The Workshops of the Thirty-Second
AAAI Conference on Artificial Intelligence

Learning the Influence Structure between Partially Observed Stochastic Processes Using IoT Sensor Data

Ritesh Ajoodha, Benjamin Rosman
School of Computer Science and Applied Mathematics
The University of the Witwatersrand, Johannesburg, and
The Council of Scientific and Industrial Research, Pretoria
South Africa

Abstract

The recent widespread of availability of sensors, as part of the IoT, presents the opportunity to learn the properties of compound distributions in practical applications. Understanding temporal distributions by observations collected from the IoT can advance many intelligent applications. In this paper we develop an algorithm to learn influence between stochastic processes using observations obtained from the IoT. The proposed method learns these processes using temporal models independently, and then attempts to recover the underlying distribution of influence between them. Experimental results are provided which demonstrate the effectiveness of our method. This approach is useful in applications that require an understanding of how partially observed high-level processes can influence each other given a set of observations at different times.

Introduction

The Internet of Things (IoT) is a network of on-line devices which are able to exchange data. These devices contain sensors which collect observations from environments. Several such observations can be aggregated to describe some partially observed phenomena whose recovery can be useful for intelligent applications. For example, intelligent transportation systems (ITS) attempts to provide traffic management that allows users to traverse transportation networks in a safe and effective manner.

Predicting the influence of traffic conditions can be achieved by aggregating features obtained from the IoT that describe transportation networks. Such features include weather observations (smart watches), the time, number of cars (smart phones), and event start-times (smart calenders). All of this data can be collected at different times of the day from on-line smart devices to tell us something about temporal phenomena (e.g. network traffic).

In this paper we provide a theoretical foundation for such temporal intelligent applications. We provide an algorithm to learn influence between partially observed stochastic processes described by IoT observations. More specifically, we wish to learn the structure and parameters of a probability distribution which describes the influence between a set of partially observed processes.

Learning a probability distribution for temporal processes can aid in performing density estimation for trajectories; or perhaps for mere knowledge discovery where we are interested to learning how processes interact (e.g. learning the influence of how the mood of workers in an office affect the performance and work-flow of activities in a corporate environment. In this case, IoT can provide sensor data from smart-watches, cell phones, etc.) to describe the hypothetical mood of individuals. This paper stems from the work of Ajoodha and Rosman (2017), who recovered the influence structure between latent variables aggregated over observations in the non-dynamic setting (Ajoodha and Rosman 2017).

Learning and representing a probability distribution over partially observed stochastic processes poses a difficult learning problem. Unlike in observable data, the likelihood function in incomplete data is not decomposable and we have to perform inference to evaluate it. This forces us to use non-linear optimisation techniques such as EM or gradient ascent (Binder et al. 1997) to optimise the likelihood function.

The novelty and intuition of our approach is to learn the optimal influence structure in layers. We firstly learn a set of independent models (as HMMs), and thereafter, optimise a structure score over possible structural configurations. Since the search for the optimal structure is done using complete data we can take advantage of efficient learning procedures and significant computational savings from the structure learning literature (Koller and Friedman 2009).

We provide the following contributions: (a) we introduce the notion of delayed influence between HMMs; (b) extend the traditional BIC score for temporal models rather than random variables; (c) provide a complete algorithm to recover the influence structure between HMMs; (d) provide a notion of a delayed structural assemble (DSA) to relate temporal models for delayed influence; (e) and finally, provide empirical evidence for the effectiveness of our method with respect to a generative ground-truth distribution.

This paper is structured as follows. The background section reviews current literature on parameter estimation for complete and incomplete data; the related work section reviews current work in structure learning; we then provide the specification of our algorithm, this includes the modified structure score and introduces a DSA; thereafter, we

present the empirical results which show the effectiveness of our method; and finally, we provide a discussion of the results in terms of IoT sensor data.

Background

A Bayesian network (Jensen 1996) is a directed acyclic graph (DAG) whose nodes represent random variables and whose edges represent the influence of one variable over another. A Bayesian network structure is often established as a set of independence assertions between these random variables that encode a joint distribution in a compact way (Pearl 1988).

In this section we review preliminary work in parameter estimation for Bayesian networks (Koller and Friedman 2009). We explore the basic definition of a dynamic Bayesian network (DBN), define a hidden Markov model (HMM), review methods to learning parameters for observable data; and finally, review learning parameters for missing data (Heckerman, Geiger, and Chickering 1995).

Dynamic Bayesian Networks

The Markov (representing the next state as independent from the past given the present) and time invariance (allowing the system dynamics to be stationary) assumptions (Koller and Friedman 2009) allow us to compactly represent a trajectory over time since we need only specify a 2-time-slice Bayesian network that consists of the initial distribution, $P(\mathcal{X})$, and a transition model, $P(\mathcal{X}'|\mathcal{X})$, where $\mathcal{X}$ is a template set of random variables. The transition model can then be unrolled into a dynamic Bayesian network (Koller and Friedman 2009).

A 2-time-slice Bayesian network for a process over the set of template variables $\mathcal{X}$ is a conditional Bayesian network over $\mathcal{X}'$ given $\mathcal{X}_I$, where $\mathcal{X}_I \subseteq \mathcal{X}$ is a set of interface variables (Koller and Friedman 2009).

The conditional Bayesian network only has parents, and hence conditional probability distributions (CPDs), for $\mathcal{X}'$. Interface variables refer to those variables that persist through the temporal aspect of the model (e.g. abstract changes in the process). Consequently for every template variable we have a template factor (a matrix of CPDs) which will be initialised as the model unfolds. For example, we can describe a transitional distribution as

$$P(\mathcal{X}'|\mathcal{X}) = P(\mathcal{X}'|\mathcal{X}_I) = \prod_{i=1}^{n} P(X_i'|Pa_{X_i'}),$$

where $Pa_{X_i'}$ is the parent set for X_i'. A dynamic Bayesian network (DBN) (Koller and Friedman 2009) is a pair $\langle \mathcal{B}_0, \mathcal{B}_{\rightarrow} \rangle$, where $\mathcal{B}_0$ is a Bayesian network over $\mathcal{X}^{(0)}$, representing the initial distribution over states and $\mathcal{B}_{\rightarrow}$ is a 2-time-slice Bayesian network for the process. For any desired time span $T \geq 0$, the distribution over $\mathcal{X}^{(0:T)}$ is defined as a unrolled Bayesian network, where, for any $i = 1, \ldots, n$, the structure and CPDs of $\mathcal{X}_i^{(0)}$ are the same as those for $\mathcal{X}_i$ in $\mathcal{B}_0$; the structure and CPDs of $\mathcal{X}_i^{(t)}$ for $t > 0$ are the same as those for $\mathcal{X}_i^{'}$ in $\mathcal{B}_{\rightarrow}$.

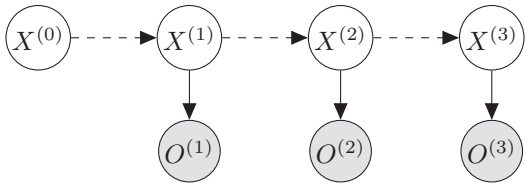

Figure 1: An illustration of a hidden Markov model with 4 timeslices. The dotted lines indicate the inter-time-slice edges for the persistent variable $X^{(t)}$. The solid lines indicate the intra-time-slice edges for each respective time-slice.

Hidden Markov Models

A hidden Markov model is a DBN (Eddy 1996). The likelihood function of the hidden Markov model, as illustrated in Figure 1, relative to the data decomposes as

$$L(\Theta : X^{0:T}, O^{0:T}) = \prod_{i,j} \theta_{X^i \rightarrow X^j}^{M[X^i \rightarrow X^j]} \prod_{i,k} \theta_{O^k|X^i}^{M[X^i, O^k]},$$

where Θ is the complete set of parameters, $M[X^i \rightarrow X^j]$ is the count of transitions from $X^i \rightarrow X^j$ for observation k in the state i, and $M[X^i, O^k]$ is the counts for X^i and O^k. For a more detailed overview of DBNs and hidden Markov models see (Murphy 2002; Bishop 2006). In the next two section we look at learning parameters for general Bayesian networks.

Learning Parameters with Observable Data

A common parameter learning method is Bayesian Estimation (BE). BE follows the Bayesian paradigm which views any event that has uncertainty as a random variable with a distribution over it. Suppose we have a data-set, $\mathcal{D} = \{\xi_1, \ldots, \xi_M\}$, where each instance contains only one observation x, then we can express the joint distribution of each observation (as well as the parameter θ which generates the data) by using the chain rule for Bayesian networks as $P(x[1], \ldots, x[M], \theta) = P(x[1], \ldots, x[M]|\theta)P(\theta)$. This gives us the prior over θ and the probability of each instance given θ,

$$P(x[1], \ldots, x[M], \theta) = P(\theta) \prod_{i=1}^{M} P(x[i]|\theta).$$

We can express the posterior of this prior, given the data, using Bayes rule:

$$P(\theta|x[1], \ldots, x[M]) = \frac{P(x[1], \ldots, x[M]|\theta)P(\theta)}{P(x[1], \ldots, x[M])}.$$

There are many choices for a prior distribution such as the Dirichlet conjugate prior (Heckerman, Geiger, and Chickering 1995) which is characterised by a set of hyperparameters $(\alpha_1, \ldots, \alpha_k)$.

Learning Parameters with Missing Data

We now consider the much more difficult problem of learning parameters from missing data. We consider latent variables since we need to learn latent variables to aggregate observations. We often consider latent variables because they

provide a sparse parameterisation of a distribution (Koller and Friedman 2009).

Estimating missing data is a well defined problem in statistics (Rubin 1976). Since the likelihood function for missing data has multiple optima, we must resort to heuristic methods. A common algorithm to optimise the likelihood function is Expectation Maximisation (EM) which attempt to learn both the missing data and the parameters iteratively. The EM algorithm generalises several algorithms including the Baum-Welch algorithm used for learning HMMs (Dempster, Laird, and Rubin 1977). The EM algorithm is a 2-step process, the E-step and the M-step. In the E-step we perform Bayesian inference to infer the data given the parameters. That is, for each instance, $\xi[m]$, and each family, X and $\mathbf{U}$, we compute $P(X, \mathbf{U}|\xi[m], \theta^t)$, where θ^t is the current setting of the parameters at iteration t. We now can compute the expected sufficient statistics, $\hat{M}$, for each combination of values $x, \mathbf{u}$ per family. We can express the sufficient statistics as

$$\hat{M}_{\theta^t}[x, \mathbf{u}] = \sum_{m=1}^{M} P(x, \mathbf{u}|\xi[m], \theta^t).$$

In the M-step we treat the expected sufficient statistics, $\hat{M}$, as real sufficient statistics and then use BE. The EM algorithm is guaranteed to monotonically improve the likelihood of the parameters relative to the data at each iteration (Caffo, Jank, and Jones 2005).

Related Work

A popular approach to structure learning is *Score-based* structure learning which requires the definition of a hypothesis space of potential network structures; defines a structure score which gauges how well the model fits the observed training data; and finally, attempts to find the highest scoring structure as an optimisation problem. However, given that the search space over models is super-exponential in size, one resorts to heuristic techniques (O'Gorman et al. 2015).

We consider score-based structure learning (Tang and Xu 2014) since it (a) considers the complete structure between models as a state in the search space; (b) preserves basic score properties (metrics to evaluate the worth of a model relative to the data) to allow for feasible computations; and (c) it provides a clear indication of the independence assertions between the concerned structures relative to the data. In this section we review score-based structure learning components. We begin by discussing structure scores; attempt to learn tree-structured networks; and then finally, move to the more difficult problem of learning graph structures.

Structure score

A well-known choice of structure score is the Bayesian information criterion score (BIC-score) (Watanabe 2013). The BIC-score is a popular choice for trading-off model complexity and fit to data (Schwarz 1978). The BIC score consists of two terms: the first describes the fit of the hypothesised structure to the data, usually the likelihood function $score_L = \ell((\hat{\theta}, \mathcal{G}) : \mathcal{D})$; and the second is a penalty term for complex networks. More formally the BIC score is given as

$$score_{BIC} = M \sum_{i=1}^{n} \mathbf{I}_{\hat{P}}(X_i; \mathbf{Pa}_{X_i}^{\mathcal{G}}) - \frac{\log M}{2} DIM[\mathcal{G}].$$

where $\mathbf{I}_{\hat{P}}(X; Y) = \sum_{x,y} \hat{P}(x, y) \log \frac{\hat{P}(x,y)}{\hat{P}(x)\hat{P}(y)}$; $\mathbf{M}$ is the number of training instances; $DIM[\mathcal{G}]$ is the number of independent parameters in the network; and $\hat{P}$ is the empirical distribution. The likelihood term maximises the likelihood ($\hat{\theta}$) given a particular graph structure ($\mathcal{G}$) relative to the data ($\mathcal{D}$).

The BIC score has the following properties (Gelman, Hwang, and Vehtari 2014). (a) As we increase the number of samples the emphasis moves from model complexity to the fit to data. In other words, as we obtain more data we are more likely to consider more complicated structures. (b) As the BIC score acquires more data it approaches the true structure (or one which is i-equivalent, that is a structure which makes the same independence assumptions). (c) The BIC score gives the same score for members of the same i-equivalence class, that is, different structures which encode the same independent assumptions (Malone 2015). We now turn our attention to using scoring functions to learn tree structures.

Learning Tree Structures

Learning a tree structured network is perhaps the simplest structure learning problem. The first application of learning a tree structured network was proposed by (Chow and Liu 1968). Having selected a structure score, we turn our attention to an optimisation problem which attempts to maximise the selected score over potential structures.

There are several reasons that one would learn tree structured networks. Firstly, there already exists powerful algorithms for efficient optimisation over high-dimensional tree structured networks; and secondly, trees provide sparse networks with manageable generalised parameterisation which reduces over-fitting.

A general algorithm to obtain a tree structure network is to compute the score of every pair of variables and then use a maximum weighted spanning tree (MWST) algorithm (Pearl 2014). One could use any standard MWST algorithm such as Prim or Kruskal in $O(n^2)$. Using a MWST allows us to summarise the most important dependences between sets of variables with one parent.

Learning Graph Structures

In the case of learning general graph-structures the structure learning problem's complexity increases significantly (Koller and Friedman 2009). More formally (Koller and Friedman 2009), for any dataset, $\mathcal{D}$, and decomposable structure score, $score$, the problem of finding the maximum scoring network, that is,

$$\mathcal{G}^* = \arg\max_{\mathcal{G} \in \mathcal{G}_{\mathbf{d}}} score(\mathcal{G} : \mathcal{D}),$$

is NP-Hard for any $d \geq 2$, where $\mathcal{G}_{\mathbf{d}} = \{\mathcal{G} : \forall i, |Pa_{X_i}^{\mathbf{G}}| \leq d\}$. In other words, finding the maximal scoring network

structure with at most d parents for each variable is NP-hard for any d greater than 2. However, using local search procedures we are able to provide a solution using heuristic hill-climbing (Fan, Yuan, and Malone 2014).

There are two main design choices that one needs to make when using a local structure search procedure: (i) the choice of search operators and (ii) search procedure.

(i) Search operators are local steps to traverse the search space. Common choices for local search operators are edge addition, reversal, and deletion.

(ii) A common choice for a search procedure is greedy hill-climbing. The greedy hill-climbing approach starts by selecting a prior network. From this prior network we iteratively try to improve the structure's score by utilising search operators. In greedy hill-climbing we always apply the change that improves the score until no improvement can be made.

In the next section we provide the first analysis of the problem of learning delayed influence between temporal models.

Learning Delayed Influence Between HMMs

Delayed influence between two stochastic processes, $A_{\rightarrow}$ and $B_{\rightarrow}$, is defined as where a change of a variable in process $A_{\rightarrow}$ at time t_1 will only result in a change in the variables at or after t_1 in $B_{\rightarrow}$. In this paper we provide the first score-based structure learning analysis of this problem to discover the complete *delayed dynamic influence network* (DDIN) structure. More formally, a DDIN is a DBN that factorises the joint distribution between a finite set of stochastic processes represented as HMMs.

We assume that the data generated from the ground-truth influence structure has the following form: $\mathcal{D}_{\langle \mathbb{I}_0, \mathbb{I}_{\rightarrow} \rangle^{\mathcal{G}}} = \{\mathcal{D}_{\langle \mathcal{H}_0^1, \mathcal{H}_{\rightarrow}^1 \rangle}, \ldots, \mathcal{D}_{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle}\}$, where $\mathcal{D}_{\langle \mathcal{H}_0^i, \mathcal{H}_{\rightarrow}^i \rangle} = \{\xi_1, \ldots, \xi_{\mathcal{M}}\}$ is a set of M instances from HMM $\langle \mathcal{H}_0^i, \mathcal{H}_{\rightarrow}^i \rangle$. These instances are observations collected from IoT sensors which describe a latent phenomenon. Each ξ_j is generated IID over time from an underlying temporal distribution, $P^*(\langle \mathbb{I}_0, \mathbb{I}_{\rightarrow} \rangle^{\mathcal{G}})$, where $\mathbb{I}_0$ is an initial network and $\mathbb{I}_{\rightarrow}$ is an unrolled network with respect to structure $\mathcal{G}$.

$\langle \mathbb{I}_0, \mathbb{I}_{\rightarrow} \rangle^{\mathcal{G}}$ contains a distribution between a set of HMM models, $\langle \mathcal{H}_0^1, \mathcal{H}_{\rightarrow}^1 \rangle, \ldots, \langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle$, with the independence assumptions specified by $\mathcal{I}_\ell(\langle \mathbb{I}_0, \mathbb{I}_{\rightarrow} \rangle^{\mathcal{G}})$. We further assume that $P^*(\langle \mathbb{I}_0, \mathbb{I}_{\rightarrow} \rangle^{\mathcal{G}})$ is induced by another model, $\mathcal{G}^*(\langle \mathbb{I}_0, \mathbb{I}_{\rightarrow} \rangle^{\mathcal{G}})$, which we refer to as the *ground-truth* structure. We will evaluate our model by recovering the local independence assertions in $\mathcal{G}^*(\langle \mathbb{I}_0, \mathbb{I}_{\rightarrow} \rangle^{\mathcal{G}})$, denoted $\mathcal{I}_\ell(\mathcal{G}^*(\langle \mathbb{I}_0, \mathbb{I}_{\rightarrow} \rangle^{\mathcal{G}}))$, by only observing the IoT sensor data, $\mathcal{D}_{\langle \mathbb{I}_0, \mathbb{I}_{\rightarrow} \rangle^{\mathcal{G}}}$.

The architecture of the proposed algorithm is given by Figure 2. We (ii) learn each HMM independently (EM); (iii) set the independence assumptions with respect to each temporal model and learn the resulting dynamic influence network; (v) compute the structure score of the model (using a scoring function and assemble for influence networks); (vi) see if we converge or if the number of iterations (i) exceeds the iteration threshold (t); (vii) apply the operator which results in the best improvement of the score with respect to the data. Steps (iii), (iv), (v), and (vii) are repeated until we can not improve the score for the structure with respect to the data or if we exceed the specified number of iterations. We then select the best network (viii). We will refer to this algorithm as the Greedy structure search (GESS).

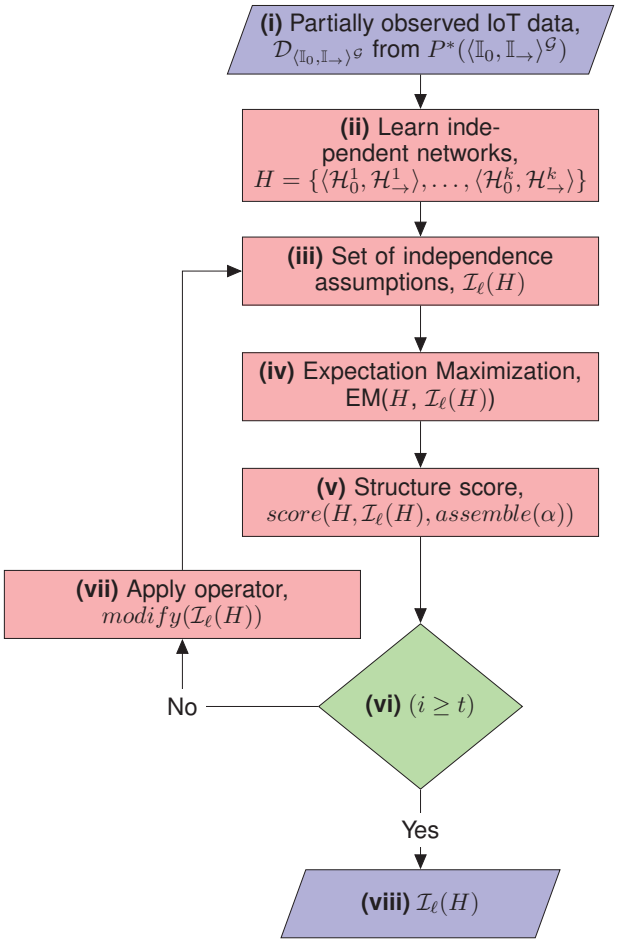

Figure 2: The architecture of the proposed algorithm.

In the next section we define the scoring function used to score DDINs; and finally, we introduce the notion of a delayed structural assemble (DSA) to relate HMMs in our DDIN.

Structure Scores for DDINs

In step (v) of Figure 2, we need to calculate the score of a DDIN. Intuitively we would like a score to measure if a set of independence assumptions between sets of HMMs is preferred, then we would get more information from having these assumptions than having different ones in $\mathcal{D}_{\langle \mathbb{I}_0, \mathbb{I}_{\rightarrow} \rangle^{\mathcal{G}}}$.

We adopt an extension of the likelihood score called the Bayesian information criterion (BIC) (Schwarz 1978) for this task. The BIC score is a derivation of the likelihood score that is biased to simpler structures, but as it acquires more data it can prefer more complex structures to describe

the distribution (Koller and Friedman 2009). In other words, it trades-off fit to data with model complexity, thereby reducing overfitting. We present the following extension of the likelihood score for a DDIN, denoted $\text{score}_{BIC}(\langle \mathbb{I}_0, \mathbb{I}_{\rightarrow} \rangle^{\mathcal{G}} : \mathcal{D}_{\langle \mathbb{I}_0, \mathbb{I}_{\rightarrow} \rangle^{\mathcal{G}}})$, in terms of the BIC score which spans sets of unrolled HMMs with respect to the Markov and time invariance assumptions:

$$M \sum_{k=1}^{K} (\sum_{t=1}^{T} (\sum_{i=1}^{N} (\mathbf{I}_{\hat{P}}(X_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}}; \mathbf{Pa}^{\mathcal{G}}_{X_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}}}) - \frac{\log M}{c} DIM[\mathcal{G}],$$

where M is the number of samples; K is the number of dependency models; T is the number of time-slices for any dependency model; N is the number of variables in each time-slice; and $DIM[G]$ is the number of independent parameters in the entire influence network.

The BIC score is simply the likelihood of the data with an added penalty term. We also notice that the BIC score for HMMs tends to trade-off the fit to $\mathcal{D}_{\langle \mathbb{I}_0, \mathbb{I}_{\rightarrow} \rangle^{\mathcal{G}}}$ with model complexity. That is, the mutual information term, $\mathbf{I}_{\hat{P}}$, grows linearly with the number of samples, and the complexity term, $\frac{\log M}{c} DIM[\mathcal{H}_0]$, grows logarithmically with the size of samples in $\mathcal{D}_{\langle \mathbb{I}_0, \mathbb{I}_{\rightarrow} \rangle^{\mathcal{G}}}$. Therefore, the larger the amount of data the more compelled the score will be to fit $\mathcal{D}_{\langle \mathbb{I}_0, \mathbb{I}_{\rightarrow} \rangle^{\mathcal{G}}}$ and thus, with enough data, prefers the set $\mathcal{I}_{\ell}(\mathcal{G}^*(\langle \mathbb{I}_0, \mathbb{I}_{\rightarrow} \rangle^{\mathcal{G}}))$. In the next subsection we address how to manage the structural relations between HMMs imposed by the DDIN.

Structure Assembles

In the previous section we introduced the BIC score for DDINs which weighted a dynamic influence network based on empirical correlation between sets of HMMs with respect to the data. Each variable in the DDIN is paired with a parent set whose members may span variables in multiple HMMs in addition to variables in its own network structure. The collective selection of parents for variables in a DDIN is decided by an imposed *delayed structural assemble* (DSA).

A DSA is a configuration which connects temporal models by their variables to other temporal models. It partly defines the parent sets for variables necessary to construct a DDIN. To intuitively capture this influence structure between temporal models we need to insert dependencies between different time-points that span various models. How far back the dependency between time-slices go depends on the influence structure of the distribution. More generally, we can describe delayed influence with respect to α-many previous time-slices for a family of temporal models. For example, Figure 4 illustrates delayed influence between two unrolled HMMs, $\langle \mathcal{A}_0, \mathcal{A}_{\rightarrow} \rangle$ and $\langle \mathcal{B}_0, \mathcal{B}_{\rightarrow} \rangle$, with $\alpha = 2$.

More generally, suppose you are given a family of HMMs and an imposed structure, where $\langle \mathcal{H}_0^0, \mathcal{H}_{\rightarrow}^0 \rangle$ represents the child with the parent set $\mathbf{Pa}^{\mathcal{G}}_{\langle \mathcal{H}_0^0, \mathcal{H}_{\rightarrow}^0 \rangle} = \{\langle \mathcal{H}_0^1, \mathcal{H}_{\rightarrow}^1 \rangle, \ldots, \langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle\}$. Further assume that $\mathcal{I}_{\ell}(\langle \mathcal{H}_0^j, \mathcal{H}_{\rightarrow}^j \rangle)$ is the same for all $j = 0, \ldots, k$. Then the *delayed* assemble network, denoted $\langle \mathcal{A}_0, \mathcal{A}_{\rightarrow} \rangle^{\mathcal{G}}$, satisfies all the independence assumptions $\mathcal{I}_{\ell}(\langle \mathcal{H}_0^i, \mathcal{H}_{\rightarrow}^i \rangle)$ $\forall\ i = 0, \ldots, k$. In addition, $\forall\ j$ and $\forall\ t$, $\langle \mathcal{A}_0, \mathcal{A}_{\rightarrow} \rangle^{(t)}$ also satisfies the following independence assumptions for each latent variable denoted L_i and some $t > \alpha \in \mathbb{Z}^+$: $\forall\ L_i^{\langle \mathcal{H}_0^0, \mathcal{H}_{\rightarrow}^0 \rangle^{(t)}}$: $(L_i^{\langle \mathcal{H}_0^0, \mathcal{H}_{\rightarrow}^0 \rangle^{(t)}} \perp\!\!\!\perp \text{NonDescendants}_{L_i^{\langle \mathcal{H}_0^0, \mathcal{H}_{\rightarrow}^0 \rangle^{(t)}}} \mid L_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}}, L_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)-1}}, \ldots, L_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)-\alpha}}, Pa_{L_i}^{\langle \mathcal{H}_0^0, \mathcal{H}_{\rightarrow}^0 \rangle^{(t)}})$. The computational complexity to analyse each step in the search space is summarised asymptotically as $O(HTX(I+\alpha)M)$.

The properties of the DSA are as follows. (a) DSA offers a sparse ensemble of dependencies with respect to α; (b) large values of α may capture a richer distribution trading off generalisation for quality density estimation; (c) the assemble provides a suitable representation for applications which we require delayed influence between processes (e.g. IoT sensors for navigation systems).

Empirical Results

In this section we present the performance of seven DDIN parameter and/or structure learning tasks with respect to the generative ground-truth DDIN's distribution. The performance is summarised in Figure 3, which shows the relative entropy to the generative ground-truth DDIN (log-scale) and execution times over the number of training samples. The results are averaged over 10 trials for the following structure learning tasks. Random structure, which used a randomly generated structure for a DDIN and learned the missing and observable parameters; No structure, which modelled each HMM as mutually independent to others and learned parameters; Tree structure, which learning a tree structure between the HMMs as well as learned parameters; BIC with GESS, which used the BIC score with GESS and learned parameters; AIC with GESS which used the AIC score with GESS and learned parameters; and finally, TSLP, which used the ground-truth structure, but relearned parameters. The second y-axis is the execution times for each learning task. In all learning tasks both the latent and observable parameters were learned from sets of synthetic sensor observations.

Discussion and Future Work

In this section we interpret the results towards using IoT sensor data for this learning task. We note that the two penalty-based learning procedures were the most successful towards recovering the ground-truth distribution. However, using no structure and tree structure generalised the ground-truth better to new instances for fewer observations (less than 200) than the two penalty-based procedures. With regard to execution time, the two penalty-based learning procedures were exponential while the others were linear. Restricting the in-degree performs faster then GESS with AIC or BIC, however under-performs during density estimation. It is also unrealistic to know the in-degree in most practical applications involving IoT sensor data.

The sensitivity of the BIC and AIC scores to judge when to restrict the structure guides the selection of independence assumptions, with roughly the same execution time, and outperforms all other methods for a large number of samples.

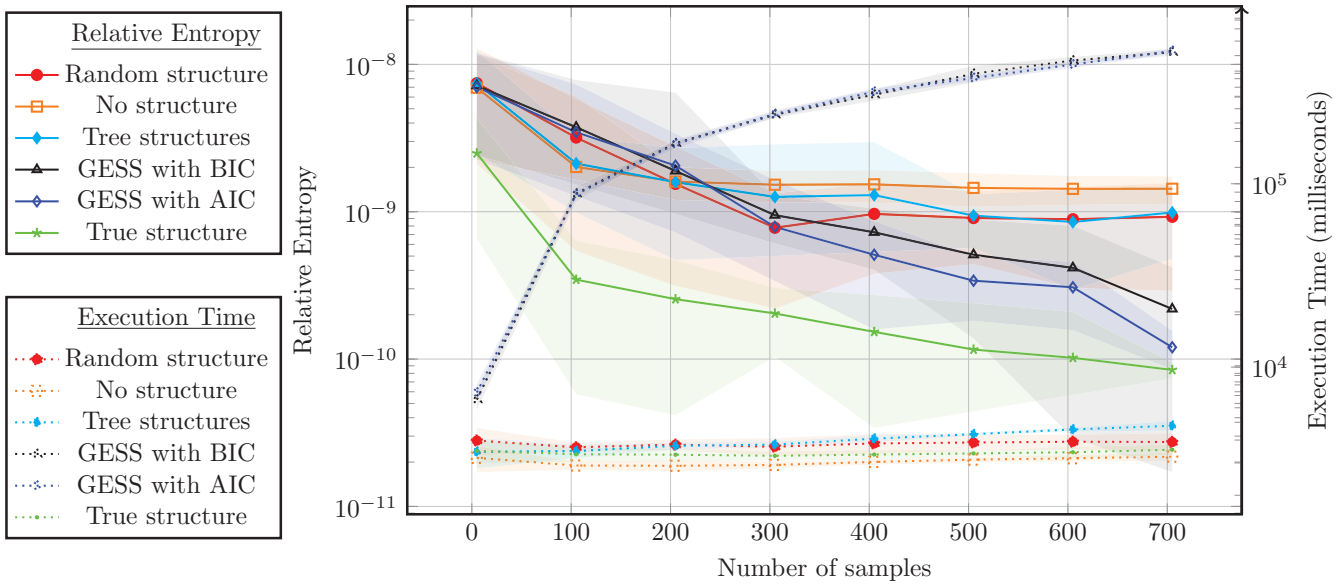

Figure 3: The performance of seven parameter and structure learning tasks. The left log-scale y-axis shows the relative entropy to the ground-truth generative DDIN; the x-axis shows the increase in sample size; and the right log-scale y-axis shows the execution time of each learning task. The parameters of the experiment is as follows. All latent variables in every HMM were learned using EM with 5 class labels, 5 observations, and 20 EM iterations to learn each latent variable. The EM recovered $65-87\%$ of the original cluster assignments. All learned variables used a Dirichlet Prior of 5. To allow for a manageable use of memory all DDINs with over 5000 independent parameters were heavily penalised. 20% of data produced was used for testing and 80% was for training. The were 50 structure search iterations used to recover each model (5 random restarts when reached local optima, and used a tabu list of length 10). The DDIN had 10 HMMs with 5 time-slices each. Finally, the ground-truth DDIN had 15 edges randomly positioned at each trail with $\alpha = 2$. Error bars are given by the shaded region.

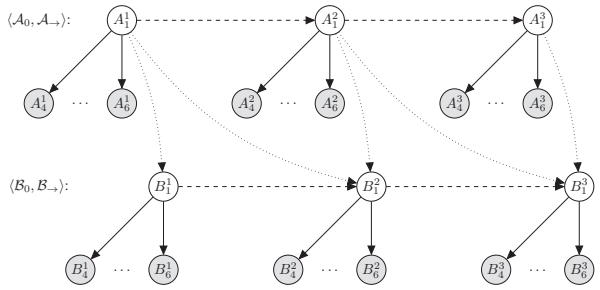

Figure 4: Two unrolled HMMs, $\langle \mathcal{A}_0, \mathcal{A}_{\rightarrow} \rangle$ and $\langle \mathcal{B}_0, \mathcal{B}_{\rightarrow} \rangle$, as represented with 3 time-slices and $\alpha = 2$. The HMMs are connected with a DSA. The unshaded variables are latent and the shaded variables are observable.

This means when we have fewer IoT observations, we are better off using sparser structures which also take a shorter time to learn. In particular, tree structures summarise the most important dependencies which allow us to perform inference faster, while random graphs provide a faster learning time but, through their density, will suffer for inference tasks.

The significance of learning a DDIN structure depends on our learning objective. If one is attempting to discover exactly the ground-truth network structure, which involves stating precisely $\mathcal{I}_{\ell}(\mathcal{G}^*(\langle \mathbb{I}_0, \mathbb{I}_{\rightarrow} \rangle^{\mathcal{G}}))$, then we should concede that there exist many *perfect maps* for $P^*(\langle \mathbb{I}_0, \mathbb{I}_{\rightarrow} \rangle^{\mathcal{G}})$ which can be recovered from $\mathcal{D}_{\langle \mathbb{I}_0, \mathbb{I}_{\rightarrow} \rangle^{\mathcal{G}}}$. Therefore, if our goal is knowledge discovery, we should instead try to recover $\mathcal{G}^*(\mathcal{H})$'s I-equivalence class. Alternatively, one could also attempt to learn the temporal influence network for *density estimation* from a set of IoT sensors, i.e. to estimate a statistical model of the underlying distribution $P^*(\langle \mathbb{I}_0, \mathbb{I}_{\rightarrow} \rangle^{\mathcal{G}})$. Such a model can be used to reason about new data instances.

One of the benefits of using Bayesian networks is the ability to encode domain expertise in a simple way. IoT data which describe real-world phenomena has mostly already been discussed and explored in other sciences. We can incorporate these finding easily in our Bayesian learning model to converge faster. For example, incorporating prior knowledge about traffic on particular transportation networks can help us make better decisions with less data to learn from.

In summary, influence networks provide rich, expressive, and intuitive understanding of influence between temporal processes deduced from IoT sensor data.

Acknowledgments

The first author gratefully acknowledges the support of the Teaching Development Grant Collaborative Project funded by the Department of Higher Education, South Africa (Ref.

APP-TDG-0020/21); *NRF Scarce Skills Doctoral Scholarship*, and *the Post-Graduate Merit Award Scholarship* for Doctoral Studies.

References

Ajoodha, R., and Rosman, B. 2017. Tracking influence between naïve bayes models using score-based structure learning. In *IEEE proceedings, Pattern Recognition Association of South Africa and Robotics and Mechatronics International Conference (PRASA-RobMech), 2017*.

Binder, J.; Koller, D.; Russell, S.; and Kanazawa, K. 1997. Adaptive probabilistic networks with hidden variables. *Machine Learning* 29(2):213–244.

Bishop, C. M. 2006. *Pattern recognition and machine learning*. springer.

Caffo, B. S.; Jank, W.; and Jones, G. L. 2005. Ascent-based monte carlo expectation–maximization. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 67(2):235–251.

Chow, C., and Liu, C. 1968. Approximating discrete probability distributions with dependence trees. *IEEE transactions on Information Theory* 14(3):462–467.

Dempster, A. P.; Laird, N. M.; and Rubin, D. B. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)* 1–38.

Eddy, S. R. 1996. Hidden markov models. *Current opinion in structural biology* 6(3):361–365.

Fan, X.; Yuan, C.; and Malone, B. M. 2014. Tightening bounds for bayesian network structure learning. In *AAAI*, 2439–2445.

Gelman, A.; Hwang, J.; and Vehtari, A. 2014. Understanding predictive information criteria for bayesian models. *Statistics and Computing* 24(6):997–1016.

Heckerman, D.; Geiger, D.; and Chickering, D. M. 1995. Learning bayesian networks: The combination of knowledge and statistical data. *Machine learning* 20(3):197–243.

Jensen, F. V. 1996. *An introduction to Bayesian networks*, volume 210. UCL press London.

Koller, D., and Friedman, N. 2009. *Probabilistic graphical models: principles and techniques*. MIT press.

Malone, B. 2015. Finding optimal bayesian network structures with constraints learned from data.

Murphy, K. P. 2002. *Dynamic bayesian networks: representation, inference and learning*. Ph.D. Dissertation, University of California, Berkeley.

O'Gorman, B.; Babbush, R.; Perdomo-Ortiz, A.; Aspuru-Guzik, A.; and Smelyanskiy, V. 2015. Bayesian network structure learning using quantum annealing. *The European Physical Journal Special Topics* 224(1):163–188.

Pearl, J. 1988. Probabilistic reasoning in intelligent systems. palo alto. *Morgan Kaufmann. PEAT, J., VAN DEN BERG, R., & GREEN, W.(1994). Changing prevalence of asthma in australian children. British Medical Journal* 308:1591–1596.

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

Rubin, D. B. 1976. Inference and missing data. *Biometrika* 581–592.

Schwarz, G. 1978. Estimating the dimension of a model. *The annals of statistics* 6(2):461–464.

Tang, Y., and Xu, Z. 2014. A score based approach towards improving bayesian network structure learning. In *Advanced Cloud and Big Data (CBD), 2014 Second International Conference on*, 39–44. IEEE.

Watanabe, S. 2013. A widely applicable bayesian information criterion. *Journal of Machine Learning Research* 14(Mar):867–897.