SrVARM: State Regularized Vector Autoregressive Model for Joint Learning of Hidden State Transitions and State-Dependent Inter-Variable Dependencies from Multi-variate Time Series

Tsung-Yu Hsieh
Pennsylvania State University
tuh45@psu.edu

Yiwei Sun
Pennsylvania State University
yus162@psu.edu

Xianfeng Tang
Pennsylvania State University
tangxianfeng@outlook.com

Suhang Wang
Pennsylvania State University
szw494@psu.edu

Vasant G. Honavar
Pennsylvania State University
vhonavar@psu.edu

ABSTRACT

Many applications, e.g., healthcare, education, call for effective methods methods for constructing predictive models from high dimensional time series data where the relationship between variables can be complex and vary over time. In such settings, the underlying system undergoes a sequence of unobserved transitions among a finite set of hidden states. Furthermore, the relationships between the observed variables and their temporal dynamics may depend on the hidden state of the system. To further complicate matters, the hidden state sequences underlying the observed data from different individuals may not be aligned relative to a common frame of reference. Against this background, we consider the novel problem of jointly learning the state-dependent inter-variable relationships as well as the pattern of transitions between hidden states from multi-variate time series data. To solve this problem, we introduce the State-Regularized Vector Autoregressive Model (SrVARM) which combines a state-regularized recurrent neural network to learn the dynamics of transitions between discrete hidden states with an augmented autoregressive model which models the inter-variable dependencies in each state using a state-dependent directed acyclic graph (DAG). We propose an efficient algorithm for training SrVARM by leveraging a recently introduced reformulation of the combinatorial problem of optimizing the DAG structure with respect to a scoring function into a continuous optimization problem. We report results of extensive experiments with simulated data as well as a real-world benchmark that show that SrVARM outperforms state-of-the-art baselines in recovering the unobserved state transitions and discovering the state-dependent relationships among variables.

KEYWORDS

Dynamic structure learning, State-regularized vector autoregressive model, State space model, Bayesian Networks, Deep Neural Networks

ACM Reference Format:
Tsung-Yu Hsieh, Yiwei Sun, Xianfeng Tang, Suhang Wang, and Vasant G. Honavar. 2021. SrVARM: State Regularized Vector Autoregressive Model for Joint Learning of Hidden State Transitions and State-Dependent Inter-Variable Dependencies from Multi-variate Time Series. In *Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3442381.3450116

1 INTRODUCTION

Many practical applications, data from individuals (indexed by i) are naturally modeled by a multi-variate time series $\mathbf{X}_i = \left[\mathbf{x}_1^i, \ldots, \mathbf{x}_T^i\right]$ where $\mathbf{x}_t^i \in \mathbb{R}^P$ is the P dimensional observation at time point t. Collection of such multi-variate time series data from a set of N individuals yields a data set $\mathcal{X} = \{\mathbf{X}_1, \ldots, \mathbf{X}_N\}$. In many real-world applications, the relationships among the variables can be complex, and change with time. Of particular interest is the setting where the data generating process undergoes a sequence of unobserved transitions among a finite set of *hidden* states. The resulting data set consists of data instances each of which takes the form of a multi-variate time series.

For example, in a health case setting, such data may include repeated measurements of physiological measurements, e.g., heart rate, body temperature, blood pressure, etc., recorded from an individual at different times. The temporal dynamics of these observed variables can change as the individual goes through unobserved transitions between unobserved (hidden) states e.g., stressed versus relaxed. Similarly, in economics, the interactions among macroeconomic variables vary as a function of the state of the economy: For example, the impacts of oil price on GDP growth are significantly different when the economy is in a high growth phase versus a low growth phase [2]; Consumer behavior shows seasonal patterns [24, 37]. In life sciences, the structure of the regulatory relationships between genes can vary across the stages of the cell cycle [27]. Coping with such applications calls for a formalism wherein the data generating process undergoes transitions between a finite number of hidden states; and the dependencies among variables are state-dependent (See Fig. 1). To further complicate matters, the sequence of hidden state transitions across individuals in the data set may not be aligned relative to a common frame of reference.

Against this background, we consider the novel problem of jointly learning the state-dependent dependencies among variables as well as the pattern of transitions between hidden states where

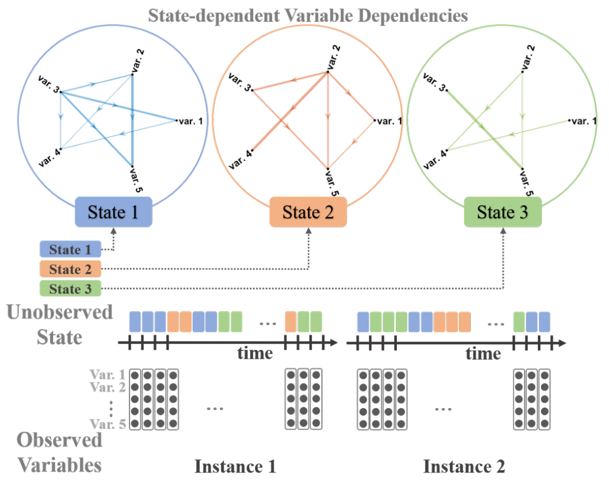


Figure 1: A toy example of dynamic systems undergoing a sequence of transitions among a set of three states. The state sequences of individual systems may not be aligned. The observed variables in the system exhibit different relationships at each state. (Best viewed in color.)

the hidden state transitions across different individuals are not necessarily aligned relative to a common frame of reference.

The key contributions of the paper are as follows:

- We consider the novel problem of jointly learning the dynamics of transitions between a finite set of hidden states and the state-dependent DAG-structured inter-variable dependencies.
- We introduce SrVARM, a state-regularized autoregressive model which combines a state-regularized recurrent neural network to learn the dynamics of transitions between discrete hidden states with an augmented autoregressive model which models the state-dependent inter-variable dependencies using state-dependent DAGs.
- We propose an efficient algorithm for training SrVARM by leveraging a recently introduced reformulation of the combinatorial problem of optimizing the DAG structure with respect to a scoring function into a continuous optimization problem.
- We present results of extensive experiments on both synthetic and real-world data sets and show that SrVARM outperforms state-of-the-art baselines.

The rest of the paper is organized as follows. Section II summarizes related work that sets the stage for SrVRAM framework developed in this paper. Section III defines the problem of learning the pattern of hidden state transitions and state-dependent relationships between observed variables from MTS data; Section IV describes the design of SrVARM and the associated learning algorithm in detail. Section V describes experimental setup and results. Section V concludes the paper with a brief summary and an outline of some promising directions for further research.

2 RELATED WORK

In this section, we review related work on structure learning, which include structure learning from static observational data, structure learning from temporal data, and state space deep neural network for time series learning.

2.1 Structure Learning from Static Data

Bayesian Networks (BN) (reviewed in [5, 10, 11, 14, 21, 30, 39, 40, 43, 55]) offer an attractive formalism for modeling the (static) dependencies among random variables. The structure of BN is a directed acyclic graph (DAG) wherein the nodes represent random variables and directed links model direct dependencies between variables. The semantics of BN ensure that each variable is conditionally independent of its non-descendents given its parents. This admits an efficient factorization of the joint distribution of the variables as a product of the probability distributions of each of the variables conditioned on their parents. Bayesian network models have found applications across a number of domains including biology [38], genetics [52], economics [16], among others. Learning a BN involves learning its DAG structure and its parameters (the probability distribution of each variable conditioned on its parents in the DAG). Methods for learning BN structure from data fall into two broad classes: *score-based* methods which use a scoring function to choose a BN structure from a set of candidate structures and *constraint-based* methods that use conditional independence tests against data to rule out a subset of candidate structures. Because exhaustive consideration of all possible DAG structures is computationally intractable [4], most algorithms either restrict the set of possible structures a priori (e.g., by assuming that each variable can directly depend on no more than k other variables where k is much smaller than the total number of variables) or by using heuristics, e.g., greedy hill-climbing with respect to a scoring function, to guide the search. Recent work has led to efficient approaches to learning the DAG structure of BNs by reformulating the NP-hard combinatorial optimization problem [4] of finding an DAG structure that optimally and parsimoniously models the dependency structure present in the data as a continuous optimization [22, 53, 57].

There is a growing body of work on the problem of learning learning the structure of dependencies between variables in multivariate time series data [28, 31, 33, 50]. For example, Xu et al. [50] propose an approach to modeling nonlinear relationships between variables from temporal data through a low-rank approximation based deep neural network; Pamfil et al. [31] model the relationships between the variables in a multi-variate time series using a structural vector autoregressive model [8] where the time-invariant structure of the relationships between variables is modeled using DAGs. However, such models are not sufficiently expressive to accommodate the state-dependent structure of dependencies among the variables of multi-variate time series data.

A recent breakthrough in learning BN structure from data has been enabled by a smooth characterization of the acyclicity constraint of DAG [53]. It reformulates the combinatorial optimization problem of choosing a DAG structure to optimize a scoring function into a continuous optimization problem that can be efficiently solved by standard numerical algorithms. Several studies have exploited this result to propose novel methods for learning the structure of BN. For example, Lachapelle et al. [22] extended the results to model nonlinear relations between variables by means of a neural network. In [57], a reinforcement learning approach was used jointly with the smooth characterization as a search strategy to find the best scoring DAG. Yu et al. [51] proposed a graph neural network based deep generative model to learn the DAG.

2.2 Structure Learning from Temporal Data

Dynamic Bayesian Networks (DBN) [7, 29, 31], a generalization of (static) BN [30] offer an attractive formalism for learning probabilistic models from multi-variate time series data, under the assumption that the data are generated by a stationary process. Works in [45, 46] propose multi-layer perceptron (MLP) and a long short-term memory (LSTM) [15] deep learning framework to learn causal graph. Xu et al. [50] present a deep neural network for scalable causal graph learning through low-rank approximation and without any predefined kernel or distribution assumptions. A recent work [31] exploits the smooth acyclic characterization to tackle structure learning from time-series data. However, these methods learn static graph from time series data and cannot fully characterize the changing dynamic. DBN have been generalized to the non-stationary setting where the conditional dependence structure of the underlying data generation process and hence the structure of the BN is permitted to evolve over time [12, 35, 36, 42]. For example, Robinson et al. [35] introduced nonstationary dynamic Bayesian networks. Song et al. [42] proposed a kernel reweighted l-1 regularized autoregressive procedure for learning time-varying dynamic Bayesian Networks. Huang et al. [19] proposed CPF-SAEM,in a sequential Monte Carlo sampling-based approach to learning the time-varying inter-variable dependencies.

2.3 State Space Deep Neural Networks

State-space approaches offer an attractive approach to modeling time series data [1, 13, 34, 41, 44, 48, 54]. Work in [34] introduce an approach to probabilistic time series forecasting that combines state space models with deep neural networks. Alaa et al. [1] develop the attention-based state space deep neural network model to learn accurate and interpretable structured representations for modeling disease trajectories. SRLSTM [48] aims enhance the transparency of leaw rned recurrent networks and their ability to model long-range dependencies. However, none of these approaches adequately address the problem of learning both the dynamics of transitions between hidden states and state-dependent inter-variable dependencies from time series data sets.

2.4 Research Gap to be Addressed

None of the existing methods address the problem of jointly learning the state-dependent dependencies among variables as well as the pattern of transitions between hidden states in a setting where the hidden state transitions across different individuals are not necessarily aligned relative to a common frame of reference.

3 PROBLEM DEFINITION

Let $\mathcal{X} = \{\mathbf{X}_1, \ldots, \mathbf{X}_N\}$ denote a set of N multivariate time series (MTS) data instances. We use $\mathbf{X}_i = \left[\mathbf{x}_1^i, \ldots, \mathbf{x}_T^i\right]$ to denote the i-th MTS data instance, where $\mathbf{x}_t^i \in \mathbb{R}^P$ is the P dimensional observation at time point t. In real-world applications, MTS observations can be highly nonstationary and the relationships among MTS variables can be dynamic and change with time. We assume that the underlying dynamical system or the MTS data generating process undergoes a sequence of unobserved transitions among a finite discrete set S of hidden states, and that the relationships between variables for each state is modeled by a state-dependent DAG, i.e., A_s for state s. For example, suppose a number of physiological measurements e.g., heart rate, body temperature, blood pressure, etc., are recorded from each individual at different time points. The resulting time series constitutes an MTS data instance. The collection of such MTS data instances, one per individual, for an MTS data set. Each individual may transition through hidden states, e.g., relaxed and stressed, over time. The relationships between the observed variables may change as a function of the hidden system state. We use $s_t^i \in \{1, \ldots, S\}$ refer to the hidden state of individual i at time t. In general, the unobserved state sequences of different individuals are not aligned with respect to a common frame of reference. Thus, s_t^i is not necessarily equal to s_t^j. In each state, for example s_t^i, the structure of inter-variable dependencies is modeled by a DAG, or equivalently, a state-dependent directed acyclic adjacency matrix $A_{s_t^i}$. In light of the above, the problem of learning the pattern of transitions between unobserved states and the state-dependent DAGs that model the inter-variable dependencies from MTS data can be formulated as follows:

Definition 3.1. **Learning the Pattern of Hidden State Transitions and State-Specific Inter-Variable Dependencies from MTS Data**. Given a MTS data set $\mathcal{X}$, learn a function that can simultaneously recovers the MTS instance-specific hidden state transition sequence s_t^i, where $t = 1, \ldots, T-1, i = 1, \ldots, N$, and the state-dependent inter-variable dependencies encoded by $A_1, \ldots, A_S$.

4 STATE-REGULARIZED VECTOR AUTOREGRESSIVE MODEL

We now proceed to introduce the proposed State Regularized Vector Autoregressive Model (SrVARM). The schematic of an SrVARM is shown in Fig. 2. The key idea behind SrVARM is to use a recurrent neural network to learn from a given MTS data set, the dynamics of transitions among hidden states and learn the state-dependent DAGs represented by acyclic autoregressive matrices that encode state-dependent inter-variable relationships. Next, we describe each component of SrVARM in detail.

4.1 State-Regularized Recurrent Network

We restrict the underlying dynamical system to undergo a sequence of discrete state transitions to better understand the extracted results as well as to control the complexity of the model to minimize chance of over-fitting the MTS data. We adopt a state-regularized recurrent neural network [48] to model the transitions among hidden states in S. We regularize the recurrent network to ensure that it approximates a model with transitions between discrete states. In what follows, to minimize notational clutter, we omit the MTS data instance index i.

Given MTS data instances, a recurrent function $f(\cdot, \cdot)$ takes current observation $\mathbf{x}_t$ and the previous hidden state vector $\mathbf{h}_{t-1}$ as inputs and outputs an intermediate latent representation $\mathbf{u}_t$

$$\mathbf{u}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t) \tag{1}$$

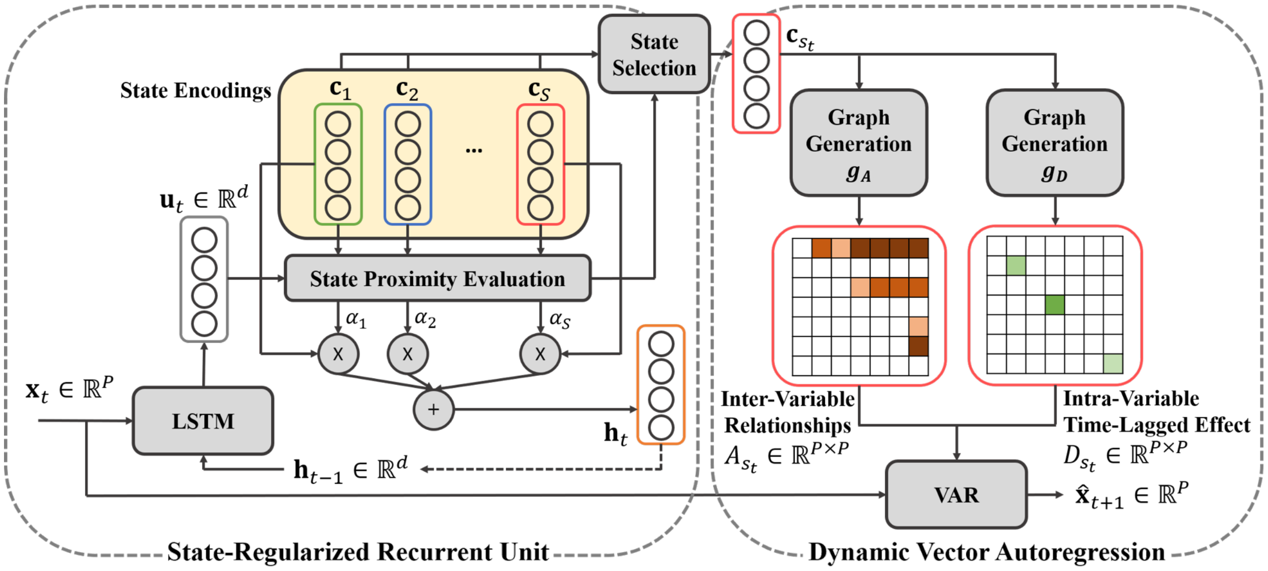

State Encodings
$\mathbf{c}_1$
$\mathbf{c}_2$
$\mathbf{c}_S$
State Selection
$\mathbf{c}_{s_t}$
$\mathbf{u}_t \in \mathbb{R}^d$
State Proximity Evaluation
α_1
α_2
α_S
$\mathbf{x}_t \in \mathbb{R}^P$
LSTM
$\mathbf{h}_t$
$\mathbf{h}_{t-1} \in \mathbb{R}^d$
Graph Generation g_A
Graph Generation g_D
Inter-Variable Relationships $A_{s_t} \in \mathbb{R}^{P\times P}$
Intra-Variable Time-Lagged Effect $D_{s_t} \in \mathbb{R}^{P\times P}$
VAR
$\hat{\mathbf{x}}_{t+1} \in \mathbb{R}^P$
State-Regularized Recurrent Unit
Dynamic Vector Autoregression

Figure 2: SrVARM model framework. The framework consists of a state-regularized recurrent unit and a dynamic vector autoregressive model. The state-regularized recurrent unit is used to model the transitions between discrete hidden states. The augmented dynamic autoregressive model learns state-dependent inter-variable dependencies subject to an acyclic constraint and encodes the relations in state-dependent DAG structures.

We use a long short-term memory network (LSTM) [15] to realize $f(\cdot,\cdot)$. The model maintains S hidden state encodings $\mathbf{C} = \mathbf{c}_1,\dots,\mathbf{c}_S$. Note that the intermediate latent representation $\mathbf{u}_t$ captures the temporal dynamics of the MTS sequence. To the next state indicator s_t is obtained from $\mathbf{u}_t$ by comparing $\mathbf{u}_t$ with the state encodings $\mathbf{c}_1,\dots,\mathbf{c}_S$ and computing a state proximity score α_{s_t} between $\mathbf{u}_t$ and each $\mathbf{c}_s \in \mathrm{C}$.

$$\alpha_{s_t} = \langle \mathbf{u}_t, \mathbf{c}_s \rangle \tag{2}$$

where $s = 1,\dots,S$ and $\langle\cdot,\cdot\rangle$ is the vector inner product. Ideally, in order for the recurrent network to mimic a finite state machine, the hidden state vector $\mathbf{h}_t$ should assume values from $\{\mathbf{c}_1,\dots,\mathbf{c}_S\}$. The assignment of $\mathbf{h}_t$ is obtained by selecting the state encoding corresponding to the maximum state proximity score, i.e.

$$s_t = arg \max_{s'_t=1,\dots,S} \alpha_{s'_t} \tag{3}$$

and let $\mathbf{h}_t = \mathbf{c}_{s_t}$. Because the argmax function renders the model not end-to-end differentiable, we approximate the assignment of $\mathbf{h}_t$ as follows: Given the state proximity scores $\boldsymbol{\alpha}_t = [\alpha_{1_t},\dots,\alpha_{S_t}]$, we apply a softmax operation to normalize the scores into a probability distribution given by:

$$\hat{\alpha}_{s_t} = \frac{\exp(\alpha_{s_t}/\tau)}{\sum_{s'=1}^{S}\exp(\alpha_{s'_t}/\tau)} \tag{4}$$

where τ is a temperature hyperparameter used to anneal the probabilistic state transition behavior. The lower the τ, the more closely the normalized scores $\hat{\boldsymbol{\alpha}}_t = [\hat{\alpha}_{1_t},\dots,\hat{\alpha}_{S_t}]$ approximate a one-hot encoding. With the normalized proximity scores, the assignment of hidden state vector is given by:

$$\mathbf{h}_t = \sum_{s=1}^{S} \hat{\alpha}_{s_t}\mathbf{c}_s \tag{5}$$

Note that, although the temperature hyperparameter τ accentuates the differences in the proximity scores, it does not quite approximate one-hot encoding of states. To see this, consider a scenario where the proximity scores for two states are equal, i.e. $\alpha_1 = \alpha_2$. In this case, we end up with $\hat{\alpha}_1 = \hat{\alpha}_2$ regardless of the value of τ. We need to ensure that the recurrent neural network indeed models transitions between a finite set of distinct discrete states. We observe that the entropy of the normalized proximity scores should be zero when the uncertainty associated with the system state approaches zero. Based on this observation, we apply a regularizerization to the normalized proximity scores as follows:

$$\mathcal{E}(\hat{\boldsymbol{\alpha}}_t) = -\sum_{s_t=1}^{S} \hat{\alpha}_{s_t}\log(\hat{\alpha}_{s_t}) \tag{6}$$

The regularizer forces the state proximity scores towards zero entropy (one-hot) encodings of states thus forcing the the recurrent network to approximate a finite state machine.

Incorporating Prior Knowledge. We observe that while the true pattern of transitions between hidden states is generally unknown, domain knowledge may suggest the relative frequencies of occupancy of each state. For example, we may expect a "normal" state to be more frequent than an "abnormal" state. When such prior knowledge is available, it is possible to leverage it to improve the accuracy of the learned transitions between hidden states. Suppose the ideal state distribution is given by $\mathbf{z} = [z_1,\dots,z_S]$. We can define a state frequency regularizer based on a Kullback-Leibler (KL) divergence between $\mathbf{z}$ and the hidden state distribution $P(s)$ as follows:

$$P(s) = \sum_{t=1}^{T-1} \hat{\alpha}_{s_t}/(T-1), s = 1,\dots,S \tag{7}$$

With the estimated hidden state distribution, the state frequency regularizer is given by:

$$KL(\{\hat{\boldsymbol{\alpha}}_t\}, \mathbf{z}) = \sum_{s=1}^{S} P(s) \log \frac{P(s)}{z_s} \tag{8}$$

where $\{\hat{\boldsymbol{\alpha}}_t\}$ is the collection of $\hat{\boldsymbol{\alpha}}_t$, $t = 1, \ldots, T-1$.

In summary, the state regularized recurrent network learns the state transition function between the hidden state. The deterministic hidden state vector $\mathbf{h}_t$ of the recurrent network is given by Eq. (5) and the state indicator s_t is obtained by identifying the state with the largest normalized proximity score.

4.2 Learning the Structure of State-Dependent Inter-Variable Dependencies

Recall that in SrVARM, the state-dependent structure of inter-variable dependencies is modeled by state-dependent DAGs, or equivalently, state-dependent directed acyclic adjacency matrices. Now we turn to the problem of learning such state-dependent directed acyclic matrices. Given an inferred system state s_t, the progression of the MTS data can be modeled using

$$\hat{\mathbf{x}}_{t+1} = W_{s_t} \mathbf{x}_t \tag{9}$$

where W_{s_t} is the autoregressive matrix associated with the state s_t.

We note that when W_{s_t} encodes the DAG structure of the inter-variable relationships, there arises a "*source-node vanishing*" problem in Eq. (9): The variables in $\hat{\mathbf{x}}_{t+1}$ that correspond to the source nodes in W_{s_t} will always be zero since there are no incoming arrows into the source nodes. Because a DAG must have at least one source node, this introduces an undesirable bias as we seek to learn the DAG structure by minimizing the difference between $\hat{\mathbf{x}}_{t+1}$ and $\mathbf{x}_{t+1}$.

To alleviate the problem, we devised an augmented autoregressive model by dividing the autoregressive matrix into two components, $W_{s_t} = A_{s_t} + D_{s_t}$, where A_{s_t} encodes the inter-variable relationships and D_{s_t} models intra-variable time-lagged effects. The first component of W_{s_t}, namely A_{s_t}, can be obtained from the state encoding $\mathbf{c}_{s_t}$ as follows:

$$A_{s_t} = g_A(\mathbf{c}_{s_t}) \tag{10}$$

where $g_A(\cdot)$ is a graph generator function which is realized using a feed-forward neural network which produces an intermediate output of dimension $\mathbb{R}^{P^2}$, followed by a reshape operation to yield an autoregressive matrix $A_{s_t} \in \mathbb{R}^{P \times P}$. The second component W_{s_t}, namely, D_{s_t}, because it encodes intra-variable time-lagged effects, is a diagonal matrix with nonzero entries only along its diagonal which can be obtained from the state encoding as follows:

$$D_{s_t} = g_D(\mathbf{c}_{s_t}) \tag{11}$$

where $g_D(\cdot)$ is a graph generator function realized using a feed-forward neural network which generates an intermediate output of dimension $\mathbb{R}^P$, and follows by an operation to turn the vector into a diagonal matrix of $\mathbb{R}^{P \times P}$. In this study, both graph generator networks, $g_A(\cdot)$ and $g_D(\cdot)$, are realized by a one-hidden-layer feed-forward neural network. However, the framework admits more complex graph generators.

In order to encode state-dependent inter-variable relationships using A_{s_t}'s, we leverage a smooth characterization of acyclic DAGs [53]

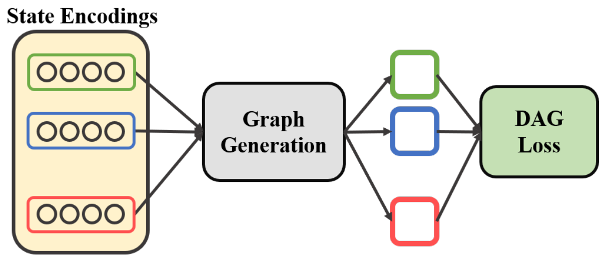


Figure 3: Enforcing acyclicity constraint on graph generation process. The acyclic constraint is evaluated on the graphs generated from state encodings $\mathbf{c}_s$'s.

to enforce acyclicity constraints on the graph generator $g_A(\cdot)$. The graph is acyclic if its adjacency matrix $A \in \mathbb{R}^{P \times P}$ satisfies:

$$\operatorname{tr}(e^{A \circ A}) - P = 0 \tag{12}$$

where $\circ$ is the Hadamard product and $\operatorname{tr}(\cdot)$ is the matrix trace. As shown in Fig 3, the penalty is applied to the adjacency matrices, $A_1, \ldots, A_S$, associated with the corresponding state encodings, $\mathbf{c}_1, \ldots, \mathbf{c}_S$. Because the penalty is non-negative [53], i.e. $\operatorname{tr}(e^{A \circ A}) - P \geq 0$, the acyclicity constraints for all state-specific adjacent matrices can be combined to yield:

$$\sum_{s=1}^{S} \left[\operatorname{tr}\left(e^{A_s \circ A_s}\right) - P \right] = 0 \tag{13}$$

Thus, SrVARM exploits the smooth acyclicity constraints to learn the dynamic autoregressive model that encodes state-dependent DAG-structured inter-variable relationships.

4.3 Objective Function of SrVARM

We now proceed to describe our algorithm for training SrVARM to recover the pattern of transitions among hidden states as well as state-dependent structure of inter-variable dependencies from an MTS data set. The optimization problem to be solved is given by:

$$\begin{aligned} \min_{\Omega} \sum_{i=1}^{N} \sum_{t=1}^{T-1} \left[\left\| \mathbf{x}_{t+1}^i - W_{s_t^i} \mathbf{x}_t^i \right\|^2 + \lambda_1 \mathcal{E}(\hat{\boldsymbol{\alpha}}_t^i) \right] + \lambda_2 KL(\{\hat{\boldsymbol{\alpha}}_{t'}^i\}, \mathbf{z}) \\ \text{s.t. } h(\Omega) = \sum_{s=1}^{S} [\operatorname{tr}\left(e^{A_s \circ A_s}\right) - P] = 0 \end{aligned} \tag{14}$$

where Ω represents all of the model parameters to be learned including $\mathbf{c}_1, \ldots, \mathbf{c}_S$ and the parameters of the recurrent network for modeling hidden state transitions and the graph generator networks. In what follows, we assume a one-step-ahead vector autoregressive model without contemporaneous effects. It is straightforward to extend the framework to use multi-step-ahead vector autoregressive model and/or contemporaneous effects.

4.4 A Training Algorithm

Eq.(14) specifies a constrained optimization problem. Hence, the standard backpropagation algorithm which performs an unconstrained optimization of neural network parameters cannot be directly used to train SrVARM. Hence, following [53], we adopt the augmented Lagrangian method and approximate Eq.(14) by an unconstrained problem given by:

$$\begin{aligned} D(\gamma) &= \min_{\Omega} \mathcal{L}_\rho(\Omega, \gamma) \\ \mathcal{L}_\rho(\Omega, \gamma) &= E(\Omega) + \frac{\rho}{2}|h(\Omega)|^2 + \gamma h(\Omega) \end{aligned} \tag{15}$$

where

$$\begin{aligned} E(\Omega) = &\sum_{i=1}^{N}\sum_{t=1}^{T-1}\left[\left\|\mathbf{x}_{t+1}^i - W_{s_t^i}\mathbf{x}_t^i\right\|^2 + \lambda_1\mathcal{E}(\hat{\boldsymbol{\alpha}}_t^i)\right] \\ &+ \lambda_2 KL(\{\hat{\boldsymbol{\alpha}}_{t'}^i\}, \mathbf{z}) \end{aligned} \tag{16}$$

The resulting optimization problem does not have a closed-form solution for deep neural networks with non-linear activation functions. Hence, we solve the resulting optimization problem by iteratively solving the primal problem $\Omega_\gamma^* = arg\min_\Omega \mathcal{L}_\rho(\Omega, \gamma)$, using gradient descent, and the dual problem, $\gamma = \gamma + \rho h(\Omega_\gamma^*)$, using gradient ascent. At each primal update step, we evaluate the primal function and the gradients with respect to model parameters and update the parameters using gradient descent for a given number of iterations. Then, we perform gradient ascent for the dual problem.

In addition, during the primal update step, an early stopping criterion is periodically checked to monitor the current best model parameters that minimize $E(\Omega)$ on a validation set and decrements a *patience* tolerance if the performance improvement over the period has not exceeded a pre-specified threshold. Once the patience tolerance, p_{tol} has been reached, the training phase is terminated and the best model parameters are retrieved. The patience tolerance countdown begins when the first set of model parameters that satisfy the acyclicity threshold, ϵ_{DAG}, are found. Once the countdown begins, the performance is considered to be improved when: (1) $h(\Omega) < \epsilon_{DAG}$; and (2) $E(\Omega)$ on validation set is decreased by at least a percentage threshold, δ, from the previous best value. Algorithm 1 summarizes the SrVARM training procedure.

Algorithm 1: Training algorithm of SrVARM

```
Input: Multi-variate time series data, training set $\mathcal{X}_T$ and
       validation set $\mathcal{X}_V$
Parameter: Maximum Lagrangian iteration $n_{Lag}$, primal
           update iteration $n_p$, examine period $k$,
           improvement threshold $\delta$, DAG improvement
           threshold $\sigma$, maximum quadratic penalty
           coefficient $\rho_{max}$
Output: Model parameters $\Omega$
Initialize $\rho = 1, \gamma = 0, stop = False, h_{old} = \infty$;
while $n_{Lag}$ not exceeded do
    $it = 1$;
    /* Primal update step */
    while $it \leq n_p$ do
        Forward pass to evaluate $\mathcal{L}_\rho(\Omega_{it-1}, \gamma)$ on $\mathcal{X}_T$;
        Backward pass to compute gradients and update
         $\Omega_{it}$;
        if $it\%k = 0$ then
            Evaluate $E(\Omega_{it})$ on $\mathcal{X}_V$;
            $stop$ = earlyStop($E(\Omega_{it}), h(\Omega_{it})$);
            /* "earlyStop" implements early stopping test */
        end
        if $stop$ then
            Terminate training
        end
        $it = it + 1$;
    end
    if $h(\Omega_{it}) > \sigma \cdot h_{old}$ and $\rho < \rho_{max}$ then
        $\rho = \rho \times 10$
    end
    $h_{old} = h(\Omega_{it})$;
    /* Dual update step */
    $\gamma = \gamma + \rho h(\Omega_{it})$
end
$\Omega$ = load best parameter set;
```

5 EXPERIMENTS AND RESULTS

We now proceed to describe the results of experiments with both synthetic as well as real-world data that are designed to evaluate the effectiveness of the proposed SrVARM framework for learning the pattern of transitions between hidden states and the state-dependent structure of inter-variable dependencies from MTS data. We aim to answer the following questions:

- Can SrVARM successfully recover the pattern of transitions among hidden states?
- How effectively does SrVARM learn the state-dependent DAGs that model the relations between observed variables?
- How does each component of SrVARM contribute to its performance?

5.1 Baseline Methods

To the best of our knowledge, this paper is the first to address the problem of simultaneously learning the pattern of transitions between discrete hidden states and the state-dependent structure of dependencies between observed variables from MTS data. Hence, we compare SrVARM with several representative baseline methods for learning the structure of dependencies and/or the pattern of transitions among hidden states:

- **SCGL** [50], a deep neural network-based graph learning method. It explores nonlinear Granger causality on both temporal and inter-variable relations without assuming any predefined kernel or distributional assumptions. It exploits low-rank approximation to ensure scalability. It is designed to learn a single static graph from MTS data.
- **DNT-static** [31], a state-of-the-art score-based structure learning algorithm that recovers (static) DAG structured dependencies between observed variables from MTS data by exploiting the smooth acyclicity constraint.
- **DNT-vary**, a direct extension of DNT-static to accommodate the learning time-varying relationships among observed variables from MTS data. It assumes that the hidden state transitions underlying the MTS data instances from different individuals are aligned relative to a common frame of reference. It learns time-varying DAGs applying DNT-static at each time point.

- **CPF-SAEM** [19], an algorithm for discovering DAG-structured dependencies using a nonlinear state-space model in the non-stationary setting. The state-space model estimation performed by a stochastic approximation of expectation maximization aided by conditional particle filers with ancestor sampling. The CPF-SAEM learns a DAG for each time point. CPF-SAEM does not restrict the hidden states to be discrete.
- **SRLSTM** [48], a state-regularized LSTM network that models the transitions among hidden states to enhance the interpretability of recurrent neural networks learned from temporal data. The SRLSTM models only the transitions between discrete hidden states transition and not the state-dependent structure of relationships among variables.

The DNT baselines are implemented with SciPy [47] using the L-BFGS-B [56] to solve the constrained optimization problem. CPF-SAEM is implemented in MATLAB and is made publicly available by the authors of [19][1]. The SCGL, SRLSTM, and SrVARM are implemented with PyTorch [32] and trained with the Adam optimizer [20] using the default parameter settings.

5.2 Evaluation Using Simulated Data

We evaluated the performance of SrVARM using experiments on simulated MTS data, where the underlying pattern of transitions between hidden states as well as the state-dependent DAG-structured dependencies between observed variables is known.

5.2.1 **Simulated Data.** We assume that the data generating process can be modeled using four hidden states , i.e. $s \in \{1, 2, 3, 4\}$, each with a state-specific DAG structure encoded by an autoregressive matrix A_s. State transitions are encoded by the sequence $\left[s_1^i, \ldots, s_{T-1}^i\right]$, where s_t^i is the state indicator for MTS data instance i at time point t. If the sequence of hidden state transitions are aligned relative to a common frame of reference and the state transition pattern is state-invariant, each MTS data sample is produced by an identical pattern of transitions between hidden states, we can set $s_t^i = s_t^j$, for all individuals i, j, and for all t. We refer to such a simulated data set as a *static pattern* data set. In contrast, when the state transition sequences from different individuals are not aligned relative to a common frame of reference, s_t^i and s_t^j are not necessarily identical. We consider four types of transition patterns, one of which is randomly chosen for each MTS data instance. We refer to the resulting simulated data set as a *dynamic pattern* data set. Table 1 shows the state transition patterns for static pattern data set and the four dynamic pattern data sets.

To construct each state-dependent autoregressive matrix, we first generate unweighted DAGs for each state and then sample random weights for the edges. For the set of source nodes common in all state-dependent DAGs, the observed variable values are modeled using predefined functions, including linear, sine, cosine, logarithm. The observed values for the rest of the variables are generated based on the corresponding sequence of hidden states and the state-dependent autoregressive matrices as follows: the following

[1]Implementation shared by the authors: https://github.com/Biwei-Huang/Causal-discovery-and-forecasting-in-nonstationary-environments

Table 1: State transition patterns for static pattern and dynamic pattern data sets.

	Static Pattern	Dynamic Pattern
State Transition Pattern	$\{1, 1, 1, 2, 2, 3, 3, 4, 4, 4$	$\begin{cases} 1, 1, 1, 2, 2, 3, 3, 4, 4, 4 \\ 2, 2, 1, 1, 1, 3, 3, 4, 4, 4 \\ 1, 1, 1, 3, 3, 2, 2, 4, 4, 4 \\ 1, 1, 1, 2, 2, 4, 4, 4, 3, 3 \end{cases}$

Table 2: Summary of the simulated data sets.

Static	Syn1	Syn2	Syn3
N	500	2000	5000
T	11	11	11
P	5	10	20
Dynamic	Syn1_d	Syn2_d	Syn3_d
N	500	2000	5000
T	11	11	11
P	5	10	20

formula,

$$\begin{aligned} \mathbf{x}_{t,p}^i &= w_p(t) \ + \epsilon, p \in \text{ source nodes} \\ \mathbf{x}_{t,p'}^i &= A_{s_{t-1}^i}\left[p', :\right] \mathbf{x}_{t-1}^i + \epsilon, p' \notin \text{ source nodes} \end{aligned} \tag{17}$$

where $\mathbf{x}_{t,p}^i$ is the p-th variable observed at time t, $w_p(\cdot)$ is the predefined function for the corresponding source node, $A_{s_t^i}[p', :]$ is the $p'-$th row of the autoregressive matrix, and ϵ is Gaussian noise. In our experiments, the parameters are set as follows: length of the temporal data $T = 11$, number of variables P varies from 5, 10, to 20, and number of instances N ranges from 500, 2000, and 5000. A summary of the simulated datasets is listed in Table 2, where static pattern dataset 1 is labeled as "Syn1" and dynamic pattern dataset 1 is labeled as "Syn1_d" and so on.

5.2.2 **Experiments with Simulated Data.** We proceed to provide details of parameter settings of the proposed model for experiments on the simulated data sets. The dimensionality of the state encoding $\mathbf{c}_s$, denoted by $d_{\mathbf{c}_s}$, is selected from $\{4, 8, 16\}$ for experiments on Syn1 and Syn1_d, from $\{8, 16, 32\}$ for experiments on Syn2 and Syn2_d, and from $\{16, 32, 64\}$ for experiments on Syn3 and Syn3_d, respectively. The number of hidden nodes of the graph generator networks of $g_A(\cdot)$ and $g_D(\cdot)$ are set to be equal and chosen from $d_{\mathbf{c}_s} \times \{10, 20\}$ on Syn1 and Syn1_d, $d_{\mathbf{c}_s} \times \{20, 40\}$ on Syn2 and Syn2_d, and $d_{\mathbf{c}_s} \times \{40, 80\}$ on Syn3 and Syn3_d, respectively. The coefficient for the one-hot distribution regularizer λ_1 is set to 0.1, and that for the state frequency regularization is selected from $\{0.1, 1\}$. Each experiment is repeated 5 times and the performance averaged over the 5 runs is reported along with its standard deviation.

5.2.3 **Recovering the Hidden State Transition Pattern.** To assess the effectiveness of different methods in recovering the patterns of transitions between hidden states, we define a measure of discrepancy between the actual and the learned hidden state sequences $\text{ACC}_{Trans} = \frac{1}{N(T-1)} \sum_{i=1}^{N} \sum_{t=1}^{T-1} \mathbb{1}(s_t^i, \hat{s}_t^i)$, where $\mathbb{1}(a, b) = 1$ if $a = b$ or 0 otherwise, and $s^i, \hat{s}_t^i$ are the ground truth

Table 3: State transition pattern recovery performance evaluated in terms of ACC_{Trans}. The higher the value, the better the performance.

	Syn1	Syn2	Syn3
SCGL	0.26 ± 0.05	0.20 ± 0.00	0.28 ± 0.04
DNT-static	0.22 ± 0.04	0.26 ± 0.05	0.24 ± 0.05
DNT-vary	0.30 ± 0.10	0.24 ± 0.05	0.36 ± 0.15
CPF-SAEM	0.26 ± 0.02	0.25 ± 0.00	0.28 ± 0.00
SRLSTM	0.63 ± 0.07	0.56 ± 0.05	0.63 ± 0.08
SrVARM	**0.80 ± 0.17**	**0.78 ± 0.11**	**0.76 ± 0.05**
	Syn1_d	Syn2_d	Syn3_d
SCGL	0.22 ± 0.04	0.20 ± 0.00	0.30 ± 0.00
DNT-static	0.30 ± 0.00	0.26 ± 0.05	0.28 ± 0.04
DNT-vary	0.26 ± 0.08	0.26 ± 0.05	0.30 ± 0.09
CPF-SAEM	0.30 ± 0.01	0.24 ± 0.00	0.29 ± 0.00
SRLSTM	0.75 ± 0.08	0.58 ± 0.08	0.64 ± 0.07
SrVARM	**0.92 ± 0.07**	**0.68 ± 0.18**	**0.73 ± 0.09**

state indicator and the estimated state indicator, respectively, for MTS instance i at time t. For SRLSTM and SrVARM, the hidden state transitions are naturally obtained from the outputs of the algorithms. For other methods, including SCGL, DNT-static, DNT-vary, and CPF-SAEM, they require post-processing to obtain state indicators. To be specific, the hidden state indicator is obtained by comparing the learned autoregressive matrices to the ground truth state-specific autoregressive matrices and selecting the state that yields the minimal mean square error between the two.

From the results summarized in Table 3 we observe that:

- When the state transition pattern static, the methods that learn a single global graph fail to recover the pattern of transitions between hidden states.
- When the state transition pattern is dynamic, the gap between DNT-vary and DNT-static becomes indistinguishable because of the lack of alignment of hidden state transitions across the different MTS data instances.
- SRLSTM and SrVARM explicitly model discrete hidden state transitions and outperform the other models under both static and dynamic settings.
- SrVARM outperforms all other baseline methods.

5.2.4 **Structure Learning Performance.** Following [53], we adopted a widely used evaluation metric, the structural hamming distance, denoted by ERR_{SHD}, to evaluate the quality of the learned DAG structures. The ERR_{SHD} measures the number of edges that differ between the learned DAG and the true DAG. This metric considers both the presence or absence of edge between each pair of nodes as well as the directionality of the edges, but not the weights associated with the edges as specified by the respective adjacency matrices. To obtain a more finegrained estimate of performance, we use the mean squared error ERR_{MSE}, between the estimated and the actual weighted adjacency matrices. For both metrics, the results are obtained by averaging over all time points and MTS data instances. For example, $\mathrm{ERR}_{MSE} = \frac{1}{N(T-1)} \sum_{i=1}^{N} \sum_{t=1}^{T-1} \|A_{s_t^i} - \hat{A}_{s_t^i}\|_F^2$. The two metrics taken together provide a comprehensive evaluation of performance on the task of learning the hidden state-dependent structure of dependencies between observed variables.

Table 4: Graph learning performance as measured by ERR_{SHD}. The lower value, the better performance.

	Syn1	Syn2	Syn3
SCGL	13.40 ± 0.89	74.50 ± 3.61	300.82 ± 8.56
DNT-static	8.48 ± 1.84	44.14 ± 1.40	185.70 ± 7.85
DNT-vary	**6.96 ± 0.27**	27.88 ± 3.69	80.68 ± 6.80
CPF-SAEM	9.83 ± 0.18	57.34 ± 0.65	152.69 ± 0.63
SrVARM	7.12 ± 0.72	**18.64 ± 1.07**	**59.68 ± 1.43**
	Syn1_d	Syn2_d	Syn3_d
SCGL	14.30 ± 0.45	73.98 ± 2.04	295.70 ± 2.99
DNT-static	7.42 ± 1.94	44.02 ± 3.95	184.78 ± 4.87
DNT-vary	6.24 ± 0.45	32.46 ± 3.84	148.83 ± 12.92
CPF-SAEM	12.61 ± 0.10	53.67 ± 0.51	207.04 ± 0.85
SrVARM	**5.96 ± 0.56**	**20.05 ± 1.49**	**60.58 ± 2.03**

Table 5: Graph learning performance as measured by ERR_{MSE}. The lower value, the better performance.

	Syn1	Syn2	Syn3
SCGL	2.21 ± 0.39	3.64 ± 0.53	6.16 ± 0.29
DNT-static	2.32 ± 1.01	13.36 ± 0.15	25.75 ± 0.32
DNT-vary	2.61 ± 0.30	7.93 ± 0.24	9.42 ± 0.36
CPF-SAEM	2.34 ± 0.26	10.75 ± 0.16	7.52 ± 0.09
SrVARM	**1.65 ± 0.11**	**1.85 ± 0.12**	**4.11 ± 0.12**
	Syn1_d	Syn2_d	Syn3_d
SCGL	2.39 ± 0.43	3.78 ± 0.31	6.53 ± 0.24
DNT-static	3.49 ± 2.31	13.30 ± 0.28	25.75 ± 0.32
DNT-vary	2.82 ± 0.26	9.87 ± 0.26	20.98 ± 0.30
CPF-SAEM	4.81 ± 0.09	9.48 ± 0.15	17.67 ± 0.13
SrVARM	**1.19 ± 0.08**	**2.04 ± 0.09**	**3.17 ± 0.09**

Table 6: State transition monitoring performance on SDB.

SDB	SRLSTM	SrVARM
ACC_{Trans}	0.61 ± 0.08	**0.67 ± 0.04**

The results of ERR_{SHD} are presented in Table 4 and that of ERR_{MSE} are reported in Table 5. The results summarized in Table 4 show that SrVARM virtually outperforms all other methods in all simulated datasets with respect to the number of directed edges that are correctly recovered, with one exception: The performance of SrVARM is slightly worse than that of DNT-vary on Syn1 with a static pattern of state transitions. SrVARM outperforms all other methods with respect to ERR_{MSE}.

5.3 Evaluation on Real-world Data

We report a case study of the SrVARM on a real-world dataset, hereafter referred to as the *stress data base* (SDB) [3]. SDB contains non-EEG physiological signals used to infer the neurological status of subjects when they are stressed as well as when they are relaxed [49]. The physiological signals that are collected using non-invasive wrist worn biosensors consist of 7 variables, including electrodermal activity (EDA), body temperature, heart rate (HR), arterial oxygen level (SpO2), among others. The onset and end times of each relax/stress state are subject-specific and the ground truth time indices are provided. (Additional details regarding the experimental and measurement setup are available in [3]). Note that in the case of SDB, the hidden states that generate the observed signals are known, making it possible for us to objectively evaluate our methods.

To demonstrate the effectiveness of SrVARM in correctly identifying the pattern of state transitions from SDB data and report the results in Table 6. We limit our comparisons to SRLSTM and SrVARM, the two methods that outperform other baselines on simulated data. SrVARM outperforms SRLSTM in recovering the hidden state sequence from SDB. Fig. 4 shows a visual comparison of the recovered sequence of state transitions against the ground truth sequence from SDB on two participants. We note that SrVARM is effective at recovering the transitions between relaxed state and stressed states.

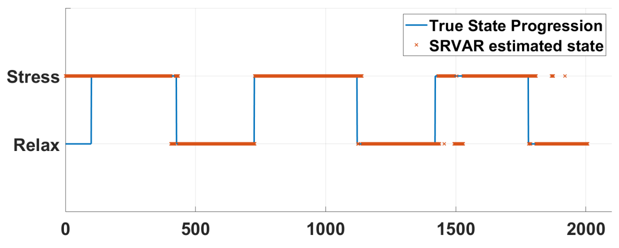
The state-dependent inter-variable relationships extracted by SrVARM from SDB data are shown in Fig. 5. An edge denotes a nonzero entry in the corresponding state-dependent autoregressive matrix encoding the DAG-structured inter-variable relationships. The thickness of each edge is proportional to the magnitude of its weight in the autoregressive matrix. When the participants are in relaxed state, the biological measurements are related by a sparsely connected DAG. In contrast, in the stressed state, the same variables are related by a more densely connected DAG. Specifically, we can observe that variables "Acc$_x$" and "Acc$_y$", representing accelerometer readings recorded by a wrist worn sensor that are indicative of physical activity, appear to impact the heart rate of the participants. We further note that body temperature appears to impact heart rate when the participants are in a stressed state, which is consistent with the finding that body temperature is an independent determinant of heart rate, causing an increase of approximately 10 beats per minute per degree centigrade increase in temperature [6]. We also note that body temperature appears to impact skin conductance (EDA), which is consistent with reports that increase in body temperature could raise skin conductance level [9].

5.4 Ablation Studies of SrVARM

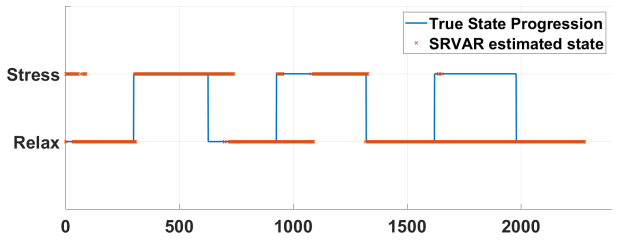
We present results of ablation studies that examine the relative contributions of some of the key design choices in SrVARM.

5.4.1 **State Frequency Regularization.** In this experiment, we examine the impact of incorporating prior state frequency knowledge by way of the state frequency regularization term in SrVARM. We test the models with and without state frequency regularizer on the imulated data set Syn1. The results shown in the left hand side of Table 7 suggest that state frequency regularization improves the performance of SrVARM as measured by its ability to recover the pattern of transitions between hidden states.

5.4.2 **Intra-variable Time-lagged Effect.** We compare the structure learning performance of SrVARM variants with and without inclusion of intra-variable time-lagged effect on the simulated data set Syn1. The results displayed in the right hand side of Table 7 show SrVARM variant which includes intra-variable time-lagged effects outperforms its counterpart that omits such effects. This suggests that modeling the intra-variable time-lagged effects helps alleviate the "source-node vanishing" which would otherwise degrade SrVARM's ability to accurately learn the state-dependent DAG-structured inter-variable relationships from MTS data.

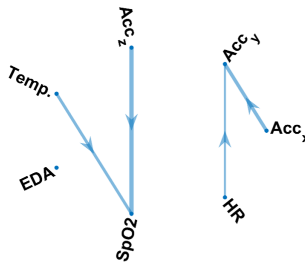


(a)

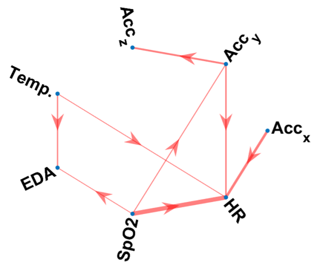


(b)

Figure 4: State transition modeling performance on SDB. (Best viewed in color).



(a) Relax State



(b) Stress State

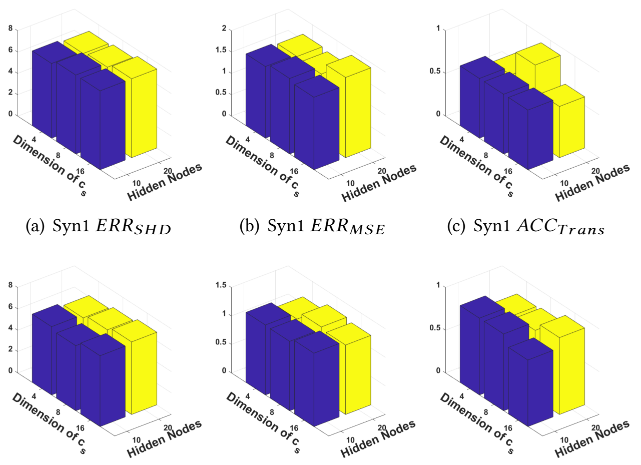
Figure 5: State-dependent inter-variable relations in relaxed/stressed states learned from SDB. An edge denotes a relationship between the connected variables and the thickness of the edge is proportional to the magnitude of the weight in the corresponding adjacency matrix.

Table 7: Effect of the regularization terms

Syn1	State Freq.	Intra-variable Time-lagged Effect	
	ACC$_{Trans}$	ERR$_{SHD}$	ERR$_{MSE}$
Without	0.30	10.00	2.20
With	**0.80**	**7.12**	**1.65**

5.5 Parameter Sensitivity Analyses

We conducted parameter sensitivity analyses to examine how the choice of tunable parameters impact the performance of SrVARM. Specifically, we considered the impact of dimensionality of the state



(a) Syn1 ERR_{SHD} (b) Syn1 ERR_{MSE} (c) Syn1 ACC_{Trans}

(d) Syn1_d ERR_{SHD} (e) Syn1_d ERR_{MSE} (f) Syn1_d ACC_{Trans}

Figure 6: Parameter sensitivity analysis on the dimension of state encoding and the number of hidden nodes in graph generation network.

encodings $\mathbf{c}_s$, the hidden layer size of graph generation networks $g_A(\cdot)$ and $g_D(\cdot)$ on the performance of SrVARM.

5.5.1 **Dimension of $\mathbf{c}_s$ and Size of Graph Generator Network.** We varied the dimensionality of state encodings and the number of hidden nodes in the graph generator networks. In each case, we examine the performance of SrVARMs trained with on simulated data sets Syn1 and Syn1_d with respect to their efficacy in recovering the pattern of hidden state transitions and the state-dependent structure of inter-variable dependencies. The dimensionality of state encoding is chosen from $\{4, 8, 16\}$, and the hidden layer size is chosen from $\{10, 20\}$. The hidden layer sizes of $g_A(\cdot)$ and $g_D(\cdot)$ are set to be equal. From the results of of our comparisons summarized in Fig. 6, it can be observed that the performance of SrVARM, as measured by ERR_{SHD} and ERR_{MSE}, are rather stable to the changes in the parameters. We conclude that SrVARM is relatively insensitive to the choice of the dimensionality of state encodings and the size of the hidden layer of the graph generator networks. In contrast, the performance of SrVARM in recovering the pattern of hidden state transitions appears to be more sensitive to the setting of these parameters.

5.5.2 **Number of States.** We examined how the performance of SrVARM on recovering the pattern of hidden state transitions is affected by the choice of number of hidden states of SrVARM. We varied the number of hidden states from 2 to 6 in increments of 2, in our experiments with simulated data sets Syn1 and Syn1_d. The results of these experiments summarized in Table 8 suggest, perhaps not surprisingly, that SrVARM perform optimally in recovering the pattern of hidden state transitions when the specified number of states equal to the actual number of hidden states of the underlying data generating process, which is 4 in the case of both the simulated data sets Syn1 and Syn1_d. We also observe that the performance with greater than the actual number of hidden states ($S = 6$) is better compared that with less than the actual number of hidden states ($S = 2$).

Table 8: Effect of the number of states.

ACC_{Trans}	$S = 2$	$S = 4$	$S = 6$
Syn1	0.40	**0.80**	0.50
Syn1_d	0.60	**0.92**	0.67

6 CONCLUSION

Many applications, e.g., healthcare, education, economics, life sciences, call for effective methods methods for constructing predictive models from high dimensional time series data where the relationship between variables can be complex, and driven by unobserved transitions among hidden system states. To address the needs of such applications, we introduced SrVARM, a novel State-Regularized Autoregressive Model, for jointly learning the state-dependent relationships among variables as well as the pattern of transitions between hidden states from multi-variate time series data. The results of our extensive experiments with simulated data as well as a real-world benchmark that show that SrVARM outperforms state-of-the-art baselines in recovering the unobserved state transitions and discovering the state-dependent relationships among variables.

Some promising directions for future research include extensions of the SrVARM framework to explainable learning of predictive models from time series data [18], functional data [17], longitudinal data [23, 25, 26] - where the time interval between measurements can be irregular for each individual and variable across individuals and only a small subset of the variables are actually measured at any given time, and the missing measurements are not missing at random.

Acknowledgements. This work was funded in part by the NIH NCATS grant UL1 TR002014 and by NSF grants 2041759, 1636795, 1909702, and 1955851, the Edward Frymoyer Endowed Professorship at Pennsylvania State University and the Sudha Murty Distinguished Visiting Chair in Neurocomputing and Data Science funded by the Pratiksha Trust at the Indian Institute of Science (both held by Vasant Honavar).

REFERENCES

[1] Ahmed M Alaa and Mihaela van der Schaar. 2019. Attentive state-space modeling of disease progression. In *NeurIPS*. 11334–11344.
[2] Mehmet Balcilar, Reneé Van Eyden, Josine Uwilingiye, and Rangan Gupta. 2017. The Impact of Oil Price on South African GDP Growth: A Bayesian Markov Switching-VAR Analysis. *African Development Review* 29, 2 (2017), 319–336.
[3] Javad Birjandtalab, Diana Cogan, Maziyar Baran Pouyan, and Mehrdad Nourani. 2016. A non-EEG biosignals dataset for assessment and visualization of neurological status. In *2016 IEEE International Workshop on Signal Processing Systems (SiPS)*. IEEE, 110–114.
[4] David Maxwell Chickering, David Heckerman, and Christopher Meek. 2004. Large-sample learning of Bayesian networks is NP-hard. *JMLR* 5, Oct (2004), 1287–1330.
[5] Rónán Daly, Qiang Shen, and Stuart Aitken. 2011. Learning Bayesian networks: approaches and issues. *The knowledge engineering review* 26, 2 (2011), 99–157.
[6] Patrick Davies and I Maconochie. 2009. The relationship between body temperature, heart rate and respiratory rate in children. *Emergency Medicine Journal* 26, 9 (2009), 641–643.
[7] Thomas Dean and Keiji Kanazawa. 1989. A model for reasoning about persistence and causation. *Computational intelligence* 5, 2 (1989), 142–150.
[8] Selva Demiralp and Kevin D Hoover. 2003. Searching for the causal structure of a vector autoregression. *Oxford Bulletin of Economics and statistics* 65 (2003), 745–767.
[9] Sigrun Doberenz, Walton T Roth, Eileen Wollburg, Nina I Maslowski, and Sunyoung Kim. 2011. Methodological considerations in ambulatory skin conductance monitoring. *International Journal of Psychophysiology* 80, 2 (2011), 87–95.

[10] Mathias Drton and Marloes H Maathuis. 2017. Structure learning in graphical modeling. *Annual Review of Statistics and Its Application* 4 (2017), 365–393.

[11] Clark Glymour, Kun Zhang, and Peter Spirtes. 2019. Review of causal discovery methods based on graphical models. *Frontiers in Genetics* 10 (2019).

[12] Marco Grzegorczyk and Dirk Husmeier. 2009. Non-stationary continuous dynamic Bayesian networks. In *Advances in Neural Information Processing Systems*. 682–690.

[13] Shixiang Shane Gu, Zoubin Ghahramani, and Richard E Turner. 2015. Neural adaptive sequential monte carlo. In *Advances in neural information processing systems*. 2629–2637.

[14] David Heckerman. 2008. A tutorial on learning with Bayesian networks. In *Innovations in Bayesian networks*. Springer, 33–82.

[15] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[16] Seyedmohsen Hosseini and Kash Barker. 2016. A Bayesian network model for resilience-based supplier selection. *International Journal of Production Economics* 180 (2016), 68–87.

[17] Tsung-Yu Hsieh, Yiwei Sun, Suhang Wang, , and Vasant G Honavar. 2021. Functional Autoencoders for Functional Data Representation Learning. In *Proceedings of the SIAM Conference on Data Mining*.

[18] Tsung-Yu Hsieh, Suhang Wang, Yiwei Sun, and Vasant G Honavar. 2021. Explainable Multivariate Time Series Classification: A Deep Neural Network Which Learns To Attend To Important Variables As Well As Time Intervals. In *Proceedings of the ACM International Conference on Web Search and Data Mining*.

[19] Biwei Huang, Kun Zhang, Mingming Gong, and Clark Glymour. 2019. Causal discovery and forecasting in nonstationary environments with state-space models. *Proceedings of machine learning research* 97 (2019), 2901.

[20] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv:1412.6980* (2014).

[21] Timo JT Koski and John Noble. 2012. A review of Bayesian networks and structure learning. *Mathematica Applicanda* 40, 1 (2012).

[22] Sébastien Lachapelle, Philippe Brouillard, Tristan Deleu, and Simon Lacoste-Julien. 2019. Gradient-Based Neural DAG Learning. In *International Conference on Learning Representations*.

[23] Thanh Le and Vasant G Honavar. 2020. Dynamical Gaussian Process Latent Variable Model for Representation Learning from Longitudinal Data. In *Proceedings of the Conference on Foundations of Data Science*.

[24] Seungsin Lee, Jungkun Park, Hyowon Hyun, Seungyup Back, Sukhyung Bryan Lee, Frances Gunn, and Jiseon Ahn. 2018. Seasonality of consumers' third-party online complaining behavior. *Social Behavior and Personality: an international journal* 46, 3 (2018), 459–470.

[25] Junjie Liang, Yanting Wu, Dongkuan Xu, and Vasant G Honavar. 2021. Longitudinal Deep Kernel Gaussian Process Regression. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*.

[26] Junjie Liang, Dongkuan Xu, Yiwei Sun, and Vasant G Honavar. 2020. LMLFM: Longitudinal Multi-Level Factorization Machine. In *Proceedings of the 34th AAAI Conference on Artficial Intelligence*.

[27] Nicholas M Luscombe, M Madan Babu, Haiyuan Yu, Michael Snyder, Sarah A Teichmann, and Mark Gerstein. 2004. Genomic analysis of regulatory network dynamics reveals large topological changes. *Nature* 431, 7006 (2004), 308–312.

[28] Helmut Lütkepohl. 2005. *New introduction to multiple time series analysis*. Springer Science & Business Media.

[29] Kevin Patrick Murphy and Stuart Russell. 2002. Dynamic bayesian networks: representation, inference and learning. (2002).

[30] Richard E Neapolitan et al. 2004. *Learning bayesian networks*. Vol. 38. Pearson Prentice Hall Upper Saddle River, NJ.

[31] Roxana Pamfil, Nisara Sriwattanaworachai, Shaan Desai, Philip Pilgerstorfer, Paul Beaumont, Konstantinos Georgatzis, and Bryon Aragam. 2020. DYNOTEARS: Structure Learning from Time-Series Data. *arXiv:2002.00498* (2020).

[32] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. PyTorch: An imperative style, high-performance deep learning library. In *NeurIPS*. 8024–8035.

[33] Jagath C Rajapakse and Juan Zhou. 2007. Learning effective brain connectivity with dynamic Bayesian networks. *Neuroimage* 37, 3 (2007), 749–760.

[34] Syama Sundar Rangapuram, Matthias W Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski. 2018. Deep state space models for time series forecasting. In *NeurIPS*. 7785–7794.

[35] Joshua W Robinson and Alexander J Hartemink. 2009. Non-stationary dynamic Bayesian networks. In *Advances in neural information processing systems*. 1369–1376.

[36] Joshua W Robinson and Alexander J Hartemink. 2010. Learning non-stationary dynamic Bayesian networks. *Journal of Machine Learning Research* 11, Dec (2010), 3647–3680.

[37] Sydney Roslow, Tiger Li, and JAF Nicholls. 2000. Impact of situational variables and demographic attributes in two seasons on purchase behaviour. *european Journal of Marketing* (2000).

[38] Karen Sachs, Omar Perez, Dana Pe'er, Douglas A Lauffenburger, and Garry P Nolan. 2005. Causal protein-signaling networks derived from multiparameter single-cell data. *Science* 308, 5721 (2005), 523–529.

[39] Mauro Scanagatta, Antonio Salmerón, and Fabio Stella. 2019. A survey on Bayesian network structure learning from data. *Progress in Artificial Intelligence* (2019), 1–15.

[40] Marco Scutari, Catharina Elisabeth Graafland, and José Manuel Gutiérrez. 2019. Who learns better Bayesian network structures: Accuracy and speed of structure learning algorithms. *International Journal of Approximate Reasoning* 115 (2019), 235–253.

[41] Gautam Singh, Jaesik Yoon, Youngsung Son, and Sungjin Ahn. 2019. Sequential Neural Processes. In *Advances in Neural Information Processing Systems*. 10254–10264.

[42] Le Song, Mladen Kolar, and Eric P Xing. 2009. Time-varying dynamic bayesian networks. In *Advances in neural information processing systems*. 1732–1740.

[43] Peter Spirtes, Clark N Glymour, Richard Scheines, and David Heckerman. 2000. *Causation, prediction, and search*. MIT press.

[44] Xianfeng Tang, Huaxiu Yao, Yiwei Sun, Charu Aggarwal, Prasenjit Mitra, and Suhang Wang. 2020. Joint modeling of local and global temporal dynamics for multivariate time series forecasting with missing values. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 5956–5963.

[45] Alex Tank, Ian Cover, Nicholas J Foti, Ali Shojaie, and Emily B Fox. 2017. An interpretable and sparse neural network model for nonlinear granger causality discovery. *arXiv preprint arXiv:1711.08160* (2017).

[46] Alex Tank, Ian Covert, Nicholas Foti, Ali Shojaie, and Emily Fox. 2018. Neural granger causality for nonlinear time series. *arXiv:1802.05842* (2018).

[47] Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. 2020. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature methods* 17, 3 (2020), 261–272.

[48] Cheng Wang and Mathias Niepert. 2019. State-Regularized Recurrent Neural Networks. In *ICML*. 6596–6606.

[49] Dilranjan S Wickramasuriya, Chaoxian Qi, and Rose T Faghih. 2018. A state-space approach for detecting stress from electrodermal activity. In *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE, 3562–3567.

[50] Chenxiao Xu, Hao Huang, and Shinjae Yoo. 2019. Scalable Causal Graph Learning through a Deep Neural Network. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 1853–1862.

[51] Yue Yu, Jie Chen, Tian Gao, and Mo Yu. 2019. DAG-GNN: DAG Structure Learning with Graph Neural Networks. In *ICML*. 7154–7163.

[52] Bin Zhang, Chris Gaiteri, Liviu-Gabriel Bodea, Zhi Wang, Joshua McElwee, Alexei A Podtelezhnikov, Chunsheng Zhang, Tao Xie, Linh Tran, Radu Dobrin, et al. 2013. Integrated systems approach identifies genetic nodes and networks in late-onset Alzheimer's disease. *Cell* 153, 3 (2013), 707–720.

[53] Xun Zheng, Bryon Aragam, Pradeep K Ravikumar, and Eric P Xing. 2018. DAGs with NO TEARS: Continuous optimization for structure learning. In *NeurIPS*. 9472–9483.

[54] Xun Zheng, Manzil Zaheer, Amr Ahmed, Yuan Wang, Eric P Xing, and Alexander J Smola. 2017. State space LSTM models with particle MCMC inference. *arXiv:1711.11179* (2017).

[55] Yang Zhou. 2011. Structure learning of probabilistic graphical models: a comprehensive survey. *arXiv preprint arXiv:1111.6925* (2011).

[56] Ciyou Zhu, Richard H Byrd, Peihuang Lu, and Jorge Nocedal. 1997. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software (TOMS)* 23, 4 (1997), 550–560.

[57] Shengyu Zhu and Zhitang Chen. 2019. Causal discovery with reinforcement learning. *arXiv:1906.04477* (2019).