Clustering multivariate time series using dynamic Bayesian networks

José Pedro Borges
jose.v.borges@tecnico.ulisboa.pt

Alexandra M. Carvalho
alexandra.carvalho@tecnico.ulisboa.pt

Susana Vinga
susanavinga@tecnico.ulisboa.pt

Instituto Superior Técnico
October 2020

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

1. Introduction

We live in an era where access to information is easier than ever [4]. There is a massive amount of new data made available across multiple knowledge areas such as biomedicine, economics and meteorology every day. This information corresponds to measurements taken over time on a set of variables, commonly denoted as multivariate time series. Working with multivariate time series brings additional layers of complexity, specifically when clustering. Clustering [1] consists of grouping objects based on their features. The algorithms are divided into partition based, model-based, density-based or grid-based according to the employed strategy [11]. All these algorithms are closely related to the concept of distance. A distance measure belongs to one of three main types, depending on the kind of similarity measure used. It can be a comparison in shape such as dynamic time warping [3], in change with models such as hidden markov models [22] or in time like the euclidean distance. Since we are working with multivariate time series measuring a distance is a challenge. An ingenious way of dealing with it is through a model-based distance approach that, instead of using directly a metric over the observations, used a similarity between the inferred models [10]. The original algorithm Ghassempour proposed made use of the fact that there are well defined distances between models. They decided to use HMM's [22] to represent their time series and computed the Kullback–Leibler divergence on the likelihoods of each HMM generating each original time series. Our approach is similar other than a few key differences with the first one being the model. We propose the use of Dynamic Bayesian Networks, DBN, instead of HMM. Since HMM are a particular case of DBN [18], we expect improved results since DBN's are able to represent variable dependencies over time. We will also experiment with two extra statistical distances other than the KLD, in an attempt to study the effects of each metric on the results for clustering.

2. Background

2.1. Temporal modelling

A Bayesian network is a graphical model that encodes the joint probability distribution of a set of n random variables [20]. Let $\mathbf{X} = (X_1, \ldots, X_n)$ denote a discrete random vector

where each variable X_i takes values over a finite set $\mathcal{X}_i = \{x_{i1}, \ldots, x_{ir_i}\}$. Moreover, let x_{ik} denotes the k-th value X_i takes.

A Bayesian network (BN) is defined by a pair $B = (G, \Theta)$, where G regards the network structure, and Θ the network parameters. The structure $G = (V, E)$ is a directed acyclic graph (DAG) with vertices V and edges E. Each vertex (node) corresponds to one of the random variables X_i, and the edges represent direct dependencies between the variables. The parameters $\Theta = \{\Theta_{ijk}\}_{i\in\{1,\ldots,n\},j\in\{1,\ldots,q_i\},k\in\{1,\ldots,r_i\}}$ encode conditional probability tables (CPT):

$$\Theta_{ijk} = P_B(X_i = x_{ik}|\Pi_{X_i} = w_{ij}), \tag{1}$$

where Π_{X_i} denotes the set of parents of X_i in G, w_{ij} is the j-th configuration of Π_{X_i}, among all possible configurations given by $\{w_{i1}, \ldots, w_{iq_i}\}$, and $q_i = \prod_{X_j\in\Pi_{X_i}} r_j$ is the total number of parent configurations.

A Bayesian network B induces a unique joint probability distribution over $\mathbf{X}$ given by:

$$P_B(X_1, \ldots, X_n) = \prod_{i=1}^{n} P_B(X_i|\Pi_{X_i}). \tag{2}$$

Having said so, the graph of a BN can be viewed as a network structure that provides the skeleton for representing the joint probability compactly in a factorized way.

An example of a BN (taken from [25]) is depicted in Figure 1. 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 dependencies encoded by the graph, the joint probability distribution of the network can be factored as

$$P(M, S, F, O, C) = P(M)P(S)P(F|M, S)P(O|F)P(C|F, S), \tag{3}$$

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.

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

$$P(\mathbf{X}[0:t]) = P(\mathbf{X}[0]) \prod_{t=0}^{T-1} P(\mathbf{X}[t+1] \mid \mathbf{X}[0:t]). \tag{4}$$

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. (4) can be simplified as:

$$P(\mathbf{X}[t+1] \mid \mathbf{X}[0:t]) = P(\mathbf{X}[t+1] \mid \mathbf{X}[t-m+1:t]). \tag{5}$$

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

$$P(\mathbf{X}[t+1] \mid \mathbf{X}[t-m+1:t]) = P(\mathbf{X}[t+z] \mid \mathbf{X}[t+z-m+1:t+z]), \tag{6}$$

for all time slices.

For this study we decided to use Dynamic Bayesian Networks as the chosen models since they are optimal to represent dependencies between variables which would allow us to extract enough useful information from a single time series to train a complex model. In order to find a suitable DBN structure, a scoring function is applied. Given a dataset $D = y_1, ..., y_N$, where $y_t = (y_{t1}, ..., y_{tN})$, for $1 \leqslant t \leqslant N$. $y_t i \epsilon D_i$, for all $1 \leqslant t \leqslant N$ and $1 \leqslant i \leqslant n$, and a scoring function ϕ, the problem of learning a BN turns into finding a BN B that maximizes the value of $\phi(B, D)$

To help learn a complex structure with limited information the score function used to learn our networks was the Log-Likelihood [12].

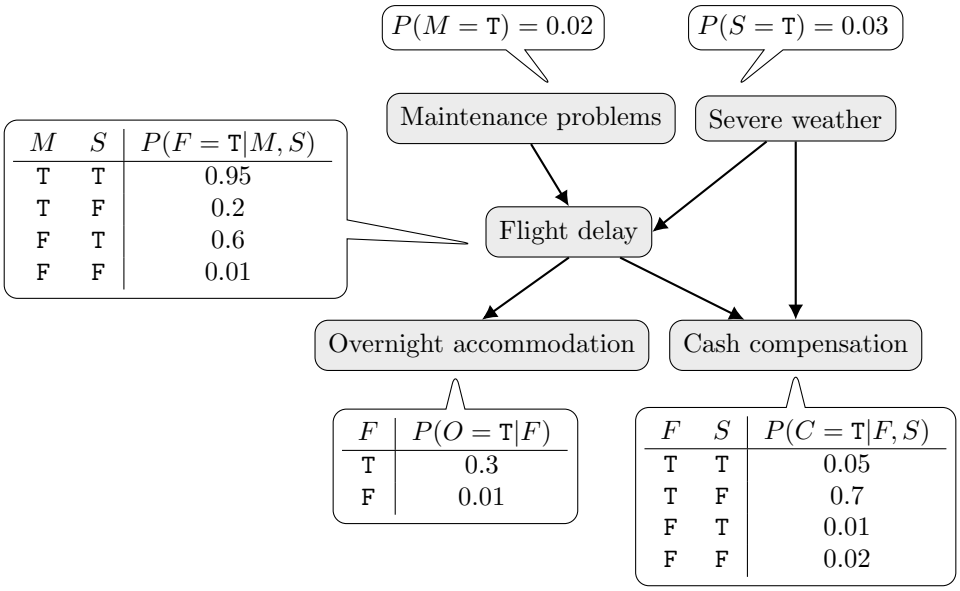

Figure 1: A BN example regarding airline regulations with conditional probability tables.

The expression is

$$LL(B|T) = \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk} \log(\frac{N_{ijk}}{N_{ij}}). \quad (7)$$

This score is usually connected to the concept of overfitting, however we benefit from a bit of overfitting since we want our model to reflect all possible intricacies present in our time series.

2.2. Distances

The comparison between DBN models can be performed through model-based distances, such as the Kullback-Leibler, the Hellinger, and the Bhattacharyya distance. The KL-distance showed promising results for HMMs [10]. 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:

$$KLD(P||Q) = \int_{-\infty}^{\infty} p(x) \times \log \frac{p(x)}{q(x)} dx. \quad (8)$$

Approaches to simplify the expression are given in [7]. 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 [10]. It is also important to note that this distance is not symmetric, so a symmetrization process is also required.

The Hellinger Distance [2] and the Bhattacharyya Distance [16], [9], are strong candidates when working with statistical models. The Hellinger distance is defined for two discrete distributions, P and Q, as:

$$HD(P, Q) = \frac{1}{\sqrt{2}} \sqrt{\sum_{i=1}^{k} (\sqrt{p(i)} - \sqrt{q(i)})^2}. \quad (9)$$

Lastly, the Bhattacharyya Distance's expression for two discrete distributions is the following

$$BD(P, Q) = -\log(\sum_{i=1}^{k} \sqrt{p(i)q(i)}). \quad (10)$$

2.3. Time-series Clustering Algorithms and Validation

A standard approach based partitioning clustering was made with the distance matrixes computed.

The idea was to cluster the data using the Partition Around Medoids algorithm [13] followed by cluster validation indexes as a form of legitimizing the results. We decided to use both intrinsic [23] and extrinsic methods [26], since we have access to previously cassified datasets. To have some redundancy, we used an array of both types of indexes, in hopes of achieving concordant results.

As for *clustering algorithms*, we applied partitioning methods such as the k-means and partition around medoids. Originally the plan was to use only PAM since it was used successfully by [10], but we decided to incorporate the k-means since it is very common and we expect it to bring coherent results since they are similar.

We used *extrinsic methods* present in an overview about comparing clusters [26] including Rand, Fowlkes-Mallows index, Jackard Index, Normalized Mutual Information, Maximum-Match-Measure. It

was important to experiment with a wide variety of metrics since clustering is a difficult task and we want to measure our algorithm's coherence as best as possible.

In terms of *intrinsic methods* we used Silhouette to measure how well matched models are to their cluster and Davies–Bouldin Index [21] to measure the ratio between intra-cluster sparsity and inter-cluster separation. The intrinsic validation methods will also be used to determine the optimal number of clusters, this will be discussed further when explaining the method.

3. Implementation

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 X_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.

CRATE Procedure The name CRATES comes from Clustering Multivariate Networks. First the data is divided into each individual database entry, $(x_1, \ldots, x_n)$. For each of these, a Tree-augmented DBN (tDBN) structure is learnt using algorithm [17].

Once the list of DBN's is successfully obtained, the next step is finding how similar two networks are. To do this we need to compute a distance, much like the previous algorithm computed the KLD. In our case either the KLD, the HD or the BD can be computed. The way that Ghassempour was able to compute the KLD went through finding how likely it was for an HMM to generate the most likely set of observations from all the other HMMs. This cannot be done with DBNs, but we can do something similar. In this case we simply take the set of observations used to train the DBN and check how likely it is for other DBNs to generate those observations. The intuiton behind this is that two similar DBNs would be likely to generate the same observations.

Algorithm 1 CRATES

Input : X: Set of network attributes
D: Dataset of MTS
ϕ: Decomposeable scoring function
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
```

4. Results

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 × 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

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 2. Each multivariate time series generated had 5 different attributes and 20 time steps.

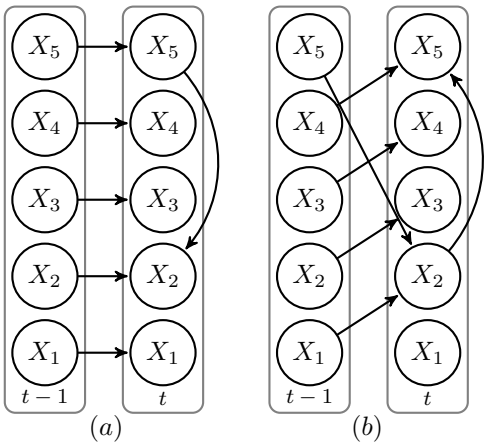

Figure 2: Transition networks of the two generated DBNs to perform the first experiment.

As we can see in Table 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 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 3, we can see represented as (b) one of 50 learnt structures for network (a) in Figures 2 and 3. 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.

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 Ex-

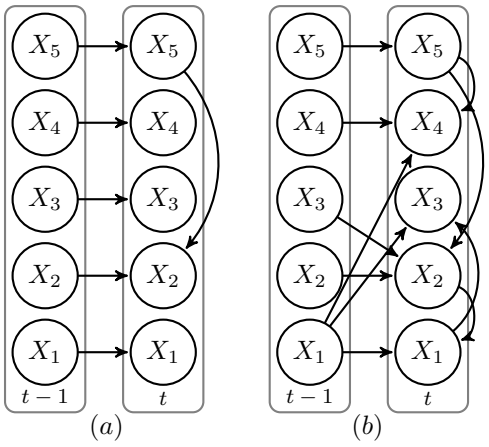

Figure 3: A way to depict the difference between the DBN from which the data was generated (a) and 1 out of the 50 learnt structures that closely resemble the original (b).

perience 1, again generating 50 subjects from each network in Figure 2, but now changing the properties of the time series. Instead of only inspecting what happens when the MTS have 5 attributes and 20 time steps, since we already know that results in perfect clustering, in Table 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 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

Table 1: First experiment on simulated data using the networks present in Figure 2. 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

Table 2: In depth test on simulated data using the two networks represented in Figure 2 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

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 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 3.

The results in Table 3 are worst than in Table 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.

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 3.

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

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 the following:

- Uwave(4478 individuals, 3 attributes, 99 time steps, 8 possible classes) [14];
- Wafer(1194 individuals, 6 attributes, 99 time steps, 2 possible classes) [19];
- CT(2858 individuals, 3 attributes, 100 time steps, 20 possible classes) [8];
- Libras(360 individuals, 2 attributes, 45 time steps, 15 possible classes) [8];

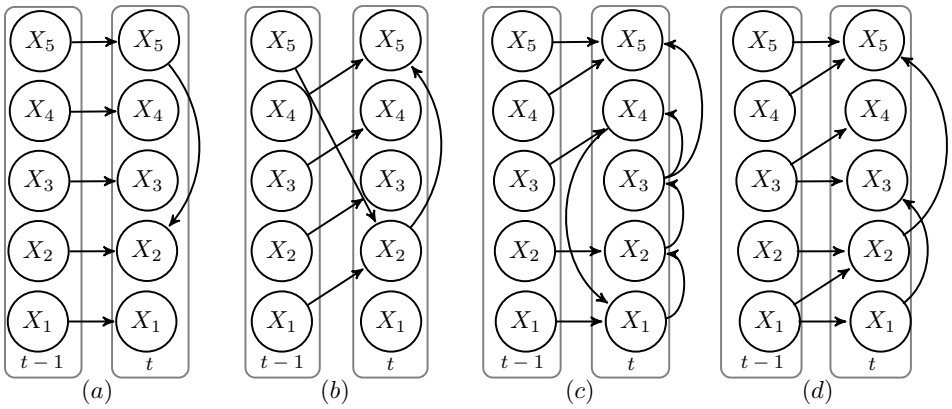

Figure 4: Transition networks of stationary first-order DBNs used to generate synthetic data to test the efficiency of the algorithm before using real data. All networks are represented with 5 features.

Table 3: In depth test on simulated data using the four networks represented in Figure 4 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

- ECG(200 individuals, 2 attributes, 39 time steps, 2 possible classes) [6];
- 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) [8];

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 [24] that uses the Triangular Global Alignment Kernel (TGAK) [5], 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 [15] and evaluating with the chosen indexes.

The reason we did not compare the results with the original presented by Ghassempour [10], was simply because we couldn't. The way the algorithm was programmed required an *a priori* knowledge 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.

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 [19] 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.

5. Conclusions

In this thesis we propose a method to cluster multivariate time series based on the HMM model based algorithm approach presented by [10]. 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.

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.

Acknowledgements

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

Table 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

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).

References

[1] S. Aghabozorgi, A. S. Shirkhorshidi, and T. Y. Wah. Time-series clustering–a decade review. *Information Systems*, 53:16–38, 2015.

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

[3] 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.

[4] 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.

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

[6] 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/`.

[7] 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.

[8] D. Dua and C. Graff. UCI machine learning repository, 2017.

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

[10] 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.

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

[12] 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.

[13] L. Kaufman and P. J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*, volume 344. John Wiley & Sons, 2009.

[14] 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.

[15] 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).

[16] 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.

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

[18] K. P. Murphy and S. Russell. Dynamic Bayesian networks: representation, inference and learning. 2002.

[19] R. T. Olszewski. Generalized feature extraction for structural pattern recognition in time-series data. Technical report, Carnegie-Mellon University, 2001.

[20] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 2014.

[21] 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.

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

[23] S. Saitta, B. Raphael, and I. F. Smith. A bounded index for cluster validity. In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pages 174–187. Springer, 2007.

[24] A. Sardá-Espinosa. Time-series clustering in r using the dtwclust package. *The R Journal*, 2019.

[25] 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.

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