Dynamic Sum-Product Networks

Mazen Melibari, Pascal Poupart, Edward Lank
{mmelibari,ppoupart,lank}@uwaterloo.ca
David R. Cheriton School of Computer Science
University of Waterloo
200 University Avenue West
Waterloo, ON, Canada
N2L 3G1

Technical Report: CS-2013-07

September 10, 2013

Inference in dynamic graphical models is known to be hard, except for models with low treewidth structure. This restricts severely the expressive power of these kinds of models. In this document we are proposing a new type of dynamic graphical model that allows one to model complex stochastic processes with unbounded treewidth while guaranteeing tractable exact inferenc e. The proposed dynamic model is an extension of a relatively new graphical model, named Sum-Product Network, that was introduced in 2011. The document also discusses the plan to develop a Bayesian non-parametric version of the model and an application in the area of activity recognition.

CONTENTS

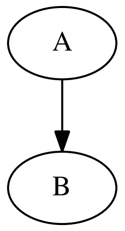
1 Introduction

Dynamic Bayesian Networks (DBNs) are one of the most popular tools for modeling sequential data and complex stochastic processes. They have been applied to a wide range of problems in different domains. Examples include information extraction, speech recognition, computer vision, and computational molecular biology. However, despite their expressive power, performing inference in DBNs –and probabilistic graphical models, in general– is known to be #P-hard [25]; the complexity of performing exact inference is exponential in the *treewidth* of the DBN's structure. Hence, in practice most DBN applications either rely on approximate inference methods or restrict themselves to only *low-treewidth*, i.e. tree-like, structures. That, consequently, severely limits the expressive power of DBNs.

In 2011, Hoifung Poon and Pedro Domingos proposed a new graphical probabilistic model named: Sum-Product Networks (SPNs) [22]. It was designed to avoid the intractability of most of the graphical models. This new class of graphical models can have unbounded treewidth while guaranteeing tractable and fast exact inference. Its inference complexity is linear in the number of the edges. It also showed significantly better results when compared to other deep learning models. A simple interpretation of an SPN is to see it as a hierarchical recursive network of low-level arithmetic operations. Hence, the structure of a SPN is nothing but alternate layers of sum and product nodes, as these are the only two required mathematical operations to answer any probabilistic inference query (sums substituted with integrals in the case of continuous variables).

In this document we are proposing a dynamic extension of Sum-Product Networks, which we name: Dynamic Sum-Product Networks (D-SPNs). D-SPNs will give the ability to model sequential and complex stochastic processes with unbounded network treewidth, while keeping the inference tractable. Similar to DBNs, this new model can be considered a factored state-space model. We will show how the main inference tasks for state-space models: monitoring, smoothing, and prediction can be done using D-SPNs. We are also proposing a Bayesian Nonparametric version of D-SPNs that requires no manual engineering to choose the right parameters for the model. Finally, we will discuss the plan to apply D-SPNs to a real-life activity monitoring application to show the performance of the proposed model and compare it to other state-of-the-art models and approximation methods.

The rest of this document is organized as follows. Section 2 reviews two



(a)

A	Θ_A
a	θ_a
$\bar{a}$	$\theta_{\bar{a}}$

A	B	$\Theta_{B\|A}$
a	b	$\theta_{a\|b}$
a	$\bar{b}$	$\theta_{a\|\bar{b}}$
$\bar{a}$	b	$\theta_{\bar{a}\|b}$
$\bar{a}$	$\bar{b}$	$\theta_{\bar{a}\|\bar{b}}$

(b)

A	B	$P(A,B)$
a	b	$\theta_a\theta_{a\|b}$
a	$\bar{b}$	$\theta_a\theta_{a\|\bar{b}}$
$\bar{a}$	b	$\theta_{\bar{a}}\theta_{\bar{a}\|b}$
$\bar{a}$	$\bar{b}$	$\theta_{\bar{a}}\theta_{\bar{a}\|\bar{b}}$

(c)

Figure 1: A simple Bayesian network, where (a) shows the network structure (b) CPTs, and (c) the joint probability distribution [6].

important concepts related to this research: (1) network polynomials, and (2) context-specific independence. A summary of the main contributions is given in section 3. Section 4 presents a survey of the three related research areas: (1) dynamic graphical models, (2) complexity of inference in graphical models, and (3) sum-product networks. The problem formulation is given in section 5. In section 6 we describe Sum-Product networks and show how they can be learned from data. Section 7 introduces our proposed solution and discuss its relation to other dynamic models. Section 8 discusses the application that we are going to apply our proposed model to. Finally, a conclusion and a summary of the research directions is given in section 9.

2 Preliminaries

2.1 Notation

This document follows the standard convention of denoting random variables using upper-case letters, e.g. X, and their instantiations using lower-case letters, e.g. x. The value *false* for a Boolean random variable X is denoted by $\bar{x}$, while x is used for the *true* value.

2.2 Network Polynomial

A discrete Bayesian network can be represented using a multi-linear function f called the network polynomial. This function can be used to perform probabilistic inference tasks through simple evaluation and differentiation procedures[6].

First, we define a set of indicators λ for all the values of the random variables in the Bayesian Network. Then we multiply each instantiation in the joint distribution by the compatible indicators and take their sums to define the network polynomial. For example, the indicators for the basic Bayesian network in figure 1 are: λ_a, $\lambda_{\bar{a}}$, λ_b, and $\lambda_{\bar{b}}$. Its network polynomial is:

$$f = \lambda_a \lambda_b \theta_a \theta_{a|b} + \lambda_a \lambda_{\bar{b}} \theta_a \theta_{a|\bar{b}} + \lambda_{\bar{a}} \lambda_b \theta_{\bar{a}} \theta_{\bar{a}|b} + \lambda_{\bar{a}} \lambda_{\bar{b}} \theta_{\bar{a}} \theta_{\bar{a}|\bar{b}} \tag{1}$$

More formally, the network polynomial for a Bayesian network is defined as follows:

$$f = \sum_x \prod_{\theta_x} \theta_x \lambda_x$$

where the summation is over the instantiations of all the random variabes, the product is over the parameters that are compatible with x and its parents, and λ_x are the compatible indicators.

Computing the probability of evidence e corresponds to evaluating the function f when the indicators are replaced with 1 or 0 based on whether they are consistent with e or not, respectively. If, for example, we observed the evidence $e = \bar{a}b$ for the network in Figure 1, we can compute its probability using Equation 1 by setting $\lambda_a = 0$, $\lambda_{\bar{a}} = 1$, $\lambda_b = 1$, and $\lambda_{\bar{b}} = 0$:

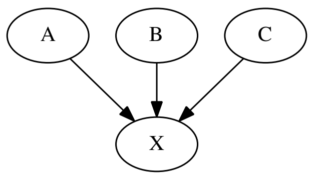
$$\begin{aligned} f(e) &= (0)(1)\theta_a \theta_{a|b} + (0)(0)\theta_a \theta_{a|\bar{b}} + (1)(1)\theta_{\bar{a}} \theta_{\bar{a}|b} + (0)(0)\theta_{\bar{a}} \theta_{\bar{a}|\bar{b}} \\ &= \theta_{\bar{a}} \theta_{\bar{a}|b} \end{aligned}$$

which is equal to $P(e)$ in the Bayesian network.

2.3 Context-Specific Independence

Consider the example in figure 2. The Bayesian network structure tells us that the random variable X depends on its three parents A, B, and C. We cannot say anything else about X from the graph. However, by examining the CPT we can see that under a specific assignment the variable X becomes independent from some of its parents. In particular, X becomes independent of B and C given that the value of $A = t$. Also, X becomes independent of C when $A = f$ and $B = t$. This type of independence is called *context-specific independence* [2].

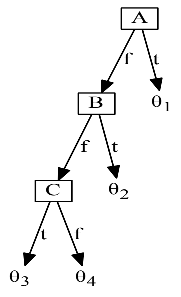
Context-specific independence is a type of independence that holds only for certain values of a set of variables. It can be represented using a tree as in figure 2(c). Nodes, edges, and leaves represent random variables, possible values, and conditional distributions, respectively. A full path from the root to a leaf gives a context in which an independence holds.



(a)

A	B	C	$P(x)$
t	t	t	θ_1
t	t	f	θ_1
t	f	t	θ_1
t	f	f	θ_1
f	t	t	θ_2
f	t	f	θ_2
f	f	t	θ_3
f	f	f	θ_4

(b)



(c)

Figure 2: An example of context-specific independence represented as a tree. (a) the Bayesian network structure, (b) the CPT for X, and (c) the tree-CPD for $P(X|A, B, C)$.

3 Overview of Contributions

In this section we give a brief overview of each of the three main proposed contributions of my work. Figure 3 summarizes the scope of the first and second contribution. The gray area in the figure shows the scope of the first contribution, which consists of developing a new dynamic graphical model. The black area is the scope of the second contribution where we are planning to develop a Bayesian Non-parametric version of the newly developed model.

3.1 A New Tractable Dynamic Graphical model

The main contribution of my research is the development of a new class of dynamic probabilistic graphical models, which we name Dynamic Sum-Product Networks (D-SPNs). As the name suggests, the new model is an extension of Sum-Product Networks [22]. It allows one to model discrete time-sliced stochastic processes. It guarantees fast and tractable exact inference even for some high-treewidth models. Moreover, D-SPNs can be learned much faster than typical probabilistic graphical models. That is essentially due to the fact that inference tasks are almost always parts of the learning process of probabilistic models. Consider for example the well known Baum-Welch algorithm for parameters learning. When applied to factored state-space models, such as DBNs, this algorithm requires at least two computationally intensive inferences tasks for each observation sequence at each learning iteration: one to

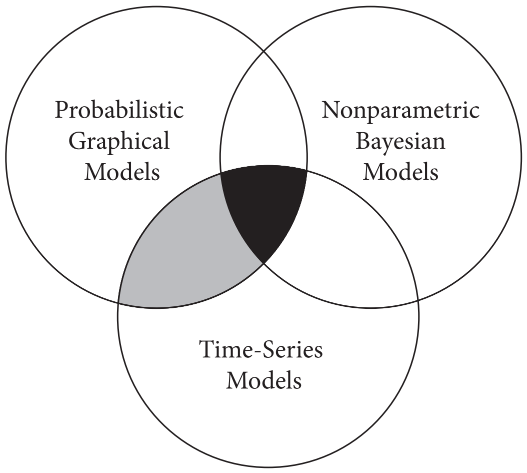


Figure 3: Research Scope. The gray and black area correspond to the first and second part of my research, respectively.

compute the forward probability and another one for the backward probability. One can clearly see that these two inferences tasks are the bottlenecks in the learning process as they their complexity is exponential in the number of states variables. If we can improve the speed of the inference algorithm, as we propose to do in D-SPNs, then we would also be able to improve the speed of the learning process.

3.2 Bayesian Nonparametric D-SPNs

As will be discussed in section 7, we augment the basic nature of SPNs by adding what we call *Summary Variables* **S**. The cardinalities of these latent variables have direct impact on the performance of the D-SPN. A low cardinality means that the model will have less degrees of freedom to represent all the necessary information about the belief state; hence, errors will propagate over time. On the other hand, high cardinality means that the model would have more parameters. This in turn makes the model more complex and requires more data to estimate the parameters accurately. Hence, the question: "How big should the latent variable state-space be?" is a crucial part of the D-SPN learning process.

The research area of Bayesian Non-Parametric models provides a pragmatic way to deal with the previous question through two models: Dirichlet Processes (DPs) [10] and its extension Hierarchical Dirichlet Processes (HDPs)

[26]. HDPs was applied successfully to HMMs [26, 12] to answer a similar question: "How many hidden states are sufficient to represent the available data?"; the model was named: Infinite Hidden Markov Model (iHMM), in reference to the fact that the parameter space of the model is allowed to grow as more data is obtained.

My second contribution is to develop a Bayesian Non-parametric version of the D-SPNs (Infinite D-SPNs), which allows the cardinality of the summary variables S_t to grow with the available data. This research will also lead to a contribution consisting of a Bayesian non-parametric version of the original static Sum-Product Networks, which can be seen as special cases of D-SPNs with one time slice.

3.3 Activity Recognition Using D-SPNs

D-SPNs with their fast inference capability are suitable for many real life applications, especially time critical ones. For example, a D-SPN model can be used in a highly rich networked sensor environment to monitor and assist individuals with Alzheimer's disease. The advantage of D-SPNs over other alternatives in these applications is that it allows one to develop complex models that incorporates all the sensors and hidden factors in the environment, while simultaneously allowing for near real-time inference. The last contribution of my research will be a real life application in the area of Activity Recognition that utilizes the power of D-SPNs.

4 Literature Review

This section gives a brief description of the most important literature that is related to this work. The three areas of research focused upon from the literature include dynamic graphic models, complexity in graphical models and the progress of Sum-Product networks. Other areas related to this work are covered in more detail in the background section.

4.1 Dynamic Graphical Models

The history of dynamic graphical models goes back to the Russian mathematician, Andry Andreyevich Markov, who was born in 1856. He established the branch of stochastic processes and studied one of the essential properties for most of the work in this area, which carries his namesake: the Markov property [1]. In [23], Rabiner and Jaung developed the first version of Hidden Markov Models, which can be seen as the first dynamic graphical model. Dean and Kanazawa coined the term Dynamic Bayesian Networks in 1989 [7]; their work is considered the first work that extends graphical models to the dynamic space. Several dynamic models that use the framework of DBNs were published after that, including Factorial HMMs [16], and Hierarchical HMM [11].

4.2 Complexity of Inference in Graphical Models

Cooper [5] was the first to formalize the complexity of inference in Bayesian Networks. Roth [25] showed latter that inference in graphical models is #P-hard, which is a class of complexity that is harder than NP-hard. A more recent work by Chandrasekaran [4] showed that in typical graphical models treewidth is the only property that can ensure tractability.

4.3 Sum-Product Networks

Sum-Product Networks were first introduced in [22]. The paper described the new model and presented a generative learning method that uses an online hard EM algorithm. A discriminative learning method was then presented in [14]. An algorithm to learn the structure of the SPN was recently published by Gens and Domingos in [15]. Their algorithm learns the structure from the data and showed comparable likelihood results to other graphical models.

5 Problem Formulation

The focus of this work is on finite discrete-time Markovian stochastic processes. Such processes are defined over sequences of countably infinite random variables. These random variables can be hidden X (can not be measured directly from the environment), or observed (evidence) E.

Dynamic Bayesian Networks (DBNs), which are extensions of Bayesian Networks, can be used to compactly represent the kind of processes that are considered in this work. Performing probabilistic inference with DBNs means that we want to compute the distribution of a set of variables A, given the values of another set of variables B. If the set B is empty then this is an inference task to compute the marginal of A, i.e. $P(A)$. On the other hand if B is not empty then we want to compute the posterior (conditional) probability, i.e $P(A|B)$.

5.1 Inference Tasks in Dynamic Models

The three main inference tasks that are of interest when dealing with dynamic graphical models are: monitoring, smoothing, and prediction [20]. Moni-

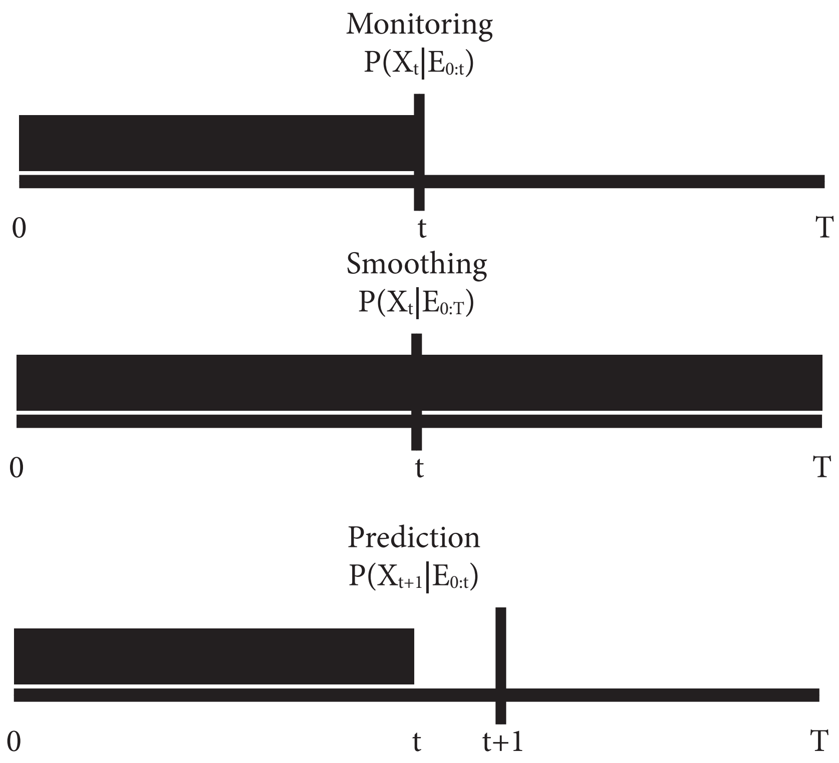


Figure 4: The three main inference tasks in state-space models. The horizontal fat bars represent the available evidences and the vertical bars represents the query that we are interested in.

toring is the task of performing inference about the current state given all the observations up until the current time slice: $P(X_t|E_{0:t})$. It is called monitoring, because we are tracking the state of the process over time. Smoothing consists of performing inference about a state in the past, given all the observations up until the current time: $P(X_\tau|E_{0:t})$, where $0 \leq \tau < t$. An alternative name for this task is: hindsight, which describes the nature of this task better. One example of this task is when investigating a case in the past and we want to use all available information, including the evidence that has happened after the occurrence of the case. Smoothing allows the investigator in this example to compute the probability of a state using both the evidence that occurs before the case as well as "through hindsight" the evidence that occurs after the case. Finally, prediction is the task of predicting the state of the process in a future time point, given all the available observations up until the current time: $P(X_\tau|E_{0:t})$, where $\tau > t$. Figure 4 visualizes these three inference tasks on a timeline.

5.2 Distance Metric

For the purpose of testing the performance of a new model, one could design an experiment using synthetic data and then compare the learned model with the ground truth model. In such a situation and in many others a measurement to compute the distance between the two probability distributions (the learned one and the ground-truth) is required. There are several statistical measurements designed for this purpose; one of which is Kullback-Leibler divergence (commonly known as KL-divergence) [18]. KL-divergence, basically, measures the amount of information lost when one probability distribution is used to approximate another. KL-divergence is formally defined as:

$$D(P||Q) = \sum_i log(\frac{P(i)}{Q(i)})P(i)$$

This measurement is going to be used throughout my research.

5.3 Problem Statement

The main goal of this work is to develop a dynamic probabilistic graphical model that can perform tractable exact inference for the three previously mentioned tasks and allows for unbounded *treewidth*. The performance of the model, in terms of speed and accuracy, shall be compared to current state-of-the-art inference techniques. Among the methods the model being developed in this work will be compared to are: Variational methods, Boyen-Koller [3], Markov Chain Monte Carlo, and Loopy Belief Propagation [19]. KL-divergence and inference time will be used to measure accuracy and speed, respectively.

6 Sum-Product Networks

Sum-Product Networks (SPNs) are rooted directed acyclic graphs that represent joint distributions over random variables. The leaves of SPNs are indicators for atomic values of the random variables. For example, a binary random variable x would have two leaves in an SPN: a leaf as an indicator for $x = 0$, and another one for $x = 1$.

The internal structure of the SPNs consists of alternating layers of sum and product nodes. A non-negative weight w is assigned to each edge of a sum

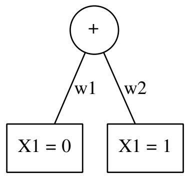


Figure 5: One binary random variable

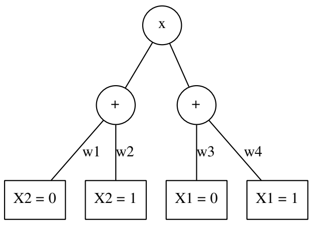


Figure 6: Two independent variables

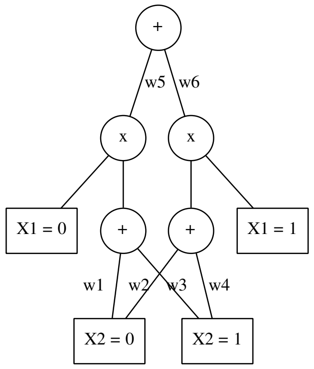


Figure 7: Two conditionally independent variables

node. These two types of nodes, the sums and products, represent the low level mathematical operations that need to be performed in order to complete any inference task.

To compute the value of the SPN, we evaluate the network from bottom-up and report the value of the SPN's root as the result of the computation. We start by assigning either 0 or 1 to the indicators. If a random variable was observed, then the indicator of the observed value will be 1 while the other indicators of this same variable will be 0. On the other hand, if a random variable has not been observed and it is a part of the inference query, we set all its indicators to 1.

The value of a product node p_i is the result of the product of its childrens' values:

$$\prod_{j \in Child(p_i)} v_j$$

where $Child(X)$ is a set of all the children of node X. The value of a sum node s_i, on the other hand, is the summation of the products of its edges' weights times the children values:

$$\sum_{j \in Child(s_i)} w_{ij} v_j$$

where w_{ij} is the weight for the edge from the sum node s_i to node j, and v_j is the value of node j.

Three basic SPNs are depicted in figures 5, 6, and 7. Figure 5 shows an SPN over one binary variable. To compute the probability of $X1 = 0$ from this

Algorithm 1 Learning an SPN [22]

```
Input: Dataset D over variables $X$.
Output: An SPN that is learned from D and represents a distribution over $X$.
  $\mathbf{S} \leftarrow GenerateDenseSPN(X)$
  $InitializeWeights(\mathbf{S})$
  repeat
    for all $d \in \mathbf{D}$ do
      UpdateWeights($\mathbf{S}$, Inference($\mathbf{S}$), $d$)
    end for
  until convergence
```

SPN, we perform the following computation:

$$P(X1 = 0) = (1 * w1) + (0 * w2)$$

The SPN in figure 6 is over two binary independent variables. Computing the probability of $X1 = 1$ and $X2 = 1$ can be done as follows:

$$P(X1 = 1, X2 = 1) = ((0 * w1) + (1 * w2)) * ((0 * w3) + (1 * w4))$$

Figure 7 shows a slightly more complex SPN. It depicts a network over two conditionally independent binary random variables. We can perform inference in this network using a computation that is similar to the previosuly mentioned one.

6.1 Learning SPNs

In the original paper that introduced SPNs [22], Poon and Domingos presented a general generative learning algorithm to learn the structure and parameters of an SPN. In general, the algorithm starts by generating a dense SPN over variables X and initializing the weights randomly. It then runs inference on each data point in the dataset and updates the weights. This process is repeated until convergence and, then, zero-weight edges are pruned and non-root nodes that have no parents are recursively removed. The resulting SPN is considered the final SPN. Algorithm 1 shows this general learning procedure.

Poon and Domingos' paper proposed two specific methods for updating weights: Gradient Descent, and Expectation-Maximization. For the gradient descent method we firstly compute the likelihood gradient using:

$$\frac{\partial\ \mathbf{S}(\mathbf{x})}{\partial w_{ij}} = \left(\frac{\partial\ \mathbf{S}(\mathbf{x})}{\partial\ \mathbf{S}_{\mathrm{i}}(\mathbf{x})}\right) .\ \mathbf{S}_{\mathrm{j}}(\mathbf{x})$$

where $\mathbf{S}(\mathbf{x})$ is an SPN over x, w_{ij} is the weight of the edge that connects node i and j, and $\mathbf{S_i}(\mathbf{x})$ is the sub-SPN for node i. It is worth noting that the terms of this equation can be efficiently computed using the inference algorithm for SPNs mentioned previously. A gradient step can then be used to update the weights of the sum nodes.

Expectation-Maximization can also be used to learn SPNs. In this procedure the $Inference$ task in Algorithm 1 is considered the E step, where we compute the marginals over the hidden variables of the SPN. The $UpdateWeights$ procedure is where the M step happens, in which we add the marginals to their values at the previous iteration and then we normalize the weights to obtain their new values.

This procedure includes internally a procedure to learn the structure of the SPN by starting from a generic SPN and then pruning all non-necessary nodes. However, the structure in this case is not completely learned from the data; it is restricted to the form of the initial generic SPN. Gens and Domingos [15] recently proposed a new learning algorithm that can learn the entire structure from the data. Essentially, their algorithm recursively tries to partition the set of variables X into independent subsets. If the partitioning succeeds we repeat the same procedure with each subset and return a product node of these partitions. Otherwise, we learn a sum node from a sub-set of the dataset and return it.

7 Proposed Solution: Dynamic-SPNs

Our proposed approach to model stochastic processes using Sum-Product Networks is to define two SPNs: one that represents the process at its initial state, and a second SPN that works as a template for the process at different time steps; this template represents the transition model of the process. The Initial-SPN and Transition-SPN can then be used to represent a stationary process of length T by instantiating the Initial-SPN and repeating the Transition-SPN $T-1$ times.

The nature of Sum-Product Networks leads to implicitly introducing latent variables between time slices (because, in SPNs, sum nodes can be seen as summing out a latent variable). We augment the implicit latent variables at each time step t by explicitly adding a set of latent random variables that work as the interface between time step t and $t+1$. The explicitly added latent variables help in collecting sufficient statistics about the state of the process at each time step. This augmentation can, simply, be seen as wrapping the

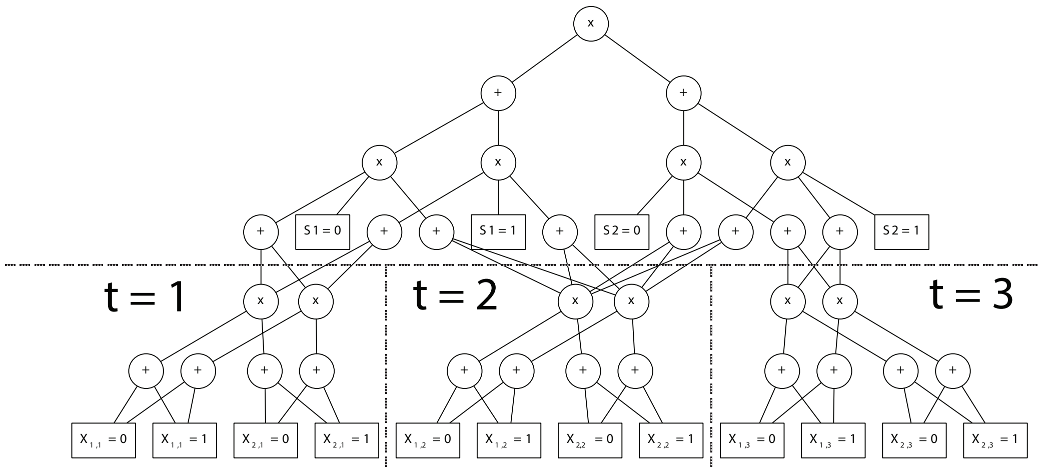


Figure 8: An example of a Dynamic Sum-Product Network. The network is over two binary random variables X_1 and X_2, and it uses one binary summary variable S. The figure shows the process for three time slices. A network like this can be used to model a simple process that has very strong correlation between two variables and requires no more than one bit of information to summarize its state.

stochastic process with a dynamic mixture model that has N components. The components are responsible for summarizing the entire state of the process at time t, including its hidden and observed variables, and propagate it to the consecutive time $t+1$. We call these latent variables, the Summary variables $\mathbf{S}$.

The use of the repeated (template-based) structure and the summary variables ensures that the proposed model can be used to represent any stationary discrete Markovian stochastic process. It also ensures that the number of edges of the resulting model is low, which has the effect of keeping the inference time complexity low. Another factor that plays an important role in keeping the model tractable is exploiting context-specific independence between the summary variables and other random variables.

More formally, a Dynamic Sum-Product Network (D-SPN) is a directed probabilistic graphical model that represents a discrete-time Markov process over a countably infinite collection of hidden $\{X_t : t \in \mathbb{N}\}$, and observed $\{E_t : t \in \mathbb{N}\}$ random variables. Analogously to the DBN, the structure of the D-SPN can be defined using a pair of SPNs $[SPN_{t=0}, SPN_{t>0}]$, where $SPN_{t=0}$ represents the initial state of the D-SPN and $SPN_{t>0}$ represents a template of the evolving process. In addition to the hidden and observed variables, a D-SPN also includes a set of summary variables $\{S_t : t \in \mathbb{N}\}$.

The network structure within a time slice, i.e. the intra-time slice edges, can vary depending on the problem. However, a basic general structure is to condition all the variables at time slice t on the summary variable S_t. This structure simplifies the process and keeps the number of edges low. The same thing also applies to the structure between time slices, i.e. the inter-time slice edges, where all the variables on a time-slice are conditioned on the previous time-slice summary variable S_{t-1}. This structure allows the summary variable to collect all the necessary information about time slice t, and also to broadcast the collected information to the next time slice $t+1$. The following equations summarize the conditional independencies that hold by this structure for the hidden X and the observed E variables:

$$P(X_t|X_{0:t-1,t+1:T}, E_{0:T}, S_{0:T}) = P(X_t|S_{t-1}, S_t) \tag{2}$$

$$P(E_t|X_{0:T}, E_{0:t-1,t+1:T}, S_{0:T}) = P(E_t|S_{t-1}, S_t) \tag{3}$$

Figure 8 shows an example of a Dynamic Sum-Product Network. The figure depicts three time slices for a process that has two binary random variables X_1 and X_2. The summary variable in this example is also binary; thus, it can only hold one bit of information. A D-SPN like this can be used to model a simple process with a strong correlation between its variables.

7.1 Relation to Other Models

D-SPNs are closely related to several dynamic probabilistic models. In particular, D-SPNs are more general than Hidden Markov Models in the sense that they can (1) represent a factored state-space stochastic process, (2) model processes that don't necessarily have natural forward dependencies, and (3) induce dependencies between observations in different time-slices.

D-SPNs also differ from Dynamic Bayesian Network in a number of ways. First, exact inference is tractable in D-SPNs even in high treewidth models. This gives us the ability to build practical models with complex interactions between variables within and among time slices. Second, the nature of SPNs leads to models that have rich latent structure within time slices. In this sense, D-SPNs can be seen as a class of temporal deep learning models. Finally, exploiting determinism and context-specific independence are inherent properties of D-SPNs. That is in contrast to typical learning and inference procedures

Algorithm 2 Learning a D-SPN

Input: Dataset **D** over variables X.
Output: A D-SPN that is learned from **D** and represents a distribution over X.

```
G ← $GenerateGeneralDynamicSPN(X)$
DSPN ← $InitializeWeights(\mathbf{G})$
repeat
    for all $d \in \mathbf{D}$ do
        $\alpha \leftarrow ForwardProbability(d, DSPN)$
        $\beta \leftarrow BackwardProbability(d, DSPN)$
        for all $i, j \in PairOfHiddenStates(DSPN)$ do
            $\gamma \leftarrow ExpectedTransition(i, j, \alpha, \beta, d, DSPN)$
            $DSPN \leftarrow UpdateWeights(\gamma, DSPN)$
        end for
    end for
until convergence
```

for Dynamic Bayesian Networks, which are based on the structure of the model and, usually, lead to intractable models.

The previous differences also apply to dynamic conditional random fields (DCRFs), which are generalizations of linear-chain conditional random fields (CRFs). However, it seems that DCRFs and D-SPNs share the same expressiveness, but not the same complexity.

One of the research directions that we will investigate is the relationship between D-SPNs and DCRFs. Also, more generally, the relationship between D-SPNs and CRFs. The work in this direction will also contribute to a better understanding of SPNs and their expressiveness.

7.2 Learning D-SPNs

This section considers the problem of learning D-SPNs from data. We first assume that the structure of the D-SPN is known and our task is to estimate the parameters using the available data. Then, we discuss an initial idea for learning the structure of D-SPNs. The dataset in this context is a set of independent and identically distributed (i.i.d.) cases, where each case is a full or partial assignment of the random variables for one or more time steps.

We are proposing a general scheme to learn the parameters of D-SPNs when the structure is known. The scheme is based on the Expectation-Maximization algorithm and can be viewed as a generalization of the learning algorithm that was presented in section 6.1. It can also be seen as a special instance of the Baum-Welch algorithm applied to D-SPNs.

Similar to algorithm 1, we start with a general D-SPN structure. Such structure could be based on the one that was described in section 7 (summarized by equations 2 and 3). We, then, initialize the weights of the selected structure. Next, for each sequence of observations $E_{0:T}^{(d)}$, where the superscript (d) indicates the index of the case in the dataset, and T is the length of the case, we compute the probability that the D-SPN, with its current weights, will end up with a specified assignment i for the hidden variables (including the summary variable) at each time step, given the observation sequence $E_{0:T}^{(d)}$. This probability is denoted by $\alpha_t(i)$ and can be computed recursively for $t = 1, 2, ...T$ using:

$$\alpha_t(i) = \begin{cases} P(E_t|X_t = i).P_{SPN_0}(X = i) & t = 0 \\ P(E_t|X_t = i).\sum_{j=1}^{N} \alpha_{t-1}(j)P(X_{t-1} = j|X_t = i, E_t) & t > 0 \end{cases} \tag{4}$$

where $P(E_t|X_t = i)$ is the probability of observing E_t when i is the assignment of the hidden variables, $P_{SPN_0}(X = i)$ is the initial probability for i, N is the combination of all the possible states, and $P(X_{t-1} = j|X_t = i, E_t)$ is the probability of ending with the assignment i for the hidden variables when the assignment of the previous time step is j. Essentially, $\alpha_t(i)$ is the forward-probability of having the assignment i at time t while all the previous assignments are consistent with the observation case $E_{0:T}^{(d)}$. Similarly, we compute the backward-probability of $E_{0:T}^{(d)}$ using:

$$\beta_t(i) = \sum_{j=1}^{N} \beta_{t+1}(j)P(E_{t+1}|X_{t+1} = j)P(X_{t+1} = j|X_t = i, E_t). \tag{5}$$

This gives us the probability that at time t we have the i assignment and the next time step will be consistent with $E_{t+1}^{(d)}$. We can then use the forwards and backwards probabilities to maximize weights of the transition model. Algorithm 2 gives an overview of the proposed scheme.

We now turn to discuss the problem of learning the structure of D-SPNs from data. Adopting the general structure that was described in section 7 helps us with restricting the inter-time slices connections to a smaller set, because the model is defined such that the summary variables are the only interfaces between time slices. However, it still leaves us with three problems:

- The intra-time slice connections: in which we need to decide how the random variables interact with each others within the time slice.
- The number of summary variables: If more than one summary variables are going to be used then a decision needs to be made about how many of them are required to sufficiently collect and propagate information between time slices. This also leads to the next problem.
- The decomposition of the stochastic process into sub-processes: Using more than one summary variable is equivalent to decomposing the stochastic process into multiple sub-processes. In this problem these sub-processes need to be identified, so that dependent subsets of random variables end up together in their own sub-process (i.e. they are sharing the same summary variable)

We are planning to pursue our research in the area of structure learning of D-SPNs in order to solve these three problems. One of the possible solutions is to extend the algorithm that was recently given in [15]. The algorithm tries to recursively learn an SPN from data by approximating the dependencies among the random variables. Our plan is to extend the algorithm by, first, applying independence tests over random variables within time-slices (first problem), then between time slices (second and third problem). A D-SPN structure could then be built according to the results of these tests.

7.3 Bayesian Non-Parametric D-SPN

As the previous example shows, the cardinality of the summary variable $|S|$ determines its capacity to accurately summarize the states of the process. Basically, choosing a low-cardinality means that the model will have less degrees of freedom and, at the learning time, it will try to fit many –maybe unrelated– states together. This in turns will introduce errors that propagate over time. On the other hand, increasing the cardinality will reduce the possibility of such errors, but it will also increase the complexity of the model.

The problem is similar to one of the most common problems in machine learning: the problem of model selection. It arises in Hidden Markov Models, where the right number of hidden states needs to be chosen. It also appears in the k-means algorithm, where the number of clusters k has to be chosen. The problem also appears in mixture models where there is a need to determine the best number of clusters that describe the available data. A simple solution for this issue is to use a method like cross-validation to evaluate models with different configurations and then pick the most suitable model. Another approach is to adopt the Bayesian Non-parametric framework, so that the model can grow with the data.

We are planning to study the ability of augmenting the summary variables even further by defining a Hierarchical Dirichlet Process (HDP) over them. An HDP is a hierarchical Bayesian non-parametric model that can be used to define infinite mixture models, where the mixture components are shared [26]. Another possible research direction in this area is to study the ability to define a Bayesian non-parametric prior over the entire structure of the network, such that the complexity of the D-SPN structure grows with data. Indian Buffet Processes [17] have already been successfully applied to several graphical models in order to make their structures grow with the data (e.g. Infinite Factorial Hidden Markov Models [13] and iDBNs [8]). A similar technique can be used to develop a version of D-SPNs that has infinite structure.

8 Application: Activity Recognition using D-SPNs

Activity recognition is the task of automatically assigning labels of actions performed by an agent to segments of sequential data. Possible actions of interest are: gestures, interactions between people, or activities like walking and sitting. The source of the data can be a set of sensors. Many different types of sensors are used for activity recognition, including: microphones, accelerometers, cameras, etc. Activity recognition has many applications in areas such as surveillance, human-computer interaction, and sports.

To evaluate our proposed model we are going to apply it to the problem of activity recognition. We are planing to use three datasets to benchmark the accuracy, tractability, and inference time of our proposed model. The datasets have different numbers and types of sensors, which will help in testing and studying the characteristics of our models in different settings. The three datasets are:

- Kasteren, et al. Benchmark [27]: The benchmark consists of three datasets and also contains the results of four models: Naive Bayes, Hidden Markov Model, Hidden Semi-Markov Model, and Conditional Random Field. These results are supposed to be used as a baseline for performance with other models. The three datasets are the sensors' readings from three different houses. All the sensors give binary outputs and the data was manually annotated.

- OPPORTUNITY Activity Recognition Benchmark [24]: The dataset consists of readings from 72 different sensors. The data was collected from four subjects that operated in a room simulating a small apartment. The sensors were attached to the subjects' body and the objects in the room, in addition to some other ambient sensors.

- SmartWalker [21] Dataset: The dataset consists from two sensor readings of two experiments that were done using a special walker equipped with several accelerometers and load-cells. The experiments were done in a controlled environment. Video recordings were used to manually annotate the data by aligning the recordings with the sensor readings.

A possible contribution out of the work in this area is to provide a tractable dynamic graphical model that can model complex interactions in sensor-rich environments.

Some of the datasets' readings are continuous values; and there is still no known way to build a tractable SPN over variables with continuous domains. One obvious solution to this issue is to convert these values to discrete intervals using a discretization method like Fayyad&Irani [9]. A research direction that could result in a contribution to both SPNs and D-SPNs is to study the possibility of defining these networks over random variables with continuous domains.

9 Conclusion and Future Plans

Dynamic Sum-Product Networks are extensions of Sum-Product Networks that allow one to model sequential and complex stochastic processes with unbounded network treewidth while keeping the inference tractable. We have proposed a specified structure that augments the implicit hidden variables in Sum-Product Networks by explicitly adding a special type of latent variables that we call summary variables. These variables help in keeping the number of edges low, which in turn keeps the model tractable. We have also presented an algorithm to learn the parameters of the proposed model and discussed a plan to develop a structure learning algorithm.

The following list summarizes the four research directions that we are planning to pursue in the future:

- **Characterizing the relationship between D-SPNs and DCRFs**: Studying and formalizing the relationship between D-SPNs and DCRFs. In particular, we are interested in answering the question of whether they share the same expressive power or not.

- **Structure learning for D-SPNs**: Developing an algorithm to learn the structure of D-SPNs, which involves three sub-tasks: (1) learn the connections between the variables within a time slice, (2) choose the number of summary variables, (3) decompose the process into sub-processes, where each process shares a summary variable.

- **Bayesian Non-Parametric D-SPNs**: Developing a Bayesian Non-Parametric extension of D-SPNs such that the cardinality of the summary variables can grow with the data. A more general extension would allow the entire structure of the D-SPN to also grow with the data.

- **Activity Recognition using D-SPNs**: Apply D-SPNs to a real-life activity monitoring application to show the performance of the proposed model and compare it to other state-of-the-art models and approximation methods.

10 References

References

[1] Kenneth P Baclawski. *Introduction to Probability with R*. CRC Press, 2011.

[2] Craig Boutilier, Nir Friedman, Moises Goldszmidt, and Daphne Koller. Context-specific independence in bayesian networks. In *Proceedings of the Twelfth international conference on Uncertainty in artificial intelligence*, pages 115–123. Morgan Kaufmann Publishers Inc., 1996.

[3] Xavier Boyen and Daphne Koller. Tractable inference for complex stochastic processes. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 33–42. Morgan Kaufmann Publishers Inc., 1998.

[4] Venkat Chandrasekaran, Nathan Srebro, and Prahladh Harsha. Complexity of inference in graphical models. *arXiv preprint arXiv:1206.3240*, 2012.

[5] Gregory F Cooper. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial intelligence*, 42(2):393–405, 1990.

[6] Adnan Darwiche. A differential approach to inference in bayesian networks. *Journal of the ACM (JACM)*, 50(3):280–305, 2003.

[7] Thomas Dean and Keiji Kanazawa. A model for reasoning about persistence and causation. *Computational intelligence*, 5(2):142–150, 1989.

[8] Finale Doshi, David Wingate, Josh Tenenbaum, and Nicholas Roy. Infinite dynamic bayesian networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 913–920, 2011.

[9] Usama Fayyad and Keki Irani. Multi-interval discretization of continuous-valued attributes for classification learning. 1993.

[10] Thomas S Ferguson. A bayesian analysis of some nonparametric problems. *The annals of statistics*, pages 209–230, 1973.

[11] Shai Fine, Yoram Singer, and Naftali Tishby. The hierarchical hidden markov model: Analysis and applications. *Machine learning*, 32(1):41–62, 1998.

[12] Emily B Fox, Erik B Sudderth, Michael I Jordan, and Alan S Willsky. An hdp-hmm for systems with state persistence. In *Proceedings of the 25th international conference on Machine learning*, pages 312–319. ACM, 2008.

[13] Jurgen V Gael, Yee W Teh, and Zoubin Ghahramani. The infinite factorial hidden markov model. In *Advances in Neural Information Processing Systems*, pages 1697–1704, 2008.

[14] Robert Gens and Pedro Domingos. Discriminative learning of sum-product networks. In *Advances in Neural Information Processing Systems*, pages 3248–3256, 2012.

[15] Robert Gens and Pedro Domingos. Learning the structure of sum-product networks. 2013.

[16] Zoubin Ghahramani and Michael I Jordan. Factorial hidden markov models. *Machine learning*, 29(2-3):245–273, 1997.

[17] Thomas Griffiths and Zoubin Ghahramani. Infinite latent feature models and the indian buffet process. 2005.

[18] Daphne Kollar and Nir Friedman. *Probabilistic graphical models: principles and techniques*. The MIT Press, 2009.

[19] Kevin P Murphy, Yair Weiss, and Michael I Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 467–475. Morgan Kaufmann Publishers Inc., 1999.

[20] Kevin Patrick Murphy. *Dynamic bayesian networks: representation, inference and learning*. PhD thesis, University of California, 2002.

[21] Farheen Omar, Mathieu Sinn, Jakub Truszkowski, Pascal Poupart, James Tung, and Allen Caine. Comparative analysis of probabilistic models for activity recognition with an instrumented walker. *arXiv preprint arXiv:1203.3500*, 2012.

[22] Hoifung Poon and Pedro Domingos. Sum-product networks: A new deep architecture. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 689–690. IEEE, 2011.

[23] Lawrence Rabiner and B Juang. An introduction to hidden markov models. *ASSP Magazine, IEEE*, 3(1):4–16, 1986.

[24] Daniel Roggen, Alberto Calatroni, Mirco Rossi, Thomas Holleczek, Kilian Forster, Gerhard Troster, Paul Lukowicz, David Bannach, Gerald Pirkl, Alois Ferscha, et al. Collecting complex activity datasets in highly rich networked sensor environments. In *Networked Sensing Systems (INSS), 2010 Seventh International Conference on*, pages 233–240. IEEE, 2010.

[25] Dan Roth. On the hardness of approximate reasoning. *Artificial Intelligence*, 82(1):273–302, 1996.

[26] Yee Whye Teh, Michael I Jordan, Matthew J Beal, and David M Blei. Hierarchical dirichlet processes. *Journal of the american statistical association*, 101(476), 2006.

[27] TLM van Kasteren, Gwenn Englebienne, and BJA Kröse. Human activity recognition from wireless sensor network data: Benchmark and software. In *Activity Recognition in Pervasive Intelligent Environments*, pages 165–186. Springer, 2011.