Accepted for publication in Knowledge and Information Systems

Discovering Excitatory Relationships using Dynamic Bayesian Networks

Debprakash Patnaik[1], Srivatsan Laxman[2], and Naren Ramakrishnan[1]
[1]Department of Computer Science, Virginia Tech, VA 24061, USA;
[2]Microsoft Research Bangalore 560080, India

Abstract. Mining temporal network models from discrete event streams is an important problem with applications in computational neuroscience, physical plant diagnostics, and human-computer interaction modeling. In this paper we introduce the notion of excitatory networks which are essentially temporal models where all connections are stimulative, rather than inhibitive. The emphasis on excitatory connections facilitates learning of network models by creating bridges to frequent episode mining. Specifically, we show that frequent episodes help identify nodes with high mutual information relationships and that such relationships can be summarized into a dynamic Bayesian network (DBN). This leads to an algorithm that is significantly faster than state-of-the-art methods for inferring DBNs, while simultaneously providing theoretical guarantees on network optimality. We demonstrate the advantages of our approach through an application in neuroscience, where we show how strong excitatory networks can be efficiently inferred from both mathematical models of spiking neurons and several real neuroscience datasets.

Keywords: Frequent episodes; dynamic Bayesian networks; computational neuroscience; spike train analysis; temporal data mining.

1. Introduction

Discrete event streams are prevalent in many applications, such as neuronal spike train analysis, physical plants, and human-computer interaction modeling. In all these domains, we are given occurrences of events of interest over a time course and the goal is to identify trends and behaviors that serve discriminatory or descriptive purposes. A Multi-Electrode Array (MEA) records spiking action

Received Apr 04, 2010
Revised Jul 07, 2010
Accepted Aug 22, 2010

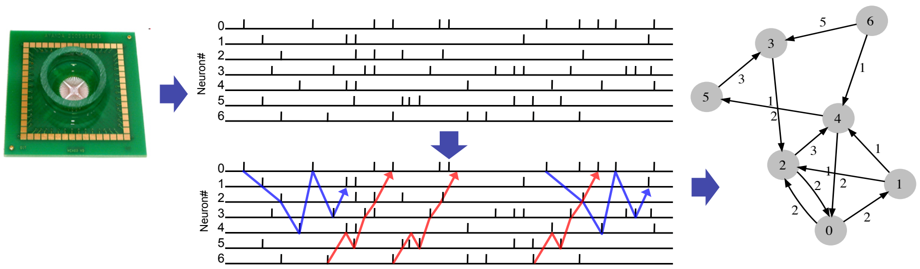


Fig. 1. Event streams in neuroscience: A multi-electrode array (MEA; left) produces a spiking event stream of action potentials (middle top). Mining repeating firings cascades (middle bottom) in the event stream helps uncover excitatory circuits (right) in the data.

potentials from an ensemble of neurons which after various pre-processing steps, yields a spike train dataset providing real-time and dynamic perspectives into brain function (see Fig. 1). Identifying sequences (e.g., cascades) of firing neurons, determining their characteristic delays, and reconstructing the functional connectivity of neuronal circuits are key problems of interest. This provides critical insights into the cellular activity recorded in the neuronal tissue. Similar motivations arise in other domains as well. In physical plants the discrete event stream denotes diagnostic and prognostic codes from stations in an assembly line and the goal is to uncover temporal connections between codes emitted from different stations. In human-computer interaction modeling, the event stream denotes actions taken by users over a period of time and the goal is to capture aspects such as user intent and interaction strategy by understanding causative chains of connections between actions.

Beyond uncovering structural patterns from discrete events, we seek to go further, and actually uncover a generative temporal process model for the data. In particular, our aim is to infer dynamic Bayesian networks (DBNs) which encode conditional independencies as well as temporal influences and which are also interpretable patterns in their own right. We focus exclusively on excitatory networks where the connections are stimulative rather than inhibitory in nature (e.g., 'event A stimulates the occurrence of event B 5ms later which goes on to stimulate event C 3ms beyond.') This constitutes a large class of networks with relevance in multiple domains, including neuroscience.

Probabilistic modeling of temporal data is a thriving area of research. The development of dynamic Bayesian networks as a subsuming formulation to HMMs, Kalman filters, and other such dynamical models has promoted a succession of research aimed at capturing probabilistic dynamical behavior in complex systems. DBNs bring to modeling temporal data the key advantage that traditional Bayesian networks brought to modeling static data, i.e., the ability to use graph theory to capture probabilistic notions of independence and conditional independence. They are now widely used in bioinformatics, neuroscience, and linguistics applications.

A contrasting line of research in modeling and mining temporal data is the counting based literature in the KDD community (Mannila, Toivonen and Verkamo, 1997; Laxman, Sastry and Unnikrishnan, 2005; Papapetrou et al., 2009). Similar to frequent itemsets, the notion of frequent episodes is the object of interest here. Identifying frequency measures that support efficient counting procedures (just as support does for itemsets) has been shown to be crucial.

It is natural to question whether these two threads, with divergent origins, can be related to one another. Many researchers have explored this question. Pavlov, et. al used frequent itemsets to place constraints on the joint distribution of item random variables and thus aid in inference and query approximation (Pavlov, Mannila and Smyth, 2003). In (Wang and Parthasarathy, 2006) probabilistic models are viewed as summarized representations of databases and demonstrate how to construct MRF (Markov random field) models from frequent itemsets. Closer to the topic of this paper, (Laxman et al., 2005) linked frequent episodes to learning Hidden Markov Models for the underlying data. Similar in scope to the above works, we present a unification of the goals of dynamic Bayesian network inference with that of frequent episode mining. Our motivations are not merely to establish theoretical results but also to inform the computational complexity of algorithms and spur faster algorithms targeted for specific domains.

Our main contributions are three-fold:

1. **New model class of excitatory networks**: Learning Bayesian networks (dynamic or not) is a hard problem and to obtain theoretical guarantees we typically have to place restrictions on network structure, e.g., assume the BN has a tree structure as done in the Chow-Liu algorithm. Here, we place restrictions on the form of the conditional probability tables (CPT) instead of network structure; this leads to a tractable class of DBNs for inference, referred to here as *Excitatory Dynamic Networks (EDNs).*
2. **New methods for learning DBNs**: We demonstrate that optimal EDNs can be learnt by creating bridges to frequent episode mining literature. In particular, the focus on EDNs allows us to relate frequent episodes to parent sets for nodes with high mutual information. This enables us to predominantly apply fast algorithms for episode mining, while relating them to probabilistic notions suitable for characterizing DBNs.
3. **New applications to spike train analysis**: We demonstrate a successful application of our methodologies to analyzing neuronal spike train data, both from mathematical models of spiking neurons and from real cortical tissue.

The paper is organized as follows. Sec. 2 gives a brief overview of DBNs. Sec. 3 develops the formalism for learning optimal DBNs from event streams. Sec. 4 defines excitatory networks and presents the theoretical basis for efficiently learning such networks. Sec. 5 introduces fixed-delay episodes and relates frequencies of such episodes with marginal probabilities of a DBN. Our learning algorithm is presented in Sec. 6, experimental results in Sec. 7 and conclusions in Sec. 8.

2. Bayesian Networks: Static and Dynamic

Formal mathematical notions are presented in the next section, but here we wish to provide some background context to past research in Bayesian networks (BNs). As is well known, BNs use directed acyclic graphs to encode probabilistic notions of conditional independence, such as that a node is conditionally independent of its non-descendants given its parents (for more details, see (Jordan, 1998)). The joint pdf encoded by a BN can be succinctly written in product form, one term for each node in the network, where the term denotes the conditional probability of the node given its parents.

The earliest known work for learning BNs is the Chow-Liu algorithm (Chow and Liu, 1968). It showed that, if we restricted the structure of the BN to a tree, then the optimal BN can be computed using a maximum spanning tree algorithm. In particular, the optimal BN is the tree that maximizes the sum of mutual information quantities between pairs of nodes. The Chow-Liu algorithm is noteworthy because it established a class of networks (trees) for which inference is tractable. More recent work, by Williamson (Williamson, 2000), generalizes the Chow-Liu algorithm to show how (discrete) distributions can be approximated using the same general ingredients as the Chow-Liu approach, namely mutual information quantities between random variables. Meila (Meila, 1999) presents an accelerated algorithm that is targeted toward sparse datasets of high dimensionality.

More generally, however, BN learning must be approximate and we must settle for lack of optimality guarantees. Most methods provide good search heuristics (Wang and Yang, 2009). There are two broad classes of algorithms for learning BNs: score-based and constraint-based. Score-based algorithms aim to search in the space of networks guided by a score function. Friedman's sparse candidate approach (Friedman, Murphy and Russell, 1998) refers to a general class of algorithms for approximate learning of BNs. The approach is iterative: in each iteration, a set of candidate parents are discovered for each node, and from these parents, a network maximizing the score function is discovered. Conditioned on this network, the algorithm then aims to add further nodes that will improve on the score. The approach terminates when no more nodes can be added or when a pre-determined number of iterations are met. One instantiation of this approach uses the BIC (Bayesian Information Criterion) as the score function. The simulated annealing approach from (Eldawlatly, Zhou, Jin and Oweiss, 2010) begins with an empty network and aims to add edges that can improve the score. With some small probability, edges that decrease the score are added as well. An underlying cooling schedule controls the annealing parameters. Constraint-based algorithms such as GSMN (Bromberg, Margaritis and Honavar, 2009) on the other hand first aim to identify conditional independence relationships across all the variables and then aim to fit the discovered relationships into a network, much like fitting a jigsaw puzzle.

As a class of networks, DBNs are a relatively newer development although specific forms of DBNs (e.g., HMMs) have a long and checkered history. Most examples of DBNs can be found in specific state space and dynamic modeling contexts (Wang, Zhang, Shen and Shi, 2008). In contrast to their static counterparts, exact and efficient inference for general classes of DBNs has not been studied well. While many of the above algorithms for BNs can be adapted toward DBNs, our aim is to go further and exploit the specific modeling contexts that DBNs were designed for.

3. Learning Optimal DBNs

Consider a finite alphabet, $\mathcal{E} = \{A_1, \ldots, A_M\}$, of event-types (or symbols). Let $s = \langle (E_1, \tau_1), (E_2, \tau_2), \ldots, (E_n, \tau_n) \rangle$ denote a data stream of n events over $\mathcal{E}$. Each $E_i, i = 1, \ldots, n$, is a symbol from $\mathcal{E}$. Each $\tau_i, i = 1, \ldots, n$, takes values from the set of positive integers. The events in s are ordered according to their times of occurrence, $\tau_{i+1} \geq \tau_i, i = 1, \ldots, (n-1)$. The time of occurrence of the last event in s, is denoted by $\tau_n = T$. We model the data stream, s, as a realization of a discrete-time random process $\mathbf{X}(t), t = 1, \ldots, T$; $\mathbf{X}(t) = [X_1(t) X_2(t) \cdots X_M(t)]'$, where $X_j(t)$ is an indicator variable for the occurrence of event type, $A_j \in \mathcal{E}$, at time t. Thus, for $j = 1, \ldots, M$ and $t = 1, \ldots, T$, we will have $X_j(t) = 1$ if $(A_j, t) \in s$, and $X_j(t) = 0$ otherwise. Each $X_j(t)$ is referred to as the *event-indicator* for event-type, A_j, at time t.

Example 1. The following is an example event sequence of $n = 7$ events over an alphabet, $\mathcal{E} = \{A, B, C, \ldots, Z\}$, of $M = 26$ event-types:

$$\langle (A, 2), (B, 3), (D, 3), (B, 5), (C, 9), (A, 10), (D, 12) \rangle \tag{1}$$

The maximum time tick is given by $T = 12$. Each $\mathbf{X}(t),\ t = 1, \ldots, 12$, is a vector of $M = 26$ indicators. Since there are no events at time $t = 0$ in the example sequence (1), we have $\mathbf{X}(1) = \mathbf{0}$. At time $t = 2$, we will have $\mathbf{X}(2) = [1000 \cdots 0]'$. Similarly, $\mathbf{X}(3) = [01010 \cdots 0]'$, and so on.

A DBN (Murphy, 2002) is a DAG with nodes representing time-indexed random variables and arcs representing conditional dependency relationships. We model the random process $\mathbf{X}(t)$ (or equivalently, the event stream s), as the output of a DBN. Each event-indicator, $X_j(t),\ t = 1, \ldots, T$ and $j = 1, \ldots M$, corresponds to a node in the network, and is assigned a set of parents, which is denoted as $\pi_j(t)$. A parent-child relationship is represented by an arc (from parent to child) in the DAG. In a DBN, nodes are conditionally independent of their non-descendants given their parents. The joint probability distribution of $\mathbf{X}(t)$ under the DBN model, can be factorized as a product of $P[X_j(t) \,|\, \pi_j(t)]$ for various j, t. In this paper we restrict the class of DBNs using the following two constraints:

A1 *[Time-bounded causality]* For user-defined parameter, $W > 0$, the set, $\pi_j(t)$, of parents for the node, $X_j(t)$, is a subset of event-indicators out of the W-length history at time-tick, t, i.e. $\pi_j(t) \subseteq \{X_k(\tau)\ :\ 1 \leq k \leq M,\ (t - W) \leq \tau < t\}$.

A2 *[Translation invariance]* If $\pi_j(t) = \{X_{j_1}(t_1), \ldots, X_{j_\ell}(t_\ell)\}$ is an ℓ-size parent set of $X_j(t)$ for some $t > W$, then for any other $X_j(t'),\ t' > W$, its parent set, $\pi_j(t')$, is simply a time-shifted version of $\pi_j(t)$, and is given by $\pi_j(t') = \{X_{j_1}(t_1 + \delta), \ldots,\ X_{j_\ell}(t_\ell + \delta)\}$, where $\delta = (t' - t)$.

While **A1** limits the range-of-influence of a random variable, $X_k(\tau)$, to variables within W time-ticks of τ, **A2** is a structural constraint that allows parent-child relationships to depend only on relative (rather than absolute) time-stamps of random variables. Further, we also assume that the underlying data generation model is stationary, so that joint-statistics can be estimated using frequency counts of suitably defined temporal patterns in the data.

A3 *[Stationarity]* For every set of ℓ event-indicators, $X_{j_1}(t_1), \ldots, X_{j_\ell}(t_\ell)$, and for every time-shift δ (either positive or negative) we have $P[X_{j_1}(t_1), \ldots, X_{j_\ell}(t_\ell)]$ $= P[X_{j_1}(t_1 + \delta),\ \ldots, X_{j_\ell}(t_\ell + \delta)]$.

The joint probability distribution, $Q[\cdot]$, under the Dynamic Bayesian Network model can be written as:

$$Q[\mathbf{X}(1), \ldots, \mathbf{X}(T)] = P[\mathbf{X}(1), \ldots, \mathbf{X}(W)] \times \prod_{t=W+1}^{T} \prod_{j=1}^{M} P[X_j(t) \mid \pi_j(t))] \tag{2}$$

Learning network structure involves learning the map, $\pi_j(t)$, for each $X_j(t)$, $j = 1, \ldots, M$ and $t > W$. Let $I[X_j(t)\ ;\ \pi_j(t)]$ denote the *mutual information* between $X_j(t)$ and its parents, $\pi_j(t)$. DBN structure learning can be posed as a problem of approximating the data distribution, $P[\cdot]$, by a DBN distribution, $Q[\cdot]$. Let $D_{KL}(P||Q)$ denote the KL divergence between $P[\cdot]$ and $Q[\cdot]$. Using **A1**, **A2** and **A3** we now show that for parent sets with sufficiently high mutual information to $X_j(t)$, $D_{KL}(P||Q)$ will be concomitantly lower.

The Kullback-Leibler divergence between the underlying joint distribution, $P[\cdot]$, of the stochastic process, and the joint distribution, $Q[\cdot]$, under the DBN model is given by

$$D_{KL}(P||Q) = \sum_{\mathcal{A}} \left(P[\mathbf{X}(1), \ldots, \mathbf{X}(T)] \times \log \frac{P[\mathbf{X}(1), \ldots, \mathbf{X}(T)]}{Q[\mathbf{X}(1), \ldots, \mathbf{X}(T)]} \right) \tag{3}$$

where $\mathcal{A}$ represents the set of all possible assignments for the T M-length random vectors, $\{\mathbf{X}(1), \ldots, \mathbf{X}(T)\}$. Denoting the entropy of $P[\mathbf{X}(1), \ldots, \mathbf{X}(T)]$ by $H(P)$, the entropy of the marginal, $P[\mathbf{X}(1), \ldots, \mathbf{X}(W)]$, by $H(P_W)$, and substituting for $Q[\cdot]$ from Eq. (2), we get

$$\begin{aligned} D_{KL}(P||Q) = & -H(P) - H(P_W) \\ & - \sum_{\mathcal{A}} \left(P[\mathbf{X}(1), \ldots, \mathbf{X}(T)] \times \sum_{j=1}^{M} \sum_{t=W+1}^{T} \log P[X_j(t) \mid \pi_j(t)] \right) \end{aligned} \tag{4}$$

We now expand the conditional probabilities in Eq. (4) using Bayes rule, switch the order of summation and marginalize $P[\cdot]$ for each j, t. Denoting, for each j, t, the entropy of the marginal $P[X_j(t)]$ by $H(P_{j,t})$, the expression for KL divergence now becomes:

$$\begin{aligned} D_{KL}(P||Q) = & -H(P) - H(P_W) \\ & - \sum_{j=1}^{M} \sum_{t=W+1}^{T} H(P_{j,t}) - \sum_{j=1}^{M} \sum_{t=W+1}^{T} I[X_j(t)\ ;\ \pi_j(t)] \end{aligned} \tag{5}$$

$I[X_j(t)\ ;\ \pi_j(t)]$ denotes the *mutual information* between $X_j(t)$ and its parents, $\pi_j(t)$, and is given by

$$I[X_j(t)\ ;\ \pi_j(t)] = \sum_{\mathcal{A}_{j,t}} \left(P[X_j(t), \pi_j(t)] \times \log \frac{P[X_j(t), \pi_j(t)]}{P[X_j(t)]\ P[\pi_j(t)]} \right) \tag{6}$$

where $\mathcal{A}_{j,t}$ represents the set of all possible assignments for the random variables, $\{X_j(t), \pi_j(t)\}$. Under the translation invariance constraint, **A2**, and the stationarity assumption, **A3**, we have $I[X_j(t)\ ;\ \pi_j(t)] = I[X_j(t')\ ;\ \pi_j(t')]$ for all $t > W$, $t' > W$. This gives us the following final expression for $D_{KL}(P||Q)$:

$$\begin{aligned} D_{KL}(P||Q) = & -H(P) - H(P_W) \\ & - \sum_{j=1}^{M} \sum_{t=W+1}^{T} H(P_{j,t}) - (T - W) \sum_{j=1}^{M} I[X_j(t)\ ;\ \pi_j(t)] \end{aligned} \tag{7}$$

where t is any time-tick satisfying $(W < t \leq T)$. We note that in Eq. (7), the entropies, $H(P)$, $H(P_W)$ and $H(P_{j,t})$ are independent of the DBN structure (i.e. they do not depend on the $\pi_j(t)$ maps). Since $(T - W) > 0$ and since $I[X_j(t) \; ; \; \pi_j(t)] \geq 0$ always, the KL divergence, $D_{KL}(P||Q)$, is minimized when the sum of M mutual information terms in Eq. (7) is maximized. Further, from **A1** we know that all parent nodes of $X_j(t)$ have time-stamps strictly less than t, and hence, no choice of $\pi_j(t), \; j = 1, \ldots, M$ can result in a cycle in the network (in which case, the structure will not be a DAG, and in-turn, it will not represent a valid DBN). This ensures that, under the restriction of **A1**, the *optimal DBN structure* (namely, one that corresponds to a $Q[\cdot]$ that minimizes KL divergence with respect to the true joint probability distribution, $P[\cdot]$, for the data) can be obtained by *independently* picking the highest mutual information parents, $\pi_j(t)$, for each $X_j(t)$ for $j = 1, \ldots, M$ (and, because of **A2** and **A3**, we need to carry-out this parents' selection step only for the M nodes in any one time slice, t, that satisfies $(W < t \leq T)$). Moreover, from Eq. (7) it is clear that if we had a lower-bound ϑ on the mutual information terms, $I[X_j(t) \; ; \; \pi_j(t)]$, it would imply an upper-bound Δ_ϑ on the KL divergence $D_{KL}(P||Q)$ between $P[\cdot]$ and $Q[\cdot]$. In other words, a good DBN-based approximation of the underlying stochastics (i.e. one with small KL divergence) can be achieved by picking, for each node, a parent-set whose corresponding mutual information exceeds a user-defined threshold.

Remark 3.1. Consider a sequence $\mathbf{X}(t) \; : \; t = 1, \ldots, T$, of time-evolving indicators for an event stream s over an alphabet of size M. Under assumptions **A1** and **A2**, the optimal DBN (one that minimizes KL divergence with the true data distribution) can be obtained by independently picking for each $X_j(t)$, $1 \leq j \leq M$ (and for some fixed $t > W$) a parent set $\pi_j(t)$ which maximizes mutual information $I[X_j(t); \pi_j(t)]$. Moreover, picking $\pi_j(t)$ such that $I[X_j(t); \pi_j(t)] > \vartheta$ implies a corresponding upper-bound Δ_ϑ on the KL divergence between the DBN and the true data distribution.

However, picking such sets with high mutual information (while yields a good approximation) falls short of unearthing useful dependencies among the random variables. This is because mutual information is non-decreasing as more random variables are added to a parent-set (leading to a fully-connected network always being optimal). For a parent-set to be interesting, it should not only exhibit sufficient correlation (or mutual information) with the corresponding child-node, but should also successfully encode the conditional independencies among random variables in the system. This can be done by checking if, *conditioned* on a candidate parent-set, the mutual information between the corresponding child-node and all its non-descendants is always close to zero. (We provide more details later in Sec. 6.3).

4. Excitatory Dynamic Networks

The structure-learning approach described in Sec. 3 is applicable to any general DBN that satisfies **A1** and **A2**. We now introduce a specialized class of networks, which we call as *Excitatory Dynamic Networks* (or EDNs) where only certain kinds of conditional dependencies among nodes are permitted. Each event-type is assumed to occur with probability *less than 0.5* in the data. This corresponds to a sparse-data assumption. A collection, Π, of random variables is said to have an *excitatory* influence on an event-type, $A \in \mathcal{E}$, if occurrence of events corresponding to the variables in Π, increases the probability of A to *greater*

Table 1. CPT structure for EDNs. $\Pi = \{X_B, X_C, X_D\}$ denotes the set of parents for node X_A. The table lists excitatory constraints on the probability of observing $[X_A = 1]$ conditioned on the different value-assignments for X_B, X_C and X_D.

	Π			$P[X_A = 1 \mid \mathbf{a}_j]$
	X_B	X_C	X_D	
$\mathbf{a}_0$	0	0	0	$\epsilon < \frac{1}{2}$
$\mathbf{a}_1$	0	0	1	$\epsilon_1 \geq \epsilon$
$\mathbf{a}_2$	0	1	0	$\epsilon_2 \geq \epsilon$
$\mathbf{a}_3$	0	1	1	$\epsilon_3 \geq \epsilon, \epsilon_1, \epsilon_2$
$\mathbf{a}_4$	1	0	0	$\epsilon_4 \geq \epsilon$
$\mathbf{a}_5$	1	0	1	$\epsilon_5 \geq \epsilon, \epsilon_1, \epsilon_4$
$\mathbf{a}_6$	1	1	0	$\epsilon_6 \geq \epsilon, \epsilon_2, \epsilon_4$
$\mathbf{a}_7(\mathbf{a}^*)$	1	1	1	$\phi > \epsilon, \frac{1}{2}, \epsilon_j \forall j$

than 0.5. We define an Excitatory Dynamic Network as a DBN in which *all* parent nodes can only exert excitatory influences on corresponding child nodes. For example, EDNs will allow relationships like "if B, C and D occur (say) 2 time-ticks apart, the probability of A increases." However, EDNs cannot encode excitations-in-absence like "when A does not occur, the probability of B occurring 3 time-ticks later increases" or inhibitions like "when A occurs, the probability of B occurring 3 time-ticks later decreases." Excitatory networks are natural in neuroscience, where one is interested in unearthing conditional dependency relationships among neuron spiking patterns. Several regions in the brain are known to exhibit predominantly excitatory relationships (Rieke, Warland, Steveninck and Bialek, 1999) and our model is targeted towards unearthing these.

The excitatory assumption manifests as constraints on the conditional probability tables associated with the DBN. Consider a node[1] X_A (an indicator variable for event-type $A \in \mathcal{E}$) and let Π denote a parent-set for X_A. In excitatory networks, the probability of A conditioned on the occurrence of all event-types in Π, should be *at least as high as* the corresponding conditional probability when only some (though not all) of the event-types of Π occur. Further, the probability of A is less than 0.5 when none of the event-types of Π occur and greater than 0.5 when all the event-types of Π occur. For example, let $\Pi = \{X_B, X_C, X_D\}$. The conditional probability table (or CPT) for A given Π is shown in Table 1. The different conditioning contexts come about by the occurrence or otherwise of each of the events in Π. These are denoted by $\mathbf{a}_j,\ j = 0, \ldots, 7$. So while $\mathbf{a}_0$ represents the all-zero assignment (i.e. none of the events B, C or D occur), $\mathbf{a}_7$ (or $\mathbf{a}^*$) denotes the all-ones assignment (i.e. all the events B, C and D occur). The last column of the table lists the corresponding conditional probabilities along with the associated excitatory constraints. The baseline propensity of A is denoted by ϵ and the only constraint on it is that it must be less than $\frac{1}{2}$. Conditioned on the occurrence of any event of Π, the propensity of A can only increase, and hence, $\epsilon_j \geq \epsilon\ \forall j$ and $\phi \geq \epsilon$. Similarly, when both C and D occur, the probability must be at least as high as that when either C or D occur alone (i.e. we must have $\epsilon_3 \geq \epsilon_1$ and $\epsilon_3 \geq \epsilon_2$). Finally, for the all-ones case, denoted by $\Pi = \mathbf{a}^*$, the conditional probability must be greater than $\frac{1}{2}$ and must also satisfy $\phi \geq \epsilon_j\ \forall j$.

[1] To facilitate simple exposition, time-stamps of random-variables are dropped from the notation in this discussion.

In the context of DBN learning, an excitatory assumption on the data implies that event-types will occur *frequently* after their respective parents (with suitable delays). This will allow us to estimate EDN structures using frequent pattern discovery algorithms (which have been a mainstay in data mining for many years). We now present a simple necessary condition on the probability (or frequency) of parent-sets in an excitatory network.

Theorem 4.1. Let X_A denote a node corresponding to the event-type $A \in \mathcal{E}$ in an Excitatory Dynamic Network (EDN). Let Π denote the parent-set for X_A in the EDN. Let ϵ^* be an upper-bound on the conditional probabilities $P[X_A = 1|\Pi = \mathbf{a}]$ for all $\mathbf{a} \neq \mathbf{a}^*$ (i.e. for all but the all-ones assignment in Π). If the mutual information $I[X_A\,;\Pi]$ exceeds $\vartheta(>0)$, then the joint probability of an occurrence of A along with all events of Π satisfies $P[X_A = 1, \Pi = \mathbf{a}^*] \geq P_{min}\Phi_{min}$, where

$$P_{min} = \frac{P[X_A = 1] - \epsilon^*}{1 - \epsilon^*} \tag{8}$$

$$\Phi_{min} = h^{-1}\left[\min\left(1, \frac{h(P[X_A = 1]) - \vartheta}{P_{min}}\right)\right] \tag{9}$$

and where $h(\cdot)$ denotes the binary entropy function $h(q) = -q\log q - (1-q)\log(1-q)$, $0 < q < 1$ and $h^{-1}[\cdot]$ denotes its pre-image greater than $\frac{1}{2}$.

Proof: Under the excitatory model we have $P[X_A = 1|\Pi = \mathbf{a}^*] > P[X_A = 1|\Pi = \mathbf{a}]\ \forall \mathbf{a} \neq \mathbf{a}^*$. First we apply ϵ^* to terms in the expression for $P[X_A = 1]$:

$$\begin{aligned} P[X_A = 1] &= P[\Pi = \mathbf{a}^*]P[X_A = 1 \mid \Pi = \mathbf{a}^*] \\ &\quad + \sum_{\mathbf{a}\neq\mathbf{a}^*} P[\Pi = \mathbf{a}]P[X_A = 1 \mid \Pi = \mathbf{a}] \\ &\leq P[\Pi = \mathbf{a}^*] + (1 - P[\Pi = \mathbf{a}^*])\epsilon^* \end{aligned}$$

This gives us $P[\Pi = \mathbf{a}^*] \geq P_{min}$ (see Fig. 2). Next, since we are given that mutual information $I[X_A\ ;\ \Pi]$ exceeds ϑ, the corresponding conditional entropy must satisfy:

$$\begin{aligned} H[X_A \mid \Pi] &= P[\Pi = \mathbf{a}^*]h(P[X_A = 1 \mid \Pi = \mathbf{a}^*]) \\ &\quad + \sum_{\mathbf{a}\neq\mathbf{a}^*} P[\Pi = \mathbf{a}]h(P[X_A = 1 \mid \Pi = \mathbf{a}]) \\ &< H[X_A] - \vartheta = h(P[X_A = 1]) - \vartheta \end{aligned}$$

Every term in the expression for $H[X_A\,|\,\Pi]$ is non-negative, and hence, each term (including the first one) must be less than $(h(P[X_A = 1]) - \vartheta)$. Using $(P[\Pi = \mathbf{a}^*] \geq P_{min})$ in the above inequality and observing that $P[X_A = 1\,|\,\Pi = \mathbf{a}^*]$ must be greater than 0.5 for an excitatory network, we now get $(P[X_A = 1\,|\,\Pi = \mathbf{a}^*] > \Phi_{min})$. This completes the proof. □

4.1. Utility of EDNs

In Sec. 3, we derived conditions for optimal DBN approximations based on mutual information criteria. We showed that if each node was associated with a parent set that had high mutual information with the node, then the resulting DBN would have small KL divergence with respect to the true data distribution.

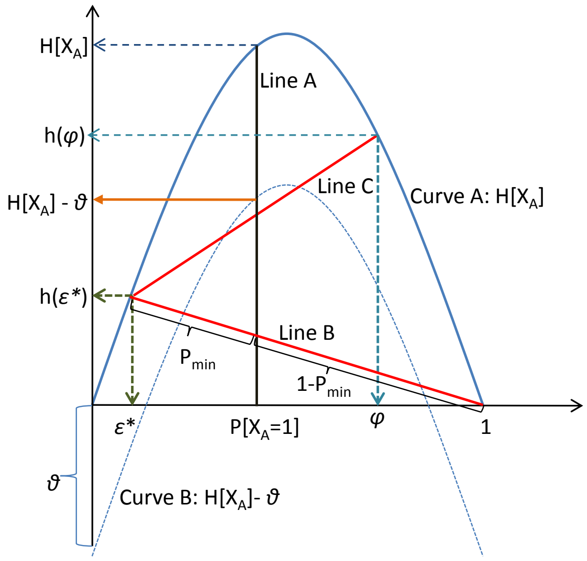


Fig. 2. An illustration of the results obtained in Theorem 4.1. x-axis of the plot is the range of $P[X_A = 1]$ and y-axis is the corresponding range of entropy $H[X_A]$. For the required mutual information criteria, the conditional entropy must lie below $H[X_A] - \vartheta$ and on line A. At the boundary condition $[\phi = 1]$, Line A splits Line B in the ratio $P_{min} : (1 - P_{min})$. This gives the expression for P_{min}.

The difficulty, however, is that searching for these sets with high mutual information is a computational expensive proposition. Excitatory Dynamic Networks, introduced in Sec. 4, alleviates this problem by focussing an a subclass of DBNs that only encode excitatory relationships among nodes.

The main result about EDNs that we can exploit to develop efficient algorithms is *Theorem 4.1*. The theorem essentially translates a lower-bound ϑ on the mutual information $I[X_A\ ;\ \Pi]$ between a node X_A and its parents Π, to a lower-bound $P_{min}\Phi_{min}$ on the joint probability $P[X_A = 1\ ,\ \Pi = \mathbf{a}^*]$ of the occurrence of events corresponding to the node and its parents. Given a node X_A, identifying all sets, Π, for which the joint probabilities of the special form $P[X_A = 1\ ,\ \Pi = \mathbf{a}^*]$ exceed a threshold falls in the realm of frequent pattern discovery in data mining. It is well-known in data mining literature that, if the data is sparse, frequent pattern discovery algorithms constitute a very efficient and effective class of methods for unearthing strong correlations in the data. In this paper, we exploit these methods for efficiently learning optimal DBNs from event streams. Specifically, we define correlations of the form $[X_A = 1\ ,\ \Pi = \mathbf{a}^*]$ as a new class of patterns called *fixed-delay episodes* and develop fast algorithms for discovering all fixed-delay episodes in the data whose frequencies exceed some threshold (See Sec. 5 for details). *Theorem 4.1* shows that a minimum mutual information constraint for a node and its parents translates to a (minimum) frequency threshold for discovering frequent fixed-delay episodes.

The lower-bound on joint probabilities $P[X_A\ ,\ \Pi = \mathbf{a}^*])$ as per *Theorem 4.1* is $P_{min}\Phi_{min}$, where P_{min} and Φ_{min} depend on two user-defined quantities (cf. Eqs. 8-

9): (i) the minimum mutual information threshold ϑ, and (ii) the upper-bound ϵ^* on the conditional probabilities in all-but-the-last-row of the CPT for X_A. It is easy to see that as the ϑ increases Φ_{min} increases and consequently, the threshold $P_{min}\Phi_{min}$ also increases. Similarly, as ϵ^* increases P_{min} decreases, as does Φ_{min}. Thus, as ϵ^* increases the threshold $P_{min}\Phi_{min}$ decreases. These relationships facilitate the use of ϵ^* and ϑ as exploratory 'knobs' for EDN learning. In general, using a lower threshold (i.e. a lower $P_{min}\Phi_{min}$) will allow us to detect even the weaker correlations, which results in estimation of a *more complete* network. However, lower thresholds translate to higher run-times, and as is well-known in pattern discovery literature, at some sufficiently low threshold the computation can become infeasible. The theoretical results allow us to systematically explore this trade-off between inferring weaker dependencies and requiring higher run-times. In Sec. 7, we describe experiments that demonstrate the effect of ϵ^* and ϑ on the performance of our algorithms. Finally, note that $P_{min}\Phi_{min}$ also depends on the marginal probability of A in the data, namely $P[X_A = 1]$. Unlike ϵ^* and ϑ (which are user-defined parameters) $P[X_A = 1]$ is estimated from the data. The relationship of $P_{min}\Phi_{min}$ with $P[X_A = 1]$, however, is a bit more complicated – the threshold can increase or decrease with $P[X_A = 1]$ depending on the values chosen for ϵ^* and ϑ.

5. Fixed-delay episodes

In Secs. 3-4 we developed the theoretical framework for estimating optimal Excitatory Dynamic Networks from time-stamped event streams. We pointed out that we can reduce the search space for the parent sets of a node by constructing candidate parent sets out of frequent fixed-delay episodes in the data (cf. Sec. 4.1). In this section, we formally define fixed-delay episodes and show how to use the frequencies of fixed-delay episodes to compute corresponding joint probabilities and mutual information quantities.

In the framework of frequent episode discovery (Mannila et al., 1997) the data is a single long stream of events over a finite alphabet (cf. Sec. 3 and *Example 1*). An ℓ-node (serial) episode, α, is defined as a tuple, $(V_\alpha, <_\alpha, g_\alpha)$, where $V_\alpha = \{v_1, \ldots, v_\ell\}$ denotes a collection of nodes, $<_\alpha$ denotes a *total order*[2] such that $v_i <_\alpha v_{i+1},\ i = 1, \ldots, (\ell - 1)$. If $g_\alpha(v_j) = A_{i_j}, A_{i_j} \in \mathcal{E}, j = 1, \ldots, \ell$, we use the graphical notation $(A_{i_1} \to \cdots \to A_{i_\ell})$ to represent α. An occurrence of α in event stream, $s = \langle (E_1, \tau_1), (E_2, \tau_2), \ldots, (E_n, \tau_n) \rangle$, is a map $h : V_\alpha \to \{1, \ldots, n\}$ such that (i) $E_{h(v_j)} = g(v_j)\ \forall v_j \in V_\alpha$, and (ii) for all $v_i <_\alpha v_j$ in V_α, the times of occurrence of the i^{th} and j^{th} events in the occurrence satisfy $\tau_{h(v_i)} \leq \tau_{h(v_j)}$ in s.

Example 2. Consider a 3-node episode $\alpha = (V_\alpha, <_\alpha, g_\alpha)$, such that, $V_\alpha = \{v_1, v_2, v_3\}$, $v_1 <_\alpha v_2$, $v_2 <_\alpha v_3$ and $v_1 <_\alpha v_3$, and $g_\alpha(v_1) = A$, $g_\alpha(v_2) = B$ and $g_\alpha(v_3) = C$. The graphical representation for this episode is $\alpha = (A \to B \to C)$, indicating that in every occurrence of α, an event of type A must appear before an event of type B, and the B must appear before an event of type C. For example, in sequence (1), the subsequence $\langle (A, 1), (B, 3), (C, 9) \rangle$ constitutes an

[2] In general, $<_\alpha$ can be any partial order over V_α. We focus on only total orders here and show how multiple total orders can be used to model DBNs of arbitrary -arity. In (Mannila et al., 1997), such total orders are referred to as *serial* episodes.

occurrence of $(A \rightarrow B \rightarrow C)$. For this occurrence, the corresponding h-map is given by, $h(v_1) = 1$, $h(v_2) = 2$ and $h(v_3) = 5$.

There are many ways to incorporate explicit time constraints in episode occurrences like the windows-width constraint of (Mannila et al., 1997). Episodes with inter-event *gap* constraints were introduced in (Patnaik, Sastry and Unnikrishnan, 2007). For example, the framework of (Patnaik et al., 2007) can express the temporal pattern "B must follow A *within* 5 time-ticks and C must follow B *within* 10 time-ticks." Such a pattern is represented using the graphical notation, $(A \xrightarrow{[0\text{–}5]} B \xrightarrow{[0\text{–}10]} C)$. In this paper, we use a simple sub-case of the inter-event gap constraints, in the form of *fixed* inter-event time-delays. For example, $(A \overset{5}{\rightarrow} B \overset{10}{\rightarrow} C)$ represents a fixed-delay episode, every occurrence of which must comprise an A, followed by a B *exactly* 5 time-ticks later, which in-turn is followed by a C *exactly* 10 time-ticks later.

Definition 5.1. An ℓ-node *fixed-delay episode* is defined as a pair, $(\alpha, \mathcal{D})$, where $\alpha = (V_\alpha, <_\alpha, g_\alpha)$ is the usual (serial) episode of (Mannila et al., 1997), and $\mathcal{D} = (\delta_1, \ldots, \delta_{\ell-1})$ is a sequence of $(\ell-1)$ non-negative delays. Every occurrence, h, of the fixed-delay episode in an event sequence s must satisfy the inter-event constraints, $\delta_j = (\tau_{h(v_{j+1})} - \tau_{h(v_j)})$, $j = 1, \ldots, (\ell-1)$. $(A_{i_1} \xrightarrow{\delta_1} \cdots \xrightarrow{\delta_{\ell-1}} A_{i_\ell})$ is our graphical notation for the inter-event episode, $(\alpha, \mathcal{D})$, where $A_{i_j} = g_\alpha(v_j)$, $j = 1, \ldots, \ell$.

Definition 5.2. Two occurrences, h_1 and h_2, of a fixed-delay episode, $(\alpha, \mathcal{D})$, are said to be *distinct*, if they do not share any events in the data stream, s. Given a user-defined, $W > 0$, *frequency* of $(\alpha, \mathcal{D})$ in s, denoted $f_s(\alpha, \mathcal{D}, W)$, is defined as the total number of distinct occurrences of $(\alpha, \mathcal{D})$ in s that terminate strictly after W.

In general, counting distinct occurrences of episodes suffers from computational inefficiencies (Laxman, 2006). Each occurrence of an episode $(A \rightarrow B \rightarrow C)$ is a substring that looks like $A * B * C$, where $*$ denotes a *variable-length* don't-care, and hence, counting all distinct occurrences in the data stream can require memory of the same order as the data sequence which typically runs very long. However, in case of fixed-delay episodes, it is easy to track distinct occurrences efficiently. For example, when counting frequency of $(A \xrightarrow{3} B \xrightarrow{5} C)$, if we encounter an A at time t, to recognize an occurrence involving this A we only need to check for a B at time $(t + 3)$ and for a C at time $(t + 8)$. In addition to being attractive from an efficiency point-of-view, we show next in Sec. 5.1 that the distinct occurrences-based frequency count for fixed-delay episodes will allow us to interpret relative frequencies as appropriate DBN marginals. (Note that the W in *Definition 5.2* is same as the length of the history window used in the constraint **A1**. Skipping occurrences terminating in the first W time-ticks makes it easy to normalize the frequency count into a probability measure).

5.1. Marginals from episode frequencies

In this section, we describe how to compute mutual information from the frequency counts of fixed-delay episodes. For this, every subset of event-indicators in the network is associated with a fixed-delay episode.

Definition 5.3. Let $\{X_j(t) : j = 1, \ldots, M;\ t = 1, \ldots, T\}$ denote the collection of event-indicators used to model event stream, $s = \langle (E_1, \tau_1), \ldots (E_n, \tau_n) \rangle$, over alphabet, $\mathcal{E} = \{A_1, \ldots, A_M\}$. Consider an ℓ-size subset, $\mathcal{X} = \{X_{i_1}(t_1), \ldots, X_{i_\ell}(t_\ell)\}$, of these indicators, and without loss of generality, assume $t_1 \leq \cdots \leq t_\ell$. Define the $(\ell - 1)$ inter-event delays in $\mathcal{X}$ as follows: $\delta_j = (t_{j+1} - t_j)$, $j = 1, \ldots, (\ell - 1)$. The fixed-delay episode, $(\alpha(\mathcal{X}), \mathcal{D}(\mathcal{X}))$, that is associated with the subset, $\mathcal{X}$, of event-indicators is defined by $\alpha(\mathcal{X}) = (A_{i_1} \to \cdots \to A_{i_\ell})$, and $\mathcal{D}(\mathcal{X}) = \{\delta_1, \ldots, \delta_{\ell-1}\}$. In graphical notation, the fixed-delay episode associated with $\mathcal{X}$ can be represented as follows:

$$(\alpha(\mathcal{X}), \mathcal{D}(\mathcal{X})) = (A_{i_1} \overset{\delta_1}{\to} \cdots \overset{\delta_{\ell-1}}{\to} A_{i_\ell}) \tag{10}$$

For computing mutual information we need the marginals of various subsets of event-indicators in the network. Given a subset like $\mathcal{X} = \{X_{i_1}(t_1), \ldots, X_{i_\ell}(t_\ell)\}$, we need estimates for probabilities of the form, $P[X_{i_1}(t_1) = a_1, \ldots, X_{i_\ell}(t_\ell) = a_\ell]$, where $a_j \in \{0, 1\}$, $j = 1, \ldots, \ell$. The fixed-delay episode, $(\alpha(\mathcal{X}), \mathcal{D}(\mathcal{X}))$, that is associated with $\mathcal{X}$ is given by *Definition 5.3* and its frequency in the data stream, s, is denoted by $f_s(\alpha(\mathcal{X}), \mathcal{D}(\mathcal{X}), W)$ (as per *Definition 5.2*) where W denotes the length of history window as per **A1**. Since an occurrence of the fixed-delay episode, $(\alpha(\mathcal{X}), \mathcal{D}(\mathcal{X}))$, can terminate in each of the $(T - W)$ time-ticks in s, the probability of an all-ones assignment for the random variables in $\mathcal{X}$ is given by:

$$P[X_{i_1}(t_1) = 1, \ldots, X_{i_\ell}(t_\ell) = 1] = \frac{f_s(\alpha(\mathcal{X}), \mathcal{D}(\mathcal{X}), W)}{T - W} \tag{11}$$

For all other assignments (i.e. for assignments that are not all-ones) we use inclusion-exclusion to obtain corresponding probabilities. Inclusion-exclusion has been used before in data mining, e.g., in (Seppanen, 2006), to obtain exact or approximate frequency counts for arbitrary boolean queries using only counts of *frequent* itemsets in the data. In our case, counting distinct occurrences of fixed-delay episodes facilitates use of the inclusion-exclusion formula for obtaining the probabilities needed for computing mutual information of different candidate parent-sets. Consider the set, $\mathcal{X} = \{X_{i_1}(t_1), \ldots, X_{i_\ell}(t_\ell)\}$, of ℓ event-indicators, and let $\mathcal{A} = (a_1, \ldots, a_\ell)$, $a_j \in \{0, 1\}$, $j = 1, \ldots, \ell$, be an assignment for the event-indicators in $\mathcal{X}$. Let $\mathcal{U} \subset \mathcal{X}$ denote the subset of indicators out of $\mathcal{X}$ for which corresponding assignments (in $\mathcal{A}$) are 1's, i. e. $\mathcal{U} = \{X_{i_j}(t_j) \in \mathcal{X} : j \text{ s.t. } a_j = 1 \text{ in } \mathcal{A}, 1 \leq j \leq \ell\}$. Inclusion-exclusion is used to compute the probabilities as follows:

$$\begin{aligned} P[X_{i_1}(t_1) &= a_1, \ldots, X_{i_\ell}(t_\ell) = a_\ell] \\ &= \sum_{\substack{\mathcal{Y} \text{ s.t.} \\ \mathcal{U} \subseteq \mathcal{Y} \subseteq \mathcal{X}}} (-1)^{|\mathcal{Y} \setminus \mathcal{U}|} \left(\frac{f_s(\mathcal{Y})}{T - W} \right) \end{aligned} \tag{12}$$

where $f_s(\mathcal{Y})$ is short-hand for $f_s(\alpha(\mathcal{Y}), \mathcal{D}(\mathcal{Y}), W)$, the frequency (cf. *Definition 5.2*) of the fixed-delay episode, $(\alpha(\mathcal{Y}), \mathcal{D}(\mathcal{Y}))$.

Finally, a note on mutual information computation. Consider a node $X_{i_\ell}(t_\ell)$ and its candidate parent set $\Pi = \{X_{i_1}(t_1), \ldots, X_{i_{\ell-1}}(t_{\ell-1})\}$. Construct the set $\mathcal{X} = \{X_{i_\ell}(t_\ell)\} \bigcup \Pi$. After computing all the necessary joint probabilities involving $X_{i_\ell}(t_\ell)$ and Π based on Eqs. (11)-(12) we can compute the mutual informa-

Procedure 1 Overall Procedure

```
Input: Alphabet $\mathcal{E}$, event stream $s = \langle(E_1, \tau_1), \ldots, (E_n, \tau_n = T)\rangle$, length $W$ of
    history window, conditional probability upper-bound $\epsilon^*$, mutual information
    threshold, $\vartheta$
Output: DBN structure (parent-set for each node in the network)
 1: for all $A \in \mathcal{E}$ do
 2:     $X_A$ := event-indicator of $A$ at any time $t > W$
 3:     Set $f_{min} = (T - W)P_{min}\Phi_{min}$, using Eqs. (8)-(9)
 4:     Obtain set, $\mathcal{C}$, of fixed-delay episodes ending in $A$, with frequencies greater
        than $f_{min}$ (cf. Sec. 6.2, Procedure 2)
 5:     for all fixed-delay episodes $(\alpha, \mathcal{D}) \in \mathcal{C}$ do
 6:         $\mathcal{X}_{(\alpha,\mathcal{D})}$ := event-indicators corresponding to $(\alpha, \mathcal{D})$
 7:         Compute mutual information $I[X_A \,;\, \mathcal{X}_{(\alpha,\mathcal{D})}]$
 8:         Remove $(\alpha, \mathcal{D})$ from $\mathcal{C}$ if $I[X_A \,;\, \mathcal{X}_{(\alpha,\mathcal{D})}] < \vartheta$
 9:     Prune $\mathcal{C}$ using conditional mutual information criteria to distinguish direct
        from indirect influences (cf. Sec. 6.3)
10:     Return (as parent-set for $X_A$) event-indicators corresponding to episodes
        in $\mathcal{C}$
```

tion $I[X_{i_\ell} \,;\, \Pi]$ using the following expression:

$$I[X_{i_\ell}(t_\ell) \,;\, \Pi] = \sum P[X_{i_\ell}(t_\ell) = a_\ell, X_{i_1}(t_1) = a_1, \ldots, X_{i_{\ell-1}}(t_{\ell-1}) = a_{\ell-1}] \times \log_2 \left[\frac{P[X_{i_\ell} = a_\ell,\ X_{i_1}(t_1) = a_1, \ldots,\ X_{i_{\ell-1}}(t_{\ell-1}) = a_{\ell-1}]}{P[X_{i_\ell}(t_\ell) = a_\ell] P[X_{i_1}(t_1) = a_1, \ldots,\ X_{i_{\ell-1}}(t_{\ell-1}) = a_{\ell-1}]} \right] \tag{13}$$

where $(a_1, \ldots, a_\ell) \in \{0, 1\}^\ell$ and the summation is over all possible values for $(a_1, \ldots, a_\ell)$. This way, all mutual information quantities that are needed for EDN learning can be computed using just the frequencies of appropriate fixed-delay episodes in the data.

6. Algorithms

6.1. Overall approach

In Secs. 3-5, we developed the formalism for learning an optimal DBN structure from event streams by using distinct occurrences-based counts of fixed-delay episodes to compute the DBN marginal probabilities. The top-level algorithm (cf. Sec. 3) for discovering the network is to fix any time $t > W$, to consider each $X_j(t)$, $j = 1, \ldots, M$, in-turn, and to find its set of parents in the network. Due to the translation invariance assumption **A2**, we need to do this *only once for each event-type* in the alphabet.

The algorithm is outlined in *Procedure 1*. For each $A \in \mathcal{E}$, we first compute the minimum frequency for episodes ending in A based on the relationship between mutual information and joint probabilities as per *Theorem 4.1* (line 3, *Procedure 1*). Then we use a pattern-growth approach (see *Procedure 2*) to discover all patterns terminating in A (line 4, *Procedure 1*). Each frequent pattern corresponds to a set of event-indicators (line 6, *Procedure 1*). The mutual information between this set of indicators and the node X_A is computed using

Procedure 2 *pattern_grow*$(\alpha, \mathcal{D}, \mathcal{L}_{(\alpha,\mathcal{D})})$

```
Input: ℓ-node episode $(\alpha, \mathcal{D}) = (A_{j_1} \overset{\delta_1}{\to} \cdots \overset{\delta_{\ell-1}}{\to} A_{j_\ell})$ and event sequence
    $s = \langle (E_1, \tau_1), \ldots, (E_n, \tau_n = T) \rangle$, Length of history window $W$, Frequency
    threshold $f_{min}$.
$\Delta = W - span(\alpha, \mathcal{D})$
for all $A \in \mathcal{E}$ do
    for $\delta = 0$ to $\Delta$ do
        if $\delta = 0$ and $(A_{j_1} > A$ or $\ell = 1)$ then
            continue
        $(\alpha', \mathcal{D}') = A \overset{\delta}{\to} \alpha$; $\mathcal{L}_{(\alpha',\mathcal{D}')} = \{\}$; $f_s(\alpha', \mathcal{D}') = 0$
        for all $\tau_i \in \mathcal{L}_{(\alpha,\mathcal{D})}$ do
            if $\exists (E_j, \tau_j)$ such that $E_j = A$ and $\tau_i - \tau_j = \delta$ then
                Increment $f_s(\alpha', \mathcal{D}')$
                $\mathcal{L}_{(\alpha',\mathcal{D}')} = \mathcal{L}_{(\alpha',\mathcal{D}')} \cup \{\tau_j\}$
        if $f_s(\alpha', \mathcal{D}') \geq f_{min}$ then
            Add $(\alpha', \mathcal{D}')$ to output set $\mathcal{C}$
            if $span(\alpha', \mathcal{D}') \leq W$ then
                pattern_grow$(\alpha', \mathcal{D}', \mathcal{L}_{(\alpha',\mathcal{D}')})$
```

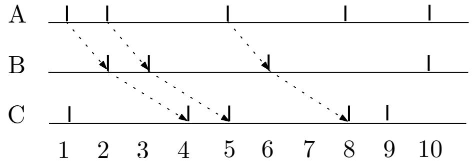


Fig. 3. An event sequence showing 3 distinct occurrences of the episode $A \overset{1}{\to} B \overset{2}{\to} C$.

inclusion-exclusion formula and only sets for which this mutual information exceeds ϑ are retained as candidate parent-sets (lines 5-8, *Procedure 1*). Finally, we prune out candidate parent-sets which have only *indirect influences* on A and return the final parent-sets for nodes corresponding to event-type A (lines 9-10, *Procedure 1*). This pruning step is based on conditional mutual information criteria (to be described later in Sec. 6.3).

6.2. Discovering fixed-delay episodes

We employ a pattern-growth algorithm (Procedure 2) for mining frequent fixed-delay episodes because, unlike *Apriori*-style algorithms, pattern-growth procedures allow use of different frequency thresholds for episodes ending in different alphabets. This is needed in our case, since, in general, *Theorem 4.1* prescribes different frequency thresholds for nodes in the network corresponding to different alphabets. The recursive procedure is invoked with $(\alpha, \mathcal{D}) = (A, \phi)$ and frequency threshold $f_{min} = (T - W)P_{min}\phi_{min}$ (Recall that in the main loop of *Procedure 1*, we look for parents of nodes corresponding to event-type $A \in \mathcal{E}$).

The pattern-growth algorithm listed in *Procedure 2* takes as input, an episode $(\alpha, \mathcal{D})$, a set of start times $\mathcal{L}_{(\alpha,\mathcal{D})}$, and the event sequence s. $\mathcal{L}_{(\alpha,\mathcal{D})}$ is a set of time stamps τ_i such that there is an occurrence of $(\alpha, \mathcal{D})$ starting at τ_i in s. For example, if at level 1 we have $(\alpha, \mathcal{D}) = (C, \phi)$, then $\mathcal{L}_{(C,\phi)} = \{1, 4, 5, 8, 9\}$ in the event sequence s shown in Fig 3. The algorithm obtains counts for all episodes like $(\alpha', \mathcal{D}')$ generated by extending $(\alpha, \mathcal{D})$ e.g. $B \xrightarrow{1} C$, ..., $A \xrightarrow{5} C$ etc. For an episode say $(\alpha', \mathcal{D}') = B \xrightarrow{2} C$, the count is obtained by looking for occurrences of event B at times $\tau_j = \tau_i - 2$ where $\tau_i \in \mathcal{L}_{(C,\phi)}$. In the example such B's at $\tau_j \in \mathcal{L}_{B \xrightarrow{2} C} = \{2, 3, 6\}$. The number of such occurrences $(= 3)$ gives the count of $B \xrightarrow{2} C$. At every step the algorithm tries to grow an episode with count $f_s > f_{min}$ otherwise stops.

6.3. Conditional MI criteria

The final step in determining the parents of a node X_A involves testing of conditional mutual information criteria (cf. line 10, *Procedure 1*). The input to this step is a set, $\mathcal{C}_A$, of candidate parent sets for each node X_A in the network, such that each candidate set $\mathcal{Y}$ has high mutual information with X_A i.e. we have $I[X_A\,;\,\mathcal{Y}] > \vartheta\ \forall \mathcal{Y} \in \mathcal{C}_A$. Consider two such sets $\mathcal{Y}$ and $\mathcal{Z}$, each having sufficient mutual information with X_A. Our conditional mutual information criterion is: *remove $\mathcal{Y}$ from the set of candidate parents (of X_A), if $I[X_A\,;\,\mathcal{Y} \mid \mathcal{Z}] = 0$*[3]. We repeat this test for every pair of episodes in $\mathcal{C}$.

To understand the utility of this criterion, there are two cases to consider: (i) either $\mathcal{Y} \subset \mathcal{Z}$ or $\mathcal{Z} \subset \mathcal{Y}$, and (ii) both $\mathcal{Y} \not\subset \mathcal{Z}$ and $\mathcal{Z} \not\subset \mathcal{Y}$. In the first case, our conditional mutual criterion will ensure that we pick the larger set as a parent only if it brings more information about X_A than the smaller set. In the second case, we are interested in eliminating sets which have a high mutual information with X_A because of indirect influences. For example, if the network were such that C excites B and B excites A, then X_C can have high mutual information with X_A, but we do not want to report X_C as a parent of X_A, since C influences A only *through* B. Our conditional mutual information criterion will detect this and eliminate X_C from the set of candidate parents (of X_A) because it will detect $I[X_A\,;\,X_C \mid X_B] = 0$.

We now summarize the conditional mutual information step in our algorithm. We pick candidates out of $\mathcal{C}_A$ in decreasing order of their respective mutual informations with X_A. A candidate set, say $\mathcal{Y}$, is declared to belong to the final parent set Π only if $I[X_A\,;\,\mathcal{Y} \mid \mathcal{Z}] = 0$ for all $\mathcal{Z} \in \mathcal{C}_A,\ \mathcal{Z} \neq \mathcal{Y}$. Note that the time needed for pruning, which is quadratic in the size of $\mathcal{C}_A$, is typically small since the number of frequent fixed-delay episodes ending in A is typically small.

7. Results

In this section, we present results both on data gathered from mathematical models of spiking neurons and real neuroscience experiments.

[3] We use a small threshold parameter to ascertain this equality.

7.1. Spiking Neuronal Network Models

Several mathematical models of spiking neurons have been proposed and studied in neuroscience (Sprekeler, Michaelis and Wiskott, 2007; Czanner, Eden, Wirth, Yanike, Suzuki and Brown, 2008; Eldawlatly et al., 2010; Patnaik et al., 2007; Sastry and Unnikrishnan, 2010; Raajay, 2009). For example, (Sprekeler et al., 2007) employ a linear Poisson model where spike train signals are generated using an inhomogeneous Poisson process, while (Czanner et al., 2008) study a point process observation model of neural spiking activity. In (Eldawlatly et al., 2010) the spike train of a neuron is expressed as a conditionally Poisson point process. A network of interacting neurons, each modeled as an inhomogeneous Poisson process was introduced in (Patnaik et al., 2007) and this model was extended to incorporate higher-order interactions in (Raajay, 2009). These models based on inter-dependent inhomogeneous Poisson processes can incorporate sophisticated relationships between neurons in the network and constitute a rich source of data for testing our DBN learning models. We briefly introduce the spiking neuronal model of (Raajay, 2009) which we use in our experimental work.

The data generation model[4] takes as input an inter-connected set of neurons. The spike train of each neuron is modeled as an inhomogeneous Poisson process and interactions with other neurons are introduced by controlling the firing rate of the neuron as a function of the input it receives from others. This rate is updated every ΔT time units (We can think of ΔT as the resolution of each time-tick). The firing rate $\lambda_i(t)$ of the i^{th} neuron at time instant $t\Delta T$ (or t^{th} time-tick) is given by

$$\lambda_i(t) = \frac{\widehat{\lambda}_1}{1 + exp(-I_i(t) + d)} \tag{14}$$

where $I_i(t)$ denotes the input received by the i^{th} neuron at time $t\Delta T$, $\widehat{\lambda}_1$ determines the (high) firing rate of the neuron in the excited state (when sufficient potential accumulates at its input) and d denotes an offset parameter that is fixed based on the user-defined rest (or quiescent) firing rate $\widehat{\lambda}_0$ of the neuron (when the input is *zero*). The simulator determines the high firing rate $\widehat{\lambda}_1$ based on the desired conditional probability ρ of firing of the neuron when it receives its full expected input (ρ is a user-defined parameter). The network inter-connect model gives it the sophistication needed for simulating higher-order interactions. The model allows for variable delays which mimic the delays in conduction pathways of real neurons. The total input $I_i(t)$ received by the i^{th} neuron at time-tick t is given by

$$I_i(t) = \sum_{j} \beta_{ij} Y_{j(t-\tau_{ij})} + \ldots + \sum_{j\ldots\ell} \beta_{ij\ldots l} Y_{j(t-\tau_{ij})} \ldots Y_{\ell(t-\tau_{i\ell})} \tag{15}$$

where $Y_{j(t-\tau_{ij})}$ is the indicator of a spike on j^{th} neuron τ_{ij} time-ticks before t and $\beta_{(.)}$'s are the weights of the corresponding synaptic interactions. The first summation in Eq. (15) models first-order interactions, while the subsequent summations model progressively higher-order interactions. For example, to model a strong first-order connection from neuron j to neuron i with appropriate synaptic

[4] Simulator courtesy Raajay Viswanathan, Electrical Engineering, Indian Institute of Science, Bangalore.

delays (i.e. to obtain a high conditional probability for the firing of i given that j has fired at an appropriate time in the past) the simulator would set a high value for the weight β_{ij}. Similarly, to model a higher-order interaction such as one that sets a high conditional probability of firing for neuron i given that neurons j, k and ℓ fired (at times determined by the corresponding synaptic delays) the simulator assigns a high value for the weight β_{ijkl}. Finally, the simulator also incorporates a short refractory period for the neurons (which is the time for which a neuron does not respond to any stimulus after it has just spiked).

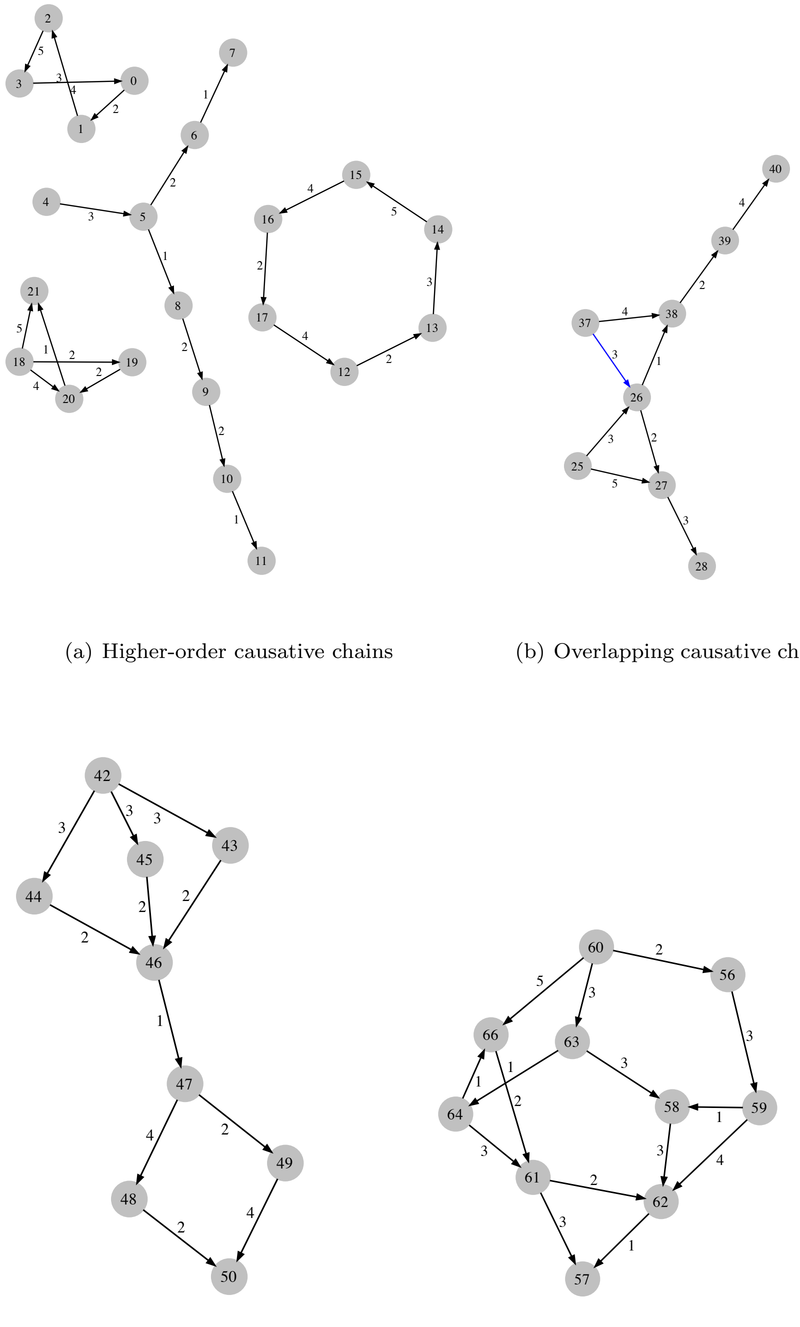
7.1.1. Types of Networks

We generate spike train data from a wide range of networks by embedding several different kinds of neuronal interactions and by varying the different user-defined parameters in the simulator.

1. **Causative chains and higher-order structures**: A higher-order chain is one where parent sets are not restricted to be of cardinality one. In the example network of Fig. 4(a), there are four disconnected components with two of them having cycles. (Recall this would be 'illegal' in a static Bayesian network formulation). Also the component consisting of nodes $\{18, 19, 20, 21\}$ exhibits higher-order interactions. The node 20 fires with high probability when node 18 has fired 4 ms before and node 19 has fired 2 ms before. Similarly node 21 is activated by nodes $\{18, 20\}$ firing at respective delays.
2. **Overlapping causative chains**: The graph shown in Fig. 4(b) has two chains $\{25, 26, 27, 28\}$ and $\{37, 26, 38, 39, 40\}$. Node 26 is common to both chains and can be independently excited by 25 or 37. Also 27 is activated by $\{25, 26\}$ together and 38 is activated by $\{26, 37\}$. Thus a firing event on 25 excites one chain while a firing event on 37 excites the other chain. This shows one possible way in which neurons can participate in multiple circuits at the same time (e.g. polychronous circuits (Izhikevich, 2006)). Depending on the stimulus sequence, the same neurons can participate in different cascades of firing events (encoding completely different pieces of information).
3. **Syn-fire chains**: Another important class of patterns studied in neuron spike trains is called synfire chains. This consists of groups of synchronously firing neurons strung together repeating over time. An example of a syn-fire chain is given in Fig. 4(c). In (Patnaik et al., 2007), it was noted that discovering such patterns required a combination of serial and parallel episode mining. But the DBN approach applies more naturally to mining such network structures.
4. **Polychronous circuits**: Groups of neurons that fire in a time-locked manner with respect to each other are referred to as polychronous groups. This notion was introduced in (Izhikevich, 2006) and gives rise to an important class of patterns. Once again, our DBN formulation is a natural fit for discovering such groups from spike train data. A polychronous circuit is shown in Fig 4(d).

7.1.2. Data sets

The data sets we used for our experiments are listed in Table 2. The name of the data set is listed in Column 1, the length of the data set (or number of time-slices in the data sequence) in Column 2, the size of the alphabet (or total number of neurons in the network) in Column 3, the rest firing rate $\widehat{\lambda}_0$ in Column 4 and the conditional probability ρ (that a neuron will fire when it receives its



(a) Higher-order causative chains

(b) Overlapping causative chains

(c) Syn-fire Chains

(d) Polychronous Circuits

Fig. 4. Four classes of DBNs investigated in our experiments.

Table 2. Data sets

Dataset Name	Data length	Alphabet Size	Rest Firing Rate $\widehat{\lambda}_0$	Conditional Probability ρ
A1	60000	100	0.02	0.9
A2	60000	125	0.02	0.9
A3	60000	150	0.02	0.9
A4	60000	175	0.02	0.9
B5	60000	100	0.01	0.9
B6	60000	100	0.015	0.9
B7	60000	100	0.02	0.9
B8	60000	100	0.025	0.9
C9	60000	100	0.02	0.8
C10	60000	100	0.02	0.85
C11	60000	100	0.02	0.9
C12	60000	100	0.02	0.95
D13	60000	100	0.02	0.9
D14	90000	100	0.02	0.9
D15	120000	100	0.02	0.9

full expected input) in Column 5. All the substructures listed in Figs. 4(a)-4(d) were inserted into all the data sets. The data sets are arranged into 4 groups. In the A-group (A1-A4) the data length is 60000 events, the rest firing rate $\widehat{\lambda}_0$ is 0.02, the conditional probability ρ is 0.9 and the size of alphabet is varied from 100 to 175. In the B-group (B5-B8) the rest firing rate $\widehat{\lambda}_0$ is varied from 0.01 to 0.025 keeping everything else constant. Similarly, in the C-group (C9-C12) the conditional probability ρ is varied from 0.8 to 0.95. Finally, in the D-group (D13-D15) the data lengths are varied from 60000-120000 events.

7.2. Competing methods

Most of the existing DBN literature on structure learning is focused on networks with first order Markov assumption and use hidden variables to circumvent this limitation (Friedman et al., 1998; Murphy, 2002). In contrast, our formulation allows variable order structure learning controlled by the parameters k and w in our algorithm. This makes baseline comparison with existing methods difficult. There is also a limit on the number of variables in the data for such algorithms (Chickering, 2003; Cooper and Herskovits, 1992). In this paper we found it most natural to compare our method with an implementation of Sparse Candidate algorithm (Friedman, Nachman and Pe'er, 1999) extended for learning DBN structures and the Simulated Annealing-based DBN learning proposed in (Eldawlatly et al., 2010). (We use the shorthands SC and SA to denote the Sparse Candidate algorithm and the Simulated Annealing-based DBN respectively). SC was originally proposed for structure learning of regular Bayesian Networks. We use a natural extension of SC for DBNs based on our assumptions in Section 3. In the Dynamic Probabilistic Networks formulation (Friedman et al., 1998), two scoring functions were proposed, namely BIC (cf. Eq. (16)) and BDe (cf. Eq. (17)) (The latter assumes Dirichlet priors).

$$BIC = \sum_{i,j_i,k_i} N_{i,j_i,k_i} \log \frac{N_{i,j_i,k_i}}{\sum_{k_i} N_{i,j_i,k_i}} - \frac{\log N}{2} \#G \qquad (16)$$

Table 3. Summary of results of SA with BDe score over the data sets of Table 2.

Parameters	Precision (%)	Recall (%)	Run-time (in secs.)
k=5,w=5	17.1 ± 6.2	42.5 ± 6.9	9197.0
k=5,w=10	8.7 ± 3.3	23.9 ± 4.8	9760.9
k=10,w=5	12.2 ± 3.7	39.0 ± 8.5	9522.7
k=10,w=10	7.5 ± 3.6	25.9 ± 10.0	9816.1

where N_{i,j_i,k_i} is number of times $X_i(t)$ takes the value k_i and its parent-set Π_i takes the value j_i, $\#G$ is the dimension of the bayesian network G (which in this case is the number of parameters in the conditional probability tables) and N is the number of time slices in the data sequence.

$$BDe = \prod_{i,j_i} \frac{\Gamma\left(\sum_{k_i} N'_{i,j_i,k_i}\right)}{\Gamma\left(\sum_{k_i} N'_{i,j_i,k_i} + N_{i,j_i,k_i}\right)} \prod_{k_i} \frac{\Gamma\left(N'_{i,j_i,k_i} + N_{i,j_i,k_i}\right)}{\Gamma\left(N'_{i,j_i,k_i}\right)} \tag{17}$$

where N_{i,j_i,k_i} is same as before, and the hyper-parameters N'_{i,j_i,k_i} are selected by assuming an empty prior network: $N'_{i,j_i,k_i} = \hat{N} \times P(X_i(t) = k_i)$ where $\hat{N}$ is called the effective data length of the prior and we set it at 10% of the actual data length. We experimented with both these scoring functions and the results were comparable in terms of quality as well as run-times. Hence, in this paper we quote SC results only for BIC. However we use BDe score for SA (as suggested by (Eldawlatly et al., 2010)). The SA method simply uses the BDe score along with simulated annealing for searching through the space of candidates. The main tunable parameter in simulated annealing is the temperature or cooling schedule. We experimented with many schedules in our implementation: $T = T_0(0.99)^k$ for $T_0 = 10^2, 10^4, 10^6$ and $T = \frac{T_0}{1+\log C.k}$ for $T_0 = 10^4, 10^6$ and $C = 1, 10, 100$. Further, at each iteration of SA we remembered the best state so far, and used the best state after around 2.5 hours as the final answer.

7.3. Performance

First we summarize the performance of SA on the data sets of Table 2. The table quotes the average precision and recall values obtained across all 15 data sets along with the corresponding average run-times. The results are reported in (mean±std. dev.) format. These are the best results we could achieve with SA in reasonable time (about 2.5 hours). For this we used the exponentially decreasing cooling schedule with parameter $T_0 = 10^6$. We were unable to extract any meaningful performance on these data sets using SA (despite trying a wide range of parameter tuning schedules). The precision and recall almost never exceeded 20% and 50% respectively even after letting each optimization run for more than 2.5 hours. These run-times were typically 5-10 times longer than the corresponding run-times for SC and EDN (which also delivered substantially better precision and recall performance). Based on our experiments, we concluded that SA was unsuitable for learning structure from our inter-dependent inhomogeneous poisson model of spiking neuronal networks. For these reasons, in this section, we provide full experimental results and comparisons only for SC and EDN based methods and omit the detailed results of SA to avoid clutter.

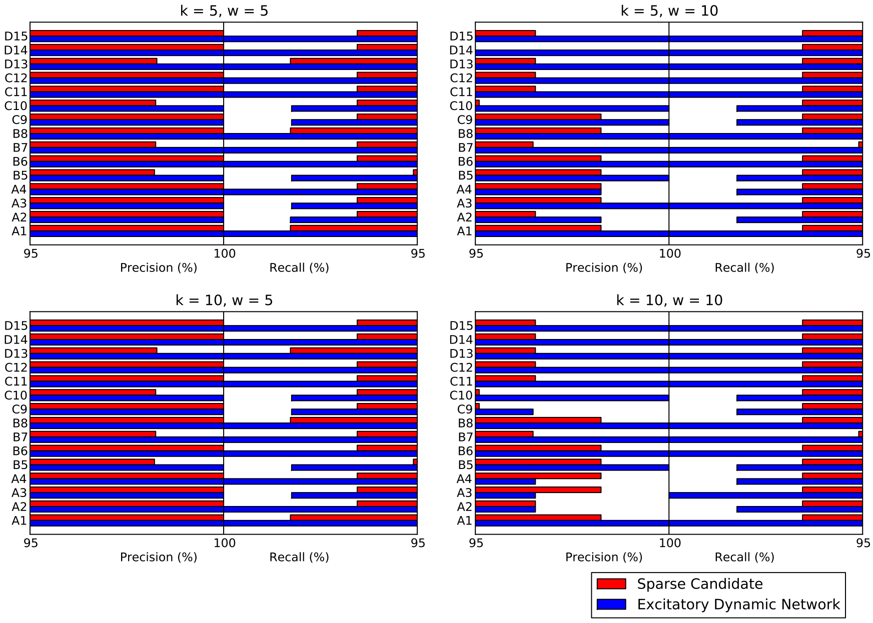


Fig. 5. Precision and recall comparison of Excitatory Dynamic Network discovery algorithm and Sparse Candidate algorithm for different window sizes, w and max. no. of parents, k.

7.3.1. Quality of inferred networks

Fig. 5 shows a comparison of the performance of EDN and SC on the 15 data sets of Table 2 for different choices of k (maximum number of parents) and w (size of history window). The precision and recall for a given data set are represented as bars along the same horizontal line (the light color bar represents SC and the dark color bar represents EDN). For example, for the $k = 5$ $w = 5$ case, both precision and recall of EDN were 100% for 10 (out of the 15) data sets. In case of SC, although precision was 100% in 11 data sets, the recall was between 95% and 98% for all the data sets. From the results reported for different k and w values, we can see that (on most data sets) EDN consistently outperformed SC in terms of precision and recall. This is not surprising because the network structures injected into these data sets were predominantly excitatory. The results also highlight the bias of EDN towards high precision performance (sometimes, at the cost of some loss in recall). This is primarily due to the conditional mutual information checks (which, in case of EDNs, is feasible because of the typically small number of frequent episodes ending in the same event-type).

Table 4 presents a comparison of SC and EDN for the different groups of networks defined in Sec. 7.1.1. We consider each of the network groups described in Figs. 4(a)-4(d) separately and report the average precision and recall values obtained by SC and EDN across all the 15 data sets of Table 2 (for $k = 5$, $w = 5$). The results show that the few edges that EDN missed were from the more complex network structures, namely, overlapping causative chains and polychronous circuits (while causative chains, higher-order structures and syn-fire chains were

Table 4. Comparison of quality of subnetworks recovered by Excitatory Dynamic Network algorithm and Sparse Candidate algorithm. (k=5,w=5)

	Excitatory Dynamic Network		Sparse Candidate	
Network group	Precision(%)	Recall(%)	Precision(%)	Recall(%)
Causative chains & Higher order structures	100.0	100.0	100.0	99.39
Overlapping causative chains	100.0	99.26	98.42	92.59
Syn-fire chains	100.0	100.0	100.0	99.39
Polychronous circuits	100.0	98.22	99.06	93.77

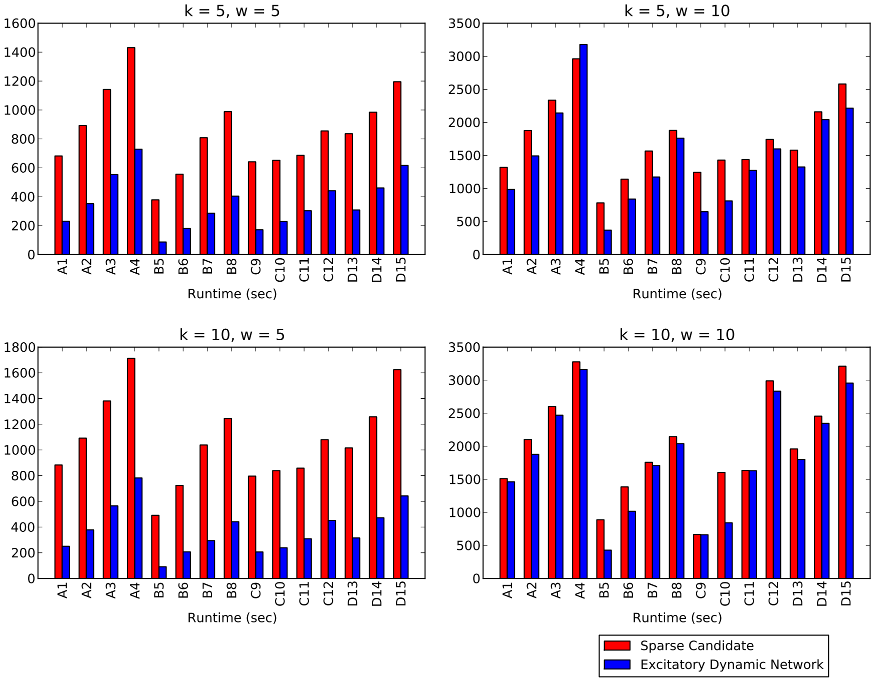


Fig. 6. Run-time comparison of Excitatory Dynamic Network discovery algorithm and Sparse Candidate algorithm for different window sizes, w and max. no. of parents, k.

detected with 100% precision and recall by EDN). The same trends can be observed for SC also, although the deterioration for overlapping causative chains and polychronous circuits is more severe here than in EDN. Similar trends were observed for the other choices of k and w as well.

7.3.2. Run-times

Fig. 6 shows the run-time comparisons for EDN and SC on the 15 data sets of Table 2. The run-time bars for ($k = 5, w = 5$) and for ($k = 10, w = 5$) show a substantial run-time advantage for EDN over SC (by a factor of at least two or more) in all the data sets. For ($k = 5, w = 10$) and ($k = 10, w = 10$) also,

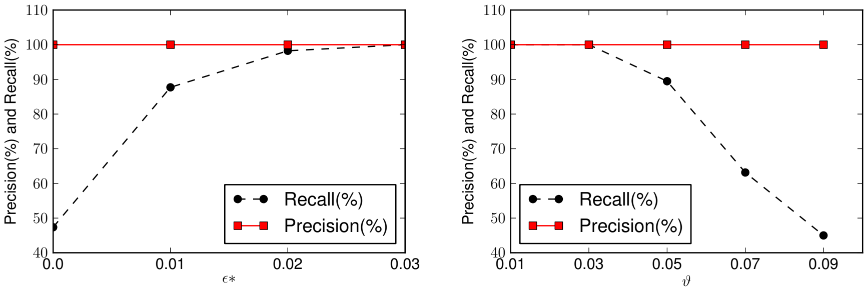


Fig. 7. Effect of $\epsilon*$ and ϑ on precision and recall of EDN method applied to dataset A1.

we observe that EDN is consistently faster than SC (although the improvement is much less). This shows that if we use a w value that is near the true size of history window in the network generating the data, EDN has a considerable computational advantage over SC (and this advantage reduces for larger values of w).

7.3.3. *Effect of parameters*

In Sec. 4.1, we discussed the role of ϑ and ϵ^* in fixing the frequency threshold ($P_{min}\Phi_{min}$) for frequent episode discovery. We argued that while the threshold decreases as ϵ^* increases, it increases as ϑ increases. In Fig. 7 we plot the effect of changing ϵ^* and ϑ on the performance of our EDN learning algorithms. The figure show results obtained on the A1 data set. Here again, we see the bias of EDN toward high precision – the plots show the precision staying at 100% even as ϵ^* and ϑ are varied. The recall however is more sensitive to the choice of parameters, showing an increasing trend with ϵ^* and a decreasing trend with ϑ. These results are consistent with our theoretical understanding of how frequency threshold changes with ϵ^* and ϑ. Eventually, for sufficiently high ϵ^* and for sufficiently low ϑ the recall starts to approach 100%. (Similar trends were observed in other data sets as well).

Finally, we describe our choice of EDN parameters that were used to generate the results reported in Figs. 5-6 and in Table 4. For A-group and D-group data sets we used $\epsilon^* = 0.03$ and $\vartheta = 0.03$. The conditional probability ρ varies in the C-group data sets, and hence, for these data sets, we used a lower mutual information threshold $\vartheta = 0.02$ and kept $\epsilon^* = 0.03$ (as before). The B-group data sets were the more tricky cases for our EDN algorithm since the base firing rates vary in this group. For B5-B6 (which have lower base firing rates) we used $\epsilon^* = 0.02$ and for B7-B8 (which have higher base firing rates) we used $\epsilon^* = 0.03$. The mutual information threshold for all the B-group data sets was set at $\vartheta = 0.03$.

7.4. Results on MEA data

Multi-electrode arrays provide high throughput recordings of the spiking activity in neuronal tissue and are hence rich sources of event data where events corre-

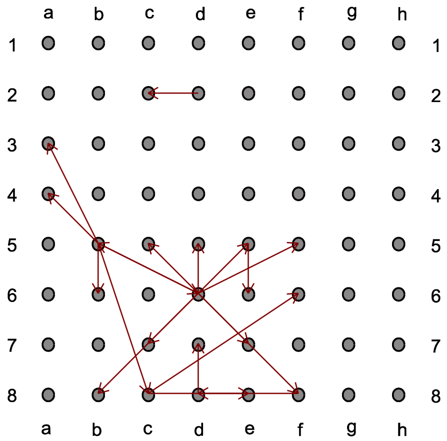


Fig. 8. DBN structure discovered from first 10 min of spike train recording on day 35 of culture 2-1 (Wagenaar et al., 2006).

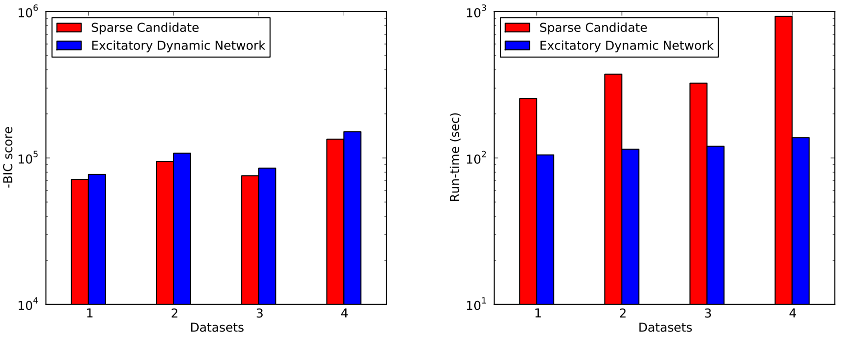
Table 5. Comparison of networks discovered by EDN algorithm and SC algorithm on 4 slices of spike train recording (each 10 min long) taken from datasets 2-1-32 to 2-1-35 (Wagenaar et al., 2006). Parameters $k = 2$ and $w = 2$ used for both EDN and SC.

	No. of common edges	No. of unique EDN edges	No. of unique SC edges
Data Slice 1	25	59 (70.24%)	30 (54.55%)
Data Slice 2	24	53 (68.83%)	31 (56.36%)
Data Slice 3	30	45 (60.00%)	26 (46.43%)
Data Slice 4	29	41 (58.57%)	27 (48.21%)

spond to specific neurons being activated. We used data from dissociated cortical cultures gathered by Steve Potter's laboratory at Georgia Tech (Wagenaar, Pine and Potter, 2006) over several days.

In our first experiment, we estimated EDNs from 10 minute-chunks of recordings using parameters $k = 2$, $w = 2$, $\vartheta = 0.001$ and $\epsilon^* = 0.02$. By way of example, we visualize one such network that we estimated in Fig. 8. This network obtained from the first 10 minutes of the spike train recording on day 35 of culture 2-1 reflects the sustained bursts observed in this culture by (Wagenaar et al., 2006). In order to establish the significance of the networks discovered we ran our algorithm on several surrogate spike trains generated by replacing the neuron labels of spikes in the real data with randomly chosen labels. These surrogates break the temporal correlations in the data while still preserving the overall summary statistics. No network structures were found in such surrogate sequences. We are currently in the process of characterizing and interpreting the usefulness of such networks found in real data.

In our second experiment using recordings from real cortical cultures, we compared the networks estimated using SC and EDN on different slices of data. The results obtained on 4 such slices are reported in Table 5. For each data slice, Column 1 of the table reports the number of edges that were common in the outputs of EDN and SC. The number of unique edges found by each of the methods is listed in columns 3 and 4. The figures in brackets describe the cor-



(a) BIC score comparison

(b) Runtime comparison

Fig. 9. Comparison of networks discovered by EDN algorithm and SC algorithm in 4 slices of spike train recording (each 10 min long) taken from datasets 2-1-32 to 2-1-35 (Wagenaar et al., 2006). Parameters used: Mutual information threshold for EDN = 0.01 and for both EDN and SC, $k = 2$ and $w = 2$ were used.

responding number of unique edges computed as a *percentage* of the total edges found by EDN and SC respectively. The results show that considerable portions of the networks discovered (30% or more of edges found by either method) were actually common under EDN and SC. However, about 50-70% of the edges reported by EDN were unique, and similarly, 45-55% of the edges reported by SC were unique. This suggests that EDN and SC discover different kinds of networks from the same data sets (which is not surprising since EDN and SC restrict the network topologies in different ways and optimize for different criteria). The common edges correspond to *strong excitatory* connections, the unique edges in EDN represent relatively *weaker excitatory* connections and the unique edges in SC correspond to other *non-excitatory* connections (such as inhibitions or excitation-in-absence). Thus, EDN and SC infer complemntary relationships from time-stamped event sequences.

Next, we assess the quality of excitatory networks discovered by EDN and SC in the MEA data. Toward this end, we use a standard scoring function (such as BIC) to compare the parent sets returned by EDN and SC. For each data set, we compute the aggregate BIC score over all nodes for which the EDN returned non-empty parent sets and compare it with the corresponding BIC score for the same nodes but with parents as per the SC network. The comparison is reported in Fig. 9. In Fig. 9(b) observe that the run-times of SC are roughly 2-6 times worse than for EDN, while the strengths of the excitatory connections discovered (in terms of aggregate BIC scores) are within 10% of the corresponding scores in SC (see Fig. 9(a)). Recall that our theoretical results guarantee that the EDN algorithm finds the *optimal* network, over the class of all excitatory networks. Our results here show that, these excitatory networks discovered are also comparable in terms of connections strengths with the output of SC (which searches over non-excitatory networks as well).

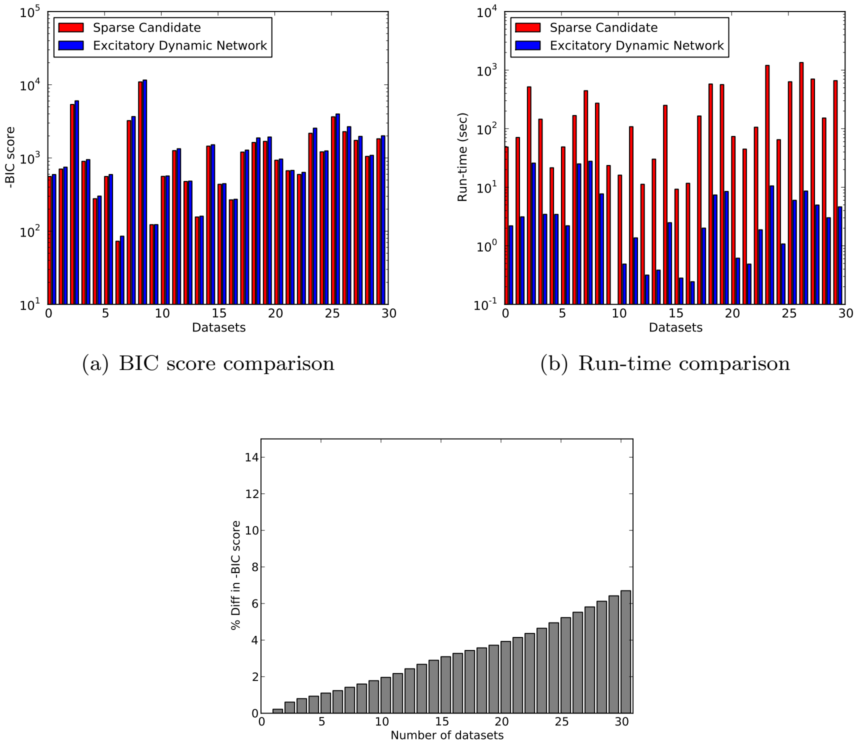
Table 6. Comparison of networks discovered by EDN algorithm and SC algorithm in calcium imaging spike train data (Source: (Takahashi et al., 2007)). Parameters: Mutual information threshold used for EDN = 0.0005 and for both EDN and SC, k=5 and w=5 were used.

	No. of	No. of unique	No. of unique
	common edges	EDN edges	SC edges
DATA_000	12	24 (66.67%)	1 (7.69%)
DATA_001	8	11 (57.89%)	10 (55.56%)
DATA_002	30	14 (31.82%)	43 (58.90%)
DATA_003	8	20 (71.43%)	5 (38.46%)
DATA_004	-	-	-
DATA_005	12	24 (66.67%)	1 (7.69%)
DATA_006	1	3 (75.00%)	1 (50.00%)
DATA_007	50	32 (39.02%)	57 (53.27%)
DATA_008	50	8 (13.79%)	49 (49.49%)
DATA_009	1	1 (50.00%)	0 (0.00%)
DATA_010	5	9 (64.29%)	3 (37.50%)
DATA_011	15	21 (58.33%)	9 (37.50%)
DATA_012	3	5 (62.50%)	1 (25.00%)
DATA_013	8	2 (20.00%)	1 (11.11%)
DATA_014	19	24 (55.81%)	13 (40.62%)
DATA_015	3	6 (66.67%)	0 (0.00%)
DATA_016	3	3 (50.00%)	0 (0.00%)
DATA_017	26	29 (52.73%)	9 (25.71%)
DATA_018	48	42 (46.67%)	46 (48.94%)
DATA_019	43	32 (42.67%)	34 (44.16%)
DATA_020	18	12 (40.00%)	7 (28.00%)
DATA_021	9	3 (25.00%)	1 (10.00%)
DATA_022	6	15 (71.43%)	2 (25.00%)
DATA_023	49	48 (49.48%)	53 (51.96%)
DATA_024	11	21 (65.62%)	1 (8.33%)
DATA_025	46	51 (52.58%)	23 (33.33%)
DATA_026	62	86 (58.11%)	82 (56.94%)
DATA_027	41	40 (49.38%)	45 (52.33%)
DATA_028	-	-	-
DATA_029	13	24 (64.86%)	4 (23.53%)
DATA_030	32	13 (28.89%)	25 (43.86%)

7.5. Results on fMCI data

Another relatively new technique of recording activity of multiple neurons simultaneously is functional Multi-neuron Calcium Imaging (fMCI). fMCI is an imaging technique where the neurons are loaded with calcium fluorophores. Their electrical activity causes fluorescence which is recorded using confocal microscopy. In the second set of experiments we analyze the recordings from rat hippocampal slices made publicly available by Yuji Ikegaya's group (Takahashi et al., 2007).

In Table 6, we present a comparison of the networks discovered by EDN and SC. Like the MEA results presented earlier (in Table 5), Table 6 has 4 columns – the first column describes the name of the data slice, the second column shows the number of edges commonly discovered by SC and EDN, the third column lists the number of unique EDN edges and the fourth column lists the number of unique SC edges. In columns 3 and 4, the numbers in brackets describe the corresponding quantities as percentages of total number of EDN and SC edges. There is no clear pattern in the result – in some data sets, like in DATA_007 and DATA_026, the proportion of common edges is reasonably high, while in some



(a) BIC score comparison

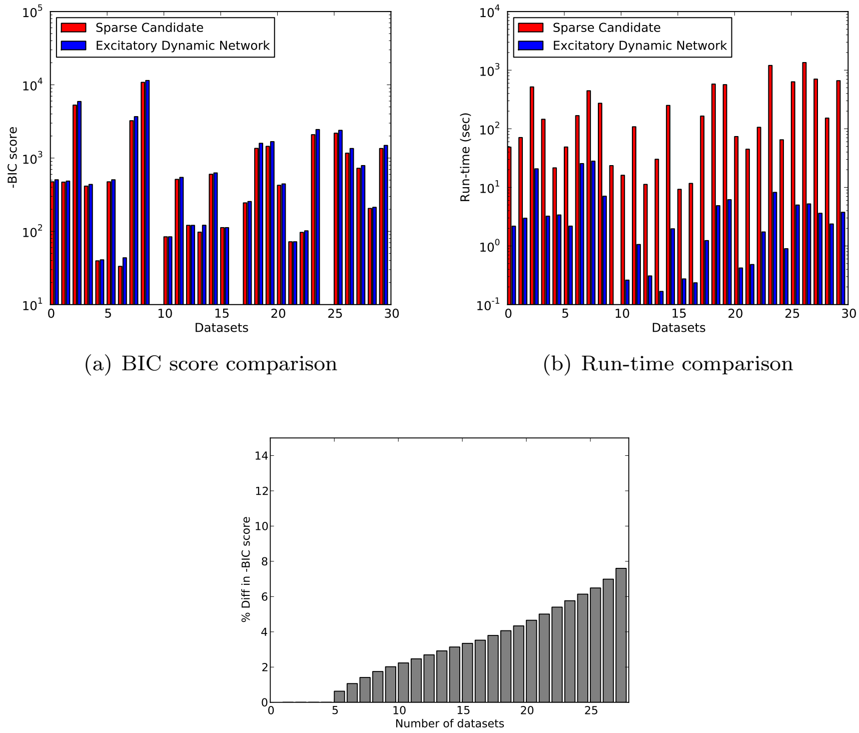
(b) Run-time comparison

(c) % Diff. in BIC score

Fig. 10. Comparison of networks discovered by EDN algorithm and SC algorithm in calcium imaging spike train data [Source: (Takahashi et al., 2007)]. Parameters used: Mutual information threshold for EDN = 0.0005 and for both EDN and SC, k=5 and w=5 were used.

others, like in DATA_001, this proportion is small (Results are not reported for DATA_004 and DATA_028 since, for these data sets, neither algorithm converged even after running for over three hours). These results again show that EDN and SC often yield different Bayesian networks. While SC discovers strong connections between nodes over an unconstrained class of networks, EDN focusses attention only on excitatory networks (a restriction that is natural in applications like neuroscience).

Finally, we compare strengths of the connections discovered as we have done in case of MEA data. For this, we compare aggregate BIC scores (and run-times) for the excitatory networks (under EDN) with the scores (and run-times) for corresponding subnetworks under SC. The results are shown in Fig. 10 for an MI threshold of 0.0005 (for EDN). Fig. 10(a) shows a bar graph for the corresponding BIC scores under both methods, while Fig. 10(c) shows the corresponding cumulative average plot. The graphs show that the BIC scores achieved under EDN and SC are consistently very close for all data sets. For example, in atleast 25 (out of 29) data sets the BIC scores differ by less than 5%. This shows that



(a) BIC score comparison

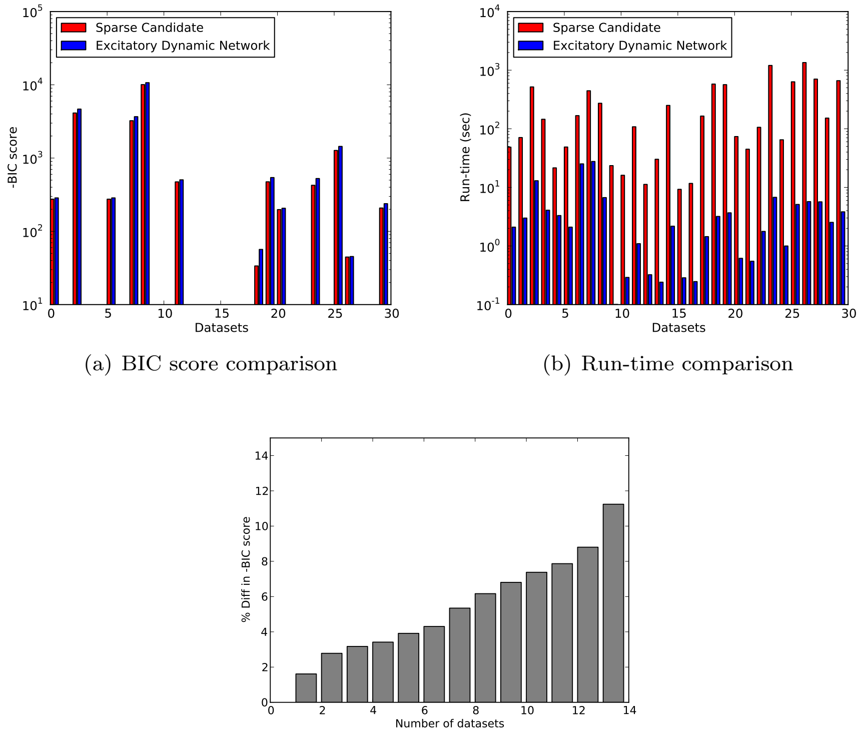
(b) Run-time comparison

(c) % Diff. in BIC score

Fig. 11. Comparison of networks discovered by EDN algorithm and SC algorithm in calcium imaging spike train data [Source: (Takahashi et al., 2007)]. Parameters used: Mutual information threshold for EDN = 0.001 and for both EDN and SC, k=5 and w=5 were used.

the excitatory networks discovered by EDN, although different from the networks obtained through SC, consist of connections that are of roughly the same strength (in terms of BIC scores). Moreover, these strong excitatory circuits are discovered very fast through EDN (compared to the run-times in SC shown in Fig. 10(b)). On the average, EDN runs more than 70 times faster than SC (with the minimum and maximum run-time advantages being 6 times and 270 times respectively).

To illustrate the effect of MI threshold on the EDN algorithm, we repeated our experiment for two higher thresholds of 0.001 and 0.002. The corresponding results are shown in Figs. 11 and 12. First, as can be expected, with increasing MI threshold, excitatory networks are discovered in fewer data sets. EDN found excitatory networks in 27 data sets at a threshold of 0.001 and in 13 data sets at a threshold of 0.002 (compared to finding excitatory networks in 29 datasets at a threshold of 0.0005). Second, the aggregate BIC scores for EDN became marginally poorer at higher thresholds. For example, the number of data sets on which BIC scores for EDN and SC differed by less than 5% fell to 22 at a threshold of 0.001 (see Fig. 11(c)) and to 6 at a threshold of 0.002 (see



(a) BIC score comparison

(b) Run-time comparison

(c) % Diff. in BIC score

Fig. 12. Comparison of networks discovered by EDN algorithm and SC algorithm in calcium imaging spike train data [Source: (Takahashi et al., 2007)]. Parameters used: Mutual information threshold for EDN = 0.002 and for both EDN and SC, k=5 and w=5 were used.

Fig. 12(c)). However, the corresponding run-time advantage of EDN became more exaggerated (with EDN being close to 90 times faster, on average, for both thresholds 0.001 and 0.002). All these results demonstrate that learning optimal excitatory networks is considerably more efficient than learning unrestricted DBNs of comparable connection-strengths. Thus, either when the application itself motivates use of excitatory networks (as is the case in neuroscience), or when learning unrestricted DBNs requires more computation than is acceptable (or both), excitatory networks (or EDNs) become relevant for learning optimal temporal networks from time-stamped event sequences.

In all of our experiments above, although we have diligently compared our approach to the Sparse Candidate algorithm, the latter is by no means the ground truth for the underlying network. On the contrary, it is interesting that we achieve comparable values for BIC (as the Sparse Candidate algorithm) without directly optimizing for it. These results lead to the radical suggestion that perhaps our approach can supplant SC as a good starting point to find DBNs in general, inspite of its seemingly narrow focus on EDNs, and with significant gains in runtime. Further work is necessary to explore this possibility, especially into the sufficiency of the model class of EDNs for practical application problems and the constraints imposed by the CPTs.

8. Discussion

Our work marries frequent pattern mining with probabilistic modeling for analyzing discrete event stream datasets. DBNs provide a formal probabilistic basis to model relationships between time-indexed random variables but are intractable to learn in the general case. Conversely, frequent episode mining is scalable to large datasets but does not exhibit the rigorous probabilistic interpretations that are the mainstay of the graphical models literature. We have presented the beginnings of research to relate these two diverse threads and demonstrated its potential to mine excitatory networks with applications in spike train analysis.

Two key directions of future work are being explored. The excitatory assumption as modeled here posits an order over the entries of the conditional probability table but does not impose strict distinctions of magnitude over these entries. This suggests that, besides the conditional independencies inferred by our approach, there could potentially be additional 'structural' constraints masquerading inside the conditional probability tables. We seek to tease out these relationships further. A second, more open, question is whether there are other useful classes of DBNs that have both practical relevance (like excitatory circuits) and which also can be tractably inferred using sufficient statistics of the form studied here.

Acknowledgment

This work is supported in part by General Motors Research, NSF grant CNS-0615181, and ICTAS, Virginia Tech. Authors thank V. Raajay and P. S. Sastry of Indian Institute of Science, Bangalore, for many useful discussions and for access to the neuronal spike train simulator of (Raajay, 2009).

References

Bromberg, F., Margaritis, D. and Honavar, V. (2009), 'Efficient markov network structure discovery using independence tests', *J. Artif. Int. Res.* **35**(1), 449–484.

Chickering, D. M. (2003), 'Optimal structure identification with greedy search', *J. Mach. Learn. Res.* **3**, 507–554.

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

Cooper, G. F. and Herskovits, E. (1992), 'A bayesian method for the induction of probabilistic networks from data', *Machine Learning* **9**, 309–347.

Czanner, G., Eden, U. T., Wirth, S., Yanike, M., Suzuki, W. A. and Brown, E. N. (2008), 'Analysis of between-trial and within-trial neural spiking dynamics', *Journal of Neurophysiology* (99), 2672–2693.

Eldawlatly, S., Zhou, Y., Jin, R. and Oweiss, K. G. (2010), 'On the use of dynamic bayesian networks in reconstructing functional neuronal networks from spike train ensembles', *Neural Computation* **22**(1), 158–189.

Friedman, N., Murphy, K. and Russell, S. (1998), Learning the structure of dynamic probabilistic networks, *in* 'Proc. UAI'98', Morgan Kaufmann, pp. 139–147.

Friedman, N., Nachman, I. and Pe'er, D. (1999), Learning bayesian network structure from massive datasets: The “Sparse Candidate” algorithm, *in* ‘5th Conf. on Uncertainty in Artificial Intelligence UAI (1999)’, pp. 206–215.

Izhikevich, E. M. (2006), ‘Polychronization: Computation with spikes’, *Neural Comput.* **18**(2), 245–282.

Jordan, M. I., ed. (1998), *Learning in Graphical Models*, MIT Press.

Laxman, S. (2006), Discovering frequent episodes: Fast algorithms, Connections with HMMs and generalizations, PhD thesis, IISc, Bangalore, India.

Laxman, S., Sastry, P. and Unnikrishnan, K. (2005), ‘Discovering frequent episodes and learning hidden markov models: A formal connection’, *IEEE TKDE* **Vol 17**(11), 1505–1517.

Mannila, H., Toivonen, H. and Verkamo, A. (1997), ‘Discovery of frequent episodes in event sequences’, *Data Mining and Knowledge Discovery* **1**(3), 259–289.

Meila, M. (1999), An accelerated chow and liu algorithm: Fitting tree distributions to high-dimensional sparse data, *in* ‘Proc. ICML’99’, pp. 249–257.

Murphy, K. (2002), Dynamic Bayesian Networks: representation, inference and learning, PhD thesis, University of California, Berkeley, CA, USA.

Papapetrou, P. et al. (2009), ‘Mining frequent arrangements of temporal intervals’, *Knowledge and Information Systems* **21**(2), 133–171.

Patnaik, D., Sastry, P. S. and Unnikrishnan, K. P. (2007), ‘Inferring neuronal network connectivity from spike data: A temporal data mining approach’, *Scientific Programming* **16**(1), 49–77.

Pavlov, D., Mannila, H. and Smyth, P. (2003), ‘Beyond independence: Probabilistic models for query approximation on binary transaction data’, *IEEE TKDE* **15**(6), 1409–1421.

Raajay, V. (2009), Frequent episode mining and multi-neuronal spike train data analysis, Master’s thesis, IISc, Bangalore.

Rieke, F., Warland, D., Steveninck, R. and Bialek, W. (1999), *Spikes: Exploring the Neural Code*, The MIT Press.

Sastry, P. S. and Unnikrishnan, K. P. (2010), ‘Conditional probability based significance tests for sequential patterns in multi-neuronal spike trains’, *Neural Computation* **22**(2), 1025–1059.

Seppanen, J. K. (2006), Using and extending itemsets in data mining: Query approximation, dense itemsets and tiles, PhD thesis, Helsinki University of Technology.

Sprekeler, H., Michaelis, C. and Wiskott, L. (2007), ‘Slowness: An objective for spike-timing-dependent plasticity?’, *PLoS Computational Biology* **3**(6).

Takahashi, N. et al. (2007), ‘Watching neuronal circuit dynamics through functional multineuron calcium imaging (fmci)’, *Neuroscience Research* **58**(3), 219 – 225.

Wagenaar, D. A., Pine, J. and Potter, S. M. (2006), ‘An extremely rich repertoire of bursting patterns during the development of cortical cultures’, *BMC Neuroscience* .

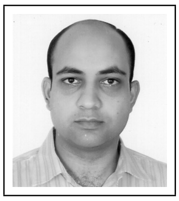
Wang, C. and Parthasarathy, S. (2006), Summarizing itemset patterns using probabilistic models, *in* ‘Proc. KDD’06’, ACM, New York, NY, USA, pp. 730–735.

Wang, K., Zhang, J., Shen, F. and Shi, L. (2008), ‘Adaptive learning of dynamic bayesian networks with changing structures by detecting geometric structures of time series’, *Knowledge and Information Systems* **17**(1), 121–133.

Wang, T. and Yang, J. (2009), ‘A heuristic method for learning bayesian networks using discrete particle swarm optimization’, *Knowledge and Information Systems* .

Williamson, J. (2000), Approximating discrete probability distributions with bayesian networks, *in* ‘Proc. Intl. Conf. on AI in Science & Technology, Tasmania’, pp. 16–20.

Author Biographies



Debprakash Patnaik is currently a Ph.D. student at the Department of Computer Science, Virginia Tech, USA. He received his bachelor degree from Veer Surendra Sai University of Technology, Orissa, India, in 2002 and masters degree from Indian Institute of Science, Bangalore, in 2006. He also worked as a researcher in the Diagnosis and Prognosis group at General Motors Research, Bangalore. His research interests include temporal data mining, probabilistic graphical models, and application in neuroscience.

Srivatsan Laxman Srivatsan Laxman is a Researcher in the CSAM Group (Cryptography, Security and Applied Mathematics) at Microsoft Research India, Bangalore. He works broadly in the areas of data mining and machine learning. Srivatsan received his PhD in Electrical Engineering from Indian Institute of Science, Bangalore.

Naren Ramakrishnan Naren Ramakrishnan is a professor and the associate head for graduate studies in the Department of Computer Science at Virginia Tech. He works with domain scientists and engineers to develop software systems for scientific knowledge discovery. Application domains in his research have spanned protein design, biochemical networks, neuro-informatics, plant physiology, wireless communications, and sustainable design of data centers. Ramakrishnan received his Ph.D. in computer sciences from Purdue University. He is an ACM Distinguished Scientist.

Correspondence and offprint requests to: Debprakash Patnaik, Department of Computer Science, Virginia Tech, Blacksburg, VA 24060, USA. Email: patnaik@vt.edu