Sparse dynamic Bayesian network estimation using generalized linear models

João Santos, DEEC, Instituto Superior Técnico

***Abstract*—Dynamic Bayesian Networks are probabilistic graphical models used to predict the evolution of stochastic processes. These models can be trained from multivariate time series data to uncover interesting temporal relationships between measured variables. However, optimal training algorithms are computationally prohibitive, inspiring the development of heuristic techniques. This dissertation introduces sDBN, an alternative training algorithm with better computational complexity. The proposed method handles both stationary and non-stationary models and is flexible to a specified Markov lag. Empirical results show that the algorithm achieves great network identification, accomplishing up to perfect F_1 scores in aritificial datasets with a considerable number of dimensions. Using simulated data, `sDBN` beats state-of-the-art dynamic Bayesian network training algorithms both in terms of structure quality and training time. Tests in benchmark public datasets show that sDBN is also competitive in time-series classification. Using Ankylosing Spondylitis patient data from Reuma.pt, a national rheumatological registry, the new method recovers intelligible models and successfully predicts disease progression.**

Keywords: multivariate time series, dynamic Bayesian networks, structure learning, high-dimensional data, data mining

I. Introduction

DATA science has become, over the last few years, ubiquitous in society. Data is currently being collected everywhere due to the digitalization of society. Using concepts from statistics and computer science, this field of study has exponentially increased in popularity in the last decade. Under this subject, two distinct areas can be identified: machine learning and data mining. Machine learning studies a set of techniques that allow for computers to learn some task. On the other hand, data mining aims to extract patterns from data and help in decision making. The focus is not on performing a task, but instead on analyzing the data and getting insights.

One type of datasets are multivariate time series (MTS), representing how a set of multiple variables evolves over time. MTS datasets frequently arise in several contexts like meteorology, robotics, economics and finance, and electronic health records. The increase in the availability of this type of data justifies the growing interest in data mining techniques for the analysis of time series.

Dynamic Bayesian networks (DBNs) are a class of mathematical models that can be used for the analysis of time-series. They are a probabilistic graphical model (PGM) and therefore can be easily schematized using a graph and provide interpretable information on the relationships between measured variables. This is especially important in medical applications, where decisions should follow a well-defined rationale instead of simply being outputs of black-box type models. Additionally, these models are flexible and allow the computation of complex probability queries through inference algorithms. Another advantage of these kinds of mathematical models is the possibility of fine-tuning using domain expert knowledge. DBNs can also be trained automatically from data, but exact optimization algorithms are computationally expensive. To counteract this, the training typically includes the use of state-of-the-art heuristic methods to restrict the search space.

Recent work on discrete Bayesian network (BN) training leverage generalized linear models (GLMs) as an approximate parametrization to discrete probability tables. GLMs are also the heart of some deep neural networks, and the recent advances in this field provide a sophisticated computational framework that can leverage modern hardware to accelerate the training procedure meaningfully. Training BNs approximating them as a collection of GLMs profits from these novel techniques and are mandatory in high-dimensional contexts.

Although plenty of literature is found on identifying static models, extending these techniques to include temporal information is still unexplored. Therefore, this article focuses on developing alternative heuristic methods for training DBNs leveraging GLMs, allowing the possibility of training of these kinds of models using high-dimensional MTS, *i.e.*, time-series with several measured variables per timestep with few observations.

This document is organized as follows. Section II establishes a theoretical background, providing a brief overview of generalized linear models, and introducing three probabilistic graphical models: Bayesian networks, dynamic Bayesian networks and Bayesian multinets. Section III exposes the new method and provides details of the developed implementation. Section IV reports empirical results that validate the algorithm using synthetic datasets, real benchmark time series and data from Portugal's national registry of Ankylosing Spondylitis patients. Section V presents some conclusions.

II. Background

A. Generalized Linear Models

A generalized linear model [1] is a model with parameters $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_n)$ that relates the response variable y with the features $\mathbf{X} = (1, x_1, \ldots, x_n)$, via a linear mapping $Z = \mathbf{X}^T\boldsymbol{\beta}$, a linking function $g(Z)$ and an assumption on the distribution of the response variables $y \sim \Psi(\theta)$, where Ψ is a distribution of the exponential family with parameter $\boldsymbol{\theta} = g(Z)$. With $\Psi(\theta) = \mathcal{N}(\boldsymbol{\theta}, \sigma^2)$ and $g(Z) = Z$, the

GLM specializes to linear regression. Different choices of distribution and link function lead to different, well-known models.

These models are estimated from observational data using a procedure called maximum likelihood estimation (MLE) [2]. This consists of solving the optimization problem given by

$$\underset{\boldsymbol{\beta}}{\text{maximize}} \quad \prod_{i=1}^{M} P(y^{(i)} \mid \boldsymbol{\beta}, \mathbf{X}^{(i)}),$$

i.e., the likelihood that the model with a given set of parameters $\boldsymbol{\beta}$ generates D. Typically, instead of minimizing the likelihood directly, the equivalent problem of minimizing the log-likelihood is often considered.

Let $\Psi(\boldsymbol{\theta}) = \text{Bernoulli}(\boldsymbol{\theta})$ and the link function be the logistic function. This generalized linear model is called logistic regression, and it is used to predict the outcome of a binary response variable from a set of features. The characteristic probability mass function of logistic regression is given by

$$P(y = k \mid \mathbf{X}, \boldsymbol{\beta}) = \left(\frac{\exp(\mathbf{X}^\top \boldsymbol{\beta})}{1 + \exp(\mathbf{X}^\top \boldsymbol{\beta})}\right)^k \left(\frac{1}{1 + \exp(\mathbf{X}^\top \boldsymbol{\beta})}\right)^{1-k} \tag{1}$$

where $k \in \{0, 1\}$.

Multinomial Logistic Regression extends logistic regression to handle the prediction of categorical distributed random variables, allowing the response variables to take r possible categories. Therefore, $\Psi(\boldsymbol{\theta})$ is set to be the categorical distribution. The link function has to specify every parameter independently. The probability mass function for multinomial logistic regression is obtained directly from the categorical distribution

$$P(y = k \mid \mathbf{X}, \boldsymbol{\beta_1}, \ldots, \boldsymbol{\beta_r}) = \frac{\exp(\mathbf{X}^\top \boldsymbol{\beta_k})}{\sum_{i=1}^{r} \exp(\mathbf{X}^\top \boldsymbol{\beta_i})}, \tag{2}$$

where $k \in \{1, \ldots, r\}$ and $\boldsymbol{\beta_i} \in \mathbb{R}^{n+1}$ are the model parameters associated with category i.

Choosing a new beta $\boldsymbol{\beta'} = \boldsymbol{\beta} + \alpha \mathbb{1}$ lead to exactly the same probability distribution. Due to this, when identifying these models from data, a model parameter is usually constrained to a given value, effectively eliminating this ambiguity.

Maximum Likelihood Estimation, although effective, is prone to overfit the model to the observed data. In order to fix this issue, typically, regularization techniques are used, adding a new term to the cost function achieving a smaller model variance at the cost of adding some bias to the prediction.

Regression using the least absolute shrinkage and selection operator (LASSO) [3] consists of imposing an ℓ_1-norm constraint on the model parameters $||\boldsymbol{\beta}||_1 \leq t$. Like Ridge, the parameter set is typically obtained by solving the equivalent unconstrained optimization problem given by

$$\underset{\boldsymbol{\beta}}{\text{minimize}} \quad -\frac{1}{M}\ell(\boldsymbol{\beta} \mid D) + \lambda||\boldsymbol{\beta}||_1, \tag{3}$$

where ℓ is the log-likelihood function.

LASSO is a special case of regularization methods using ℓ_q-norms because it yields a sparse parameter vector, *i.e.*, some irrelevant or redundant parameters are set exactly to 0. This property is desirable because it applies feature selection while training the model, remaining a convex optimization problem.

Group LASSO [4] is a regularization technique used to obtain sparse solutions where predictors are grouped, and a group is either considered to explain the response variable or discarded, setting all the features in the group to zero. Let the model parameters $\boldsymbol{\beta}$ be split in J groups of coefficients. These groups may have different lengths.

Let $\boldsymbol{\gamma} \in \mathbb{R}^n$ be an arbitrary vector of dimension $n \geq 1$ and $\mathbf{K}$ be a symmetric positive definite matrix of dimension $d \times d$. Consider the norm

$$||\boldsymbol{\gamma}||_{\mathbf{K}} = \sqrt{\boldsymbol{\gamma}^\top \mathbf{K} \boldsymbol{\gamma}}.$$

Sparse group regularization is applied by solving the unconstrained optimization problem given by

$$\underset{\boldsymbol{\beta}}{\text{minimize}} \quad -\frac{1}{M}\ell(\boldsymbol{\beta} \mid D) + \lambda \sum_{i=1}^{J} ||\boldsymbol{\beta_j}||_{\mathbf{K_i}}, \tag{4}$$

where $\mathbf{K_1}, \ldots, \mathbf{K_J}$ are positive definite matrices of the appropriate dimension to be applied with the norm of the group. Typical values for the $\mathbf{K_i}$ matrices are identity matrices, transforming the regularization term in the sum of simple ℓ_2-norms for each group.

B. Bayesian networks

A Bayesian network is a representation of a joint probability distribution over a set of random variables using a directed graph [5]. It can be defined as a triple $B = (\boldsymbol{X}, G, \boldsymbol{\theta})$, where $\boldsymbol{X} = (X_1, X_2, \ldots, X_n)$ is a vector of random variables, $G = (\boldsymbol{X}, E)$ is a directed acyclic graph (DAG) which nodes are random variables, and the edges encode dependencies between them and $\boldsymbol{\theta}$ is a set of parameters that quantify the conditional probability tables (CPDs) of the network.

Let $pa(X_i)$ denote the parent nodes of the random variable X_i as defined in G. The joint probability distribution is then defined by the network B as

$$P_B(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P_B(X_i \mid pa(X_i)). \tag{5}$$

The structure G of the Bayesian network uniquely defines the joint probability over all random variables. However, how each conditional probability distribution (CPD) is computed remains open. Discrete Bayesian networks assume a categorical distribution and use tables to parameterize the CPDs, defining $\boldsymbol{\theta} = \{\theta_{ijk}\}$, where $i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, q_i\}$ and $k \in \{1, \ldots, r_i\}$. The parameters θ_{ijk} are defined by

$$\theta_{ijk} = P_B(X_i = x_{ik} \mid pa(X_i) = \boldsymbol{w_{ij}}), \tag{6}$$

and specify the probability of X_i assuming the value x_{ik}, given a parent configuration $\boldsymbol{w_{ij}}$. Accordingly, q_i is the number of different configurations for the parents of X_i, and r_i is the number of different values that X_i takes in the dataset. This way, the parameters have to be specified for every possible case, and there is no formula defined to obtain them.

There are two steps in learning a Bayesian network – learning the structure of the G graph and learning the parameters set $\boldsymbol{\theta}$.

The parameters can be estimated using a maximum likelihood estimation (MLE) approach, and are given by

$$\hat{\theta}_{ijk} = \frac{N_{ijk}}{\sum_{k=1}^{r_i} N_{ijk}} = \frac{N_{ijk}}{N_{ij}}, \tag{7}$$

where N_{ijk} is the number of times a configuration of $X_i = x_{ik}$ and $pa(X_i) = \boldsymbol{w_{ij}}$ appears in the dataset, and N_{ij} is the number of times a configuration of $pa(X_i) = \boldsymbol{w_{ij}}$ appears in the dataset, disregarding the value of the RV X_i itself.

Regarding structure learning, there are three main categories of algorithms – constraint based, score based and hybrid.

The first methods try to estimate the conditional independence between random variables from data using statistical hypothesis testing. The PC algorithm [6] is considered state-of-the-art in this technique.

The second category, which is more used, relies on a score function that evaluates the fit of a structure to the data used. The strategy for training is to search for the best structure, given a particular score, from the space of all possible structures, therefore solving the optimization problem

$$\begin{aligned} \underset{G}{\text{maximize}} \quad & \phi(G \mid D) \\ \text{subject to} \quad & G \text{ is a DAG}, \end{aligned} \tag{8}$$

where ϕ is a scoring function, G is the structure of the network, and D is a dataset with observations of the process to model. The constraint imposes that the problem is non-convex and this is an NP-hard problem [7].

Hybrid methods combine these methodologies and usually work by restricting the search space based on a measure of independence between nodes.

To solve the optimization problem in Eq. (8), a score and a heuristic search procedure must be defined. A basic score metric that can be considered is the log-likelihood (LL). Estimating the parameters using their MLE estimation as shown in Eq. (7), the likelihood of the structure can be used as a comparison measure. The log-likelihood score of a structure G for a given dataset D is defined as

$$\phi_{\text{LL}}(G, D) = \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk} \log \frac{N_{ijk}}{N_{ij}}. \tag{9}$$

A common problem using the log-likelihood score is overfitting since it usually returns a graph with all the nodes connected that explains the given data exceptionally well but fails to generalized to unseen data. This is unwanted behavior, and therefore a new score that penalizes complex network structures was introduced based on model selection criteria. The minimum descriptor length (MDL) [8] metric is defined by

$$\phi_{\text{MDL}}(G, D) = \phi_{\text{LL}}(G, D) - \frac{1}{2}|B| \log M, \tag{10}$$

where $|B|$ is the number of elements in the set $\boldsymbol{\theta}$ and is given by

$$|B| = \sum_{i=1}^{n} (r_i - 1) q_i. \tag{11}$$

One common way to conduct the searching procedure is to use greedy hill-climbing. Starting with an empty structure (a graph with no edges), each iteration executes the operation leading to the best scoring function increment. Typically, three operations are considered possible: edge addition, edge removal, or edge reversal. Naturally, since the interest is finding an acyclic graph, operations that would result in a cycle are not allowed. The algorithm stops when no edge addition results in a positive increment.

More recently, ℓ_1-norm based optimization was applied to structure learning in an attempt to obtain sparse Gaussian Bayesian networks [9]. Gu *et al.* [10] extend this approach to general discrete Bayesian networks.

C. Dynamic Bayesian networks

Dynamic Bayesian Networks are an extension of Bayesian networks used to model temporal processes. Like Bayesian networks, these probabilistic graphical models define a joint probability distribution over a set of random variables. However, these variables are assumed to change over time due to an underlying process and are observed in several discrete time instants.

They are defined by a pair $D = (B_0, B_{\rightarrow})$ [11]. Let $\boldsymbol{X}(t) = (X_1(t), X_2(t), \ldots, X_n(t))$ denote a vector of the random variables at time $t \in [0, T]$, also known as a slice. B_0 is a Bayesian network defined over $\boldsymbol{X}(0)$, called the prior network. $B_{\rightarrow}$ is another Bayesian network defined over $\boldsymbol{X}(0:T) = \boldsymbol{X}(0) \cup \boldsymbol{X}(1) \cup \ldots \cup \boldsymbol{X}(T)$, called the transition network. In the transition network, it is always true that,

$$P(X_i(t) \mid \boldsymbol{X}(0:T)) = P(X_i(t) \mid \boldsymbol{X}(0:t)), \tag{12}$$

i.e., a random variable realization must not be dependent on future realizations.

The underlying stochastic process is said to be Markovian if future values of the random variables only depend on a fixed number of previous timesteps, *i.e.*,

$$\begin{aligned} P(\boldsymbol{X}(t+1) \mid \boldsymbol{X}(0:t+1)) = \\ P(\boldsymbol{X}(t+1) \mid \boldsymbol{X}[\,t-(\upsilon-1):t+1\,]), \end{aligned} \tag{13}$$

where υ is the Markov lag.

A random process is said to be stationary if $P(\boldsymbol{X}(t) \mid \boldsymbol{X}(0:t))$ is the same for all t. This assumption allows extrapolating the probability distribution of future timesteps by "unrolling" the network, repeating the same structure for future timesteps.

Similar to Bayesian networks, the network parameters are easily learned once a network structure has been found. Therefore, the hard part is to find the structure that best explains the data.

In general, DBNs structure learning is as hard as in BNs, because acyclicity has to be ensured in the intra-slice connections. Inter-slice connections can't generate cycles due to the constraint in Eq. (12). Monteiro *et al.* [12] introduce a polynomial-time algorithm for DBNs (`tDBN`) was achieved, restricting the intra-slice connections to a tree (just one parent) and admitting only a fixed number of parents from previous slices.

D. Bayesian multinets

Let $P(X_1, \ldots, X_N)$ be a probability distribution and $H_1, \ldots, ..., H_k$ be a collection of mutually disjoint sets of realization of the random variables in P. A Bayesian multinet is a set of k Bayesian networks, where each network B_i is a comprehensive local network associated with H_i, *i.e.*, a Bayesian network of $P(X_1, \ldots, X_N \mid H_i)$ [13]. This multinet allows the definition of a joint probability distribution

$$P(X_1, \ldots, X_N) = \sum_{i \in \mathcal{B}} P_{B_i}(X_1, \ldots, X_N \mid H_i)\, P(H_i), \quad (14)$$

where $\mathcal{B} = \{i, \ldots, k : H_i \subseteq \mathbf{X}\}$, $\mathbf{X} = \{X_1, \ldots, X_N\}$.

Bayesian multinets can be used to perform classification of given data [14]. Let C be the class random variable of some realization, and $X_1, \ldots, X_N$ be N random variables that correspond to the feature variables used in the classification. To classify the observation, the probability $P(C \mid X_1, \ldots, X_N)$ must be computed, and the class is the realization of C that has the highest probability.

To achieve this with Bayesian multinets, in a supervised learning context, a Bayesian network must be learned for each class, using a training dataset. The dataset must be split by class, and for each group, automatic techniques for learning Bayesian networks can be used to learn a network over the features $X_1, \ldots, X_N$. According to Eq. (14), the multinet defines the joint distribution

$$P(C = c_i, X_1, \ldots, X_N) = P(C = c_i)\, P_{B_i}(X_1, \ldots, X_N),$$

where B_i is the Bayesian network learned for class c_i. This distribution may then be used to compute each class probability. For a dataset with k classes,

$$P(C = c_i \mid X_1, \ldots, X_N) = \frac{P(C = c_i, X_1, \ldots, X_N)}{\sum_{l=1}^{k} P(C = c_l, X_1, \ldots, X_N)}. \quad (15)$$

Summing up, classification using a learned Bayesian multinet is done by computing the joint probability assuming each class is true and choosing the class that has the highest probability.

III. Implementation

Extending the work of Schmidt *et al.* [9] and Gu *et al.* [10] to include temporal information, this thesis proposes a three-step method for training DBNs called `sDBN`. First, edge penalization weights are estimated by fitting an unregularized multinomial logistic loss to the dataset, as suggested by the adaptative lasso method. Then, a sparse skeleton, a set of possible edges for our network, is estimated from our dataset, penalizing every possible edge using the weights computed in the previous step. Finally, a resulting network is obtained by directing the skeleton using greedy hill-climbing, complying with the temporal restrictions. In the first two steps of the training pipeline, a parametrization based on multinomial logistic regression is used to reduce the dimension of the parameter set and enable gradient based optimization with regularization, promoting sparse structures and enabling training in high-dimensional contexts.

A. Network Parametrization

Let $B = (\mathbf{X}, G, \boldsymbol{\beta})$ be a discrete Bayesian network, where $\mathbf{X} = (X_1, \ldots, X_n)$ is a vector of discrete random variables, G is any given DAG and $\boldsymbol{\beta}$ is a set of parameters. It is assumed that a discrete random variable X_j can take a value in the set $\{0, \ldots, r_j - 1\}$. In order to increase the logistic model flexibility, each random variable X_j is encoded by a set of $r_j - 1$ dummy variables that can take only two possible levels and work in an one-hot enconding configuration. For convenience, let $d_i = r_i - 1$.

Let X_j denote a random variable that belongs to our network. Let $\boldsymbol{\beta_{jli}} \in \mathbb{R}^{d_i}$ denote the vector of parameters associated with the influence of the value of X_i in the level l of X_j. If the multinomial logistic regression is used to parameterize the probability distribution of this node then

$$P(X_j = l \mid pa(X_j)) = \frac{\exp(\bar{\mathbf{x}}^\top \boldsymbol{\beta_{jl\cdot}})}{\sum_{m=1}^{r_j} \exp(\bar{\mathbf{x}}^\top \boldsymbol{\beta_{jm\cdot}})}, \quad (16)$$

where $\bar{\mathbf{x}} = \begin{bmatrix} 1 & \mathbf{x} \end{bmatrix}^\top \in \mathbb{R}^{c+1}$ is an augmented evidence vector and $\boldsymbol{\beta_{jl\cdot}} = \begin{bmatrix} \boldsymbol{\beta_{jl0}}^\top & \boldsymbol{\beta_{jl1}}^\top & \cdots & \boldsymbol{\beta_{jln}}^\top \end{bmatrix}^\top \in \mathbb{R}^{c+1}$ is the parameter vector associated with level l of X_j. To address the identifiability issues of the parameters, as discussed for regular multinomial logistic regression, it is enforced $\beta_{j\,0\,0} = 0$.

Let $\boldsymbol{\beta_{j\cdot i}} = \begin{bmatrix} \boldsymbol{\beta_{j0i}}^\top & \boldsymbol{\beta_{j1i}}^\top & \cdots & \boldsymbol{\beta_{jd_ji}}^\top \end{bmatrix}^\top \in \mathbb{R}^{r_j d_i}$, be the vector of parameters of the edge $X_i \to X_j$. Analyzing the multinomial approximation, observe that

$$X_i \notin pa(X_j) \Leftrightarrow \boldsymbol{\beta_{j\cdot i}} = \mathbf{0}, \quad (17)$$

from the conditional of the probability in Eq. (16). This equivalence allows determining a sparse graphical structure from the set of parameters if many are set exactly to zero.

In the context of dynamic Bayesian networks, the dataset used in training has a value for every random variable in each timestep. In non-stationary networks, the number of timesteps in the transition network is equal to the number of measured timesteps in the dataset. That does not happen in stationary networks. Stationary networks imply that edges are kept the same across timesteps. In practice, this is achieved by training just one timestep and then repeating the edges for every timestep needed for modeling, unrolling the network. The original dataset with measurements across all timesteps must be collapsed down to accommodate the needs for stationary network training without loss of information.

Each timestep has a collection of its own parameters $\boldsymbol{\beta}(t)$. The parameter notation from the Bayesian network approximation can be applied directly to DBNs. Then, $\boldsymbol{\beta_{jl\cdot}}(t) \in \mathbb{R}^{cT+1}$ is the vector of coefficients associated with the level l of the random variable $X_j(t)$. Accordingly, and in DBNs introducing a way to identify inter-temporal edges, $\boldsymbol{\beta_{j\cdot i}}(\tau, t)$ denotes the parameters associated with the edge $X_i(\tau) \to X_j(t)$.

For training of non-stationary networks, the evidence vector can be thought of as a collection of static evidence vectors for each timestep, *i.e.*,

$$\begin{aligned} \mathbf{x} &= (\mathbf{x}(0), \ldots, \mathbf{x}(T)) \\ &= (x_{0,1}(0), \ldots, x_{n,r_n-1}(0), \ldots, x_{0,1}(T), \ldots, x_{n,r_n-1}(T)). \end{aligned}$$

To preserve temporal causality (Eq. (12)), the parameter vector $\boldsymbol{\beta_{jl\cdot}}(t)$ has to be restricted in such a way that forbids

parents from future timesteps. Plus, edges disrespecting the Markov lag should also be disregarded. To fulfill its imposed constraints, the vector is forced to be zero everywhere except when specifying an edge from an allowed parent (in this case, it is allowed to be freely optimized). Thus,

$$\begin{aligned}\boldsymbol{\beta_{jl}}.(t) = (\beta_{jl0}(t), 0, \ldots, \boldsymbol{\beta_{jl1}}(t-\upsilon, t), \ldots, \boldsymbol{\beta_{jln}}(t-\upsilon, t),\\ \ldots, \boldsymbol{\beta_{jl1}}(t,t), \ldots, \boldsymbol{\beta_{jln}}(t,t), \ldots, 0),\end{aligned} \tag{18}$$

where υ is the Markov lag of the network.

In the specific case of stationary dynamic Bayesian networks, the transition model is assumed to be always the same. To train these kinds of models, it is only needed to determine the intra-temporal edges of one timestep and edges going into that timestep from every possible past instant according to the specified lag. Random variables from other timesteps are added as nodes to the transition network, allowing the formation of inter-temporal edges, but they only act as a stub, are not trained, and are not considered actual network nodes. A valid dynamic Bayesian network is only obtained when unrolling the transition network (joined with the prior network).

The dataset has to be collapsed in such a way that results in $\upsilon + 1$ timesteps, and all the information that may lead to an inclusion of an edge remains present. Let $\mathbf{D}$ be a $M \times cT$ matrix obtained by stacking all evidence vectors of the input dataset, *i.e.*,

$$\mathbf{D} = \begin{bmatrix} \mathbf{x}^{(0)} \\ \vdots \\ \mathbf{x}^{(M)} \end{bmatrix} = \begin{bmatrix} \mathbf{x}^{(0)}(0) & \cdots & \mathbf{x}^{(0)}(T) \\ \vdots & \vdots & \vdots \\ \mathbf{x}^{(M)}(0) & \cdots & \mathbf{x}^{(M)}(T) \end{bmatrix}.$$

The associated collapsed dataset with lag υ, $\mathbf{D}^\dagger$ is a $M(T-\upsilon+1) \times c(\upsilon+1)$ matrix constructed as

$$\mathbf{D}^\dagger = \begin{bmatrix} \mathbf{x}^{(0)}(0) & \mathbf{x}^{(0)}(1) & \cdots & \mathbf{x}^{(0)}(\upsilon) \\ \mathbf{x}^{(0)}(1) & \mathbf{x}^{(0)}(2) & \cdots & \mathbf{x}^{(0)}(\upsilon+1) \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{x}^{(0)}(T-\upsilon) & \mathbf{x}^{(0)}(T-\upsilon+1) & \cdots & \mathbf{x}^{(0)}(T) \\ \mathbf{x}^{(1)}(0) & \mathbf{x}^{(1)}(1) & \cdots & \mathbf{x}^{(1)}(\upsilon) \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{x}^{(M)}(T-\upsilon) & \mathbf{x}^{(M)}(T-\upsilon+1) & \cdots & \mathbf{x}^{(M)}(T). \end{bmatrix}$$

Therefore, the new evidence vector used in the training of the stationary network can be thought of simply as

$$\mathbf{x}^\dagger = (x_{0,1}(0), \ldots, x_{n,r_n-1}(0), \ldots, x_{0,1}(\upsilon), \ldots, x_{n,r_n-1}(\upsilon)).$$

Note that the timesteps inside the parenthesis correspond to the collapsed dataset pseudo-timesteps and not the original input dataset.

Every parameter vector is forced to zero, except the ones associated with the last timestep. Those are the only ones that need to be optimized and may take any value except, naturally, the parameter associated with the edge to itself, which is set to zero. Therefore,

$$\boldsymbol{\beta_{jl}}.(t) = \begin{cases} 0 & , \text{if } t \neq \upsilon \\ (\boldsymbol{\beta_{jl0}}(t), \ldots, \boldsymbol{\beta_{jln}}(t,t)) & , \text{if } t = \upsilon. \end{cases} \tag{19}$$

B. Loss Function

Foremost, our loss is desired to be decomposable, *i.e.*, it can be separated into a sum of components for each random variable that only depends on it and its parents. This decomposition implies that the training of the prior network can be done separately from the transition network since no node in the prior network can admit a parent from the transition network due to the restriction in Eq. (12). In fact, if our score is decomposable and the acyclicity constraint on the structure is dropped, the parent set for each node can be estimated independently. A negative log-likelihood score has this property.

Secondly, for this method to be applicable to high-dimensional data and to suit better "real" networks arguably with a small number of edges, a sparse structure is preferred. To achieve this, a group lasso regularization term should be applied to the parameters related to every possible edge of the structure, both intra-temporal and inter-temporal, respecting the temporal causality and Markov lag.

Combining them both, regularizing the log-likelihood score with a group lasso penalty, a loss function can be constructed, summing the cost for each timestep, resulting in

$$f = \sum_{t=0}^{T} \left[-\frac{1}{M}\ell(\boldsymbol{\beta}(t)) + \sum_{\tau \in W(t)} \sum_{j=1}^{n} \sum_{i=1}^{n} \lambda_{ij}(\tau, t) \|\boldsymbol{\beta_{j \cdot i}}(\tau, t)\| \right], \tag{20}$$

where $\ell(\boldsymbol{\beta}(t))$ is a shorthand notation to represent the log-likelihood of a set of parameters associated to the timestep t, M is the size of the dataset, $\lambda_{ij}(\tau, t)$ are the regularization weights, associated with the edge $X_i(\tau) \to X_j(t)$ and $W(t) = \{\tau \in \mathbb{N}_0 : t - \upsilon \leq \tau \leq t\}$ is a set of previous timesteps within the specified Markov lag υ.

The regularization weights are obtained using the strategy of the so-called adaptative LASSO [15], and constitutes the first step of the proposed pipeline. First, an unregularized optimization problem

$$\underset{\tilde{\boldsymbol{\beta}}(t)}{\text{minimize}} \quad -\frac{1}{M}\ell(\tilde{\boldsymbol{\beta}}(t)), \tag{21}$$

is solved for each timestep to find the set of relevant parameters $\tilde{\boldsymbol{\beta}}(t)$. Then, these parameters are used to compute the adaptative regularization weights using the expression

$$\lambda_{ij}(\tau, t) = \lambda \frac{1}{\|\tilde{\boldsymbol{\beta}}_{\boldsymbol{j \cdot i}}(\tau, t)\|^\gamma},$$

where λ and γ are hyper-parameters.

C. Optimization Method

To find a skeleton to the network, a parameter set $\boldsymbol{\beta}$ needs to be estimated by minimizing the loss function. To achieve this, a coordinate descent algorithm was employed following Gu *et al.* [10]. This algorithm was chosen because of its simplicity, facilitating the development of new software, and its recognized use in this context, for example, in the `sparsebn` R package. Since the proposed loss is decomposable, every node may be independently optimized.

The coordinate descent algorithm method consists of optimizing the cost function along one dimension, keeping all the others constant, and iterating throughout the dimensions until convergence to a solution point is attained. Some coordinates are forced to be zero due to the constraints imposed by the DBN. Since these restrictions are always applied only to a single coordinate, the algorithm can simply set those to zero and skip their optimization, never moving in directions that may cause illegal structures. In fact, it is as if the multinomial parametrization never depended on those coordinates. They are only on the vector due to notation convenience and to keep the parametrization general. These effectively speeds up the training process, as many coordinates can be skipped.

D. Directing the skeleton

Solutions to the minimization of the function in Eq. (20) define a skeleton for our prior and transition network through the equivalence in Eq. (17), but in order for it to define a Bayesian network structure, it has to be a directed acyclic graph. To obtain one, and following the suggestion from the MMHC algorithm, a greedy hill-climbing searching procedure is conducted, starting with an empty structure, but instead of checking every possible edge addition, reversal, and deletion on each iteration, it checks only in the set of allowed edges. In addition, in the transition network, constraining the structures to DBNs and imposing the temporal causality further decreases the allowed actions on each iteration. These constraints reduce the complexity of greedy hill-climbing substantially, especially in high dimensional contexts. Algorithm 1 summarizes this method to direct the skeleton, being a slightly modified version of traditional hill-climbing.

Algorithm 1 DBN restricted greedy hill-climbing

Input: A temporal dataset D and a set of allowed edges E

Output: A restricted local optimal structure for a DBN G

loop
  $S^* \leftarrow -\infty$
  for all timestep $t \in [0, T]$ **do**
    for all timestep τ such that $t - \upsilon \leq \tau \leq t$ **do**
      for all addition or removal, resulting in a DAG, of edge $X_i(\tau) \rightarrow X_j(t)$ in E **do**
        $G' \leftarrow$ result of applying operation to G
        if $\phi_{LL}(G') - \phi_{LL}(G) > S^*$ **then**
          $S^* \leftarrow \phi_{LL}(G') - \phi_{LL}(G)$
          $G^* \leftarrow G'$
        end if
      end for
    end for
    for all reversal, resulting in a DAG, of every edge $X_i(t) \rightarrow X_j(t)$ in E **do**
      $G' \leftarrow$ result of applying operation to G
      if $\phi_{LL}(G') - \phi_{LL}(G) > S^*$ **then**
        $S^* \leftarrow \phi_{LL}(G') - \phi_{LL}(G)$
        $G^* \leftarrow G'$
      end if
    end for
  end for
  if $S^* < 0$ **then**
    break
  end if
  $G \leftarrow G^*$
end loop

E. Implementation Details

A C++ implementation of the proposed method was developed and is available as open-source software on github.com/JBSants/sDBN. The stopping criterion used was based on the best improvement of all the coordinates. If no coordinate changes more than $\epsilon = 10^{-4}$, then it is determined that the algorithm has converged. The unregularized problem in Eq. (21) is solved using `liblbfgs`, a C implementation of the Limited-memory BFGS algorithm [16]. A parallel version using the message passing interface (MPI) was developed using the `OpenMPI` library. The implementation offers Intel® MKL acceleration.

IV. Results

A. Synthetic Datasets

Transition networks were trained using data sampled from known, randomly generated, stationary dynamic Bayesian networks. The resulting transition structures were compared to the true one. True positive edges appear in the estimated network and correspond to correct edges in the original network. False positive edges show in the resulting network but are not part of the original structure. False negative (FN) edges take part in the ground truth network but were not identified by the algorithm. Although direct comparison is possible and seen in the literature, the algorithm may output a network that is different from the original structure but still indistinguishable from an observational standpoint, *i.e.*, both networks have the same score. To mitigate this issue, the edges of the original network are labeled as reversible or unreversible [7]. Reversible edges that appear reversed on the estimated network are still counted as true positives.

A total of 12 transition networks with a Markov lag of 1 were generated for 20, 50, and 300 nodes per timestep, four networks for each size with an increasingly larger number of v-structures. Training datasets were sampled with 1000 observations for each network, and test datasets were sampled with 250 observations. All nodes were considered to have three possible discrete states. A path of 80 stationary transition networks was requested with a Markov lag of 1. The path follows a geometric grid of λ values, chosen to originate a significant variation on the outputted networks scores. The backtracking algorithm parameters were set to $\alpha_0 = 1$, $\eta = 0.5$ and $\delta = 0.1$, typical values found in the literature.

Aditionally, the performance metrics were compared between the proposed algorithm and the `tDBN` algorithm. Three different structure of stationary networks types were sampled and used: tree structures with inter-temporal edges restricted to a maximum of 1, 2, and 3 parents from previous timesteps; inter-temporal only structures with random parents also re-

stricted to a maximum of 1, 2, and 3 parents; random structures with a fixed number of v-structures similar to those used in the other experiments. All structures were generated with Markov lag 1. For each structure type, the algorithms were run five times using different original networks. `sDBN` was used to train a path of 80 networks following a geometric grid of λ values. `tDBN` was run using the log-likelihood score and limited to 2 parents from the previous timesteps, except for runs using structures with three parents from the previous timesteps.

Regarding the first experiment, Fig. 1 plots the F_1 and ϕ_{LL} scores for each obtained network in the geometric grid of λ values. By observing the F_1 evolution over the grid of λ values, it can be established that trained networks follow comparable trends. For higher regularization, the score starts very low. These networks are characterized by high precision values, but the F_1 score is very low due to the low recall. This means that the recovered edges are, in fact, present in the original network, but only a small portion of the original network is recovered. Progressing on the grid, the score attains a maximum on the optimal λ value and then begins to drop. This drop is explained by drastically lower precision values. This means that the algorithm starts to select edges that are not present in the original network, evidencing insufficient regularization. The results suggest that the `sDBN` algorithm can correctly recover transition structures with high accuracy. The best F_1 score tends to decrease as network complexity increases. The majority of tested structures scored a F_1 score greater than 0.90.

The log-likelihood score on the test dataset follows the same tendency as the F_1 score. The λ value with maximum log-likelihood does not coincide with the one for maximum F_1, and almost always scored strictly lower relative to the best retrieved F_1. Nevertheless, the maximum log-likelihood still serves as a valid approximation to the optimal λ, with the majority of networks scoring a F_1 higher than 0.90 using this criterion. On the tested structures, 9 out of 12 runs present a likelihood estimated network with F_1 within 5% of the best. This suggests that cross-validation may be used to estimate the optimal λ.

Table I compares precision, recall, F_1 and ellapsed time between the `sDBN` and the `tDBN` algorithms. All experiments were run on an Intel® Xeon® E3-1220 v3 @ 3.10GHz quad-core CPU running Ubuntu Server. The `sDBN` algorithm was run with Intel® MKL acceleration enabled. The `tDBN` algorithm was run using OpenJDK version 11.0.11.

The results show that `sDBN` beats `tDBN + LL` on structure recovery. These experiments statistically reject the hypothesis (Wilcoxon test [17], $p = 1 \times 10^{-12}$) that both algorithms perfom equally well in terms of the F_1 metric. Therefore, it is fair to extrapolate that, on the tested structure types, `sDBN` performs better than `tDBN`.

Consistently `sDBN` outperforms `tDBN` considerably in terms of running time. Even for $n = 50$, `sDBN` outputs the entire path of 80 networks in under 30 minutes. On the other hand, `tDBN` takes more than 4 hours to train a single network.

`sDBN` has weaker assumptions, better theoretical complexity, lower practical running time and can achieve good structure identification. Unfortunately, it has the additional overhead of optimal λ estimation, if domain knowledge is not available.

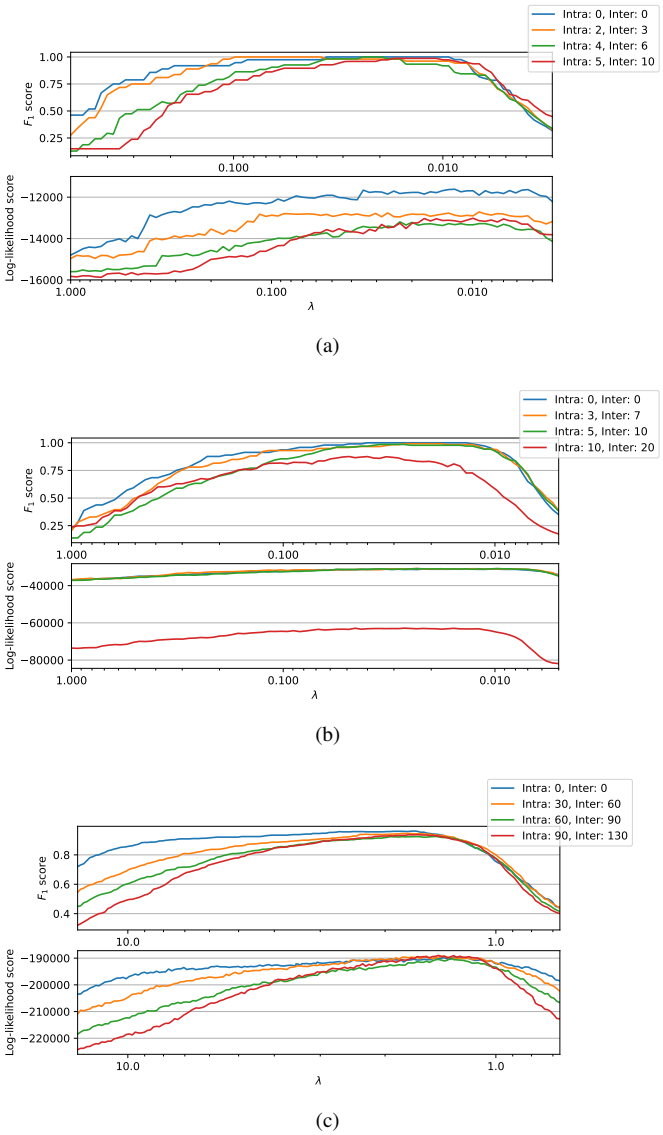

Fig. 1. Evolution of the performance metrics F_1, ϕ_{LL} and ϕ_{MDL} evaluated along a resulting path of 80 networks using a geometric grid of λ values. Generated networks have 20 (a), 50 (b) and 300 (c) nodes per timestep.

B. MTS benchmark datasets

To test the performance of this learning procedure on real data, a classification task was performed on a collection of public multivariate time-series from the UCI Machine Learning Repository [18] and the UCR Time Series Classification Archive [19].

Each dataset was split into train and test datasets using stratified cross-validation with five folds. For each dataset, a DBN multinet was trained and then used for prediction on a test dataset. The DBNs were trained considering a Markov lag of 1 and a stationary process. In the directing process, nodes were restricted to have a maximum of 3 parents. The

TABLE I
PRECISION, RECALL, F_1 SCORE AND ELAPSED TIME t FOR VARIOUS TYPES OF NETWORKS COMPARING THE sDBN AND tDBN ALGORITHM

	sDBN				tDBN + LL			
Type	Pre	Rec	F_1	t / s	Pre	Rec	F_1	t / s
Complete Tree + Inter-temporal (n = 20)								
$p = 1$	**97.5** (3.2)	93.3 (3.1)	**95.3** (1.8)	**147.5** (14.0)	66.1 (0.0)	**100** (0.0)	79.6 (0.0)	264.6 (4.2)
$p = 2$	**96.0** (3.2)	89.6 (6.3)	**92.6** (4.2)	**138.6** (14.1)	82.7 (4.3)	**100** (0.0)	90.5 (2.6)	264.7 (5.4)
$p = 3$	**93.6** (4.2)	87.6 (4.7)	**90.5** (3.9)	**113.3** (15.5)	73.4 (6.5)	**100** (0.0)	84.5 (4.4)	4338.8 (78.4)
Complete Tree + Inter-temporal (n = 50)								
$p = 1$	**95.8** (2.2)	96.2 (3.0)	**96.0** (1.6)	**952.7** (64.4)	66.4 (0.0)	**100** (0.0)	79.8 (0.0)	16474.4 (161.2)
$p = 2$	**91.6** (2.6)	91.7 (2.4)	**91.6** (2.1)	**862.0** (56.6)	81.3 (3.3)	**100** (0.0)	89.7 (2.0)	16502.0 (58.0)
$p = 3$	**89.7** (2.5)	82.9 (2.6)	86.1 (1.7)	**775.7** (51.3)	88.1 (3.2)	**90.8** (1.6)	**89.3** (1.3)	16517.8 (53.3)
Inter-temporal only (n = 20)								
$p = 1$	**100** (0.0)	**100** (0.0)	**100** (0.0)	**168.4** (15.1)	33.9 (0.0)	**100** (0.0)	50.6 (0.0)	254.8 (4.7)
$p = 2$	**99.4** (1.2)	**100** (0.0)	**99.7** (0.6)	**148.4** (17.7)	49.8 (1.7)	**100** (0.0)	66.5 (1.5)	261.0 (4.3)
$p = 3$	**98.4** (2.1)	99.5 (1.1)	**98.9** (1.3)	**132.2** (10.1)	45.3 (2.2)	**100** (0.0)	62.3 (2.1)	4273.6 (61.7)
Inter-temporal only (n = 50)								
$p = 1$	**100** (0.0)	**100** (0.0)	**100** (0.0)	**1293.5** (80.2)	33.6 (0.0)	**100** (0.0)	50.3 (0.0)	16286.2 (51.0)
$p = 2$	**96.4** (5.3)	98.7 (1.4)	**97.5** (3.4)	**1043.5** (178.9)	51.5 (2.5)	**100** (0.0)	68.0 (2.2)	16332.9 (72.9)
$p = 3$	**98.0** (1.0)	**97.6** (1.4)	**97.8** (0.9)	**854.1** (96.3)	53.8 (3.0)	86.4 (1.0)	66.3 (2.3)	16616.2 (425.6)
Variable v-structures (n = 20)								
$v = (2, 3)$	**97.2** (3.9)	**100** (0.0)	**98.6** (2.1)	**172.8** (21.6)	41.7 (1.4)	98.5 (1.9)	58.6 (1.2)	265.3 (3.3)
$v = (4, 6)$	**99.4** (1.2)	**98.8** (1.5)	**99.1** (0.8)	**150.4** (15.0)	50.8 (0.0)	92.6 (1.4)	65.6 (0.4)	261.5 (8.3)
$v = (5, 10)$	**98.9** (1.3)	**96.7** (1.0)	**97.8** (1.0)	**151.4** (20.0)	58.0 (1.3)	94.6 (3.4)	71.9 (1.1)	261.6 (2.9)

TABLE II
AVERAGE PERFORMANCE METRICS ON MTS BENCHMARK DATASETS

Dataset	sDBN		tDBN + MDL	
	Acc	F_1	Acc	F_1
ArabicDigits	**78.1**	**78.2**	75.5	75.6
CharacterTrajectories	85.8	84.2	**86.3**	**85.1**
ECG	**76.5**	**74.7**	75.0	72.3
JapaneseVowels	86.7	86.7	**89.1**	**88.8**
Libras	**63.9**	**62.2**	62.2	60.1
NetFlow	81.9	75.9	**84.4**	**78.4**
UWave	92.3	92.3	**92.7**	**92.7**
Wafer	89.8	63.5	**90.5**	**64.5**

predicted class is predicted computing the joint query for each network and considering the class associated with the highest probability (see Eq. (15)). To select the best λ value, a cross-validation procedure with 10 folds was used in the training data. Using each fold as a test dataset, a path of 80 networks was trained, and then the ϕ_{LL} score was maximized for each network.

Table II reports the average performance metrics on the test dataset, comparing sDBN and tDBN with MDL. Using sDBN, ECG, NetFlow and Wafer scored an average AUC metric, resulting from the integration of each single ROC curve and posterior averaging, of 0.836, 0.852 and 0.854, respectively. On the other hand, tDBN scored 0.812, 0.870 and 0.887, respectively.

On 5 out of 8 datasets, tDBN shows better performance than sDBN. Still, on all datasets, the proposed method performs within 3% of the accuracy of the state-of-the-art method. AUC scores are also similar, but with advantage for tDBN + MDL, scoring higher in 2 out of 3 datasets.

Poor sDBN performance can be associated with difficulty to select the optimal λ value. The benchmark datasets do not have a large number of observations, which can result in an unsatisfactory cross-validation estimate. This is consistent with results in synthetic datasets, as the best network in the path scored in the test dataset often did not coincide with the optimal one. Also, in these experiments, each network that belongs to the multinet classifier was trained using the same geometric grid. However, different classes datasets may have different optimal λ values, and therefore the request grid may not be suitable for every class, which can contribute to reduced classification performance.

Overall results show very competitive accuracies. Although tDBN wins most cases, both in terms of accuracy and F_1 score, the winning margins are not large. According to the Wilcoxon statistical test, the available results are not sufficient to reject the null hypothesis that the two classifiers have different mean accuracy (p=0.898). The results validate the use of sDBN algorithm in real data as it still can output better classifiers when faced with a state-of-the-art DBN training algorithm.

C. Rheumatological health records dataset

Ankylosing Spondylitis (AS) is a condition considered to be a rheumatological disease [20]. This disease is characterized by inflammation in skeleton joints and can lead to total fusion of the axial skeleton and consequent loss of spinal mobility.

Reuma.pt or the Rheumatic Diseases Portuguese Register, is a database of rheumatological patients health records in Portugal created by the Portuguese Society of Rheumatology[21]. To evaluate the sDBN performance on real data, a subset of records from the Reuma.pt AS dataset was cleaned and pre-processed. Table III summarizes the kept features and their associated discretization scheme. Data discretization was largely based on the proposed categories by Martins [22].

Provided data had to be cleaned as it is subject to filling errors and missing values. Missing values were filled using simple filling methods. For each patient, values from previous appointments were first forward carried to later appointments. After that, a backfill method was used to carry back observed

TABLE III
SUMMARY OF THE FEATURES OF REUMA.PT

Feature name	Description
BASDAI	A metric to assess disease activity based on a patient answered questionnaire.
BASDAI Q1-Q6	Questionnaire filled for evaluating the BASDAI index.
BASFI	A metric to assess the level of a patient disability based on a self answered questionnaire.
BASFI Q1-Q10	Questionnaire filled for evaluating the BASFI index.
PCR	C-reactive protein (CRP) / mg/L.
VS	Erythrocyte sedimentation rate (ESR) / mm/hour.
TemCorticActivo	Whether the patient is medicated with corticosteroids.
ArtTumefactas	Number of swelled articulations.
ArtDolorosas	Number of painful articulations.
EVAMedico	VAS from the physician prespective.
EVADoente	VAS from the patient prespective.
ASDAS	A composite index, computed using the C-reactive protein value, applied to evaluate disease activity.
ASDASvs	A composite index, computed using the ESR value, applied to evaluate disease activity.
NEW_BioActivo	Biological agent currently used for treatment.

values to previous appointments. Subjects were also filtered by the number of appointments, discarding patients with less than four and more than 80 observations. Additionally, patients were filtered by applied treatment. Appointments in which the biological agent was changed were dropped, as well as subsequent medical evaluations. Treatment plans were the patient started with no treatment using biological agents, but then one was applied are kept.

Stationary DBNs with Markov lag of 1 were trained using the rheumatological dataset. The dataset used has a variable number of appointments per patient. Therefore, stationary DBNs are better for this data as only transitions are used to train the networks. The best lambda value according to each score was obtained using 10 fold cross-validation. Each cross-validation run used a path of 80 networks with a geometric grid of λ values with arbitrarily chosen limits to obtain a reasonable amount of edges. The λ with the best average MDL score was selected.

Fig. 2 shows the results. The network has a high prevalence of identity edges, *i.e.*, edges originating in the same node but on the previous timestep. This is expected as the value of a particular clinical variable should influence its condition in the next medical consultation.

The algorithm identified clusters of nodes around the BASDAI and BASFI indices. These nodes correspond to the individual questions on each questionnaire. This proves the algorithm's ability to find good relationships between variables as the indices are determined using the individual questions, leading to an evident relationship. This also shows that resulting networks have intra-temporal v-structures, which are not allowed by the `tDBN` algorithm and is a clear advantage of the proposed algorithm.

The same happens for the ASDAS, and ASDAS-ESR indices as the corresponding clinical indicators applied in the calculation (CRP and ESR, respectively) are tied to both indices. ASDAS is also associated with the VAS level from the

Fig. 2. Transition network trained with data from the Reuma.pt dataset considering only patients that were treated by a single or none biological agent.

patient perspective and Q2 from the BASDAI questionnaire. In fact, these two variables also have a considerable influence on ASDAS [23], corroborating that the algorithm outputs a good structure. These results are not consistent with Martins [22] as, in most networks, a relationship is found between ASDAS and VAS from the doctor's perspective instead. Since ASDAS is calculated with the patient perspective, the shown results are more reasonable.

No network included a relationship with corticosteroids use. Martins [22] also found that corticosteroids use is very disconnected from the considered features. This suggests that the use of these drugs does not influence disease progression. In fact, the use of steroids can provide pain relief, but it has not been proved to be effective with AS [20].

For a quantitative study, a classification task was designed aiming to predict the treatment outcome. To this end, stationary DBN were trained on a transition dataset between two consecutive appointments. For comparison purposes, networks were trained with the `tDBN` algorithm. A class variable was designed discretizing the ASDAS as a binary feature – low or high disease activity – being 3.5 the threshold between the two.

To predict treatment outcome, the resulting network was queried for the probability of the class variable in the latter timestep using evidence only for the first timestep. The predicted class is the node level with the highest MAP probability, given the evidence.

Table IV shows performance metrics for the trained classifiers. All classification algorithms accurately predicted the disease activity in the medical consultation, with average accuracies greater than 80%. Given that a balanced binary dataset was used, this shows that these classifiers can be helpful in aiding clinical decisions.

TABLE IV
AVERAGE CLASSIFIER PERFORMANCE ON THE REUMA.PT DATASET

Algorithm	Accuracy	F_1 score	AUC
sDBN + LL	0.875 (0.048)	0.877 (0.050)	0.913 (0.039)
sDBN + MDL	0.838 (0.151)	0.856 (0.105)	0.846 (0.125)
tDBN + LL	0.873 (0.050)	0.876 (0.050)	0.920 (0.036)
tDBN + MDL	**0.882** (0.065)	**0.885** (0.064)	**0.923** (0.037)

In this particular classification task, tDBN with MDL is the best classifier. Not only has greater accuracy, but it also has no additional overhead of discovering the optimal λ. Also, it was trained admitting only one parent from the previous timestep, the fastest configuration possible. Therefore, sDBN does not prove itself useful in this context, as the dimensionality is still tractable to be trained by other methods with better results. The lower performance of sDBN can be linked to difficulty in choosing the optimal λ. Additional studies have to be done to determine if increasing the number of folds on the cross-validation step to find the optimal network can lead to a better result. These results in medical datasets are consistent with the results found with multinet classification on MTS benchmark datasets.

V. CONCLUSIONS

The proposed method achieves great structure identification in low and high-dimensional contexts. This work reports excellent results in DBNs with up to 300 nodes per timestep in artificial datasets, dimensions simply unattainable by state-of-the-art techniques like tDBN in a reasonable time. The new algorithm also identified suitable transition networks on rheumatological data that uncover known relationships between dataset features. These structures are full of intra-temporal v-structures that optimal tree algorithms can't recover.

The validation results in this document are by no means extensive. These experiments focused on training stationary networks with Markov lag 1, so additional work can be done to test the algorithm for networks with different configurations. Also, non-stationary network training is possible, but it is not tested. Finding public high-dimensional temporal datasets proved to be a challenge. Therefore, further studies to assess the performance of the algorithm in these kinds of datasets are needed.

The proposed algorithm conducts a full optimization procedure on the multinomial logistic parametrization, but only the support of the parameter vector is considered when directing the network. It should be possible to improve the method by stopping the optimization if the active set did not change in a certain number of iterations.

REFERENCES

[1] J. A. Nelder and R. W. M. Wedderburn, "Generalized Linear Models," *Journal of the Royal Statistical Society. Series A (General)*, vol. 135, no. 3, pp. 370–384, 1972, publisher: [Royal Statistical Society, Wiley].

[2] R. A. Fisher and E. J. Russell, "On the mathematical foundations of theoretical statistics," *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, vol. 222, no. 594-604, pp. 309–368, Jan. 1922, publisher: Royal Society.

[3] R. Tibshirani, "Regression Shrinkage and Selection via the Lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996, publisher: [Royal Statistical Society, Wiley].

[4] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 68, no. 1, pp. 49–67, Feb. 2006.

[5] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1988.

[6] P. Spirtes, C. Glymour, and R. Scheines, *Causation, Prediction, and Search*, 2nd ed. MIT press, 2000.

[7] M. Chickering, D. Geiger, and D. Heckerman, "Learning Bayesian networks: Search methods and experimental results," in *Proceedings of the Fifth International Workshop on Artificial Intelligence and Statistics*, January 1995.

[8] G. Schwarz, "Estimating the Dimension of a Model," *Annals of Statistics*, vol. 6, no. 2, pp. 461–464, Mar. 1978, publisher: Institute of Mathematical Statistics.

[9] M. Schmidt, A. Niculescu-Mizil, and K. Murphy, "Learning graphical model structure using L1-regularization paths," in *Proceedings of the 22nd national conference on Artificial intelligence - Volume 2*, ser. AAAI'07. Vancouver, British Columbia, Canada: AAAI Press, Jul. 2007, pp. 1278–1283.

[10] J. Gu, F. Fu, and Q. Zhou, "Penalized estimation of directed acyclic graphs from discrete data," *Statistics and Computing*, vol. 29, no. 1, pp. 161–176, Jan. 2019.

[11] 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. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., Jul. 1998, pp. 139–147.

[12] J. L. Monteiro, S. Vinga, and A. M. Carvalho, "Polynomial-time algorithm for learning optimal tree-augmented dynamic Bayesian networks," in *Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence (UAI'15)*, 2015, pp. 622–631.

[13] D. Geiger and D. Heckerman, "Knowledge representation and inference in similarity networks and Bayesian multinets," *Artificial Intelligence*, vol. 82, no. 1, pp. 45–74, Apr. 1996.

[14] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian Network Classifiers," *Machine Learning*, vol. 29, no. 2, pp. 131–163, Nov. 1997. [Online]. Available: https://doi.org/10.1023/A:1007465528199

[15] H. Zou, "The Adaptive Lasso and Its Oracle Properties," *Journal of the American Statistical Association*, vol. 101, no. 476, pp. 1418–1429, Dec. 2006, publisher: Taylor & Francis _eprint: https://doi.org/10.1198/016214506000000735.

[16] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization," *Mathematical Programming*, vol. 45, no. 1, pp. 503–528, Aug. 1989.

[17] F. Wilcoxon, "Individual Comparisons by Ranking Methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945, publisher: [International Biometric Society, Wiley].

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

[19] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G. Batista, "The ucr time series classification archive," July 2015.

[20] F. Mahmood and P. Helliwell, "Ankylosing Spondylitis: A review," *European Medical Journal*, pp. 134–139, Nov. 2017.

[21] H. Canhão, A. Faustino, F. Martins, and J. E. Fonseca, "Reuma.pt — the rheumatic diseases portuguese register," *Acta Reumatológica Portuguesa*, vol. 36, pp. 45–56, 2011.

[22] A. R. L. B. Martins, "Dynamic Bayesian networks for clinical decision support on ankylosing spondylitis patients under biological therapies," Master's thesis, Insituto Superior Técnico, Lisboa, 2021.

[23] C. Lukas, R. Landewé, J. Sieper, M. Dougados, J. Davis, J. Braun, S. v. d. Linden, D. v. d. Heijde, and f. t. A. o. S. i. Society, "Development of an ASAS-endorsed disease activity score (ASDAS) in patients with ankylosing spondylitis," *Annals of the Rheumatic Diseases*, vol. 68, no. 1, pp. 18–24, Jan. 2009, publisher: BMJ Publishing Group Ltd Section: Clinical and epidemiological research.