Abstract—Multivariate time series is a very active topic in the research community and many machine learning tasks are being used in order to extract information from this type of data. However, in real-world problems, the data has missing values and may lack prior information about the groups, which may difficult the application of machine learning techniques to extract information. Hence, this thesis will focus on the tasks of imputation and clustering.

Many imputation methods of time series are based on regression methods; however, these type of methods cannot capture the information between the variables of multivariate categorical time series. As such this thesis proposes a new imputation method that uses the Dynamic Bayesian Networks.

The task of clustering tries to group similar time series. State of the art methods can be divided in two types: the ones that use a custom distance and then use a traditional clustering method like k-means; and the ones that use models to cluster similar time series. However, most of these methods do not take into consideration the relationship between variables. As such, this thesis proposes a new clustering method based on Dynamic Bayesian Multinets to cluster similar time series.

The devised methods are assessed in synthetic data, showing that it outperforms the state of the art methods, in both tasks. Further experimental tests are done using real datasets, where it is shown that the proposed methods are more consistent than the state of the art methods in their tasks.

I. INTRODUCTION

Nowadays the world is full of data, due to better sensors, fast internet and more computational power to storage all of that data. This data is very useful to extract information about the world but to do that we need algorithms that are capable of finding features that could bring useful information. However, this is a non-trivial task and to solve this problem the field of machine learning emerged. The term data mining appear to describe the problem of finding useful information in large data sets by integrating methods from many fields like machine learning, statistics and database systems, spatial or temporal data analysis, pattern recognition, and image and signal processing.

In recent years many works have been done to use machine learning techniques in order to extract useful information from data. For example, Lee et al. [17] try to find patterns in hurricanes trajectories in order to better forecast the location of a hurricane landfall. Moreover, these techniques find applications in various areas that range from economics to sport. Finally, Zhao et al. [27] introduced a novel representation of temporal data in electronic health records with the objective of improving the prediction tasks in the biomedical domain.

The main objective of this thesis is the analysis of multivariate time series, in particularity the imputation of missing values and the task of clustering. To do these tasks it will be used Dynamic Bayesian Networks which can represent in a compact way relations between random variables (Pearl [20]). As such this thesis follows the work of Monteiro et al. [19] and Sousa and Carvalho [23] by extending their models. The proposed methods are implemented and assessed in synthetic and real data. The real datasets are from UCI Machine Learning Repository [6] and UCR Time Series Classification Archive [3].

II. THEORETICAL BACKGROUND
A. Bayesian Networks

Let’s assume that, in what follows, all random variables are discrete and have a finite domain. So let $X$ be a random variable that takes values over the finite set $\mathcal{X}$. Moreover, let $X = (X_1, ..., X_n)$ be a n-dimensional random vector, where each $X_i$ takes values in $Val(X_i) = \{x_{i1}, ..., x_{ir_i}\}$, being $r_i$ the number of values that $X_i$ can take. Pearl [20] describes a Bayesian Network (BN) as being an annotated Directed Acyclic Graph (DAG) that encodes a joint probability distribution over a set of random variables.

Definition 1. A n-dimensional Bayesian Network (BN) is a triple $B = (\mathcal{X}, G, \Theta)$ where:

- $\mathcal{X}$ is an n-dimentional finite random vector where each random variable $X_i$ ranges over a finite domain $D_i$. Henceforward it is denoted the joint domain by $D = \prod_{i=1}^{n} D_i$.
- $G = (N, E)$ is a DAG with nodes $N = \{X_1, ..., X_n\}$ and edges $E$ representing direct dependencies between variables.
- $\Theta$ encodes the parameters $\{\theta_{ijk}\}_{i \in \{1,...,n\}, j \in D_i, k \in D_i}$ of a network, given by:

$$\theta_{ijk} = P_B(X_i = x_{ik} | \Pi_{X_i} = w_{ij})$$

where $\Pi_{X_i}$ denotes the set of parents of the node $X_i$ in the DAG $G$, $x_{ik}$ is the k-th values of $X_i$ and $w_{ij}$ is the j-th configuration of $\Pi_{X_i}$. Moreover, let $q_i$ denote the number of total configurations of the parents of the node $X_i$, $q_i = \prod_{X_j \in \pi_i} r_j$.

A BN B defines the joint distribution over $\mathcal{X}$:

$$P_B(X_1, ..., X_n) = \prod_{i=1}^{n} P_B(X_i \mid \Pi_{X_i}).$$ (1)
So given the previous definition it can be said that the DAG G encodes independence assumptions, where each random variable $X_i$ is only dependent of the descendants nodes and independent of its non-descendants, given its parents. The task of learning BNs can be reduced to the problem of finding the structure $G$ and the parameters $\Theta$ such as the BN $B = (X, G, \Theta)$ best matches the dataset $D$. Assuming that the data $D$ is complete, i.e., there is no missing values and hidden variables, and that is given by a set of $N$ i.i.d. instances, $D = \{x_1, ..., x_N\}$. There are many learning methods but the approach that will be discussed in this thesis is the score-based learning. It consists of defining a scoring function, $\phi$, that measures how well the network $B$ describes the dataset $D$, and a search procedure (Heckerman et al. [13]). As such the learning task can be stated as an optimization problem:

$$\max_{B \in \mathcal{B}_n} \phi(B, D),$$

where $\mathcal{B}_n$ corresponds to the set of all the BNs with $n$ variables.

B. Dynamic Bayesian Networks

Dynamic Bayesian Network (DBN) extends the representation of BN to temporal processes. So let assume, for simplicity, the discretization of time in time slices $\{0, ..., T\}$. Let $X[t] = (X_1[t], ..., X_n[t])$ be a random vector that denotes the values of the set of random variables $X$ over the time $t$. Moreover let $X[t_1 : t_2]$ denote the set of random vectors $X[t]$ over $t_1 \leq t \leq t_2$. Finally let the joint probability distribution over the trajectory of a process from $X[0]$ to $X[T]$, $P(X[0], ..., X[T])$, be abbreviated with $P(X[0 : T])$, and given by:

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

This is very complex to compute, so in order to simplify these computations one common approach is to assume that the process is Markovian in $X$.

Definition 2. A stochastic process is said to satisfy the $m$-th order Markov assumption if, for all $t \geq 0$:

$$P(X[t+1] | X[0 : t]) = P(X[t+1] | X[t-m+1 : t]),$$

where in this case $m$ is called the Markov lag of the process.

Moreover, another assumption that simplifies the computation of the joint probability distribution is to consider the process to be stationary. This assumption is adequate when the training data is small.

Definition 3. A stochastic process is said to be stationary if:

$$P(X[t+1] | X[t])$$

is equal for all time slices $t \in \{0, ..., T-1\}$.

a) Non-Stationary first-order Markov DBN: The representation of this model over $T$ time-slices consists of:

- A prior network $B^0$, which specifies the distribution over the initial states $X[0];$
- A set of transition networks $B_i^{t+1}$ over the variables $X[t] \cup X[t+1]$ that specifies the transition probability $P(X[t+1] | X[t]),$ for all $t$.

To be noted that a stationary first-order Markov DBN has only one prior network and one transition network that is the same for all transitions, this can be seen in Figure 1. By looking at the “unrolled” network, Figure 1b it is important to highlight that the transition networks encode two type of dependencies. The inter-slices dependencies that connect variables from one time-slice to the next one, and the intra-slices dependencies that connect variables from the same time-slice.

Figure 1: Example of a stationary first-order Markov DBN.

1) Learning Dynamic Bayesian Networks: Learning DBNs can be divided into two categories, the fully observed process and the partially observed case. The first one occurs when there are no hidden variables that interact with the observed ones and there are no missing values in the observed data. In this case, the problem of learning a DBN can be simplified by applying the methods of learning BNs to each transition of time (Friedman et al. [10]). In the second case, many problems arise, for example, the scoring functions no longer are decomposable. In order to solve these problems Friedman et al. [9] proposed the Structural Expectation-Maximization (SEM) iterative method to learn a BN when hidden variable and/or missing values are present in the data, a description of this method can be seen in Section III. Focusing on the first case, Vinh et al. [26] proposed a polynomial-time algorithm that learns an optimal DBN using mutual information tests. More recently, Monteiro et al. [19] proposed a polynomial-time algorithm that learns the intra-slices and the inter-slices connections of a transition network. To do that the search space for the intra-slice connections is restricted to tree augmented networks, i.e, acyclic networks that each variable has only one parent from the same time slice, but can have a finite number of parents from the previous time slices. The resultant network is denoted by tDBN, where the letter t reflects the
search space considered. Recently, Sousa and Carvalho \[25\] extended this latter method by increasing the search space to networks where the intra-slice network has in-degree at most \(\kappa\) and is consistent with the Breadth-First Search (BFS) order of the DBN, and the inter-slice network has in-degree at most \(p\). The resultant network of this method is denoted by bcDBN.

C. Bayesian Multinets

Although BNs are extremely useful, allowing a wide spectrum of independence assumptions to be encoded by its model, they suffer from one deficiency, their inability to represent some values but not necessary for all of their values. In order to solve this deficiency Geiger and Heckerman \[11\] introduces the Bayesian Multinets (BM).

Definition 4. Let \(P(C, X_1, \ldots, X_n)\) be a probability distribution and \(A_i\) be a subset of the values of \(C\) and the event \(\{A_i\}\) stand for “one of the hypothesis in \(A_i\) hold true”. A DAG \(G_i\) is a comprehensive local network of \(P\) associated with \(A_i\), if \(G_i\) is a BN of \(P(C, X_1, \ldots, X_n | \{A_i\})\). The \(c\) set of local networks is called a Bayesian Multinet (BM) of \(P\). When each \(A_i\) is a singleton, the resulting BM is hypothesis-specific.

Later, Friedman et al. \[9\] defines a Bayesian classifier that is based on BMs. The idea behind this classifier was the generalization of the Tree-Augmented Naive (TAN) Bayesian classifier. In the latter classifier the relations between variables are forced to be the same for all classes, whereas in the BMs augmenting edges are learned for each class, in this case a tree structure. To implement this classifier first the data set is divided by classes, then for each class \(c_i \in Val(C)\), where \(C\) is the variable that represents the class/label, a BN \(B_i\) for the attributes \(\{X_1, \ldots, X_n\}\) is learned. The resulting probability distribution \(P_{B_i}(X_1, \ldots, X_n)\) approximates the joint distribution of the attributes for a given class, \(P_D(X_1, \ldots, X_n | C = c_i)\). Finally a multinet is formally a tuple \(M = (P_C, B_1, \ldots, B_k)\), where the \(P_C\) is the distribution on \(C\), and \(B_i\) is a BN over all the variables for \(1 \leq i \leq c = Val(C)\).

A multinet \(M\) defines a joint distribution:

\[
P_M(C, X_1, \ldots, X_n) = P(C) \cdot P_{B_i}(X_1, \ldots, X_n) \quad \text{when} \quad C = c_i.
\]

(6)

To be noted in the process of training the classifier \(P(C)\) is set to be the frequency of each class in the training data \(P_D(C)\). Finally, the result of the classification task is the class that maximizes the posterior probability \(P_M(C | X_1, \ldots, X_n)\).

Another use to the BMs is to perform the task of clustering. In order to do that Pham and Ruz \[21\] uses the Classification Maximum Likelihood (CML) criterion, where the maximization of this criterion is derived under the Classification Expectation Maximization (CEM). First, it is assumed that the data is generated by a mixture of \(H\) BN, that can be described by:

\[
\begin{align*}
P(X) &= \sum_{h=1}^{H} \alpha_h f_h(X) \\
\sum_{h=1}^{H} \alpha_h &= 1, \quad \alpha_h \geq 0
\end{align*}
\]

(7)

where \(\alpha_h\) are the mixing coefficients and \(f_h\) are the mixing components distributions defined by the BNs. So, to perform the clustering task the mixture parameters \(\theta\), composed by the mixing coefficients and the mixing distributions, and the indicator vector \(z_l = (z_l^h, h = 1, \ldots, H)\), where \(z_l^h = 1\) if \(x_l(1 \leq l \leq N)\) belongs to the \(h\)-th component, will be chosen in order to maximize the CML (Celeux and Govaert \[2\]):

\[
C_{ML}(\theta, z_1, \cdots, z_N | x_1, \cdots, x_N) = \sum_{h=1}^{H} \sum_{x_l \in P_h} \log f_h(x_l) + n_h \log \alpha_h,
\]

(8)

where \(P_h\) corresponds to the partitions (clusters) of the data examples \(x_1, \cdots, x_N\) where the indicator values \(z_1, \cdots, z_N : P_h(x_l | z_l^h = 1)\) and \(n_h = \#P_h\). This equation can be optimize using the CEM algorithm, by Celeux and Govaert \[1\]. The outline of CEM optimization follows the following steps.

a) Expectation (E-Step): Compute, for \(l = 1, \ldots, N\) and \(h = 1, \ldots, H\), the posterior probability of \(x_l\) belongs to \(P_h\), which is given by:

\[
t_h(x_l) = \frac{\alpha_h \sum_{l=1}^{n} P_h(x_l | \Pi_{x_l})}{\sum_{h=1}^{H} \alpha_h \sum_{i=1}^{n} P_h(x_l) | \Pi_{x_l}}
\]

(9)

b) Classification (C-Step): Assign \(x_l\) to the cluster that provides the maximum posterior, by creating a new partition \(P'\).

c) Maximization (M-Step): For all clusters compute the maximum likelihood estimates of \(\theta\) using the clusters \(P'\) as subsamples. By doing this it is ensured that the CML criteria is maximized. The eq. \[8\] can be rewritten as:

\[
C_{ML} = \sum_{h=1}^{H} \sum_{x_l \in P_h} \log \prod_{i=1}^{n} P_h(x_l | \Pi_{x_l}) + n_h \log \alpha_h,
\]

(10)

where the mixture parameters are disjointed, so each sum of the r.h.s of the equation can be maximized independently. The maximization of the first term of the r.h.s of the eq. \[10\] will result in a new BN for each \(P_h\), this BN is learned by maximizing the Log-Likelihood (LL). Taking into account the constraint:

\[
\sum_{h=1}^{H} \alpha_h = 1, \quad \alpha_h \geq 0,
\]

(11)

the maximization of the second term yields:

\[
\alpha'_h = \frac{n_h}{N} = \#P_h' \frac{1}{N}.
\]

(12)

Moreover, Celeux and Govaert \[1\] denotes that the initial iterations of the CEM algorithm are dependent on the initial partition, in which it may cause convergence to local optima. A way of reducing this problem is to replace the C-Step with a stochastic step, S-Step. In the S-Step, each \(x_l\) is assigned at random to one of the clusters \(P_1, \ldots, P_H\) with probability \(t_h(x_l)\).
III. STRUCTURAL EM

In many cases, the assumption that the training data are fully observed is simply unrealistic. Since this assumption is crucial for learning the structure and the parameters of a BN some changes to the learning process need to be made.

A. Parameter Estimation

The first learning task that will be considered is the parameter estimation task. As in the case of the complete data, the approach that will be used is the Maximum Likelihood Estimation (MLE). So given a network structure \( G \), and the form of the Conditional Probability Distributions (CPDs), it is only necessary to compute the parameters \( \Theta \) to define the distribution \( P(X \mid \Theta) \). It is also given a data set \( D \) that consists of \( M \) partial instances of \( X \), so it is needed to compute the values \( \Theta \) that maximize the log-likelihood function:

\[
\hat{\Theta} = \arg \max_{\Theta} \log L(\Theta : D).
\]

However, as seen in Koller and Friedman [15], in the presence of incomplete data, the likelihood does not decompose. So, it is required to optimize a highly non-linear and multimodal function over a high-dimensional space. To tackle this problem it will be used a specialized approach to optimize the likelihood function like the Expectation Maximization (EM).

Unlike the complete data case, where sufficient statistics are collected for each CPD to then compute the parameters that maximize the likelihood with respect to these statistics, in the case of missing data there is no access to the full sufficient statistics. In order to have access to them, one can take a simple approach of filling the missing values arbitrarily. Some strategies to fill these missing values are using some default value or choosing a random value. The problem with this approach is that the filled value will introduce bias in the learned parameters. Another approach tries to solve two different problems at once, these problems being the learning of the parameters and imputation of the missing values. To be noted that each of these tasks is very easy when the solution to the other is present.

Using the EM algorithm this “chicken and egg” problem can be solved. It starts by choosing some arbitrary starting point, this can be either a choice of parameters or some assignment to the missing values, assuming that it begins with a parameter assignment, then the algorithms repeat two steps. The first step is to use the current parameters in order to complete the data, using probabilistic inference. The second step is to use the completed data as if it were observed data and compute the new set of parameters. So, given a set of parameters \( \Theta \) and a partial instance, the posterior of all possible assignments to the missing value of that instance can be calculated. The EM algorithm then uses this probabilistic completion of different data instances to estimate the expected value of the sufficient statistics. In Koller and Friedman [15] it is demonstrated that each iteration of this method increases the log-likelihood, that this process converges and that the convergence point is always a local maximum of the likelihood function.

Next, detailed explanation of the EM algorithm is given. Assuming the general BN with table-CPDs, an initial assignment for the parameters \( \Theta^0 \). \( X \) being all the child variables, \( W \) all the parent variables and \( o \) being the dataset composed by \( M \) data instances. The algorithm iterates over the following steps.

1) **Expectation (E-Step):** In this step the Expected Sufficient Statistics (ESS) are computed, this is done by using the current parameters \( \Theta^t \).
   - For each family \( X, W \) and for each data case \( o[m] \), compute the joint probability \( P(X, W \mid o[m], \Theta^t) \).
   - Compute the ESS for each \( x, w \),

\[
M_{\theta}^t[x, w] = \sum_{m} P(x, w \mid o[m], \theta^t).
\]

2) **Maximization (M-Step):** Given the ESS, it performs maximum likelihood estimation, with respect to them, in order to compute a new set of parameters,

\[
\theta^{t+1} \mid o = \frac{M_{\theta}^t[x, w]}{M_{\theta}^t[\cdot, w]}.
\]

In Algorithm 1 and Algorithm 2 the E-step and for the Parameter EM are given.

**Algorithm 1 Compute the expected sufficient statistics**

1: procedure Compute-ESS(\( G, \Theta, D \))
2: for each \( i = 1, \cdots, n \) do \( \triangleright \) Initialization of data structures
3: for each \( x_i, w_i \in Val(X_i, \Pi_{X_i}^G) \) do
4: \( M_i[x_i, w_i] \leftarrow 0 \)
5: for each \( m = 1, \cdots, M \) do \( \triangleright \) Collect probabilities from all instances
6: Run inference on \( \langle G, \Theta \rangle \) using evidence \( o[m] \)
7: for each \( i = 1, \cdots, n \) do
8: for each \( x_i, w_i \in Val(X_i, \Pi_{X_i}^G) \) do
9: \( M_i[x_i, w_i] \leftarrow M_i[x_i, w_i] + P(x_i, w_i \mid o[m]) \)
10: return \( \{M_i[x_i, w_i] \mid \forall i = 1, \ldots, n, \forall x_i, w_i \in Val(X_i, \Pi_{X_i}^G)\} \)

**Algorithm 2 Expectation-Maximization algorithm for Bayesian Network(using table-CPDs)**

1: procedure Expectation-Maximization(\( G, \Theta^0, D \))
2: for each \( t = 0, \cdots, \) until convergence do \( \triangleright \) E Step
3: \( \{M_i[x_i, w_i] \} \leftarrow \) Compute-ESS(\( G, \Theta^t, D \)) \( \triangleright \) E Step
4: for each \( i = 1, \cdots, n \) do \( \triangleright \) M Step
5: for each \( x_i, w_i \in Val(X_i, \Pi_{X_i}^G) \) do
6: \( \theta^{t+1} \mid x_i, w_i \leftarrow \frac{M_i[x_i, w_i]}{M_{\theta}^t[\cdot, w]} \)
7: return \( \Theta^t \)
B. Structure Learning

The intuition behind Structural EM algorithm is the same that was applied to solve the problem of learning the parameters of a BN when there is missing data. Like the parameter estimation, there is two main steps, the expectation, where a complete data set is generated, and a maximization, where the network structure is learned. The main difference between the Structural Expectation-Maximization (SEM) and the parameter estimation is that the maximization step, in the Structural Expectation-Maximization (SEM), besides learning the parameters, the network structure is also learned. Moreover, Koller and Friedman [16] state that by using the Minimum Description Length (MDL) score it is guaranteed that, in each iteration, the learned structure is better than the one used in the previous iteration. From this statement, it results that the SEM algorithm will monotonically improve the score. The code of the algorithm is given in Algorithm 3

Algorithm 3 Structural EM algorithm for Bayesian Networks

1: procedure STRUCTURAL-EM($G^0$, $\Theta^0$, $D$)
2: for each $t = 0, \ldots$, until convergence do
3: $\Theta^t \leftarrow$ Expectation-Maximization($G^t$, $\Theta^t$, $D$) $\triangleright$ Optional parameter learning step
4: $G^{t+1} \leftarrow$ Structure-Learn($D^*_G$, $\Theta^t$) $\triangleright$ Run EM to generate the ESS for $D^*_G$, $\Theta^t$
5: $\Theta^{t+1} \leftarrow$ Estimate-Parameters($D^*_G$, $\Theta^t$, $G^{t+1}$)
6: return $G^t$, $\Theta^t$

IV. PROPOSED METHOD

One common problem with real datasets is missing values, because many methods have the assumption of full observability, thus finding ways to work with missing values becomes crucial. One of the most used approaches to solve this problem is to drop the observations with missing values, however, when the dataset has few observations this approach can lead to loss of information. Another approach that one can take is to impute the missing values. In this approach, the missing values are “filled” using some method, like an interpolation. Since the focus of this thesis is the multivariate categorical time series, the most common methods for interpolation does not apply. So in order to impute the missing values, this thesis proposes a method that uses the SEM algorithm, devised by Friedman [8], to learn the structure of the data with missing values. However, because the algorithm learns BNs, it cannot model a time series, as such the algorithm was changed for the purpose of learning DBNs. As before, the search space was restricted to tDBN [19] and bcDBN [25]. The SEM algorithm can be divided with two big steps the parameter learning and the structure learning, and because the dataset has missing values one step cannot be learned without the other. As such, this algorithm starts by generating a DBN randomly. Then the “true” parameters of the fixed network can be learned. This is done in an iterative process where first the ESS are computed and then the new set of parameters are computed. This is done until convergence. With the parameters learned the algorithm then learns a new structure and repeats this process until the convergence criterion is met. Finally with the DBN given by the SEM, the imputation algorithm then generate again a new dataset without missing values, however instead of having all the possible combinations of values that could fill the missing values, this dataset fills the missing values with the combinations that maximizes the posterior probability. In section III can be seen a description of the SEM algorithm which will be the base to develop the imputation algorithm.

Algorithm 4 Missing values imputation via a DBN

1: procedure IMPUTATION-DBN($D$)
2: $G^0$, $\Theta^0$ $\leftarrow$ Generate a random DBN
3: $G$, $\Theta$ $\leftarrow$ Structural-EM($G^0$, $\Theta^0$, $D$)
4: for each observation in $D$ do
5: for each transition in the observation do
6: if transition has missing values then
7: Generate all possible combinations of values for the missing values
8: for each new combination of values do
9: Calculate the posterior probability
10: Select the generated combination that maximizes the posterior probability and impute the missing values
11: Add the observation to $D$
12: return $D$

Although clustering of time series being widely spoken in the research community, there is less work done in categorical time series. Ghassempour et al. [12] recently proposed an algorithm to cluster multivariate time series with variables taking both categorical or continuous values. To do that he uses a Hidden Markov Model (HMM) to model the data. However, HMMs are particular cases of DBNs and have independence assumptions that in some cases could be wrong. In order to remove these independence assumptions, this thesis proposes an algorithm to cluster data using a Dynamic Bayesian Multinet (DBM). This model is based on the BMs model which is described in Section II-C and allows the representation of asymmetries in the data. Given a dataset $D$ and $H$ clusters this method starts by generating $H$ DBNs, one associated to each cluster, then associate each observation to the cluster/DBN that best describes it. This step is done by calculating the probability of the observation belonging to each cluster. Having all observations being associated with a cluster, the algorithm then learns a new DBN for each cluster and repeats the second step. These steps are repeated until the score given by Equation (10) converges or some stopping criteria are met. As was said in Section II-C when using the CEM algorithm, the initial partitions depend on the initial partition so in order to reduce this problem this clustering method does the stochastic-step for the first one hundred steps and then it changes to the classification-step. Easily can be seen that if the DBNs are learned using the LL scoring function the CML criteria is maximized. To estimate the number of clusters it
Once more in this algorithm the search space of the DBNs are several results that support its use in this context [4, 22].

approximations do not hold for finite-mixture models, there strong evidence. Although the regularity conditions of the BIC and differences exceeding 10 can be viewed as representing [4] states that the largest the BIC score better the model score to compare models. Moreover, Dasgupta and Raftery based on the work of Fraley and Raftery [7], that uses the BIC score is the MDL score multiplied by two. This approach is To be noted that the Bayesian Information Criterion (BIC) is used the approximation of the log Bayes factor that will be given in Equation (16).

\[
\phi_{BIC}(B : D) = 2 \cdot \phi_{LL}(G : D) - \log(N) |B| \tag{16}
\]

To be noted that the Bayesian Information Criterion (BIC) score is the MDL score multiplied by two. This approach is based on the work of Fraley and Raftery [7], that uses the BIC score to compare models. Moreover, Dasgupta and Raftery [4] states that the largest the BIC score better the model and differences exceeding 10 can be viewed as representing strong evidence. Although the regularity conditions of the BIC approximations do not hold for finite-mixture models, there are several results that support its use in this context [4] [22].

Once more in this algorithm the search space of the DBNs was restricted to tDBN [19] and bDBN [25]. In algorithm 5 a pseudo-code of this procedure can be seen.

**Algorithm 5 Clustering data using a DBM**

1. procedure Clustering-DBM(H, D)
2. Generate H random DBNs \( H \) corresponds to the number of clusters
3. while the score increases do \( \triangleright \) The score is given in Equation (10)
4. for each observation \( o \) in \( D \) do
5. for each \( h = 1, ..., H \) do
6. Compute the posterior probability of \( o \) belongs to \( P_h \)
7. Add the observation \( o \) to the cluster \( h \) that maximizes the posterior
8. for each cluster \( h \) do
9. Learn a new DBN
10. Calculate the new score
11. return The clusters in the data

Lastly, is important to note that given the embarrassingly parallel property of the computation of the edges weights and the optimal set of parents, when learning a DBN, the learning algorithm was changed in order to allow parallel computations. In the current implementations, the algorithm distributes the computations by the cores, allowing faster computations of the DBN.

**V. Experimental Results**

**A. Imputation on simulated data**

To assess the merits of the SEM algorithm as an imputation method, multivariate time series were randomly generated using generated DBNs. Then various datasets were generated, where the characteristics like the number of observations and the number of variables were changed. Finally, the missing values were generated with respect to two parameters, the first is the percentage of subjects with missing values and the second is the percentage of missing values corresponding to a subject. With the purpose of comparing this imputation method with state of the art methods, first these methods were used to impute the datasets with missing values, then the number of errors between the original dataset, without missing values, and the imputed datasets were counted. To facilitate the visualization, the results of these experiments are grouped in Figure 2.

When analysing the results presented in Figure 2 it can be concluded that the imputation done by SEM algorithm has fewer errors when comparing with other imputation methods like Last Observation Carried Forward (LOCF), Mode and the Amelia [15]. This result was expected because the data were generated using DBNs, however, it important to highlight that despite the SEM algorithm has fewer errors, it has errors. One justification of this can be the fact that the imputation chosen by the algorithm is the one that maximizes the probability of the observation.

**B. Imputation on real data**

In order to evaluate the performance of the SEM algorithm as an imputation method it was used 10 datasets from UCI Machine Learning Repository [6] and UCR Time Series Classification Archive [3]. Moreover, because the implementation of the SEM algorithm only works with categorical time series and most of these datasets are composed by real-valued time series a discretization of these time series must be done. The discretization was done using the SAX algorithm, Lin et al. [18], it is important to note that it was used only an alphabet size of four, a maximum size of the time series of one hundred time steps and an independently discretization of each dimension of multivariate time series.

With the resulting discretized datasets, in order to test the performance of the imputation methods, it was removed values. Once more, these values were removed with respect to two parameters, the percentage of observation with missing values and the percentage of missing values per missing observation.

Moreover, in order to use the SEM algorithm some assumptions need to be made, these are the stationarity of the time series and the first-order Markov assumption. Analysing the results, Figure 2 it can be concluded that, in most datasets, the imputation done with SEM algorithm has fewer errors than the other methods.

Given these results, a Wilcoxon signed ranks test was performed, in order to compare the SEM algorithm with others imputation methods. As such, the results were grouped by
methods and by the percentage of observations with missing values, Table I. The use of a Wilcoxon signed ranks test is justified by the fact that is simple and a robust non-parametric test for statistical comparisons (Demšar [5]).

![Figure 3: Imputation errors for real datasets.](image)

Table I: Results from the Wilcoxon signed ranks test between SEM algorithm and other methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>% of observations with missing values</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOCF</td>
<td>10</td>
<td>0.017 42</td>
</tr>
<tr>
<td>LOCF</td>
<td>20</td>
<td>0.012 46</td>
</tr>
<tr>
<td>LOCF</td>
<td>30</td>
<td>0.032 20</td>
</tr>
<tr>
<td>LOCF</td>
<td>40</td>
<td>0.125 79</td>
</tr>
<tr>
<td>Mode</td>
<td>10</td>
<td>0.005 89</td>
</tr>
<tr>
<td>Mode</td>
<td>20</td>
<td>0.001 95</td>
</tr>
<tr>
<td>Mode</td>
<td>30</td>
<td>0.001 95</td>
</tr>
<tr>
<td>Mode</td>
<td>40</td>
<td>0.001 95</td>
</tr>
<tr>
<td>Amelia</td>
<td>10</td>
<td>0.001 95</td>
</tr>
<tr>
<td>Amelia</td>
<td>20</td>
<td>0.001 95</td>
</tr>
<tr>
<td>Amelia</td>
<td>30</td>
<td>0.001 95</td>
</tr>
<tr>
<td>Amelia</td>
<td>40</td>
<td>0.001 95</td>
</tr>
</tbody>
</table>

When analysing results from Table I it is easy to note that almost all p-values have a value below 0.05, which indicates that the null-hypothesis, that the two algorithms perform equally well, is discarded. However, it is important to note that the imputation errors of LOCF are similar to the imputation errors of the SEM algorithm which is a strange result. One explanation for this result can be the number of symbols used to discretize the time series, because since 4 symbols were used the discretized time series may not vary that much over time, which leads to a lower imputation error when using the LOCF.

C. Clustering on simulated data

Once more, in order to compare the implemented algorithm with state of the art methods, synthetic datasets were generated, with different numbers of observations. These datasets consist of time series generated from 2 DBNs with five variables, each one of them with an alphabet size of two and ten time steps. Moreover, it is used a mixture of HMMs [14] and hierarchical clustering [24] to compare the clustering result of DBM. Because the datasets are synthetic, the number of clusters is known, so the nature of this experiment is to analyse the capability of these methods to find groups in data.

To do that, this experiment consists in clustering the data using these methods, then evaluate the resultant clusters using external Clustering Validity Indices (CVIs); the external CVIs are used because the DBN that generate each observation is known. It is important to highlight that because the HMMs have a parameter that defines the number of hidden states, in this experiment, it was used three possible numbers of hidden states because it was shown by experimenting that three represents a good trade-off between good results and computing time. The DBNs were learned using the MDL score, a search space restricted to tDBNs, a number of maximum parents of one and five initializations.

When analysing the results of this experiment, Figure 4 it is easily noted that the DBM obtains better results and in some cases can separate perfectly the observations. However, this result was expected, because the dataset was generated by DBNs. Furthermore, it can be seen that the clustering performance of the mixture of HMMs increases with the number of states, nevertheless, in the real world finding the number of hidden states is not an easy task. Another conclusion that one can take, is that the hierarchical clustering obtains the worst and the most unreliable results, varying abruptly from dataset to dataset. Finally, with this experiment, one can conclude that the DBMs can better capture relations within the data, which could lead to a better separation.

![Figure 4: External CVIs for the clustering results.](image)

To test the capability of the BIC score to select the number of clusters another experiment was done. This experiment consists of generating various datasets with a different number of clusters within, and then run the DBM model for each dataset with the number of DBNs varying from one to ten saving the BIC score. The datasets were generated using different random generated DBNs, as such, it can be said that each DBN represents a cluster. Each DBN was generated using five variables, each on of them with an alphabet size of two, and ten time steps. Moreover, each of these datasets has a different number of DBNs. The results can be seen in Figure 5b where Figure 5a shows the evolution of the BIC score, when changing the number of clusters, and Figure 5b shows the difference in the BIC score between models with consecutive number of clusters. Looking to the results, Figure 5 can be said that selecting the maximum number of clusters where the differences in the BIC score represents strong evidence, Figure 5b is a good heuristic. As it can be seen, by using this method all the selected clusters correspond to the correct number of clusters. Due to the fact that learning DBMs depends on the initialization, which is random, sometimes the score can be worst than expected. An
easy way to avoid this problem is to do several initializations and then use the result that obtained better score.

(a) BIC score for various models.
(b) Differences in BIC scores for various models.

Figure 5: Clustering analysis of different datasets.

In order to get an inside look at the SEM iterations, a new experiment was carried out. It starts by generating two DBNs depicted in Figure 6 and then samples two thousand instances from each of them. Then, it creates a new dataset grouping together the sampled instances of the two DBNs and a class file with the information of the class/cluster of each instance.

Finally, the clustering method is applied to this dataset, in order to get an inside look of the clustering method, the learned DBNs at each iteration are saved. To facilitate the analysis, the dependencies that are in the original network, but not in the learned network, are displayed in gray, the ones that have the opposite direction are in blue, and the ones that are in the learned network but not in the original are dashed. As it was expected, the clustering method starts by generating a random DBM, Figure 7 as such the intra and inter-dependencies are different from the real ones, in this case having much more dependencies that the original network. In the first iteration, Figure 8, the first thing that one can note is that many of the dependencies disappear and some of the original dependencies are learned. In the second iteration, Figure 9, the algorithm learned the original networks, however, it only stops in the third iteration, Figure 10. The explanation to this can be that despite having the original structure the parameters of the network could improve in this iteration.

D. Clustering on real data

Seeing that this clustering method obtains good results using simulated data, the next experiment consists of using real datasets, for the task of classification, to analyse if the clustering results contain information about the class of the observations. It was used two datasets from the UCI Machine Learning Repository [6] and UCR Time Series Classification Archive [5] which, once more, were discretized using the same procedure used in Section V-B. Moreover, to facilitate the analysis, for each dataset only two classes were selected, this is done because it makes the confusion matrices simpler and reduces the computation time. Also, it is important to note that the same assumptions made in Section V-B are also made in these datasets. The first dataset used was the Spoken Arabic Digit, even though the number of classes within the dataset is known the number of clusters within the dataset is not known. So the first step of this experiment is to analyse the evolution of the BIC score, as was done in Section V-B. As it can be seen in Figure 11a the analysis of the BIC score retrieve that the number of clusters within the data that maximizes the BIC score is two. This result seems promising since there are only two classes in the dataset. So, in order to assert if these clusters contain any information about the class the label of the observations and the assigned cluster a confusion matrix was constructed. Analysing the confusion matrix, Figure 11b, it can be easily seen that the clustering result can capture information about the class of the observations, separation almost all the observations correctly.

Thereafter, in order to compare this method with state of the art methods, it was used the same methods that were used in Section V-C. Because the number of clusters is not known, two different heuristics were used to select the number of clusters. With the hierarchical clustering, the number of clusters was
selected by choosing the one that maximizes the difference in height in the dendrogram. In the mixture of HMM, the number of clusters and the number of states of the model were selected by choosing the model that maximizes the BIC score.

Figure 12: Confusion matrix for the Spoken Arabic Digit dataset with only 2 classes.

Looking to the confusion matrix of the state of the art methods, Figure 12, it can be concluded that the methods that can better capture information about the class of the observation are the mixture of HMM and the DBM. The hierarchical clustering is the one that obtains worst results. This similar behavior between the mixture of HMM and DBM is explained by the fact that the DBN model generalizes the HMM by allowing dependencies between variables.

The second dataset used was the uWave Gesture Library dataset, once more the same steps that were done in the first dataset were used in the second. However, unlike the first dataset, the analysis of the evolution of the BIC score shows that five clusters maximize the BIC score, Figure 13a. Although this result is different from the number of classes in the dataset, a more detailed analysis is needed. Looking at the confusion matrix, Figure 13b, one can conclude that the clustering result can extract information about the class of the observations.

Comparing this confusion matrix with the resulting confusion matrix of the state of the art methods, Figure 14 it can be concluded that the hierarchical clustering using the Dynamic Time Warping (DTW) distance and the DBM obtains better results compared with the other methods.

VI. CONCLUSION

First, this thesis proposes one method that imputes missing data. To do that it implements the SEM algorithm in order to learn a DBN in the presence of missing values. The imputation of missing values is then done by choosing the values that best fit the learnt model. Because learn DBN is NP-hard the search space was reduced to tDBN and bcDBN structures. The imputation of the simulated datasets, using stationary tDBN, achieved very good results showing to be competitive with state of the art methods. Moreover, the results obtained in the real datasets show that the proposed method is as good or better than the state of the art methods.

The second method that this thesis proposes tries to solve the task of clustering, by grouping similar observations. It extends the BM model in order to allow the use of DBNs, the resultant model is denoted by DBM. Like the imputation method, the search space is also restricted to tDBN and bcDBN structures. The results obtained using the generated datasets validate this clustering method and proves that this method outperforms the state of the art methods when the data is model by a DBN. When using real datasets this method can retrieve information about the class of the observations. Also, it is important to note that comparing with state of the art methods the proposed method is the most consistent obtaining the best overall result in the tested datasets. However, this method as the limitation of only working with discrete data and the complexity being exponential with the variables.

The developed algorithms can be improved in various ways. The first one is to automatically generate various initializations of the EM algorithm and then select the one that has the greater score. Because the result of the EM algorithm is dependent on the initialization it is important to run various initializations in order to get the best result. In the experiments, these initializations were done by using an R wrapper that calls the java application and then stores its result.

Another improvement to these algorithms regards the type of variables that they algorithm support, currently these algorithms only support categorical variables. So a possible
improvement could be the support of categorical and real-valued variables. This would be very useful because when a real-valued time series is discretized there is some loss of information. Moreover, there is some work in learning hybrid BN [23].

Regarding the experiment of the imputation of a real dataset, the obtained results indicate that the LOCF is almost as good as the proposed method. This can be explained by the fact that the discretization was done by using an alphabet with size equal to four, therefore the time series might not vary in time. So a possible improvement to this experiment is using an alphabet with a greater size which would allow the discretized time series to have more variation.

REFERENCES