidden Markov models. *International journal of environmental research and public health*, 11(3):2741–2763, 2014.

[7] J.-G. Lee, J. Han, and K.-Y. Whang. Trajectory clustering: a partition-and-group framework. pages 593–604. ACM, 2007.

[8] M. Vlachos, G. Kollios, and D. Gunopulos. Discovering similar multidimensional trajectories. pages 673–684. IEEE, 2002.

[9] A. F. Feder, S. Kryazhimskiy, and J. B. Plotkin. Identifying signatures of selection in genetic time series. *Genetics*, 196(2):509–522, 2014.

[10] J. Sen and T. D. Chaudhuri. Decomposition of time series data of stock markets and its implications for prediction–an application for the indian auto sector. 2016.

[11] T. W. Liao. Clustering of time series data—a survey. *Pattern recognition*, 38(11):1857–1874, 2005.

[12] D. J. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA, 1994.

[13] C. Cassisi, P. Montalto, M. Aliotta, A. Cannata, and A. Pulvirenti. Similarity measures and dimensionality reduction techniques for time series data mining. *Advances in data mining knowledge discovery and applications*, pages 71–96, 2012.

[14] B.-H. Juang. On the hidden markov model and dynamic time warping for speech recognition—a unified view. *Bell Labs Technical Journal*, 63(7):1213–1243, 1984.

[15] M. N. Do. Fast approximation of Kullback-Leibler distance for dependence trees and hidden Markov models. *IEEE signal processing letters*, 10(4):115–118, 2003.

[16] V. Perduca and G. Nuel. Exact computation of Kullback-Leibler distance for hidden Markov trees and models. *arXiv preprint arXiv:1112.3257*, 2011.

[17] R. Beran et al. Minimum hellinger distance estimates for parametric models. *The annals of Statistics*, 5(3):445–463, 1977.

[18] B. Mak and E. Barnard. Phone clustering using the bhattacharyya distance. In *Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP'96*, volume 4, pages 2005–2008. IEEE, 1996.

[19] K. Fukunaga. *Introduction to statistical pattern recognition*. Elsevier, 2013.

[20] J. Han, J. Pei, and M. Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

[21] L. Kaufman and P. J. Rousseeuw. Partitioning around medoids (program pam). *Finding groups in data: an introduction to cluster analysis*, pages 68–125, 1990.

[22] M.-S. Yang. A survey of fuzzy clustering. *Mathematical and Computer modelling*, 18(11):1–16, 1993.

[23] S. Wagner and D. Wagner. *Comparing clusterings: an overview*. Universität Karlsruhe, Fakultät für Informatik Karlsruhe, 2007.

[24] W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.

[25] L. Morey and A. Agresti. An adjustment to the rand statistic for chance agreement. *The Classification Society Bulletin*, 5:9–10, 1981.

[26] E. B. Fowlkes and C. L. Mallows. A method for comparing two hierarchical clusterings. *Journal of the American statistical association*, 78(383):553–569, 1983.

[27] S. Emmons, S. Kobourov, M. Gallant, and K. Börner. Analysis of network clustering algorithms and cluster quality metrics at scale. *PloS one*, 11(7):e0159161, 2016.

[28] A. Strehl and J. Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research*, 3(Dec):583–617, 2002.

[29] L. Ana and A. K. Jain. Robust data clustering. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 2, pages II–II. IEEE, 2003.

[30] P. J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[31] S. Petrovic. A comparison between the silhouette index and the davies-bouldin index in labelling ids clusters. In *Proceedings of the 11th Nordic Workshop of Secure IT Systems*, pages 53–64. Citeseer, 2006.

[32] L. Rabiner and B. Juang. An introduction to hidden Markov models. *ieee assp magazine*, 3(1):4–16, 1986.

[33] Y.-L. Lin and G. Wei. Speech emotion recognition based on HMM and SVM. In *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, volume 8, pages 4898–4901. IEEE, 2005.

[34] P. Smyth. Clustering sequences with hidden Markov models. In *Advances in neural information processing systems*, pages 648–654, 1997.

[35] J. Pearl. Probabilistic reasoning in intelligent systems: networks of plausible inference. 1988.

[36] M. Sousa and A. M. Carvalho. Learning consistent tree-augmented dynamic bayesian networks. In *International Conference on Machine Learning, Optimization, and Data Science*, pages 179–190. Springer, 2018.

[37] G. F. Cooper. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial intelligence*, 42(2-3):393–405, 1990.

[38] P. Dagum and M. Luby. Approximating probabilistic inference in bayesian belief networks is np-hard. *Artificial intelligence*, 60(1):141–153, 1993.

[39] D. M. Chickering. Learning bayesian networks is np-complete. In *Learning from data*, pages 121–130. Springer, 1996.

[40] D. Heckerman, D. Geiger, and D. M. Chickering. Learning bayesian networks: The combination of knowledge and statistical data. *Machine learning*, 20(3):197–243, 1995.

[41] C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE transactions on Information Theory*, 14(3):462–467, 1968.

[42] J. L. Monteiro, S. Vinga, and A. M. Carvalho. Polynomial-time algorithm for learning optimal tree-augmented dynamic bayesian networks.

[43] I. Visser and M. Speekenbrink. depmixS4: An R package for hidden Markov models. *Journal of Statistical Software*, 36(7):1–21, 2010. URL `http://www.jstatsoft.org/v36/i07/`.

[44] M. Maechler, P. Rousseeuw, A. Struyf, M. Hubert, and K. Hornik. *cluster: Cluster Analysis Basics and Extensions*, 2019. R package version 2.1.0 — For new features, see the 'Changelog' file (in the package source).

[45] J. Liu, L. Zhong, J. Wickramasuriya, and V. Vasudevan. uwave: Accelerometer-based personalized gesture recognition and its applications. *Pervasive and Mobile Computing*, 5(6):657–675, 2009.

[46] R. T. Olszewski. Generalized feature extraction for structural pattern recognition in time-series data. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA SCHOOL OF COMPUTER SCIENCE, 2001.

[47] D. Dua and C. Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

[48] H. A. Dau, E. Keogh, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, Yanping, B. Hu, N. Begum, A. Bagnall, A. Mueen, G. Batista, and Hexagon-ML. The ucr time series classification archive, October 2018. `https://www.cs.ucr.edu/~eamonn/time_series_data_2018/`.

[49] A. Sardá-Espinosa. Time-series clustering in r using the dtwclust package. *The R Journal*, 2019. doi: 10.32614/RJ-2019-023.

[50] M. Cuturi. Fast global alignment kernels. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 929–936, 2011.