Profiling ALS disease progression through data mining techniques

Tiago Leão
Instituto Superior Técnico
Universidade de Lisboa
Lisbon, Portugal
tiago.miguel.leao@tecnico.ulisboa.pt

Alexandra M. Carvalho
Instituto de Telecomunicações
Instituto Superior Técnico
Universidade de Lisboa
Lisbon, Portugal
alexandra.carvalho@tecnico.ulisboa.pt

Sara C. Madeira
LASIGE
Faculdade de Ciências
Universidade de Lisboa
Lisbon, Portugal
sacmadeira@ciencias.ulisboa.pt

***Abstract*—Amyotrophic lateral sclerosis (ALS) is a neurodegenerative disease that causes a fast functional decline of the patients. This Thesis tackles ALS disease progression using dynamic Bayesian networks (DBNs), a machine learning model that graphically displays the joint probability distribution of dynamic (time-dependent) random variables. To include static (time-independent) information in DBNs, the sdtDBN framework is proposed, which learns optimal DBNs with static and dynamic variables, having polynomial-time complexity in the number of variables. The sdtDBN framework can also introduce prior knowledge (by restricting the networks' relations) and make inference in learned sdtDBNs. The disease progression is assessed using observations of 1214 ALS patients, studying all patients and also dividing them into three progression groups, being sdtDBNs employed to predict the patients' functional decline and to determine correlations between clinical indicators. The predictions provide promising results, with accuracies generally above 75%. The correlations found present an intuitive overview of the interactions among variables. The Thesis ends by answering three clinical questions using sdtDBNs, providing an analysis with clinical impact. All assessments presented show that sdtDBNs can properly profile ALS disease progression, motivating its use as a clinical tool.**

***Index Terms*—amyotrophic lateral sclerosis, data mining, disease progression, dynamic Bayesian networks, polynomial-time algorithm, time-dependent and time-independent variables**

1. Introduction

AMYOTROPHIC lateral sclerosis (ALS) is a neurodegenerative disease that quickly affects the loss of motor neurons of the patients [1]. The disease's symptoms are usually associated to limb weakness and difficulties in activities such as speaking and breathing, being one of the most used criteria to diagnose ALS the El Escorial criteria, whose result is a degree of probability for a patient to have ALS [1].

There is not a known cure for ALS, so, treatments focus on slowing the progression of the disease. The functional decline of a patient is assessed with a standardized set of tests that compose the ALS functional rating scale (ALS-FRS), currently in its revised version (ALS-FRS-R) [2].

As the main cause of death among ALS patients is respiratory failure [3], doctors' major concern is to determine when ventilatory support is needed, which is usually provided through non-invasive ventilation (NIV). However, no standard criteria specify the proper moment to apply NIV to a patient. Therefore, obtaining statistical information regarding the patients' progression is extremely beneficial, giving doctors a tool that helps them apply NIV at the proper time. In this work, ALS disease progression is described using dynamic Bayesian networks (DBNs), a machine learning model that includes in its framework the temporal component of clinical indicators and is easily interpretable.

1.1. Previous work

Multiple works tackle several aspects of the learning procedure of DBNs, for example, presenting an overview regarding learning and making inference on DBNs [4], explaining how to learn the structure of DBNs from complete or incomplete data [5], or debating how to determine DBNs from time-series data [6]. Some works focus specifically on learning non-stationary DBNs, for instance, determining the transition probabilities with the l_1-regularized least square linear regression method [7], employing the Markov chain Monte Carlo method (MCMC) [8], or adapting the maximum-minimum hill-climbing algorithm (MMHC) to the temporal domain [9].

Regarding the contributions of this Thesis, it is extended the tDBN framework [10], which is an algorithm that optimally learns the inter-slice and intra-slice structures and parameters of DBNs, naming the resulting DBNs as tree-augmented DBNs (tDBNs), because the intra-slice connectivity is restricted to trees, while also allowing each node of the DBNs to have a maximum number of parents from previous time-slices (hence the "augmented"). Extensions to the tDBN work have already been tried. An example is the extension of the intra-slice connectivity of tDBNs to forests [11], using an ordering of the optimal branching of tDBNs consistent with a breadth-first search.

With regard to the application of data mining techniques to study the progression of ALS patients, most studies focus on the survival of patients, using population-based statistical

approaches, such as the Kaplan-Meier method and Cox models [12]. When employing machine learning techniques, two of the most used models are random forests [13] and neural networks [14]. Some studies focus on determining when NIV should be applied to the patients [15, 16].

In the medical field, DBNs are often used to describe the progression of a disease and predict several outcomes [17]. However, the use of DBNs to model the progression of ALS patients is a relatively new topic, which has only started to be explored in recent literature [18].

1.2. Approach of this work

The ALS dataset analyzed in this Thesis has observations of patients' static (time-independent) and dynamic (time-dependent) indicators. However, the standard DBN framework does not include static variables. Therefore, this work proposes a DBN framework that includes static and dynamic variables in its structure, also allowing a user to introduce prior knowledge (by inserting restrictions in the networks) and to make inference in learned DBNs. After creating the mentioned DBN framework, the ALS dataset is addressed. First, data is preprocessed, then, ALS disease progression is tackled with the developed DBN framework.

The remaining of this document is organized as follows. **Section 2** presents the ALS dataset used in the Thesis. **Section 3** provides the background needed regarding data mining and DBNs. **Section 4** explains the proposed DBN framework. **Section 5** shows the assessment done using ALS data. **Section 6** presents the conclusions of the Thesis and some ideas for future work.

2. ALS dataset

The dataset used in this work is the Portuguese ALS dataset from the Translational Clinic Physiology Unit at Hospital de Santa Maria, IMM, Lisbon. The dataset was created in 1995 and its latest update was in March 2020. The current version of the dataset stores medical observations of 1374 ALS patients and has around 50 static and 100 dynamic features. An overview of the main features of the Portuguese ALS dataset is presented in Table 1.

TABLE 1. MAIN FEATURES OF THE PORTUGUESE ALS DATASET (ORANGE: NUMERICAL; PINK: CATEGORICAL).

Main static features	**Demographic information**	Gender	Age at onset	Body mass index (BMI) at onset	
	Medical and family history	Family history of motor neuron disease (MND)		Retired at diagnosis	
	Onset evaluation	Onset form	UMN vs LMN	EI Escorial reviewed criteria	
		Diagnostic delay		Cirurgical interventions before	
	Genetic biomarkers	Expression of C9orf72 mutations			
Main dynamic features	**Functional scores**	ALS-FRS	ALS-FRSsUL	ALS-FRSb	R
		ALS-FRS-R	ALS-FRSsLL	ALS-FRSr	
	Respiratory tests	Vital capacity (VC)	Maximal inspiratory pressure (MIP)		
		Forced VC (FVC)	Maximal expiratory pressure (MEP)		
		P0.1, PO2, PCO2 concentrations	Maximal sniff nasal inspiratory pressure (SNIP)		
			Peak expiratory flow (PEF)		
		Forced expiratory volume (FEV)			
		Depressions in O2 saturation		Mean O2 saturation	
	Respiratory status	Starting date of non-invasive ventilation (NIV)			
	Neurophysiological tests	Phrenic nerve response amplitude (PhrenMeanAmpl)			
		Phrenic nerve response latency (PhrenMeanLat)			
		Phrenic nerve response area (PhrenMeanArea)			
	Other physical values	Cervical extension		Cervical flexion	

The functional scores of Table 1 consist of evaluations specific to the ALS treatment. ALS-FRS is a standard set of tests and ALS-FRS-R is an improvement on ALS-FRS, with more respiratory tests. The remaining functional scores of Table 1 are sub-scores of ALS-FRS-R. Each functional score is composed by questions that evaluate the functional conditions of a patient. Each question is answered with an integer number from 0 to 4, where 0 is the worst condition and 4 is the best. ALS-FRS is the sum of ten questions. ALS-FRS-R is the sum of the first nine questions of ALS-FRS, plus three extra ones about specific respiratory indicators.

In this Thesis, most data used for learning the DBNs is retrieved from a pretreated version of the latest update of the ALS dataset (March 2020), provided by a student from the same research group in which this Thesis is inserted. The pretreated version used has records of 1214 patients, with measures of 9 static features and 31 dynamic features. It has average values of 5,52 records (consultations) per patient, 8,04 static features measured per patient and 21,94 dynamic features measured per consultation, with median values of 4 records per patient, 8 static features measured per patient and 21 dynamic features measured per consultation.

3. Theoretical background

3.1. Data mining

Data mining can be defined as the process of discovering useful patterns from large amounts of data [19], where the process must be at least semi-automatic [20]. Data mining is one step of the full process of knowledge discovery from data (KDD) [19], summarized in Fig. 1.

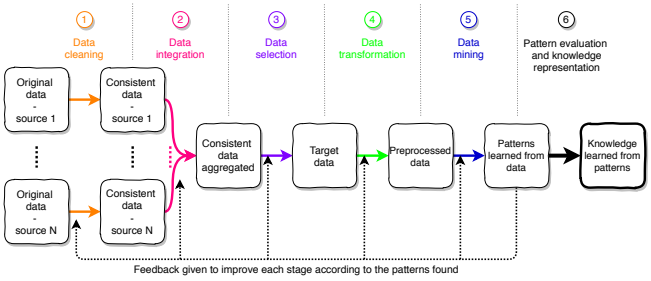


Figure 1. Overview of the process of knowledge discovery from data.

Steps 1 to 4 of the process presented in Fig. 1 compose the *data preprocessing stage*, which is a significant and possibly time-consuming activity [21]. Some of the most important procedures of the preprocessing stage are the filling of missing values [22], the detection of outliers [19], the selection of features [20], and the normalization [22] and discretization [19] of data. The preprocessing stage prepares the data for the mining operation, done in this work using dynamic Bayesian networks, described in Section 3.2.

3.2. Dynamic Bayesian networks

Bayesian networks (BNs) are a class of probabilistic graphical models [23]. Def. 1 rigorously presents the concept of Bayesian network.

Definition 1. *A BN is a triple $B = \{\boldsymbol{X}, \boldsymbol{G}, \boldsymbol{\theta}\}$, where:*

- $\boldsymbol{X} = \{X_1, \ldots, X_n\}$ *is a vector of n random variables.*
- $\boldsymbol{G} = \{\boldsymbol{X}, \boldsymbol{E}\}$ *is a directed acyclic graph (DAG) with nodes $\boldsymbol{X}$ and edges $\boldsymbol{E}$. Each node represents a random variable and each edge denotes a conditional dependency relation between a pair of nodes. For simplicity of notation, each X_i represents both the random variable and the respective node in $\boldsymbol{G}$. The parents of X_i in $\boldsymbol{G}$ are represented by $\boldsymbol{pa}(X_i)$.*
- $\boldsymbol{\theta}$ *is the set of all conditional probabilities needed to encode the joint probability distribution of $\boldsymbol{X}$. If the variables are discrete with each variable X_i having at most r_i states, $\boldsymbol{\theta} = \{\theta_{ijk}\}$, where*

$$\theta_{ijk} = P_B(X_i = x_{ik} \mid \boldsymbol{pa}(X_i) = w_{ij}). \quad (1)$$

In Eq. (1), $i \in \{1, \ldots, n\}$, $k \in \{1, \ldots, r_i\}$ and $j \in \{1, \ldots, q_i\}$, with $q_i = \prod_{X_l \in \boldsymbol{pa}(X_i)} r_l$. To specify $\boldsymbol{\theta}$, there must be provided the probabilities for each node X_i to take each of the possible r_i values, given each of the possible parents' configurations w_{ij}.

In a BN, a node, given its parents, is conditionally independent of all other variables in the BN. Therefore, the joint probability distribution of all nodes $\boldsymbol{X}$ of a BN is given by

$$P_B(X_1, \ldots, X_n) = \prod_{i=1}^{n} P_B(X_i \mid \boldsymbol{pa}(X_i)). \quad (2)$$

An extremely important task in a BN is the prediction of the values of certain unobserved nodes, given the values of some observed nodes, which is called making inference. Inference in BNs is based on Bayes' rule, applying it in the way described by

$$p(h \mid e) = \frac{p(e \mid h)p(h)}{p(e)}, \quad (3)$$

where e denotes the observed nodes and h the unobserved nodes. Exact inference in BNs is NP-hard [24], so, the only feasible options are to either apply restrictions to the BNs, or to make approximate inference.

Another crucial task in a BN consists in learning its structure and parameters, from certain observations. Learning the BN B that best fits some observations S means finding the graph $\boldsymbol{G}$ and the parameters $\boldsymbol{\theta}$ that best describe S. Using a scoring function $\phi(B, S)$, for which $\phi(B_1, S) > \phi(B_2, S)$ if a BN B_1 describes S better than a BN B_2, finding the BN B that best describes S, from a certain search-space B_n, is an optimization problem given by

$$B_{\text{Best fit}} = argmax_{B_i \in B_n} \phi(B_i, S). \quad (4)$$

An important property of a scoring function is its decomposability, which assures that changes in a certain variable of a BN only affect a certain component of the score. This property can be expressed as

$$\phi(B, S) = \sum_{i=1}^{n} \phi_i((X_i \mid \boldsymbol{pa}(X_i)), S). \quad (5)$$

Minimizing the size of an optimal code induced by the BN B when encoding the data S originates the log-likelihood (LL) score, whose expression is given by

$$\phi_{LL}(B, S) = \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk} log\left(\frac{N_{ijk}}{N_{ij}}\right), \quad (6)$$

where n is the number of nodes of the BN B, being r_i and q_i as defined in Eq. (1). N_{ijk} is the number of times in S where X_i takes its k-th value x_{ik} and $\boldsymbol{pa}(X_i)$ take their j-th configuration w_{ij}, while N_{ij} is the total number of times in S where $\boldsymbol{pa}(X_i)$ take their j-th configuration w_{ij}.

The LL score is prone to overfitting. The minimum description length (MDL) score tries to avoid this phenomenon, penalizing complex network structures by making

$$\phi_{MDL}(B, S) = \phi_{LL}(B, S) - \frac{1}{2} log(N) \times |B|, \quad (7)$$

where N is the total number of observations in S, and $|B| = \sum_{i=1}^{n} (r_i - 1)q_i$ is the number of parameters of B, with r_i and q_i as defined in Eq. (1).

Regarding the search-space of Eq. (4), if B_n is composed by all possible BNs with n nodes, the problem of Eq. (4) is NP-hard. One solution is to restrict the search-space, being a common restriction to only consider tree structures, where the Chow & Liu algorithm [25] can efficiently find a BN that maximizes the LL score, and may also be adapted to maximize any other decomposable scoring function [26]. Another solution is to perform an approximate search, where a popular example is the structural expectation-maximization algorithm (SEM) [27].

After learning the structure of a BN, its parameters $\boldsymbol{\theta}$ should also be determined, which is generally done using the observed frequency estimates (OFE), by making $\hat{\theta}_{ijk} = \frac{N_{ijk}}{N_{ij}}$, where N_{ijk} and N_{ij} are as defined in Eq. (6).

Dynamic Bayesian networks (DBNs) provide an extension of BNs to the dynamic/temporal domain. As notation, $X_i[t]$ denotes the random variable that, in timestep t, is associated to the feature X_i. The initial timestep is always $t = 0$, the last timestep is $t = T$, and $X_i[a : b]$ denotes all random variables associated to X_i between timesteps a and b. The trajectory of a random variable X_i consists of the values assigned to X_i, in each intermediate timestep, from a certain initial to a certain final timestep. Given the previous notation, Def. 2 presents the rigorous definition of DBN.

Definition 2. *A DBN is a pair $(B_0\ ,\ \boldsymbol{B_{\rightarrow}})$, where:*

- B_0 *is a prior BN that defines $P(\boldsymbol{X}[0])$, which is the probability distribution over the variables in $t = 0$.*
- $\boldsymbol{B_{\rightarrow}}$ *consists of the transition networks, being composed by the set of all $B_{\rightarrow}[0 : t]$, for $t \in \{1, \ldots, T\}$. Each $B_{\rightarrow}[0 : t]$ defines the distribution over the variables in timestep t, given all the trajectories that the variables may take between timesteps 0 and $t - 1$. Rigorously, each $B_{\rightarrow}[0 : t]$ is defined as*

$$B_{\rightarrow}[0 : t] = P(\boldsymbol{X}[t] \mid \boldsymbol{X}[0 : t - 1]). \quad (8)$$

A DBN defines the joint probability distribution of all possible trajectories of all features, which is given by

$$P(\boldsymbol{X}[0 : T]) = B_0 \prod_{t=1}^{T} B_{\rightarrow}[0 : t], \quad (9)$$

where the chain rule is used, being B_0 and $\boldsymbol{B}_{\rightarrow}$ as specified in Def. 2. Applying Eq. (8), Eq. (9) can also be expressed as

$$P(\boldsymbol{X}[0:T]) = P(\boldsymbol{X}[0]) \prod_{t=1}^{T} P(\boldsymbol{X}[t] \mid \boldsymbol{X}[0:t-1]). \quad (10)$$

Often two assumptions are made when learning a DBN. The first assumption is the m^{th}-order Markov assumption, which consists in stating that the distribution of variables in timestep t only depends on the values of variables from up to m timesteps before t, thus simplifying Eq. (10) into

$$P(\boldsymbol{X}[0:T]) = P(\boldsymbol{X}[0]) \prod_{t=1}^{T} P(\boldsymbol{X}[t] \mid \boldsymbol{X}[t-m:t-1]). \quad (11)$$

The second simplifying assumption is the stationary assumption, which assumes that, in a DBN respecting the m^{th}-order Markov assumption, the transition networks $P(\boldsymbol{X}[t] \mid \boldsymbol{X}[t-m:t-1])$ are the same for all timesteps t.

Inference in DBNs is an extension of inference in BNs. The description done in Eq. (3) remains valid, with e and h incorporating the temporal component of DBNs.

Structure and parameter learning in DBNs also extends the methods used in BNs. Usually, greedy search procedures or the MCMC algorithm are used. Other approaches restrict the intra-slice structures and extend optimal algorithms of BNs, such as the Chow & Liu algorithm, to DBNs [10].

Regarding parameter learning, the parameters θ_{ijk} (see Def. 1) can be extended to the temporal domain, by defining

$$\theta_{ijk}[t] = P_{B_d}(X_i[t] = x_{ik} \mid \boldsymbol{pa}(X_i[t]) = w_{ij}[t]), \quad (12)$$

where B_d is a DBN, with known structure, and the remaining terms are as defined in Def. 1, with the temporal component properly added. Provided Eq. (12), the OFE can be defined in DBNs as $\hat{\theta}_{ijk}[t] = \frac{N_{ijk}[t]}{N_{ij}[t]}$, being $N_{ijk}[t]$ the number of times in the dynamic observations D where the node $X_i[t]$ of B_d takes its k-th value x_{ik} and the nodes in $\boldsymbol{pa}(X_i[t])$ of B_d take their j-th configuration $w_{ij}[t]$, while $N_{ij}[t]$ is the total number of times in D where the nodes in $\boldsymbol{pa}(X_i[t])$ of B_d take their j-th configuration $w_{ij}[t]$.

4. Contributions on DBNs methods

4.1. The sdtDBN framework

As stated in Section 1.1, the tDBN framework [10] is used in this Thesis. To properly analyze the ALS dataset, there is the need of including static attributes/variables in the DBNs, which is done in this work by extending tDBNs to sdtDBNs (*tDBNs with static and dynamic variables*).

Introducing some notation, $\boldsymbol{Y}$ denotes the static attributes of the DBNs, with S being the static observations, and $\boldsymbol{X}[t]$ represents the dynamic attributes in timestep t, with D being all dynamic observations, and D_t^{t+j} the observations of dynamic attributes between timesteps t and $t+j$. All attributes are discrete, with a finite number of states, having the DBNs n_{static} static attributes and n dynamic attributes in each timestep, with a maximum of T timesteps. Each node $X_i[t]$ has a maximum of p dynamic parents from the m previous timesteps (m is the Markov lag) and a maximum of b static parents. The decomposable scoring function is denoted as ϕ, with local terms given by ϕ_i, for each node $X_i[t]$. Finally, $P_{\leq\gamma}(\boldsymbol{A})$ represents all possible subsets, with cardinality at most γ, of a certain set $\boldsymbol{A}$.

Given all presented notation, it is possible, for a node $X_i[t+1]$ of an sdtDBN, to get the maximum possible score of ϕ_i, when not considering any connection among nodes in timestep $t+1$. This maximization is given by

$$\begin{aligned} s_i = \max_{\boldsymbol{X}_{dp}, \boldsymbol{Y}_{sp}} \quad & \phi_i(\boldsymbol{X}_{dp} \cup \boldsymbol{Y}_{sp}, D_{t+1-m}^{t+1} \cup S) \\ \text{s.t.} \quad & \boldsymbol{X}_{dp} \in P_{\leq p}(\boldsymbol{X}[t+1-m] \cup \cdots \cup \boldsymbol{X}[t]), \\ & \boldsymbol{Y}_{sp} \in P_{\leq b}(\boldsymbol{Y}), \end{aligned} \quad (13)$$

where the $\boldsymbol{Y}_{sp}$ and $\boldsymbol{X}_{dp}$ that maximize Eq. (13) are the optimal sets of static and dynamic parents of $X_i[t+1]$, when not considering any connections in timestep $t+1$.

As the intra-slice connectivity is restricted to trees [10], each node $X_i[t+1]$ of an sdtDBN may have, at most, one parent from timestep $t+1$. For a certain node $X_i[t+1]$, it may be obtained the maximum score of ϕ_i, when considering that a connection $X_j[t+1] \rightarrow X_i[t+1]$ is in the DBN structure. This maximization is given by

$$\begin{aligned} s_{ij} = \max_{\boldsymbol{X}_{dp}, \boldsymbol{Y}_{sp}} \quad & \phi_i(\boldsymbol{X}_{dp} \cup \boldsymbol{Y}_{sp} \cup X_j[t+1], D_{t+1-m}^{t+1} \cup S) \\ \text{s.t.} \quad & \boldsymbol{X}_{dp} \in P_{\leq p}(\boldsymbol{X}[t+1-m] \cup \cdots \cup \boldsymbol{X}[t]), \\ & \boldsymbol{Y}_{sp} \in P_{\leq b}(\boldsymbol{Y}), \end{aligned} \quad (14)$$

where the $\boldsymbol{Y}_{sp}$ and $\boldsymbol{X}_{dp}$ that maximize Eq. (14) are the optimal sets of static and dynamic parents of $X_i[t+1]$, considering that $X_j[t+1] \rightarrow X_i[t+1]$ is in the DBN structure.

Given Eqs. (13) and (14), it is possible to express the benefit e_{ij} of including $X_j[t+1]$ as a parent of $X_i[t+1]$, instead of just having $X_i[t+1]$ with the optimal static parents from $\boldsymbol{Y}$ and the optimal dynamic parents from $\boldsymbol{X}[t+1-m] \cup \cdots \cup \boldsymbol{X}[t]$. This benefit is expressed as

$$e_{ij} = s_{ij} - s_i. \quad (15)$$

The procedure for finding an optimal sdtDBN employs the same rationale of the tDBN structure learning algorithm [10], adding the influence of static variables in the computation of the e_{ij} terms using Eqs. (13), (14) and (15). For a certain timestep $t+1$, first, a complete directed graph with nodes $\boldsymbol{X}[t+1]$ is created, using each e_{ij} as the weight of the respective edge $X_j[t+1] \rightarrow X_i[t+1]$. Then, using Edmonds' algorithm [28], a maximum spanning tree is computed in the created graph. The edges belonging to the spanning tree compose the intra-slice connectivity of timestep $t+1$. Using each intra-slice edge, the static and dynamic parents of each node are extracted from Eq. (14), except for the root node of the tree, for which Eq. (13) should be used. Applying the previous procedure for all timesteps, an optimal sdtDBN is found. Algorithm 1 presents the sdtDBN learning methodology, using Algorithm 2 to determine the optimal sets of static and dynamic parents and the e_{ij} terms.

Algorithm 1: Structure learning of m^{th}-order Markov non-stationary sdtDBNs.

Input :
- $\boldsymbol{X}$: the n dynamic attributes of the DBN.
- $\boldsymbol{Y}$: the n_{static} static attributes of the DBN.
- T: the total number of timesteps of the DBN.
- D: dataset with observations for each dynamic node of the DBN.
- S: dataset with observations for each static node of the DBN.
- ϕ: a decomposable scoring function.

Output :
- An optimal non-stationary sdtDBN.

1 **for** *each timestep* $t+1$ *between* m *and* T **do**
2 Construct a complete directed graph in $\boldsymbol{X}[t+1]$.
3 Determine the weights of all edges $X_j[t+1] \rightarrow X_i[t+1]$, $i \neq j$, using Algorithm 2, also extracting the optimal sets of static and dynamic parents of each node $X_i[t+1]$.
4 Apply a maximum branching algorithm in the graph of line 2 using the weights determined in line 3, in order to have a maximum spanning tree considering the weights of line 3.
5 Extract the static and dynamic parents of each node $X_i[t+1]$ from the spanning tree determined in line 4 and the sets of parents determined in line 3.
6 Join all transitions obtained in the **for** loop, to get the complete description of the determined sdtDBN.

The procedure of Algorithm 1 can be changed to learn stationary sdtDBNs, with only one transition network that achieves the maximum global score considering the observations of all timesteps. To do so, the for loop in line 1 of Algorithm 1 should be just one iteration (instead of a loop), line 6 of Algorithm 1 can be removed (only one transition is determined), and, in lines 7 and 19 of Algorithm 2, the sub-dataset D_{t+1-m}^{t+1} should be replaced by the whole dataset D.

The optimality of Algorithm 1 is provided in Theorem 1.

Theorem 1. *Algorithm 1 finds globally optimal sdtDBNs.*

Proof. As Algorithm 2 searches all parents' combinations to obtain the matrix E, the optimality per timestep of Algorithm 1 is inferred from the correctness of Edmonds' maximum branching algorithm [28]. Given the optimality per timestep, the global optimality of Algorithm 1 is also shown, because a certain iteration of the for loop of Algorithm 1 cannot affect the structures found in other iterations. □

The computational complexity of Algorithm 1 is given in Theorem 2.

Theorem 2. *The worst-case complexity of Algorithm 1 is polynomial in the number of dynamic attributes n and the number of static attributes n_{static}, exponential in the number of dynamic parents p and the number of static parents b, linear in the number of dynamic observations $N_{dynamic}$ and the number of static observations N_{static}.*

Algorithm 2: Edge weights and optimal parents for m^{th}-order Markov non-stationary sdtDBNs.

Input :
- $t+1$: the current timestep.
- m: the Markov lag of the DBN.
- $\boldsymbol{X}[t+1-m] \cup \cdots \cup \boldsymbol{X}[t+1]$: sets of nodes from the current timestep and the m previous timesteps, each timestep having n nodes.
- $\boldsymbol{Y}$: set of n_{static} static nodes.
- p: upper-bound on the number of dynamic parents from the m previous timesteps.
- b: upper-bound on the number of static parents.
- D_{t+1-m}^{t+1}: dataset with observations for each dynamic node, from timesteps $t+1-m$ to $t+1$.
- S: dataset with observations for each static node.
- ϕ_i(parentNodes, dataset): local terms of the decomposable scoring function ϕ, for each node $X_i[t+1]$.

Output :
- $E_{[n \times n]}$: matrix with edge weights e_{ij}.
- *dynamicParentsPast*$_{[n]}$, *staticParentsPast*$_{[n]}$: for each $X_i[t+1]$, the optimal set of dynamic parents from the m previous timesteps and the optimal set of static parents, not considering connections in timestep $t+1$.
- *dynamicParents*$_{[n \times n]}$, *staticParents*$_{[n \times n]}$: for each $X_i[t+1]$, the optimal set of dynamic parents from the m previous timesteps and the optimal set of static parents, when considering each possible connection $X_j[t+1] \rightarrow X_i[t+1]$ in the sdtDBN.

1 *allDynamicParentSets* $\leftarrow P_{\leq p}(\boldsymbol{X}[t+1-m] \cup \cdots \cup \boldsymbol{X}[t])$
2 *allStaticParentSets* $\leftarrow P_{\leq b}(\boldsymbol{Y})$
3 **for** $X_i[t+1]$ **in** $\boldsymbol{X}[t+1]$ **do**
4 *bestScore* $\leftarrow -\infty$
5 **for** $\boldsymbol{X}_{dp}$ **in** *allDynamicParentSets* **do**
6 **for** $\boldsymbol{Y}_{sp}$ **in** *allStaticParentSets* **do**
7 *currentScore* $\leftarrow \phi_i(\boldsymbol{X}_{dp} \cup \boldsymbol{Y}_{sp}, D_{t+1-m}^{t+1} \cup S)$
8 **if** *currentScore* $>$ *bestScore* **then**
9 *bestScore* $\leftarrow$ *currentScore*
10 *dynamicParentsPast*$_i \leftarrow \boldsymbol{X}_{dp}$
11 *staticParentsPast*$_i \leftarrow \boldsymbol{Y}_{sp}$
12 **for** $X_j[t+1]$ **in** $\boldsymbol{X}[t+1]$ **do**
13 $E_{ij} \leftarrow -$*bestScore*
14 **for** $X_i[t+1]$ **in** $\boldsymbol{X}[t+1]$ **do**
15 **for** $X_j[t+1]$ **in** $\boldsymbol{X}[t+1]$ **do**
16 *bestScore* $\leftarrow -\infty$
17 **for** $\boldsymbol{X}_{dp}$ **in** *allDynamicParentSets* **do**
18 **for** $\boldsymbol{Y}_{sp}$ **in** *allStaticParentSets* **do**
19 *currentScore* $\leftarrow$ $\phi_i(\boldsymbol{X}_{dp} \cup \boldsymbol{Y}_{sp} \cup X_j[t+1], D_{t+1-m}^{t+1} \cup S)$
20 **if** *currentScore* $>$ *bestScore* **then**
21 *bestScore* $\leftarrow$ *currentScore*
22 *dynamicParents*$_{ij} \leftarrow \boldsymbol{X}_{dp}$
23 *staticParents*$_{ij} \leftarrow \boldsymbol{Y}_{sp}$
24 $E_{ij} \leftarrow E_{ij}$ + *bestScore*

Proof. The bottleneck of each iteration of Algorithm 1 is when Algorithm 2 is used, being the bottleneck of Algorithm 2 the nested loops starting in line 14.

The loops in lines 14 and 15 have complexity $\mathcal{O}(n)$ each.

As $|P_{\leq p}(\boldsymbol{X}[t+1-m] \cup \cdots \cup \boldsymbol{X}[t])| = \sum_{i=0}^{p} \binom{nm}{i}$, the loop in line 17 has complexity $\mathcal{O}((nm)^p)$, because $\sum_{i=0}^{p} \binom{nm}{i} < \sum_{i=0}^{p} (nm)^i \in \mathcal{O}((nm)^p)$.

Regarding static parents, as $|P_{\leq b}(\boldsymbol{Y})| = \sum_{i=0}^{b} \binom{n_{static}}{i}$, the loop in line 18 has complexity $\mathcal{O}(n_{static}^b)$, because $\sum_{i=0}^{b} \binom{n_{static}}{i} < \sum_{i=0}^{b} n_{static}^i \in \mathcal{O}(n_{static}^b)$.

The worst-case complexity of determining ϕ_i in line 19 happens when a node has $b+p+1$ parents. Assuming that each attribute can take at most λ different values, each conditional probability distribution has a space complexity of $\mathcal{O}(\lambda^{b+p+2})$. Each configuration of a probability distribution is evaluated in each observation of D_{t+1-m}^{t+1} and S, which have space complexities of, respectively, $|D_{t+1-m}^{t+1}| \times (m+1) \times n$ and $|S| \times n_{static}$. Joining all previous components, line 19 has a total complexity of $\mathcal{O}(|D_{t+1-m}^{t+1}| \times (m+1) \times n \times |S| \times n_{static} \times \lambda^{b+p+2})$.

Algorithm 1 uses Algorithm 2 a total of $\mathcal{O}(T)$ times in the loop. Defining $N_{dynamic} = \sum_{t=0}^{T-1} |D_{t+1-m}^{t+1}|$ and $N_{static} = |S|$ as the total dimensions of the dynamic and static datasets, respectively, and joining all aforementioned complexities of Algorithm 2, the total complexity of Algorithm 1 can be expressed as

$$\mathcal{O}(n^{p+3} n_{static}^{b+1} m^{p+1} \lambda^{b+p+2} N_{dynamic} N_{static}), \qquad (16)$$

which proves Theorem 2. □

To properly study the ALS data, there is the need of including prior knowledge in the sdtDBNs, which can be done by restricting the relations of the learned networks. For a certain relation in an sdtDBN, a restriction may force that relation either to exist or not to exist. To cover all possible situations, there must be considered three types of relations:

(i) Between dynamic nodes in different timesteps.
(ii) Between dynamic nodes in the same timestep.
(iii) Between dynamic nodes and static nodes.

Restrictions in relations of **types (i)** and **(iii)** are included in the learning algorithm by changing the proper search-spaces, in lines 1 and 2 of Algorithm 2, respectively. If a node $X_i[t+1]$ has some mandatory and some forbidden parents from previous timesteps, then, when analyzing the possible parents of $X_i[t+1]$, all subsets of $P_{\leq p}(\boldsymbol{X}[t+1-m] \cup \cdots \cup \boldsymbol{X}[t])$ that contain at least one forbidden parent, or do not contain all mandatory parents, are removed from the search-space. The rationale for incorporating restrictions in relations of **type (iii)** is similar.

Restrictions in relations of **type (ii)** are included by biasing the weights E_{ij}, determined in Algorithm 2. For each timestep, Algorithm 1 applies a maximum branching algorithm using the weights E_{ij}, and the resulting maximum spanning tree contains the intra-slice connectivity of the corresponding timestep. Therefore, restrictions in relations of **type (ii)** can be introduced in Algorithm 2 by biasing the weights E_{ij}, which is done by replacing line 24 of Algorithm 2 with the procedure presented in Algorithm 3.

Algorithm 3: Procedure to include restrictions in relations of **type (ii)** in sdtDBNs, biasing the weights E_{ij} with *bigNumber*, which is a positive number several orders of magnitude higher than the weights E_{ij}.

if $X_j[t+1] \rightarrow X_i[t+1]$ *is mandatory* **then**
  $E_{ij} \leftarrow E_{ij}$ + *bestScore* + *bigNumber*
else if $X_j[t+1] \rightarrow X_i[t+1]$ *is forbidden* **then**
  $E_{ij} \leftarrow E_{ij}$ + *bestScore* - *bigNumber*
else
  $E_{ij} \leftarrow E_{ij}$ + *bestScore*

By directly changing the search-spaces, any restriction in relations of **types (i)** and **(iii)** is always respected. Restrictions in relations of **type (ii)** are always respected due to the soundness of Edmonds' maximum branching algorithm [28]. The learning procedure of sdtDBNs remains optimal given the restrictions, because restricting the search-spaces only avoids evaluating forbidden parents sets, and the bias on a certain E_{ij} is done after getting all parents sets associated with node $X_i[t+1]$ as a child node, being these sets obtained as in the unrestricted algorithm (see Algorithm 2).

The computational complexity of the learning algorithm with restrictions depends on how much the restrictions in relations of **types (i)** and **(iii)** reduce the search-spaces. Restrictions in relations of **type (ii)** do not change the complexity of the learning algorithm, as biasing the E_{ij} terms does not change the complexity of line 24 of Algorithm 2.

After defining the sdtDBN framework and including restrictions in the networks, the predictive capabilities of the model must be tackled, in order to allow a user to make inference in learned sdtDBNs.

Denoting *staticObs* and *dynObs* as, respectively, the static and dynamic observations provided by a user, Algorithm 4 presents the depth-first search (DFS) approach for estimating the value of a certain node $X_j[t]$ of an sdtDBN.

Algorithm 4: DFS approach to estimate the value of a certain node $X_j[t]$ of an sdtDBN.

function `getVal(`$X_j[t]$*, staticObs, dynObs*`)`**:**
  if $X_j[t]$ *does not have parents in the network* **then**
    return *False*
  for *sParent* **in** *static parents of* $X_j[t]$ **do**
    if *sParent* **not observed in** *staticObs* **then**
      return *False*
  for *dParent* **in** *dynamic parents of* $X_j[t]$ **do**
    if *dParent* **not observed in** *dynObs* **then**
      ret = `getVal(`*dParent, staticObs, dynObs*`)`
      if *ret* == *False* **then**
        return *False*
  Randomly determine/sample a value for $X_j[t]$, using the probabilities of its distribution. Update *dynObs* with the estimated value.
  return *True*

In Algorithm 4, the dynamic parents of each node should be searched in a topological order, to guarantee that, when Algorithm 4 is recursively called for a certain node, its parent from the same timestep already has its value estimated.

When determining the distribution of a node $X_i[t]$, Algorithm 4 is used to estimate the values of the parents of $X_i[t]$ without observations in *dynObs*. Regarding complexity, the worst-case scenario happens when only the dynamic nodes in the first m timesteps have observations in *dynObs* and every node used in the DFS has $p+b+1$ parents (p dynamic from the previous timestep, b static and 1 from the same timestep). In this situation, Algorithm 4 is called p times (for each parent of $X_i[t]$), analyzing $\sum_{k=0}^{t-1-m} p^k$ nodes in each call. To estimate a node's value, its $p+b+1$ parents are checked. Given the previous explanation, the complexity of determining the distribution of a node $X_i[t]$ of an sdtDBN is

$$\mathcal{O}\left(p \sum_{k=0}^{t-1-m} p^k\right) \times \mathcal{O}(p+b+1) \approx \mathcal{O}(p^{t-m+1} + bp^{t-m}). \tag{17}$$

4.2. Graphical user interface and publicly available implementations

For the sdtDBN framework to be available to all kinds of users, including non-experts in computer science, there is provided a graphical user interface (GUI) for the sdtDBN program. The GUI is composed by seven tabs, which allow a user to exploit all capabilities of the sdtDBN framework.

The sdtDBNs are implemented in Java, being the source code available at https://github.com/ttlion/sdtDBN_code. The latest executable version, provided in a JAR file, can be obtained at https://ttlion.github.io/sdtDBN. The GUI is implemented in Python, being the source code available at https://github.com/ttlion/sdtDBNsGUI_code. The latest standalone executable versions of the GUI can be obtained at https://ttlion.github.io/sdtDBNsGUI (for Windows and Linux). The websites with the executable versions of the programs also provide examples detailing how to insert the inputs of the programs and interpret the several outputs.

5. Assessment of the ALS dataset

5.1. Data preprocessing

The sdtDBNs (see Section 4) are applied to tackle ALS disease progression, being learned using a pretreated version of the ALS dataset (see Section 2). This pretreated version is called "ALS dataset" throughout Section 5, and must be preprocessed, to be in the proper format for learning sdtDBNs.

The preprocessing of static data consists of the selection (done by eliminating variables with high quantity of missing data) and discretization (done as proposed by ALS experts) of data, which are presented in Table 2. The preprocessing of dynamic data starts by creating the sub-datasets before and after NIV, splitting each patient's data according to NIV having been applied, or not, when the data was obtained. The procedure of Fig. 2 is then applied to each sub-dataset.

TABLE 2. SELECTION AND DISCRETIZATION OF THE VARIABLES OF THE ALS DATASET (ORANGE: STATIC; PINK: DYNAMIC). THE VARIABLES WITH † ARE ONLY USED IN THE ANALYSES BEFORE NIV.

Variable	Discretization (label: respective elements)
Gender	1: male; 2: female
BMI	1: [0,20[; 2: [20,25[; 3: [25,30[; 4: [30,+∞[
Familiar History MND	1: yes; 2: no; 3: unknown
Age at onset (years)	1: [0,30[; 2: [30,50[; 3: [50,70[; 4: [70,+∞[
Disease duration (months)	1: [0,6]; 2:]6,12]; 3:]12,18]; 4:]18,36]; 5:]36,+∞[
El Escorial reviewed criteria	1: definitive; 2: probable; 3: possible; 4: progressive muscular atrophy
Onset form	1: spinal; 2: bulbar; 3: respiratory/axial; 4: mixed; 5: frontotemporal degeneration
C9orf72	1: yes; 2: no; 3: unknown
ALS-FRS	1: {0,...,11}; 2: {12,...,23}; 3: {24,...,35}; 4: {36,...,40}
ALS-FRSb	1: {0,1,2,3}; 2: {4,5,6,7}; 3: {8,9,10,11}; 4: {12}
ALS-FRSsUL	1: {0,1,2,3}; 2: {4,5,6,7}; 3: {8,9,10,11}; 4: {12}
ALS-FRSsLL	1: {0,1,2,3}; 2: {4,5,6,7}; 3: {8,9,10,11}; 4: {12}
R	1: {0,1,2,3}; 2: {4,5,6,7}; 3: {8,9,10,11}; 4: {12}
ALS-FRS-R questions	1: {0}; 2: {1}; 3: {2}; 4: {3}; 5: {4}
FVC†	1: [0,40[; 2: [40,60[; 3: [60,80[; 4: [80,100]
MIP†	1: [0,40[; 2: [40,60[; 3: [60,100]
MEP†	1: [0,40[; 2: [40,60[; 3: [60,80[; 4: [80,100]
PhrenMeanAmpl†	1: [0; 0,4[; 2: [0,4;+∞[

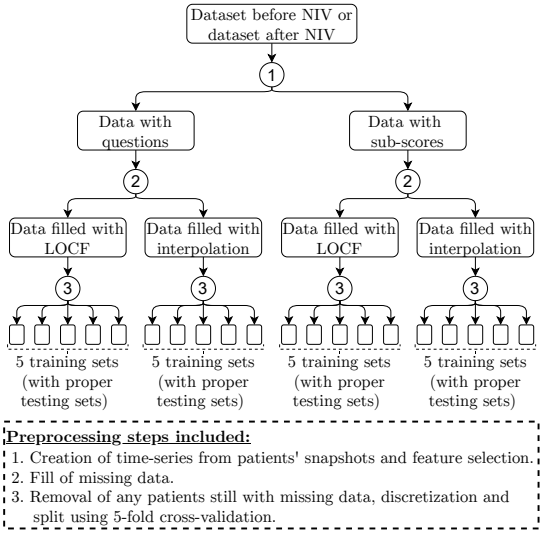


Figure 2. Preprocessing of the datasets before and after NIV.

Fig. 2 shows that, since the dataset has snapshots of the patients' consultations, but sdtDBNs describe time-series, data is converted to time-series, assuming that consecutive timesteps are separated by three months. The ALS-FRS-R sub-scores and questions are split into distinct sub-datasets, for the sdtDBNs not to learn relations among a score and its questions, being the respiratory tests included in all datasets before NIV. Table 2 shows the dynamic features selected. Missing data is filled using two methods, last observation carried forward (LOCF) and linear interpolation, in order to improve the reliability of the results. Patients with missing data after the filling procedure are removed from the dataset, as they have at least one variable without any observation. The discretization of dynamic data (see Table 2) is done after filling data, for the interpolation to be done with the real values. It is used 5-fold cross-validation to get the training and

testing sets with dynamic data, being each corresponding set with static data obtained by selecting, from the static dataset, the data of the patients in the respective dynamic set.

5.2. Study of the whole ALS dataset and division into progression groups

This section studies the whole ALS dataset and also gets the differences among the several kinds of patients, by splitting them into three progression groups, according to their *progression rates*. A patient's *progression rate* is obtained by

$$progression\ rate = \frac{48 - \text{ALS-FRS-R}_{\text{First consultation}}}{\alpha}, \quad (18)$$

where α is the number of months between the initial symptoms and the first consultation, and 48 is used as it is the maximum possible value of ALS-FRS-R. The patients with the lowest 25% and the highest 25% *progression rates* compose, respectively, the slow and fast progression groups. The remaining patients compose the average progression group.

Applying the preprocessing from Section 5.1 to each of the four mentioned groups (the whole dataset and the three progression groups), there are obtained, per group, ten training sets (see Fig. 2) for each of the following **four scenarios**: **(i)** before NIV with questions; **(ii)** before NIV with sub-scores; **(iii)** after NIV with questions; **(iv)** after NIV with sub-scores. For each scenario of each group, three assessments are done, which are presented next.

The first assessment predicts the values of the questions and sub-scores of ALS-FRS-R using sdtDBNs. From each training set, stationary sdtDBNs are learned (to avoid overfitting the training sets), using $\{m = 1, p = 2, b = 1, T = 8\}$ (see Section 4) with the LL and MDL scores.

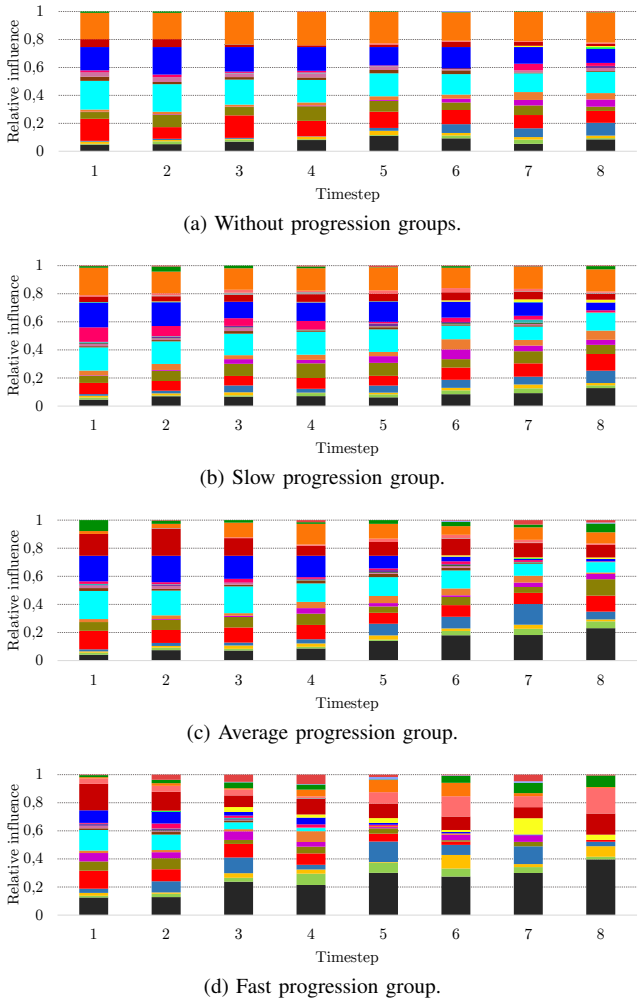
For a certain variable $X[t]$ in a scenario W of a group G, it is determined the accuracy of the predictions (the fraction of correct predictions of $X[t]$). There are also determined the sensitivity and the AUC, by classifying a value as negative if it is higher than Q_1 and as positive if not, where Q_1 is the first quartile of the values of $X[t]$ considering all testing sets of scenario W of group G (the sensitivity and the AUC are not determined if Q_1 is the highest possible value of $X[t]$).

Table 3 shows the results of the predictions of sdtDBNs learned with MDL in the scenarios with sub-scores, providing the mean values of the results of all timesteps. The results are, in general, above 75%, which demonstrates that sdtDBNs are competitive with state-of-the-art works [13, 15, 16]. The division into progression groups does not improve the performance, because the models overfit the training data, due to the low quantity of data of each group.

The second assessment graphically determines the influence of each variable in each timestep of the sdtDBNs, based on the reasoning that a variable X has more influence in a timestep t than a variable Y if X is a parent of more variables in t than Y. From the training sets filled with LOCF (only LOCF is used, as the results with interpolation are similar), non-stationary sdtDBNs are learned using the LL score with $\{m = 1, p = 2, T = 8\}$ and either $b = 1$ or $b = 2$ ($b = 2$ is only used in the scenarios with sub-scores, to learn sdtDBNs in reasonable time). The learned sdtDBNs are restricted so that a variable can never be a parent of itself in the next timestep (those relations are intuitively known).

For each scenario of each group, it is counted the number of children that each variable has in every timestep of the respective sdtDBNs. The counts done in each scenario of each group are normalized per timestep, being presented in stacked bar charts, where the larger the bar of a variable, the higher the variable's influence in the corresponding timestep.

The influence of each variable in the scenarios before NIV with questions is given in Fig. 3, which shows that the influence of the variables varies according to the progression groups. Some interesting observations regarding Fig. 3 are the high influence of P9, MIP, MEP and disease duration in slow progressors, the high influence of P1, P9, MEP, BMI and disease duration in average progressors, and the high influence of P1, BMI and age at onset in fast progressors.



(a) Without progression groups.

(b) Slow progression group.

(c) Average progression group.

(d) Fast progression group.

Figure 3. Influence of each variable in every timestep of the sdtDBNs learned in the scenarios before NIV with questions.

TABLE 3. RESULTS OF THE PREDICTIONS OF THE SDTDBNS LEARNED, WITH THE MDL SCORE, IN THE SCENARIOS WITH SUB-SCORES.

Metric		Accuracy (%)				Sensitivity (%)				AUC (%)			
Group		None	Slow	Average	Fast	None	Slow	Average	Fast	None	Slow	Average	Fast
Before NIV	ALS-FRS	79,72	82,40	77,43	76,61	77,74	84,50	71,04	70,10	86,57	89,16	82,82	84,87
	ALS-FRSb	83,12	88,15	79,59	78,33	86,92	83,39	83,23	74,07	91,65	90,09	90,75	86,54
	ALS-FRSsUL	79,02	80,04	76,54	71,87	79,39	78,70	71,54	70,58	88,21	87,39	84,51	83,46
	ALS-FRSsLL	78,53	80,89	73,99	68,33	86,07	78,60	71,12	64,14	89,79	87,55	84,57	80,47
	R	83,46	87,29	79,30	78,46	61,32	x	53,49	70,82	77,52	x	72,55	79,17
After NIV	ALS-FRS	81,15	78,34	79,50	82,82	78,91	79,70	75,42	74,91	88,49	87,31	86,82	86,89
	ALS-FRSb	82,20	81,90	82,28	81,14	89,00	86,53	81,24	88,22	93,88	92,24	89,87	93,89
	ALS-FRSsUL	81,50	81,22	80,30	81,00	84,58	80,73	80,20	85,59	90,36	88,52	88,91	90,19
	ALS-FRSsLL	82,10	82,52	81,32	75,58	83,29	82,03	81,49	80,99	90,50	90,76	89,80	89,38
	R	81,17	80,10	82,03	78,03	75,88	72,26	76,80	79,96	83,72	81,96	84,03	86,11

The third assessment determines the correlations among variables in the disease progression. The reasoning used is that, given any three variables X, Y and Z of an sdtDBN, X is more correlated with Y than with Z if there are more edges between X and Y than between X and Z, throughout all timesteps of the sdtDBN. As the graphical display of sdtDBNs is used, there are learned in this third assessment the same sdtDBNs learned in the second assessment.

For each scenario of every group, it is counted the number of edges that each variable has with each of the other variables in the learned sdtDBNs (independently of the edges' directions, as both $X \rightarrow Y$ and $Y \rightarrow X$ indicate a correlation between variables X and Y). The counts done in each scenario of each group are normalized per variable and presented in tables where each column has the normalized values of a variable (thus representing the correlations of that variable with all other variables). The correlations between the sub-scores after NIV are given in Table 4.

TABLE 4. CORRELATIONS BETWEEN SUB-SCORES OF THE SDTDBNS LEARNED IN THE SCENARIOS AFTER NIV WITH SUB-SCORES.

(a) Without progression groups.

	ALS-FRS	ALS-FRSb	ALS-FRSsUL	ALS-FRSsLL	R
ALS-FRS	—	27,21%	23,88%	21,08%	7,38%
ALS-FRSb	39,38%	—	30,18%	28,83%	37,81%
ALS-FRSsUL	29,69%	25,93%	—	27,93%	27,29%
ALS-FRSsLL	24,12%	22,79%	25,70%	—	27,52%
R	6,80%	24,07%	20,23%	22,16%	—
Sum per column:	100%	100%	100%	100%	100%

(b) Slow progression group.

	ALS-FRS	ALS-FRSb	ALS-FRSsUL	ALS-FRSsLL	R
ALS-FRS	—	27,96%	25,27%	25,00%	17,70%
ALS-FRSb	36,88%	—	34,34%	36,18%	33,43%
ALS-FRSsUL	24,38%	25,12%	—	20,73%	23,88%
ALS-FRSsLL	25,63%	28,12%	22,03%	—	25,00%
R	13,13%	18,80%	18,36%	18,09%	—
Sum per column:	100%	100%	100%	100%	100%

(c) Average progression group.

	ALS-FRS	ALS-FRSb	ALS-FRSsUL	ALS-FRSsLL	R
ALS-FRS	—	28,26%	26,90%	14,96%	12,27%
ALS-FRSb	39,79%	—	28,71%	29,92%	37,96%
ALS-FRSsUL	33,61%	25,48%	—	32,87%	23,61%
ALS-FRSsLL	15,67%	22,25%	27,56%	—	26,16%
R	10,93%	24,01%	16,83%	22,24%	—
Sum per column:	100%	100%	100%	100%	100%

(d) Fast progression group.

	ALS-FRS	ALS-FRSb	ALS-FRSsUL	ALS-FRSsLL	R
ALS-FRS	—	40,90%	48,73%	43,77%	25,91%
ALS-FRSb	38,74%	—	27,97%	27,76%	53,44%
ALS-FRSsUL	23,33%	14,13%	—	14,95%	5,26%
ALS-FRSsLL	24,95%	16,70%	17,80%	—	15,38%
R	12,98%	28,27%	5,51%	13,52%	—
Sum per column:	100%	100%	100%	100%	100%

Some conclusions regarding Table 4 are the high correlation of ALS-FRSb with the scores after NIV and the low correlation of R with the scores after NIV (see the ALS-FRSb and R rows, respectively). The diagonals of the tables of the third assessment are never considered, as the sdtDBNs are restricted for a variable never to be a parent of itself.

5.3. Analysis of some clinically relevant questions

For the assessments using sdtDBNs to have clinical impact, this section presents three clinical questions (proposed by ALS experts) and how they are answered using sdtDBNs. ***Question 1: which variables are the most associated to a patient needing NIV?*** To answer this question, it is added to the dataset a dynamic variable named NIV. In a consultation of a patient, this variable is 1 if the patient already has NIV, and 0 otherwise. The question is answered using the third assessment of Section 5.2 to determine the correlation of the NIV variable with each variable of the dataset. The results provide high correlation of NIV with ALS-FRS and BMI. ***Question 2: which variables are the most important in each year of the patients' progression?*** Assuming that the years of a patient's progression start in the first consultation, this question is answered using the second assessment of Section 5.2, applying the graphical reasoning to years instead of timesteps. As consecutive timesteps are separated by three months (see Section 5.1), the conversion from timesteps to years is direct. The results present high influence of ALS-FRSsUL and ALS-FRSsLL in slow and average progressors and high influence of ALS-FRSb in fast progressors. ***Question 3: considering the sub-score of ALS-FRS with the lowest value in the first consultation of a patient, how are its relations with the remaining variables, throughout that patient's progression?*** To get the answer to this question, patients are divided into three sub-groups, according to the sub-score with the lowest value in the first consultation being ALS-FRSb, ALS-FRSsUL or ALS-FRSsLL. For each sub-group, the third assessment of Section 5.2 is applied. The third question is answered by selecting, from the table obtained for each sub-group, the column of the sub-score with the lowest value in the first consultation (which varies per sub-group). The previous procedure is applied to each progression group (see Section 5.2). The results show, in general, high correlation of ALS-FRSb with FVC, and high correlation of ALS-FRSsUL and ALS-FRSsLL with MEP.

6. Conclusions and future work

This Thesis proposes the sdtDBN framework (see Section 4), which learns optimal DBNs with static and dynamic variables. The inclusion of static variables is a huge improvement on standard DBNs (which only include dynamic variables), being particularly relevant when studying medical datasets, which usually have static indicators, such as the patients' gender. As sdtDBNs cannot predict values of static variables (they never have parents in learned sdtDBNs), future work can extend the sdtDBN framework, for sdtDBNs to learn the optimal parents of static variables, which may not be a simple task (for instance, considering all combinations of dynamic nodes from all timesteps is unfeasible).

The sdtDBNs are employed to assess ALS disease progression (see Section 5). The prediction of the functional decline of patients shows that sdtDBNs are competitive with state-of-the-art models. The analyses of the graphical display of sdtDBNs present interesting relations among variables, which vary according to the patients' progression groups. Finally, the answers to the questions proposed by ALS experts provide results with clinical relevance. As future work, sdtDBNs can be applied in the preprocessing procedure (for example, using sdtDBNs to fill missing data) and in the division of patients into progression groups (for instance, representing the progression groups using a static variable). Future work can also impose restrictions in the sdtDBNs different from the ones used in Section 5.2, and assess outcomes not covered in this work.

References

[1] M. A. Van Es, O. Hardiman, A. Chiò, A. Al-Chalabi, R. J. Pasterkamp, J. H. Veldink, and L. H. Van den Berg, "Amyotrophic lateral sclerosis," *The Lancet*, vol. 390, no. 10107, pp. 2084–2098, 2017.

[2] N. Simon, M. Turner, S. Vucic, A. Al-Chalabi, J. Shefner, C. Lomen-Hoerth, and M. Kiernan, "Quantifying disease progression in amyotrophic lateral sclerosis," *Annals of Neurology*, vol. 76, no. 5, pp. 643—657, 2014.

[3] C. Heffernan, C. Jenkinson, T. Holmes, H. Macleod, W. Kinnear, D. Oliver, N. Leigh, and M. Ampong, "Management of respiration in MND/ALS patients: An evidence based review," *Amyotrophic Lateral Sclerosis*, vol. 7, no. 1, pp. 5–15, 2006.

[4] K. Murphy, "Dynamic Bayesian networks: Representation, inference and learning," Ph.D. dissertation, University of California, Berkeley, California, USA, 2002.

[5] N. Friedman, K. Murphy, and S. Russell, "Learning the structure of dynamic probabilistic networks," in *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, ser. UAI'98. Morgan Kaufmann Publishers Inc., 1998, pp. 139–147.

[6] H. Lähdesmäki and I. Shmulevich, "Learning the structure of dynamic Bayesian networks from time series and steady state measurements," *Machine Learning*, vol. 71, no. 2–3, pp. 185–217, 2008.

[7] L. Song, M. Kolar, and E. P. Xing, "Time-varying dynamic Bayesian networks," in *Proceedings of the 22nd International Conference on Neural Information Processing Systems*, ser. NIPS'09. Curran Associates Inc., 2009, pp. 1732–1740.

[8] J. W. Robinson and A. J. Hartemink, "Learning non-stationary dynamic Bayesian networks," *Journal of Machine Learning Research*, vol. 11, pp. 3647–3680, 2010.

[9] G. Trabelsi, P. Leray, M. Ben Ayed, and A. Alimi, "Dynamic MMHC: A local search algorithm for dynamic Bayesian network structure learning," in *Advances in Intelligent Data Analysis XII*. Springer, Berlin, Heidelberg, 2013, pp. 392–403.

[10] J. L. Monteiro, S. Vinga, and A. M. Carvalho, "Polynomial-time algorithm for learning optimal tree-augmented dynamic Bayesian networks," in *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*, ser. UAI'15. AUAI Press, 2015, pp. 622–631.

[11] M. Sousa and A. M. Carvalho, "Polynomial-time algorithm for learning optimal BFS-consistent dynamic Bayesian networks," *Entropy*, vol. 20, no. 4, 2018.

[12] B. Marin, P. Couratier, S. Arcuti, M. Copetti, A. Fontana, M. Nicol, M. Raymondeau, G. Logroscino, and P.-M. Preux, "Stratification of ALS patients' survival: a population-based study," *Journal of Neurology*, vol. 263, no. 1, pp. 100–111, 2015.

[13] A. Taylor, C. Fournier, M. Polak, L. Wang, N. Zach, M. Keymer, J. D. Glass, and D. Ennist, "Predicting disease progression in amyotrophic lateral sclerosis," *Annals of Clinical and Translational Neurology*, vol. 3, no. 11, pp. 866–875, 2016.

[14] H. K. Van der Burgh, R. Schmidt, H.-J. Westeneng, M. A. de Reus, L. H. Van den Berg, and M. P. Van den Heuvel, "Deep learning predictions of survival based on MRI in amyotrophic lateral sclerosis," *NeuroImage: Clinical*, vol. 13, pp. 361–369, 2017.

[15] A. V. Carreiro, P. M. Amaral, S. Pinto, P. Tomás, M. de Carvalho, and S. C. Madeira, "Prognostic models based on patient snapshots and time windows: Predicting disease progression to assisted ventilation in amyotrophic lateral sclerosis," *Journal of Biomedical Informatics*, vol. 58, pp. 133–144, 2015.

[16] S. Pires, M. Gromicho, S. Pinto, M. de Carvalho, and S. C. Madeira, "Predicting non-invasive ventilation in ALS patients using stratified disease progression groups," in *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE, 2018, pp. 748–757.

[17] M. Van Gerven, B. Taal, and P. J. Lucas, "Dynamic Bayesian networks as prognostic models for clinical patient management," *Journal of Biomedical Informatics*, vol. 41, no. 4, pp. 515–529, 2008.

[18] A. Zandonà, R. Vasta, A. Chiò, and B. Di Camillo, "A dynamic Bayesian network model for the simulation of amyotrophic lateral sclerosis progression," *BMC Bioinformatics*, vol. 20, no. S4, 2019.

[19] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, 3rd ed. Morgan Kaufmann Publishers Inc., 2011.

[20] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical Machine Learning Tools and Techniques*, 4th ed. Morgan Kaufmann Publishers Inc., 2016.

[21] P. Cabena, P. Hadjinian, R. Stadler, J. Verhees, and A. Zanasi, *Discovering Data Mining: From Concept to Implementation*. Prentice Hall, 1998.

[22] D. Pyle, *Data Preparation for Data Mining*. Morgan Kaufmann Publishers Inc., 1999.

[23] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[24] G. F. Cooper, "The computational complexity of probabilistic inference using Bayesian belief networks (research note)," *Artificial Intelligence*, vol. 42, no. 2–3, pp. 393–405, 1990.

[25] C. Chow and C. Liu, "Approximating discrete probability distributions with dependence trees," *IEEE Transactions on Information Theory*, vol. 14, no. 3, pp. 462–467, 1968.

[26] A. M. Carvalho, "Scoring functions for learning Bayesian networks," INESC-ID, Lisbon, Portugal, Tech. Rep., 2009.

[27] N. Friedman, "The Bayesian structural EM algorithm," in *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, ser. UAI'98. Morgan Kaufmann Publishers Inc., 1998, pp. 129–138.

[28] J. Edmonds, "Optimum branchings," *Journal of Research of the National Bureau of Standards*, vol. 71B, no. 4, pp. 233–240, 1967.