Probabilistic Sentential Decision Diagrams: Learning with Massive Logical Constraints

Doga Kisa DOGA@CS.UCLA.EDU
Guy Van den Broeck GUYVDB@CS.UCLA.EDU
Arthur Choi AYCHOI@CS.UCLA.EDU
Adnan Darwiche DARWICHE@CS.UCLA.EDU
University of California, Los Angeles, CA 90095 USA

Abstract

We propose the Probabilistic Sentential Decision Diagram (PSDD): A complete and canonical representation of probability distributions defined over the models of a given propositional theory.[1] Each parameter of a PSDD can be viewed as the (conditional) probability of making a decision in a corresponding Sentential Decision Diagram (SDD). The SDD itself is a recently proposed complete and canonical representation of propositional theories. PSDDs are tractable representations, and further, the parameters of a PSDD can be efficiently estimated, in closed form, from complete data. We empirically evaluate the quality of PSDDs learned from data, when we have knowledge, a priori, of the domain logical constraints.

1. Introduction

The interplay between logic and probability has been of great interest throughout the history of AI. One of the earliest proposals in this direction is Nilsson's probabilistic logic (Nilsson, 1986), which aimed at augmenting first-order logic with probabilities. This has prompted similar approaches, including, for example, Halpern (1990). The focus of these approaches, however, was mainly semantical, yielding no effective schemes for realizing them computationally. More recently, the area of lifted probabilistic inference has tackled this interplay, while employing a different compromise (Poole, 2003). In these efforts, the focus has been mostly on restricted forms of first-order logic (e.g., function-free and finite domain), but with the added advantage of efficient inference (e.g., algorithms whose complexity is polynomial in the domain size).

On the propositional side, the thrust of the interplay has been largely computational. An influential development in this direction has been the realization that enforcing certain properties on propositional representations, such as decomposability and determinism, provides one with the power to answer probabilistic queries efficiently. This development was actually based on two technical observations. First, that decomposable and deterministic representations allow one to perform weighted model counting efficiently. Second, that probabilistic reasoning can be reduced to weighted model counting. This development, which has its first roots in Darwiche (2002), has been underlying an increasing number of probabilistic reasoning systems in the last decade. This is especially true for representations that employ both logical and probabilistic elements (e.g., Chavira et al. (2006) and Fierens et al. (2011)). Moreover, the technique has been extended recently to certain first-order representations as well (Van den Broeck et al., 2011).

This paper is concerned with an orthogonal contribution to this interplay between propositional logic and probability theory. The problem we tackle here is that of developing a representation of probability distributions in the presence of massive, logical constraints. That is, given a propositional logic theory which represents domain constraints, our goal is to develop a representation that induces a unique probability distribution over the models of the given theory. Moreover, the proposed representation should satisfy requirements that are sometimes viewed as necessary for the practical employment of such representations.

[1] This is an updated version of a paper to appear in the Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR) (Kisa et al., 2014). This paper is slightly abbreviated, but provides a new set of experimental results. The full version of the paper, with proofs, is available at http://reasoning.cs.ucla.edu/, under "publications"

L	K	P	A	Students
0	0	1	0	6
0	0	1	1	54
0	1	1	1	10
1	0	0	0	5
1	0	1	0	1
1	0	1	1	0
1	1	0	0	13
1	1	1	0	8
1	1	1	1	3

Table 1. Student enrollment data.

These include a clear semantics of the representation parameters; an ability to reason with the representation efficiently; and an ability to learn its parameters from data, also efficiently.

Our proposal is called a *Probabilistic Sentential Decision Diagram (PSDD).* It is based on the recently proposed *Sentential Decision Diagram (SDD)* for representing propositional theories (Darwiche, 2011; Xue et al., 2012; Choi & Darwiche, 2013). While the SDD is comprised of *logical* decision nodes, the PSDD is comprised of *probabilistic* decision nodes, which are induced by supplying a distribution over the branches of a logical decision node. Similar to SDDs, the PSDD is a canonical representation, but under somewhat more interesting conditions. Moreover, computing the probability of a term can be done in time linear in the PSDD size. In fact, the probability of each and every literal can be computed in only two passes over the PSDD. It is particularly notable that the local parameters of a PSDD have clear semantics with respect to the global distribution induced by the PSDD. We will also show that these parameters can be learned efficiently from complete data.

This paper is structured as follows. We start by a concrete discussion on some of the applications that have driven the development of PSDDs and follow by an intuitive exposure of PSDDs and their salient features. We next provide a formal treatment of the syntax, semantics and properties of PSDDs. This allows us to present our algorithm for learning PSDD parameters from complete data. The paper concludes with some experimental results showing the promise of PSDDs in learning probability distributions under logical constraints. Additional discussions and experiments appear in the original version of this paper (Kisa et al., 2014) (to be presented at KR 2014), including a discussion on related work.

2. Motivation

PSDDs were inspired by the need to learn probability distributions that are subject to domain constraints. Take for example a computer science department that organizes four courses: Logic (L), Knowledge Representation (K), Probability (P), and Artificial Intelligence (A). Students are asked to enroll for these courses under the following restrictions: (1) a student must take at least one of Probability or Logic, (2) probability is a prerequisite for AI, and (3) the prerequisite for KR is either AI or Logic.

The department may have data on student enrollments, as in Table 1, and may wish to learn a probabilistic model for reasoning about student preferences. For example, the department may need to know whether students are more likely to satisfy the prerequisite of KR using AI or using Logic.

A mainstream approach for addressing this problem is to learn a probabilistic graphical model, such as a Bayesian network. In this case, a network structure is constructed manually or learned from data. The structure is then turned into a Bayesian network by learning its parameters from the data. Other graphical models can also be used. This includes, for example, Markov networks or their variations.

What is common among all these approaches is that they lack a principled and effective method for accommodating the domain constraints into the learning process—that is, ensuring, for example, that a student with a profile $A \wedge K \wedge L \wedge \neg P$, or a profile $\neg A \wedge K \wedge \neg L \wedge P$, has zero probability in the learned model. In principle, the zero parameters of a graphical model can capture logical constraints, although a fixed model structure will not in general accommodate arbitrary logical constraints. We could introduce additional structure into the model to capture such constraints, using, e.g., the method of virtual evidence (Pearl, 1988; Mateescu & Dechter, 2008). However, incorporating constraints in this manner will in general lead to a highly-connected network, making inference intractable. Even if inference remained tractable, such an approach is not ideal as we now have to learn a distribution that is conditioned on the constraints. This would require new learning algorithms (e.g., gradient methods) for performing parameter estimation as traditional methods may no longer be applicable. For example, in Bayesian networks, the closed-form parameter estimation algorithm under complete data will no longer be valid in this case.

The domain constraints of our example can be ex-

pressed using the following CNF.

$$\begin{aligned} P \vee L \\ A \Rightarrow P \\ K \Rightarrow A \vee L \end{aligned} \tag{1}$$

Even though there are 16 combinations of courses, the CNF says that only 9 of them are valid choices. An approach that observes this information must learn a probability distribution that assigns a zero probability to every combination that is not allowed by these constraints.

None of the standard learning approaches we are familiar with has been posed to address this problem. The complication here is not strictly with the learning approaches, but with the probabilistic models that are amenable to being learned under these circumstances. In particular, these models are not meant to induce probability distributions that respect a given set of logical constraints.

The simple problem we posed in this section is exemplary of many real-world applications. We mention in particular *configuration* problems that arise when purchasing products, such as cars and computers. These applications give users the option to configure products, but subject to certain constraints. Data is abundant for these applications and there is a clear economic interest in learning probabilistic models under the given constraints. We also mention *reasoning about physical systems,* which includes verification and diagnosis applications. Here, propositional logic is typically used to encode some system functionality, while leaving out some system behaviors which may have a non-deterministic nature (e.g., component failures and probabilistic transitions). There is also an interest here to learn probabilistic models of these systems, subject to the given constraints.

Our goal in this paper is to introduce the PSDD representation for addressing this particular need. We will start by an intuitive (and somewhat informal) introduction to PSDDs, followed by a more formal treatment of their syntax, semantics and the associated reasoning and learning algorithms.

3. PSDDs

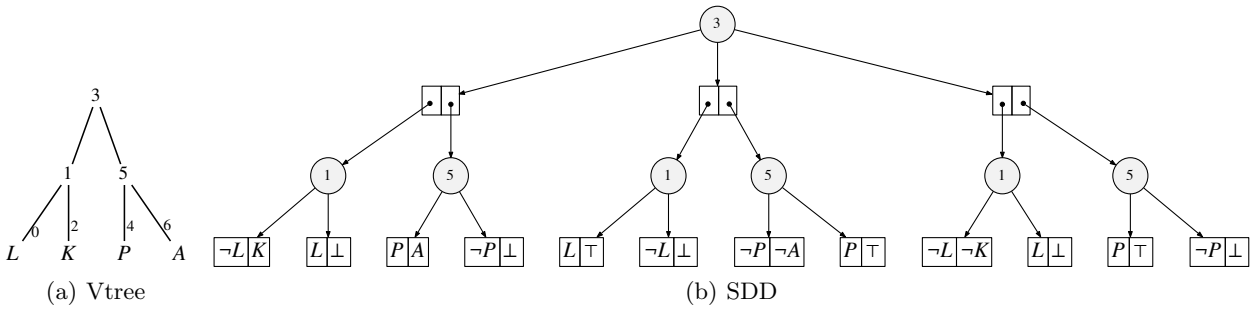
We will refer to domain constraints as the *base* of a probability distribution. Our proposed approach starts by representing this base as a *Sentential Decision Diagram (SDD)* as in Figure 1 (Darwiche, 2011; Xue et al., 2012; Choi & Darwiche, 2013). An SDD is determined by a *vtree,* which is a full binary tree with leaves corresponding to the domain variables (Pipatsrisawat & Darwiche, 2008). The choice of a particular SDD can then be thought of as a choice of a particular vtree. We will later discuss the impact of this choice on the represented distribution. For now, however, we will develop some further understanding of SDDs as they are the backbones of our probability distributions.

SDDs. An SDD is either a *decision node* or a *terminal node.* A terminal node is a literal, the constant $\top$ (true) or the constant $\bot$ (false). A decision node is a disjunction of the form $(p_1 \wedge s_1) \vee \ldots \vee (p_n \wedge s_n)$, where each pair (p_i, s_i) is called an *element.* A decision node is depicted by a circle and its elements are depicted by paired boxes. Here, $p_1, \ldots, p_n$ are called *primes* and $s_1, \ldots, s_n$ are called *subs.* Primes and subs are themselves SDDs. Moreover, the primes of a decision node are always consistent, mutually exclusive and exhaustive. The SDD in Figure 1 has seven decision nodes. The decision node to the far left has two elements $(\neg L, K)$ and $(L, \bot)$. It represents $(\neg L \wedge K) \vee (L \wedge \bot)$, which is equivalent to $\neg L \wedge K$. There are two primes for this node $\neg L$ and L. The two corresponding subs are K and $\bot$.

Structure. An SDD can be viewed as a *structure* that induces infinitely many probability distributions (all having the same base). By *parameterizing* an SDD, one obtains a PSDD that induces a particular probability distribution.

Parameters. Figure 2 depicts a PSDD which is obtained by parameterizing the SDD in Figure 1. Both decision and terminal SDD nodes are parameterized, but we focus here on decision nodes. Let n be a decision node having elements $(p_1, s_1), \ldots, (p_n, s_n)$. To parameterize node n is to provide a distribution $\theta_1, \ldots, \theta_n$. Intuitively, θ_i is the probability of prime p_i given that the decision of node n has been implied. We will formalize and prove this semantics later. We will also provide an efficient procedure for learning the parameters of a PSDD from complete data. The PSDD parameters in Figure 2 were learned using this procedure from the data in Table 2. The table also depicts the probability distribution induced by the learned PSDD.

Independence. The PSDD structure is analogous to a Bayesian network structure in the following sense. The latter can be parameterized in infinitely many ways, with each parameterization inducing a particular probability distribution. Moreover, all the induced distributions satisfy certain independences that can be inferred from the underlying Bayesian network structure. The same is true for PSDDs. Each parameterization of a PSDD structure yields a unique probability distribution. Moreover, all the induced distributions



(a) Vtree (b) SDD

Figure 1. A vtree and SDD for the student enrollment problem. Numbers in circles correspond to vtree nodes.

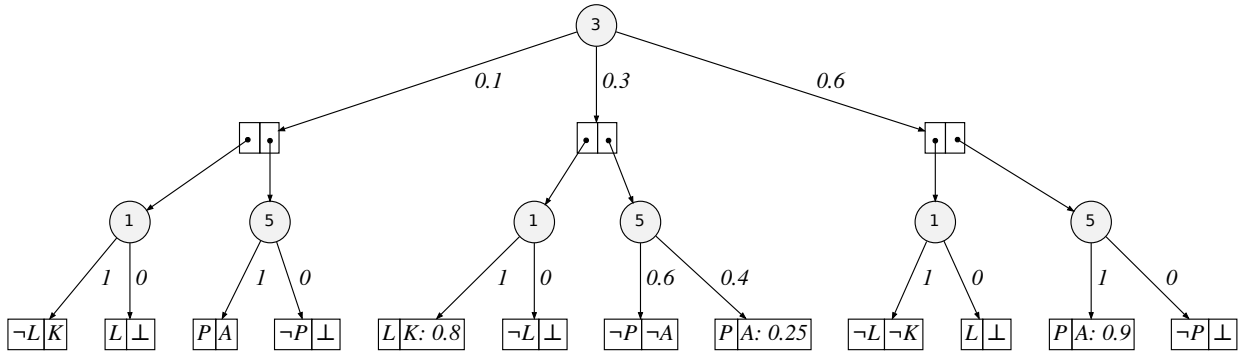


Figure 2. A PSDD for the student enrollment problem, which results from parameterizing the SDD in Figure 1. The parameters were learned from the dataset in Table 1 (also shown in Table 2).

L	K	P	A	Students	Learned PSDD Distribution	
0	0	1	0	6	$0.6 \cdot 0.1$	6.0%
0	0	1	1	54	$0.6 \cdot 0.9$	54.0%
0	1	1	1	10	0.1	10.0%
1	0	0	0	5	$0.3 \cdot 0.2 \cdot 0.6$	3.6%
1	0	1	0	1	$0.3 \cdot 0.2 \cdot 0.4 \cdot 0.75$	1.8%
1	0	1	1	0	$0.3 \cdot 0.2 \cdot 0.4 \cdot 0.25$	0.6%
1	1	0	0	13	$0.3 \cdot 0.8 \cdot 0.6$	14.4%
1	1	1	0	8	$0.3 \cdot 0.8 \cdot 0.4 \cdot 0.75$	7.2%
1	1	1	1	3	$0.3 \cdot 0.8 \cdot 0.4 \cdot 0.25$	2.4%

Table 2. Student enrollment data and learned distribution.

satisfy independences that can be inferred from the PSDD structure. We showed in (Kisa et al., 2014), however, that PSDD independence is more refined than Bayesian network independence as it allows one to express more qualified independence statements.

4. The Syntax and Semantics of PSDDs

PSDDs are based on *normalized* SDDs in which every node n is associated with (normalized for) a vtree node v according to the following rules (Darwiche, 2011). If n is a terminal node, then v is a leaf node which contains the variable of n (if any). If n is a decision node, then its primes (subs) are normalized for the left (right) child of v. If n is the root SDD node, then v is the root vtree node.

The SDD in Figure 1 is normalized. Each decision node in this SDD is labeled with the vtree node it is normalized for. We are now ready to define the syntax of a PSDD.

Definition 1 (PSDD Syntax) *A PSDD is a normalized SDD with the following parameters.*

- *For each decision node* $(p_1, s_1), \ldots, (p_k, s_k)$ *and prime* p_i*, a positive parameter* θ_i *is supplied such that* $\theta_1 + \ldots + \theta_k = 1$ *and* $\theta_i = 0$ *iff* $s_i = \bot$.
- *For each terminal node* $\top$*, a positive parameter* θ *is supplied such that* $0 < \theta < 1$.

A terminal node $\top$ with parameter θ will be denoted by $X\!:\!\theta$, where X is the variable of leaf vtree node that $\top$ is normalized for. Other terminal nodes (i.e., $\bot$, X and $\neg X$) have fixed, implicit parameters (discussed later) and will be denoted as is. A decision PSDD node will be denoted by $(p_1, s_1, \theta_1), \ldots, (p_k, s_k, \theta_k)$. Graphically, we will just annotate the edge into element (p_i, s_i) with

$\mathbf{x}$	$\mathbf{y}$	$\mathsf{Pr}_{p_i}(\mathbf{x})$	$\mathsf{Pr}_{s_i}(\mathbf{y})$	θ_i	$\mathsf{Pr}_n(\mathbf{xy})$
P	A	1	0.25	0.4	0.1
P	$\neg A$	1	0.75	0.4	0.3
$\neg P$	A	1	0	0.6	0.0
$\neg P$	$\neg A$	1	1	0.6	0.6

Table 3. Distribution of node $n = (\neg P, \neg A)(P, \top)$.

the parameter θ_i. Figure 2 provides examples of this notation.

We next define the distribution of a PSDD, inductively. That is, we first define the distribution induced by a terminal node. We then define the distribution of a decision node in terms of the distributions induced by its primes and subs.

Definition 2 (PSDD Distribution) *Let n be a PSDD node that is normalized for vtree node v. Node n defines a distribution Pr_n over the variables of vtree v as follows. If n is a terminal node, and v has variable X, then*

n	$\mathsf{Pr}_n(X)$	$\mathsf{Pr}_n(\neg X)$
$X : \theta$	θ	$1 - \theta$
$\bot$	0	0
X	1	0
$\neg X$	0	1

If n is a decision node $(p_1, s_1, \theta_1), \ldots, (p_k, s_k, \theta_k)$ and v has left variables $\mathbf{X}$ and right variables $\mathbf{Y}$, then

$$\mathsf{Pr}_n(\mathbf{xy}) \stackrel{def}{=} \mathsf{Pr}_{p_i}(\mathbf{x}) \cdot \mathsf{Pr}_{s_i}(\mathbf{y}) \cdot \theta_i$$

for i where $\mathbf{x} \models p_i$.

Applying this definition to the PSDD of Figure 2 leads to the distribution in Table 2 for its root node. The following table depicts the distribution induced by a non-root node in this PSDD, which appears in the middle of Figure 2.

The SDD node associated with a PSDD node n is called the *base* of n and is denoted by $[n]$. When there is no ambiguity, we will often not distinguish between a PSDD node n and its base $[n]$.

A PSDD assigns a strictly positive probability to a variable instantiation iff the instantiation satisfies its base. This can be seen, for example, in Table 3. This is also the first key property of PSDDs.

Theorem 1 (Base) *Consider a PSDD node n that is normalized for vtree node v. If $\mathbf{Z}$ are the variables of vtree v, then $\mathsf{Pr}_n(\mathbf{z}) > 0$ iff $\mathbf{z} \models [n]$.*

We will now discuss the second key property of PSDDs, which reveals the *local* semantics of PSDD parameters.

Theorem 2 (Parameter Semantics) *Let n be a decision node $(p_1, s_1, \theta_1), \ldots, (p_k, s_k, \theta_k)$. We have $\theta_i = \mathsf{Pr}_n([p_i])$.*

Consider the PSDD in Figure 2 and its decision node n in Table 3. Prime $\neg P$ of this node has parameter 0.6. According to Theorem 2, we must then have $\mathsf{Pr}_n(\neg P) = 0.6$, which can be verified in Table 3. Similarly, $\mathsf{Pr}_n(P) = 0.4$.

The third key property of PSDDs is the relationship between the local distributions induced by its various nodes (node distributions) and the global distribution induced by its root node (PSDD distribution)—for example, the relationship between the distribution of node n in Table 3 and the PSDD distribution given in Table 2.

Node distributions are linked to the PSDD distribution by the notion of *context.*

Definition 3 (Context) *Let $(p_1, s_1), \ldots, (p_k, s_k)$ be the elements appearing on some path from the SDD root to node n.*[2] *Then $p_1 \wedge \ldots \wedge p_k$ is called a sub-context for node n and is feasible iff $s_i \neq \bot$. The context is a disjunction of all sub-contexts and is feasible iff some sub-context is feasible.*

Consider Figure 1. The three decision nodes normalized for vtree node $v = 5$ have the contexts $\neg L \wedge K$, L and $\neg L \wedge \neg K$. Moreover, the terminal nodes normalized for vtree $v = 6$ have the contexts:

- A: $\neg L \wedge K \wedge P$
- $\neg A$: $L \wedge \neg P$
- $\bot$: $(\neg L \wedge K \wedge \neg P) \vee (\neg L \wedge \neg K \wedge \neg P) = (\neg L \wedge \neg P)$
- $\top$: $(L \wedge P) \vee (\neg L \wedge \neg K \wedge P) = (L \vee \neg K) \wedge P$.

Contexts satisfy interesting properties.

Theorem 3 (Context) *A node is implied by its context and the underlying SDD. Nodes normalized for the same vtree node have mutually exclusive and exhaustive contexts. The sub-contexts of a node are mutually exclusive. A context/sub-context is feasible iff it has a strictly positive probability.*

Contexts give a global semantics to node distributions.

[2] That is, $n = p_k$ or $n = s_k$.

Theorem 4 (Node Distribution) *Consider a PSDD r and let n be one of its nodes. If γ_n is a feasible sub-context or feasible context of node n, then* $\mathsf{Pr}_n(.) = \mathsf{Pr}_r(. \mid \gamma_n)$.

Contexts give a *global* semantics to parameters.

Corollary 1 (Parameter Semantics) *Consider a PSDD r and node n with feasible sub-context or feasible context γ_n.*

- *If n is a terminal node $X{:}\theta$, then $\theta = \mathsf{Pr}_r(X \mid \gamma_n)$.*
- *If n is a decision node $(p_1, s_1, \theta_1), \ldots, (p_k, s_k, \theta_k)$, then $\theta_i = \mathsf{Pr}_r([p_i] \mid \gamma_n)$ for $i = 1, \ldots, k$.*

This corollary says that the parameters of a node are conditional probabilities of the PSDD distribution.

In (Kisa et al., 2014), we show that PSDDs are *complete* as they are capable of representing any probability distribution. We also show that PSDDs are *canonical* under a condition known as compression. More precisely, we show that there is a unique compressed PSDD for each distribution and vtree. This is particularly important for learning PSDDs (structure and parameters) as it reduces the problem of searching for a PSDD into the problem of searching for a vtree.

5. Learning with PSDDs

In the original paper (Kisa et al., 2014), we further discussed the conditional independencies of PSDDs, and algorithms for reasoning with them. In particular, given a PSDD r and an instantiation $\mathbf{e}$ of some variables (*evidence*), we provided an algorithm for computing the probability of this evidence $\mathsf{Pr}_r(\mathbf{e})$. We also presented an algorithm for computing the conditional probability $\mathsf{Pr}_r(X \mid \mathbf{e})$ for every variable X. Both algorithms run in time which is linear in the PSDD size (and hence, PSDDs are a tractable representation).

We now present an algorithm for learning the parameters of a PSDD from a *complete* dataset. We start first with some basic definitions. An instantiation of all variables is called an *example*. There are 2^n distinct examples over n propositional variables. A complete dataset is a multi-set of examples.[3] That is, an example may appear multiple times in a dataset. Given a PSDD structure (a normalized SDD), and a complete dataset, our goal is to learn the value of each PSDD parameter. More precisely, we wish to learn *maximum likelihood* parameters: ones that maximize the probability of examples in the dataset.

[3] In an *incomplete* dataset, an example corresponds to an instantiation of some variables (not necessarily all).

We will use Pr_θ to denote the distribution induced by the PSDD structure and parameters θ. The *likelihood* of these parameters given dataset $\mathbf{D}$ is defined as

$$L(\theta|\mathbf{D}) = \prod_{i=1}^{N} \mathsf{Pr}_\theta(\mathbf{d}_i),$$

where $\mathbf{d}_i$ ranges over all N examples in dataset $\mathbf{D}$. Our goal is to find the maximum likelihood parameters

$$\theta^{ml} = \operatorname*{argmax}_\theta L(\theta|\mathbf{D}).$$

We will use $\mathbf{D}\#(\alpha)$ to denote the number of examples in dataset $\mathbf{D}$ that satisfy propositional sentence α. For a decision node $n = (p_1, s_1, \theta_1), ..., (p_k, s_k, \theta_k)$ with context γ_n, we propose the following estimate for parameter θ_i:

$$\theta_i^{ml} = \frac{\mathbf{D}\#(p_i, \gamma_n)}{\mathbf{D}\#(\gamma_n)}. \tag{2}$$

For terminal node $n = X{:}\theta$ with context γ_n, we propose the following estimate for parameter θ:

$$\theta^{ml} = \frac{\mathbf{D}\#(X, \gamma_n)}{\mathbf{D}\#(\gamma_n)}. \tag{3}$$

We can now show the following.

Theorem 5 *The parameter estimates of Equations 2 and 3 are the only estimates that maximize the likelihood function.*

Our parameter estimates admit a *closed-form*, in terms of the counts $\mathbf{D}\#(\alpha)$ in the data. One can compute these estimates using a single pass through the examples of a dataset. Moreover, each distinct example can be processed in time linear in the PSDD size.[4] These are very desirable properties for a parameter learning algorithm. These properties are shared with Bayesian network representations, but are missing from many others, including Markov networks.

When learning probabilistic graphical models, one makes a key distinction between learning structures

[4] A dataset may not include every example that is consistent with the domain constraints. If this is the case, a parameter θ for prime p may be estimated to zero, even though its sub s may not be $\bot$; see Theorem 1. To address this, one can assume a pseudo-count for each distinct example, which can be thought of as providing a prior distribution on parameters. In our experiments, we assumed a pseudo-count of $2/\mathsf{mc}$ for each distinct example, where mc is the model count of the SDD. This corresponds to a very weak prior since, in aggregate, these pseudo-counts contribute a total count that is equivalent to two real examples in the dataset.

versus learning parameters (the former being harder in general). While learning PSDD structures is beyond the scope of this paper, the experimental results we present next do use a basic method for learning structures. In particular, since we compile logical constraints into an SDD (i.e., a PSDD structure), the compilation technique we use is effectively "learning" a structure. We used the publicly available SDD package for this purpose (`http://reasoning.cs.ucla.edu/sdd/`). The SDD package tries to dynamically minimize the size of the compiled SDD and, as a result, tries to minimize the number of PSDD parameters.

6. Empirical Evaluation

In this Section, we empirically evaluate our parameter estimation algorithm for PSDDs. This section presents extended results, compared to the preliminary empirical results presented in a paper to be presented at KR 2014 (Kisa et al., 2014). In particular, our goal here is to illustrate how much the quality of a probabilistic model, learned from data, can be impacted by the ability to exploit background knowledge in the form of domain constraints. More specifically, we illustrate how we can learn PSDDs that represent the data better, compared to other approaches that are unable to incorporate such background knowledge.

Here, we simulate datasets from suites of highly-deterministic Bayesian networks, which can be viewed as already encoding domain constraints. For example, many Bayesian networks used in practice are constructed based on domain knowledge, using domain experts. The zero parameters that appear in such networks can further be assumed to be based on known domain constraints (i.e., the domain expert who constructed such a network would only assign a probability of zero to an event, only if that event were impossible, as opposed to just being unlikely).

Using such benchmarks, we simulate datasets of increasing size, from which we will learn a variety of probabilistic graphical models. In the case of PSDDs, we exploit the given domain constraints that we described above (in all cases, we discard the original structure used to simulate the datasets). We can extract the propositional theory represented by the zeros of the Bayesian network as a CNF,[5] which we in turn compile to an SDD. This SDD is further dynamically minimized, which reduces its size, and hence, reduces the number of parameters in the corresponding PSDD. We compare the PSDDs that we learn with other probabilistic models: Chow-Liu trees (CLTs), Aritmetic Circuits for Bayesian Networks (ACBNs) (Lowd & Domingos, 2008), and Sum-Product Networks (SPNs) (Gens & Domingos, 2013).[6] ACBNs and SPNs, in particular, are recently proposed representations, with publicly available implementations. However, these systems are not equipped to take advantage of the domain knowledge that PSDDs can. Our goal is to highlight that this is indeed an advantage one would want to exploit, when such background knowledge is available.

The first set of benchmarks we considered were taken from the UAI-08 inference evaluation.[7] These included `blockmap` and `students` networks from the relational benchmarks, and `90-30` networks from the grids benchmarks. All of these benchmarks are notable because they have a high degree of determinism (roughly 90% of parameters in `students` and `90-30` networks are zero, and roughly 99% are zero in `blockmap`). Figure 3 highlights the results, of learning PSDDs (with the corresponding domain knowledge), and CLTs, ACBNs, and SPNs (which can not exploit the domain knowledge). Each plot point represents an average of k different networks, with 10 datasets simulated from each (3 networks for `blockmap`, 1 for `students`, and 10 for `90-30`). Training and testing sets were simulated indepedently, each of the same size. As we can see, the ability of PSDDs to exploit domain knowledge is significant here, as it can learn much more accurate models (higher test-set likelihoods), even when there is a very small amount of data (the smallest data sets considered were of size 128). These results suggest that when there is a significant amount of domain constraints (as was the case in these benchmarks), then PSDDs could be a compelling option, compared to alternatives that do not exploit such knowledge.

Figure 4 presents results for another set of benchmarks. These benchmarks correspond to real-world Bayesian networks, where we injected determinism at random, in order to simulate high amounts of domain constraints. In particular, we iterated over every CPT column (for non-root nodes), and made it 0/1 with probability 0.9 (we selected the one value at random). Here, each plot point represents an average of (at least)

[5] For each zero parameter $\theta_{x|\mathbf{u}}$ in the Bayesian network, we have in our CNF a clause that is found by negating the instantiation $x\mathbf{u}$.

[6] CLTs and ACBNs were learned using the Libra library available at `http://libra.cs.uoregon.edu/`. SPNs were learned using the SPN package available at `http://spn.cs.washington.edu/learnspn/`. The Libra library allow for a limit on the number of edges used for ACBNs. The number of edges in an SDD's corresponding AC is at most 3 times the size of the SDD, hence we enforced this limit on ACBNs, for a fairer comparison.

[7] At `http://graphmod.ics.uci.edu/uai08/`

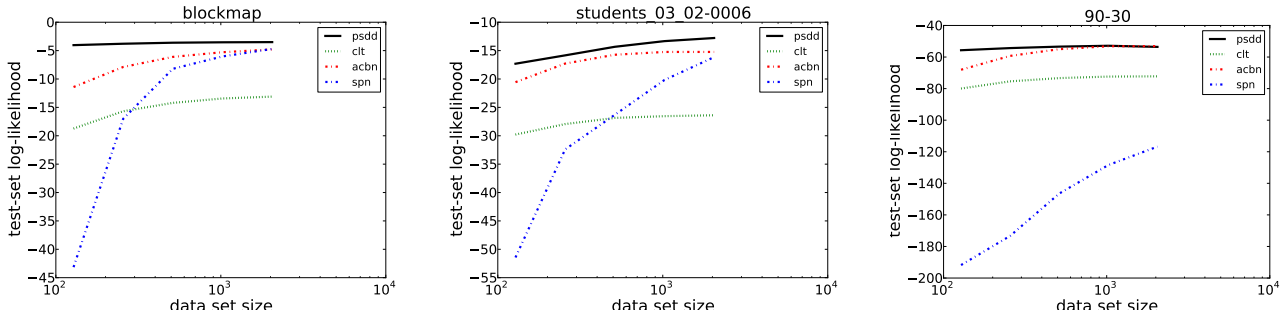


Figure 3. We observe the increasing quality of different models learned (y-axis), as we increase dataset size (x-axis).

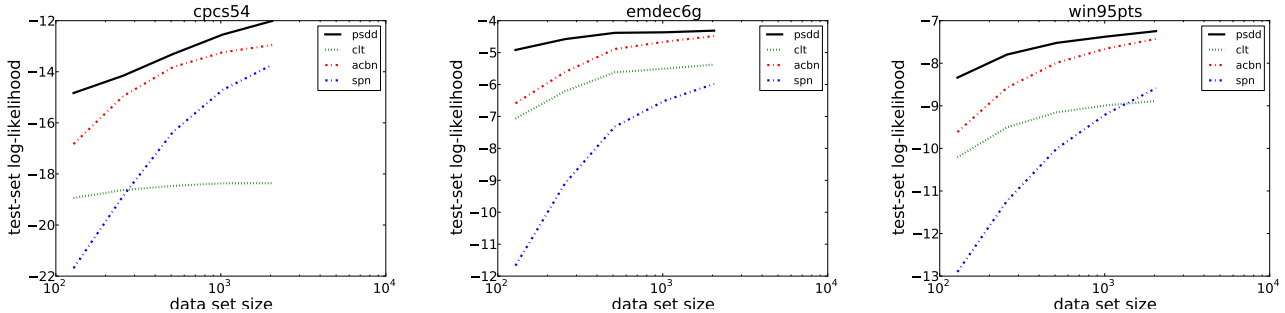


Figure 4. We observe the increasing quality of different models learned (y-axis), as we increase dataset size (x-axis).

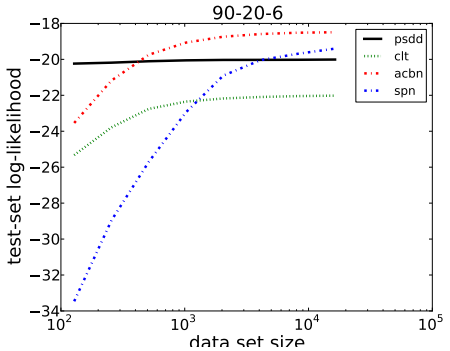


Figure 5. We observe the increasing quality of models learned (y-axis), as we increase dataset size (x-axis).

8 networks constructed by injecting determinism at random, with 10 datasets simulated from each. We see a similar story here, as before.

Finally, consider Figure 5, where we selected a benchmark from the `grids` suite, and learned networks from a wider range of dataset sizes (from sizes 128 to 16,384). Here, we see that PSDDs have an advantage with smaller data set sizes. However, for this particular example, we see at larger data set sizes that the underlying SDD is not expressive enough to capture this distribution (the likelihood stops increasing). While the PSDD parameters are learned from the data, we fix the structure of the underlying SDD (which is constructed once from the domain constraints). Fixing the structure makes certain assumptions about the conditional independencies of the PSDD distribution (which may not be reflected in the data). In contrast, ACBNs and SPNs perform structure learning, and hence they learn better models given enough data, in this case. However, as PSDDs are canonical representations (and hence can represent any probability distribution), there is still a significant opportunity for learning the structure of PSDDs, even when there is not a large amount of domain constraints.

7. Conclusion

We presented the PSDD as a representation of probability distributions that respect a given propositional theory. The PSDD is a complete and canonical representation, with parameters that are interpretable as conditional probabilities. The PSDD encodes context-specific independences, which can be derived from its structure. The PSDD is a tractable representation, allowing one to compute the probability of any term in time linear in its size. The PSDD has unique maximum likelihood parameters under complete data, which can be learned efficiently using a closed form. Preliminary experimental results suggest that the PSDD can be quite effective in learning distributions under domain constraints.

References

Bozga, Marius and Maler, Oded. On the representation of probabilities over structured domains. In *Computer Aided Verification*, pp. 261–273. Springer, 1999.

Chang, Ming-Wei, Ratinov, Lev-Arie, Rizzolo, Nicholas, and Roth, Dan. Learning and inference with constraints. In *AAAI*, pp. 1513–1518, 2008.

Chavira, Mark, Darwiche, Adnan, and Jaeger, Manfred. Compiling relational Bayesian networks for exact inference. *International Journal of Approximate Reasoning*, 42(1-2):4–20, 2006.

Choi, Arthur and Darwiche, Adnan. Dynamic minimization of sentential decision diagrams. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence (AAAI)*, pp. 187–194, 2013.

Darwiche, Adnan. A logical approach to factoring belief networks. In *Proceedings of KR*, pp. 409–420, 2002.

Darwiche, Adnan. SDD: A new canonical representation of propositional knowledge bases. In *IJCAI*, pp. 819–826, 2011.

De Raedt, Luc, Frasconi, Paolo, Kersting, Kristian, and Muggleton, Stephen (eds.). *Probabilistic inductive logic programming: theory and applications*. 2008.

Fierens, Daan, Van den Broeck, Guy, Thon, Ingo, Gutmann, Bernd, and De Raedt, Luc. Inference in probabilistic logic programs using weighted CNF's. In *Proceedings of UAI*, pp. 211–220, 2011.

Gens, Robert and Domingos, Pedro. Learning the structure of sum-product networks. In *ICML*, pp. 873–880, 2013.

Getoor, L. and Taskar, B. (eds.). *An Introduction to Statistical Relational Learning*. MIT Press, 2007.

Gogate, Vibhav and Dechter, Rina. SampleSearch: A scheme that searches for consistent samples. In *International Conference on Artificial Intelligence and Statistics*, pp. 147–154, 2007.

Halpern, J.Y. An analysis of first-order logics of probability. *Artificial Intelligence*, 46(3):311–350, 1990.

Jaeger, Manfred. Probabilistic decision graphs—combining verification and AI techniques for probabilistic inference. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 12:19–42, 2004.

Jaeger, Manfred, Nielsen, Jens D, and Silander, Tomi. Learning probabilistic decision graphs. *International Journal of Approximate Reasoning*, 42(1):84–100, 2006.

Kisa, Doga, den Broeck, Guy Van, Choi, Arthur, and Darwiche, Adnan. Probabilistic sentential decision diagrams. In *Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, 2014.

Lai, Y-T and Sastry, Sarma. Edge-valued binary decision diagrams for multi-level hierarchical verification. In *Proceedings of the 29th ACM/IEEE Design Automation Conference*, pp. 608–613, 1992.

Lowd, Daniel and Domingos, Pedro. Learning arithmetic circuits. In *UAI*, pp. 383–392, 2008.

Mateescu, Robert and Dechter, Rina. Mixed deterministic and probabilistic networks. *Annals of Mathematics and Artificial Intelligence*, 54(1-3):3–51, 2008.

Nilsson, N.J. Probabilistic logic. *Artificial intelligence*, 28 (1):71–87, 1986.

Pearl, Judea. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, Inc., San Mateo, California, 1988.

Pipatsrisawat, Knot and Darwiche, Adnan. New compilation languages based on structured decomposability. In *AAAI*, pp. 517–522, 2008.

Poole, David. First-order probabilistic inference. In *Proceedings of IJCAI*, pp. 985–991, 2003.

Poon, Hoifung and Domingos, Pedro. Sound and efficient inference with probabilistic and deterministic dependencies. In *AAAI*, volume 6, pp. 458–463, 2006.

Poon, Hoifung and Domingos, Pedro. Sum-product networks: A new deep architecture. *Proc. 12th Conf. on Uncertainty in Artificial Intelligence*, pp. 337–346, 2011.

R.I. Bahar, E.A. Frohm, C.M. Gaona, G.D. Hachtel, E. Macii, A. Pardo, and F. Somenzi. Algebraic Decision Diagrams and Their Applications. In *IEEE /ACM International Conference on CAD*, pp. 188–191, 1993.

Richardson, M. and Domingos, P. Markov logic networks. *Machine learning*, 62(1):107–136, 2006.

Sanner, Scott and McAllester, David A. Affine algebraic decision diagrams (AADDs) and their application to structured probabilistic inference. In *Proc. International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1384–1390, 2005.

Van den Broeck, Guy, Taghipour, Nima, Meert, Wannes, Davis, Jesse, and De Raedt, Luc. Lifted probabilistic inference by first-order knowledge compilation. In *Proceedings of IJCAI*, pp. 2178–2185, 2011.

Xue, Yexiang, Choi, Arthur, and Darwiche, Adnan. Basing decisions on sentences in decision diagrams. In *AAAI*, pp. 842–849, 2012.