The Sum-Product Theorem: A Foundation for Learning Tractable Models

Abram L. Friesen AFRIESEN@CS.WASHINGTON.EDU
Pedro Domingos PEDROD@CS.WASHINGTON.EDU
Department of Computer Science and Engineering, University of Washington, Seattle, WA 98195 USA

Abstract

Inference in expressive probabilistic models is generally intractable, which makes them difficult to learn and limits their applicability. Sum-product networks are a class of deep models where, surprisingly, inference remains tractable even when an arbitrary number of hidden layers are present. In this paper, we generalize this result to a much broader set of learning problems: all those where inference consists of summing a function over a semiring. This includes satisfiability, constraint satisfaction, optimization, integration, and others. In any semiring, for summation to be tractable it suffices that the factors of every product have disjoint scopes. This unifies and extends many previous results in the literature. Enforcing this condition at learning time thus ensures that the learned models are tractable. We illustrate the power and generality of this approach by applying it to a new type of structured prediction problem: learning a nonconvex function that can be globally optimized in polynomial time. We show empirically that this greatly outperforms the standard approach of learning without regard to the cost of optimization.

1. Introduction

Graphical models are a compact representation often used as a target for learning probabilistic models. Unfortunately, inference in them is exponential in their treewidth (Chandrasekaran et al., 2008), a common measure of complexity. Further, since inference is a subroutine of learning, graphical models are hard to learn unless restricted to those with low treewidth (Bach & Jordan, 2001; Chechetka & Guestrin, 2007), but few real-world problems exhibit this property. Recent research, however, has shown that probabilistic models can in fact be much more expressive than this while remaining tractable (Domingos et al., 2014). In particular, sum-product networks (SPNs) (Gens & Domingos, 2013; Poon & Domingos, 2011) are a class of deep probabilistic models that consist of many layers of hidden variables and can have unbounded treewidth. Despite this, inference in SPNs is guaranteed to be tractable, and their structure and parameters can be effectively and accurately learned from data (Gens & Domingos, 2012; 2013; Rooshenas & Lowd, 2014).

In this paper, we generalize and extend the ideas behind SPNs to enable learning tractable high-treewidth representations for a much wider class of problems, including satisfiability, MAX-SAT, model counting, constraint satisfaction, marginal and MPE inference, integration, nonconvex optimization, database querying, and first-order probabilistic inference. The class of problems we address can be viewed as generalizing structured prediction beyond combinatorial optimization (Taskar et al., 2005), to include optimization for continuous models and others. Instead of approaching each domain individually, we build on a long line of work showing how, despite apparent differences, these problems in fact have much common structure (e.g., Bistarelli et al. (1997); Dechter (1999); Aji & McEliece (2000); Wilson (2005); Green et al. (2007); Dechter & Mateescu (2007); Bacchus et al. (2009)); namely, that each consists of summing a function over a semiring. For example, in the Boolean semiring the sum and product operations are disjunction and conjunction, and deciding satisfiability is summing a Boolean formula over all truth assignments. MPE inference is summation over all states in the max-product semiring, etc.

We begin by identifying and proving the sum-product theorem, a unifying principle for tractable inference that states a simple sufficient condition for summation to be tractable in any semiring: that the factors of every product have disjoint scopes. In "flat" representations like graphical models and conjunctive normal form, consisting of a single product of sums, this would allow only trivial models; but in deep representations like SPNs and negation normal form it provides remarkable flexibility. Based on the sum-product theorem, we develop an algorithm for learning representations that satisfy this condition, thus guaranteeing that the learned functions are tractable yet expressive. We demonstrate the power and generality of our approach by applying it to a new type of structured prediction problem: learning

Proceedings of the 33^{rd} International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48.

a nonconvex function that can be optimized in polynomial time. Empirically, we show that this greatly outperforms the standard approach of learning a continuous function without regard to the cost of optimizing it. We also show that a number of existing and novel results are corollaries of the sum-product theorem, propose a general algorithm for inference in any semiring, define novel tractable classes of constraint satisfaction problems, integrable and optimizable functions, and database queries, and present a much simpler proof of the tractability of tractable Markov logic.

2. The sum-product theorem

We begin by introducing our notation and defining several important concepts. We denote a vector of variables by $\mathbf{X} = (X_1, \ldots, X_n)$ and its value by $\mathbf{x} = (x_1, \ldots, x_n)$ for $x_i \in \mathcal{X}_i$ for all i, where $\mathcal{X}_i$ is the domain of X_i. We denote subsets (for simplicity, we treat tuples as sets) of variables as $\mathbf{X}_A, \mathbf{X}_a \subseteq \mathbf{X}$, where the domains $\mathcal{X}_A, \mathcal{X}_a$ are the Cartesian product of the domains of the variables in $\mathbf{X}_A, \mathbf{X}_a$, respectively. We denote (partial) assignments as $\mathbf{a} \in \mathcal{X}_A$ and restrictions of these to $\mathbf{X}_B \subset \mathbf{X}_A$ as $\mathbf{a}_B$. To indicate compatibility between $\mathbf{a} \in \mathcal{X}_A$ and $\mathbf{c} \in \mathcal{X}_C$ (i.e., that $\mathbf{a}_j = \mathbf{c}_j$ for all $X_j \in \mathbf{X}_A \cap \mathbf{X}_C$), we write $\mathbf{a} \sim \mathbf{c}$. The *scope* of a function is the set of variables it takes as input.

Definition 1. *A* commutative semiring $(R, \oplus, \otimes, 0, 1)$ *is a nonempty set R on which the operations of sum ($\oplus$) and product ($\otimes$) are defined and satisfy the following conditions: (i) $(R, \oplus)$ and $(R, \otimes)$ are associative and commutative, with identity elements $0, 1 \in R$ such that $0 \neq 1$, $a \oplus 0 = a$, and $a \otimes 1 = a$ for all $a \in R$; (ii) $\otimes$ distributes over $\oplus$, such that $a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$ for all $a, b, c \in R$; and (iii) 0 is absorbing for $\otimes$, such that $a \otimes 0 = 0$ for all $a \in R$.*

We are interested in computing summations $\bigoplus_{\mathbf{x} \in \mathcal{X}} F(\mathbf{x})$, for $(R, \oplus, \otimes, 0, 1)$ a commutative semiring and $F : \mathcal{X} \to R$ a function on that semiring, with $\mathcal{X}$ a finite set (but see Section 5.4 for extensions to continuous variables). We refer to such a function as a sum-product function.

Definition 2. *A* sum-product function *(SPF) over $(R, \mathbf{X}, \Phi)$, where R is a semiring, $\mathbf{X}$ is a set of variables, and Φ is a set of constant ($\phi_l \in R$) and univariate functions ($\phi_l : \mathcal{X}_j \to R$ for $X_j \in \mathbf{X}$), is any of the following: (i) a function $\phi_l \in \Phi$, (ii) a product of SPFs, or (iii) a sum of SPFs.*

An SPF $S(\mathbf{X})$ computes a mapping $S : \mathcal{X} \to R$ and can be represented by a rooted directed acyclic graph (DAG), where each leaf node is labeled with a function $\phi_l \in \Phi$ and each non-leaf node is labeled with either $\oplus$ or $\otimes$ and referred to as a sum or product node, respectively. Two SPFs are *compatible* iff they compute the same mapping; i.e., $S_1(\mathbf{x}) = S_2(\mathbf{x})$ for all $\mathbf{x} \in \mathcal{X}$, where $S_1(\mathbf{X})$ and $S_2(\mathbf{X})$ are SPFs. The *size* of an SPF is the number of edges in the graph. The DAG rooted at each node $v \in S$ represents a sub-SPF $S_v : \mathcal{X}_v \to R$ for $\mathbf{X}_v \subseteq \mathbf{X}$. Notice that restricting the leaf functions ϕ_l to be univariate incurs no loss of generality because any mapping $\psi : \mathcal{X} \to R$ is compatible with the trivial SPF $F(\mathbf{X}) = \bigoplus_{\mathbf{x} \in \mathcal{X}} (\psi(\mathbf{x}) \otimes \bigotimes_{i=1}^{n} [X_i = \mathbf{x}_i])$, where the indicator function $[.]$ has value 1 when its argument is true, and 0 otherwise (recall that 0 and 1 are the semiring identity elements). SPFs are similar to arithmetic circuits (Shpilka & Yehudayoff, 2010), but the leaves of an SPF are functions instead of variables. Darwiche (2003) used arithmetic circuits as a data structure to support inference in Bayesian networks over discrete variables. An important subclass of SPFs are those that are decomposable.

Definition 3. *A product node is* decomposable *iff the scopes of its children are disjoint. An SPF is decomposable iff all of its product nodes are decomposable.*

Decomposability is a simple condition that defines a class of functions for which inference is tractable.

Theorem 1 (Sum-product theorem)**.** *Every decomposable SPF can be summed in time linear in its size.*

Proof. The proof is recursive, starting from the leaves of the SPF. Let $S(\mathbf{X})$ be a decomposable SPF on commutative semiring $(R, \oplus, \otimes, 0, 1)$. Every leaf node can be summed in constant time, because each is labeled with either a constant or univariate function. Now, let $v \in S$ be a node, with $S_v(\mathbf{X}_v)$ the sub-SPF rooted at v and $Z_v = \bigoplus_{\mathcal{X}_v} S_v(\mathbf{X}_v)$ its summation. Let $\{c_i\}$ be the children of v for $c_i \in S$, with sub-SPFs $S_i(\mathbf{X}_i)$ for $\mathbf{X}_i \subseteq \mathbf{X}_v$ and summations Z_i. Let $\mathcal{X}_{v\setminus i}$ be the domain of variables $\mathbf{X}_v \setminus \mathbf{X}_i$. If v is a sum node, then $Z_v = \bigoplus_{\mathcal{X}_v} \bigoplus_i S_i(\mathbf{X}_i) = \bigoplus_i \bigoplus_{\mathcal{X}_v} S_i(\mathbf{X}_i) = \bigoplus_i \bigoplus_{\mathcal{X}_{v\setminus i}} \bigoplus_{\mathcal{X}_i} S_i(\mathbf{X}_i) = \bigoplus_i Z_i \otimes (\bigoplus_{\mathcal{X}_{v\setminus i}} 1)$. If v is a product node, then any two children c_i, c_j for $i, j \in \{1, \ldots, m\}$ have disjoint scopes, $\mathbf{X}_i \cap \mathbf{X}_j = \varnothing$, and $Z_v = \bigoplus_{\mathcal{X}_v} \bigotimes_i S_i(\mathbf{X}_i) = \bigoplus_{\mathcal{X}_1} \bigoplus_{\mathcal{X}_{v\setminus 1}} \bigotimes_i S_i(\mathbf{X}_i) = \bigoplus_{\mathcal{X}_1} S_1(\mathbf{X}_1) \otimes \bigoplus_{\mathcal{X}_{v\setminus 1}} \bigotimes_{i=2}^{m} S_i(\mathbf{X}_i) = \bigotimes_i \bigoplus_{\mathcal{X}_i} S_i(\mathbf{X}_i) = \bigotimes_i Z_i$. The above equations only require associativity and commutativity of $\oplus$ and associativity and distributivity of $\otimes$, which are properties of a semiring. Thus, any node can be summed over its domain in time linear in the number of its children, and S can be summed in time linear in its size. □

We assume here that $\bigoplus_{\mathcal{X}_{v\setminus i}} 1$ can be computed in constant time and that each leaf function can be evaluated in constant time, which is true for all semirings considered. We also assume that $a \oplus b$ and $a \otimes b$ take constant time for any elements a, b of semiring R, which is true for most common semirings. See the supplement[1] for details.

The complexity of summation in an SPF can be related to other notions of complexity, such as treewidth, the most common and relevant complexity measure across the domains we consider. To define the treewidth of an SPF, we

[1] http://homes.cs.washington.edu/~pedrod/papers/mlc16sp.pdf

first define junction trees (Lauritzen & Spiegelhalter, 1988; Aji & McEliece, 2000) and a related class of SPFs.

Definition 4. *A* junction tree *over variables* $\mathbf{X}$ *is a tuple* (T, Q)*, where* T *is a rooted tree,* Q *is a set of subsets of variables, each vertex* $i \in T$ *contains a subset of variables* $\mathbf{C}_i \in Q$ *such that* $\cup_i \mathbf{C}_i = \mathbf{X}$*, and for every pair of vertices* $i, j \in T$ *and for all* $k \in T$ *on the (unique) path from* i *to* j*,* $\mathbf{C}_i \cap \mathbf{C}_j \subseteq \mathbf{C}_k$*. The separator for an edge* $(i, j) \in T$ *is defined as* $\mathbf{S}_{ij} = \mathbf{C}_i \cap \mathbf{C}_j$*.*

A junction tree provides a schematic for constructing a specific type of decomposable SPF called a tree-like SPF (a semiring-generalized version of a construction from Darwiche (2003)). Note that a tree-like SPF is not a tree, however, as many of its nodes have multiple parents.

Definition 5. *A* tree-like SPF *over variables* $\mathbf{X}$ *is constructed from a junction tree* $\mathcal{T} = (T, Q)$ *and functions* $\{\psi_i(\mathbf{C}_i)\}$ *where* $\mathbf{C}_i \in Q$ *and* $i \in T$*, and contains the following nodes: (i) a node* ϕ_{vt} *with indicator* $\phi_t(X_v) = [X_v = t]$ *for each value* $t \in \mathcal{X}_v$ *of each variable* $X_v \in \mathbf{X}$*; (ii) a (leaf) node* a_i *with value* $\psi_i(\mathbf{c}_i)$ *and a product node* c_i *for each value* $\mathbf{c}_i \in \mathcal{X}_{\mathbf{C}_i}$ *of each cluster* $\mathbf{C}_i$*; (iii) a sum node* s_{ij} *for each value* $\mathbf{s}_{ij} \in \mathcal{X}_{\mathbf{S}_{ij}}$ *of each separator* $\mathbf{S}_{ij}$*, and (iv) a single root sum node* s*.*

A product node c_j *and a sum node* s_{ij} *are compatible iff their corresponding values are compatible; i.e.,* $\mathbf{c}_j \sim \mathbf{s}_{ij}$*. The nodes are connected as follows. The children of the root* s *are all product nodes* c_r *for* r *the root of* T*. The children of product node* c_j *are all compatible sum nodes* s_{ij} *for each child* i *of* j*, the constant node* a_j *with value* $\psi_j(\mathbf{c}_j)$*, and all indicator nodes* ϕ_{vt} *such that* $X_v \in \mathbf{C}_j$*,* $t \sim \mathbf{c}_j$*, and* $X_v \notin \mathbf{C}_k$ *for* k *any node closer to the root of* $\mathcal{T}$ *than* j*. The children of sum node* s_{ij} *are the compatible product nodes* c_i *of child* i *of* j *connected by separator* $\mathbf{S}_{ij}$*.*

If S is a tree-like SPF with junction tree (T, Q), then it is not difficult to see both that S is decomposable, since the indicators for each variable all appear at the same level, and that each sum node s_{jk} computes $S_{s_{jk}}(\mathbf{S}_{jk}) = \bigoplus_{(\mathbf{c} \in \mathcal{X}_{\mathbf{C}_j}) \sim \mathbf{s}_{jk}} \psi_j(\mathbf{c}) \otimes [\mathbf{C}_j = \mathbf{c}] \otimes \big(\bigotimes_{i \in \mathrm{Ch}(j)} S_{s_{ij}}(\mathbf{c}_{\mathbf{S}_{ij}}) \big)$, where the indicator children of c_j have been combined into $[\mathbf{C}_j = \mathbf{c}]$, $\mathrm{Ch}(j)$ are the children of j, and $i, j, k \in T$ with j the child of k. Further, $S(\mathbf{x}) = \bigotimes_{i \in T} \psi_i(\mathbf{x}_{\mathbf{C}_i})$ for any $\mathbf{x} \in \mathcal{X}$. Thus, tree-like SPFs provide a method for decomposing an SPF. For a tree-like SPF to be compatible with an SPF F, it cannot assert independencies that do not hold in F.

Definition 6. *Let* $F(\mathbf{U})$ *be an SPF over variables* $\mathbf{U}$ *with pairwise-disjoint subsets* $\mathbf{X}, \mathbf{Y}, \mathbf{W} \subseteq \mathbf{U}$*. Then* $\mathbf{X}$ *and* $\mathbf{Y}$ *are* conditionally independent *in* F *given* $\mathbf{W}$ *iff* $F(\mathbf{X}, \mathbf{Y}, \mathbf{w}) = F(\mathbf{X}, \mathbf{w}) \otimes F(\mathbf{Y}, \mathbf{w})$ *for all* $\mathbf{w} \in \mathcal{W}$*, where* $F(\mathbf{X}) = \bigoplus_{\mathcal{Y}} F(\mathbf{X}, \mathbf{Y})$ *for* $\{\mathbf{X}, \mathbf{Y}\}$ *a partition of* $\mathbf{U}$*.*

Similarly, a junction tree $\mathcal{T} = (T, Q)$ is incompatible with F if it asserts independencies that are not in F, where variables X and Y are conditionally independent in $\mathcal{T}$ given $\mathbf{W}$ if $\mathbf{W}$ separates X from Y. A set of variables $\mathbf{W}$ *separates* X and Y in $\mathcal{T}$ iff after removing all vertices $\{i \in T : \mathbf{C}_i \subseteq \mathbf{W}\}$ from T there is no pair of vertices $i, j \in T$ such that $X \in \mathbf{C}_i$, $Y \in \mathbf{C}_j$, and i, j are connected.

Inference complexity is commonly parameterized by *treewidth*, defined for a junction tree $\mathcal{T} = (T, Q)$ as the size of the largest cluster minus one; i.e., $tw(\mathcal{T}) = \max_{i \in T} |\mathbf{C}_i| - 1$. The treewidth of an SPF S is the minimum treewidth over all junction trees compatible with S. Notice that these definitions of junction tree and treewidth reduce to the standard ones (Kask et al., 2005). If the treewidth of S is bounded then inference in S is efficient because there must exist a compatible tree-like SPF that has bounded treewidth. Note that the trivial junction tree with only a single cluster is compatible with every SPF.

Corollary 1. *Every SPF with bounded treewidth can be summed in time linear in the cardinality of its scope.*

Due to space limitations, all other proofs are provided in the supplement. For any SPF, tree-like SPFs are just one type of compatible SPF, one with size exponential in treewidth; however, there are many other compatible SPFs. In fact, there can be compatible (decomposable) SPFs that are exponentially smaller than any compatible tree-like SPF.

Corollary 2. *Not every SPF that can be summed in time linear in the cardinality of its scope has bounded treewidth.*

Given existing work on tractable high-treewidth inference, it is perhaps surprising that the above results do not exist in the literature at this level of generality. Most relevant is the preliminary work of Kimmig et al. (2012), which proposes a semiring generalization of arithmetic circuits for knowledge compilation and does not address learning. Their main results show that summation of circuits that are both decomposable and either deterministic or based on an idempotent sum takes time linear in their size, whereas we show that decomposability alone is sufficient, a much weaker condition. In fact, over the same set of variables, deterministic circuits may be exponentially larger and are never smaller than non-deterministic circuits (Darwiche & Marquis, 2002; Kimmig et al., 2012). We note that while decomposable circuits can be made deterministic by introducing hidden variables, this does not imply that these properties are equivalent.

Even when restricted to specific semirings, such as those for logical and probabilistic inference (e.g., Darwiche (2001; 2003); Poon & Domingos (2011)), some of our results have not previously been shown formally, although some have been foreshadowed informally. Further, existing semiring-specific results (discussed further below) do not make it clear that the semiring properties are all that is required for tractable high-treewidth inference. Our results are thus simpler and more general. Further, the sum-product theorem provides the basis for general algorithms for inference in arbitrary SPFs (Section 3) and for learning tractable high-treewidth representations (i.e., decomposable SPFs) in any semiring (Section 4).

3. Inference in non-decomposable SPFs

Inference in arbitrary SPFs can be performed in a variety of ways, some more efficient than others. We present an algorithm for summing an SPF that adapts to the structure of the SPF and can thus take exponentially less time than constructing and summing a compatible tree-like SPF (Bacchus et al., 2009), which imposes a uniform decomposition structure. SPF S with root node r is summed by calling SUMSPF(r), for which pseudocode is shown in Algorithm 1. SUMSPF is a simple recursive algorithm for summing an SPF (note the similarity between its structure and the proof of the sum-product theorem). If S is decomposable, then SUMSPF simply recurses to the bottom of S, sums the leaf functions, and evaluates S in an upward pass. If S is not decomposable, SUMSPF decomposes each product node it encounters while summing S.

Algorithm 1 Sum an SPF.

Input: node v, the root of the sub-SPF $S_v(\mathbf{X}_v)$
Output: *sum*, which is equal to $\bigoplus_{\mathbf{v}\in\mathcal{X}_v} S_v(\mathbf{v})$

1: **function** SUMSPF(v)
2:   **if** $\langle v, \textit{sum}\rangle$ in cache **then return** *sum*
3:   **if** v is a sum node **then** // $\mathbf{X}_{v\setminus c} = \mathbf{X}_v \setminus \mathbf{X}_c$
4:     *sum* $\leftarrow \bigoplus_{c\in \text{Ch}(v)} \text{SUMSPF}(c) \otimes \bigoplus_{\mathcal{X}_{v\setminus c}} 1$
5:   **else if** v is a product node **then**
6:     **if** v is decomposable **then**
7:       *sum* $\leftarrow \bigotimes_{c\in \text{Ch}(v)} \text{SUMSPF}(c)$
8:     **else**
9:       *sum* $\leftarrow$ SUMSPF(DECOMPOSE(v))
10:   **else** // *v is a leaf with constant a or function ϕ_v*
11:     **if** v is a constant **then** *sum* $\leftarrow a$
12:     **else** *sum* $\leftarrow \bigoplus_{x_j\in\mathcal{X}_j} \phi_v(x_j)$
13:   cache $\langle v, \textit{sum}\rangle$
14:   **return** *sum*

Decomposition can be achieved in many different ways, but we base our method on a common algorithmic pattern that already occurs in many of the inference problems we consider, resulting in a general, semiring-independent algorithm for summing any SPF. DECOMPOSE, shown in Algorithm 2, chooses a variable X_t that appears in the scope of multiple of v's children; creates $|\mathcal{X}_t|$ partially assigned and simplified copies S_{v_i} of the sub-SPF S_v for X_t assigned to each value $x_i \in \mathcal{X}_t$; multiplies each S_{v_i} by an indicator to ensure that only one is ever non-zero when S is evaluated; and then replaces v with a sum over $\{S_{v_i}\}$. Any node $u \in S_v$ that does not have X_t in its scope is re-used across each S_{v_i}, which can drastically limit the amount of duplication that occurs. Furthermore, each S_{v_i} is simplified by

Algorithm 2 Decompose a product node.

Input: product node v, with children $\{c\}$
Output: node s, such that its children are decomposable with respect to X_t and S_s, S_v are compatible

function DECOMPOSE(v)
  $X_t \leftarrow$ choose var. that appears in multiple $\mathbf{X}_c$
  $\mathbf{X}_{v\setminus t} \leftarrow \mathbf{X}_v \setminus \{X_t\}$
  $s \leftarrow$ create new sum node
  for all $x_i \in \mathcal{X}_t$ **do**
    create simplified $S_{v_i}(\mathbf{X}_{v\setminus t}) \leftarrow S_v(\mathbf{X}_{v\setminus t}, x_i)$
    set v_i as child of s // *v_i is the root of S_{v_i}*
    set $f(X_t) = [X_t = x_i]$ as child of v_i
  set s as a child of each of v's parents
  remove v and all edges containing v
  return s

removing any nodes that became 0 when setting $X_t = x_i$. Variables are chosen heuristically; a good heuristic minimizes the amount of duplication that occurs. Similarly, SUMSPF heuristically orders the children in lines 4 and 7. A good ordering will first evaluate children that may return an absorbing value (e.g., 0 for $\otimes$) because SUMSPF can break out of these lines if this occurs. In general, decomposing an SPF is hard, and the resulting decomposed SPF may be exponentially larger than the input SPF, although good heuristics can often avoid this. Many extensions to SUMSPF are also possible, some of which we detail in later sections. Understanding inference in non-decomposable SPFs is important for future work on extending SPF learning to even more challenging classes of functions, particularly those without obvious decomposability structure.

4. Learning tractable representations

Instead of performing inference in an intractable model, it can often be simpler to learn a tractable representation directly from data (e.g., Bach & Jordan (2001); Gens & Domingos (2013)). The general problem we consider is that of learning a decomposable SPF $S : \mathcal{X} \rightarrow R$ on a semiring $(R, \oplus, \otimes, 0, 1)$ from a set of i.i.d. instances $T = \{(\mathbf{x}^{(i)}, y^{(i)})\}$ drawn from a fixed distribution $D_{\mathcal{X}\times R}$, where $y^{(i)} = \bigoplus_{\mathcal{Z}} F(\mathbf{x}^{(i)}, \mathbf{Z})$, F is some (unknown) SPF, and $\mathbf{Z}$ is a (possibly empty) set of unobserved variables or parameters, such that $S(\mathbf{x}^{(i)}) \approx y^{(i)}$, for all i. After learning, $\bigoplus_{\mathcal{X}} S(\mathbf{X})$ can be computed efficiently. In the sum-product semiring, this corresponds to summation (or integration), for which estimation of a joint probability distribution over $\mathbf{X}$ is a special case.

For certain problems, such as constraint satisfaction or MPE inference, the desired quantity is the argument of the sum. This can be recovered (if meaningful in the current semiring) from an SPF by a single downward pass that recursively selects all children of a product node and the (or a) active child of a sum node (e.g., the child with the small-

Algorithm 3 Learn a decomposable SPF from data.

Input: a dataset $T = \{(\mathbf{x}^{(i)}, y^{(i)})\}$ over variables $\mathbf{X}$
Input: integer thresholds $t, v > 0$
Output: $S(\mathbf{X})$, an SPF over the input variables $\mathbf{X}$

function LEARNSPF($T, \mathbf{X}$)
  if $|T| \leq t$ or $|\mathbf{X}| \leq v$ **then**
    estimate $S(\mathbf{X})$ such that $S(\mathbf{x}^{(i)}) \approx y^{(i)}$ for all i
  else
    decompose $\mathbf{X}$ into disjoint subsets $\{\mathbf{X}_i\}$
    if $|\{\mathbf{X}_i\}| > 1$ **then**
      $S(\mathbf{X}) \leftarrow \bigotimes_i$ LEARNSPF($T, \mathbf{X}_i$)
    else
      cluster T into subsets of similar instances $\{T_j\}$
      $S(\mathbf{X}) \leftarrow \bigoplus_j$ LEARNSPF($T_j, \mathbf{X}$)
  return $S(\mathbf{X})$

est value if minimizing). Learning for this domain corresponds to a generalization of learning for structured prediction (Taskar et al., 2005). Formally, the problem is to learn an SPF S from instances $T = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}$, where $\mathbf{y}^{(i)} = \arg\oplus_{\mathbf{y} \in \mathcal{Y}} F(\mathbf{x}^{(i)}, \mathbf{y})$, such that $\arg\oplus_{\mathbf{y} \in \mathcal{Y}} S(\mathbf{x}^{(i)}, \mathbf{y}) \approx \mathbf{y}^{(i)}$, for all i. Here, $\mathbf{x}^{(i)}$ can be an arbitrarily structured input and inference is over the variables $\mathbf{Y}$. Both of the above learning problems can be solved by the algorithm schema we present, with minor differences in the subroutines. We focus here on the former but discuss the latter below, alongside experiments on learning nonconvex functions that, by construction, can be efficiently optimized.

As shown by the sum-product theorem, the key to tractable inference is to identify the decomposability structure of an SPF. The difficulty, however, is that in general this structure varies throughout the space. For example, as a protein folds there exist conformations of the protein in which two particular amino acids are energetically independent (decomposable), and other conformations in which these amino acids directly interact, but in which other amino acids may no longer interact. This suggests a simple algorithm, which we call LEARNSPF (shown in Algorithm 3), that first tries to identify a decomposable partition of the variables and, if successful, recurses on each subset of variables in order to find finer-grained decomposability. Otherwise, LEARNSPF clusters the training instances, grouping those with analogous decomposition structure, and recurses on each cluster. Once either the set of variables is small enough to be summed over (in practice, unary leaf nodes are rarely necessary) or the number of instances is too small to contain meaningful statistical information, LEARNSPF simply estimates an SPF $S(\mathbf{X})$ such that $S(\mathbf{x}^{(i)}) \approx y^{(i)}$ for all i in the current set of instances. LEARNSPF is a generalization of LearnSPN (Gens & Domingos, 2013), a simple but effective SPN structure learning algorithm.

LEARNSPF is actually an algorithm schema that can be instantiated with different variable partitioning, clustering, and leaf creation subroutines for different semirings and problems. To successfully decompose the variables, LEARNSPF must find a partition $\{\mathbf{X}_1, \mathbf{X}_2\}$ of the variables $\mathbf{X}$ such that $\bigoplus_{\mathcal{X}} S(\mathbf{X}) \approx (\bigoplus_{\mathcal{X}_1} S_1(\mathbf{X}_1)) \otimes (\bigoplus_{\mathcal{X}_2} S_2(\mathbf{X}_2))$. We refer to this as *approximate* decomposability. In probabilistic inference, mutual information or pairwise independence tests can be used to determine decomposability (Gens & Domingos, 2013). For our experiments, decomposable partitions correspond to the connected components of a graph over the variables in which correlated variables are connected. Instances can be clustered by virtually any clustering algorithm, such as a naive Bayes mixture model or k-means, which we use in our experiments. Instances can also be split by conditioning on specific values of the variables, as in SUMSPF or in a decision tree. Similarly, leaf functions can be estimated using any appropriate learning algorithm, such as linear regression or kernel density estimation.

In Section 6, we present preliminary experiments on learning nonconvex functions that can be globally optimized in polynomial time. However, this is just one particular application of LEARNSPF, which can be used to learn a tractable representation for any problem that consists of summation over a semiring. In the following section, we briefly discuss common inference problems that correspond to summing an SPF on a specific semiring. For each, we demonstrate the benefit of the sum-product theorem, relate its core algorithms to SUMSPF, and specify the problem solved by LEARNSPF. Additional details and semirings can be found in the supplement. Table 1 provides a summary of some of the relevant inference problems.

5. Applications to specific semirings

5.1. Logical inference

Consider the Boolean semiring $\mathcal{B} = (\mathbb{B}, \vee, \wedge, 0, 1)$, where $\mathbb{B} = \{0, 1\}$, $\vee$ is logical disjunction (OR), and $\wedge$ is logical conjunction (AND). If each variable is Boolean and leaf functions are literals (i.e., each $\phi_l(X_j)$ is X_j or $\neg X_j$, where $\neg$ is logical negation), then SPFs on $\mathcal{B}$ correspond exactly to negation normal form (NNF), a DAG-based representation of a propositional formula (sentence) (Barwise, 1982). An NNF can be exponentially smaller than the same sentence in a standard (flat) representation such as conjunctive or disjunctive normal form (CNF or DNF), and is never larger (Darwiche & Marquis, 2002). Summation of an NNF $F(\mathbf{X})$ on $\mathcal{B}$ is $\bigvee_{\mathcal{X}} F(\mathbf{X})$, which corresponds to propositional satisfiability (SAT): the problem of determining if there exists a satisfying assignment for F. Thus, the tractability of SAT for decomposable NNFs follows directly from the sum-product theorem.

Corollary 3 (Darwiche, 2001). *The satisfiability of a decomposable NNF is decidable in time linear in its size.*

Satisfiability of an arbitrary NNF can be determined either

Table 1. Some of the inference problems that correspond to summing an SPF on a specific semiring, with details on the variables and leaf functions and a core algorithm that is an instance of SUMSPF. $\mathbb{B} = \{0, 1\}$. $\mathbb{N}$ and $\mathbb{R}$ denote the natural and real numbers. Subscript $+$ denotes the restriction to non-negative numbers and subscript $(-)\infty$ denotes the inclusion of (negative) ∞. $\mathbf{U}_m$ denotes the universe of relations of arity up to m (see Section E of the supplement). $\mathbb{N}[\mathbf{X}]$ denotes the polynomials with coefficients from $\mathbb{N}$. See the supplement for information on MPE-SAT (Sang et al., 2007) and Generic-Join (Ngo et al., 2014).

Domain	Inference task	Semiring	Variables	Leaf functions	SUMSPF
Logical inference	SAT	$(\mathbb{B}, \vee, \wedge, 0, 1)$	Boolean	Literals	DPLL
	#SAT	$(\mathbb{N}, +, \times, 0, 1)$	Boolean	Literals	#DPLL
	MAX-SAT	$(\mathbb{N}_{-\infty}, \max, +, -\infty, 0)$	Boolean	Literals	MPE-SAT
Constraint satisfaction	CSPs	$(\mathbb{B}, \vee, \wedge, 0, 1)$	Discrete	Univariate constraints	Backtracking
	Fuzzy CSPs	$([0, 1], \max, \min, 0, 1)$	Discrete	Univariate constraints	-
	Weighted CSPs	$(\mathbb{R}_{+,\infty}, \min, +, \infty, 0)$	Discrete	Univariate constraints	-
Probabilistic inference	Marginal	$(\mathbb{R}_+, +, \times, 0, 1)$	Discrete	Potentials	Recursive conditioning
	MPE	$(\mathbb{R}_+, \max, \times, 0, 1)$	Discrete	Potentials	
Continuous functions	Integration	$(\mathbb{R}_+, +, \times, 0, 1)$	Continuous	Univariate functions	-
	Optimization	$(\mathbb{R}_\infty, \min, +, \infty, 0)$	Continuous	Univariate functions	RDIS
Relational databases	Unions of CQs	$(\mathbf{U}_m, \cup, \bowtie, \varnothing, \mathbf{1}_R)$	Sets of tuples	Unary tuples	Generic-Join
	Provenance	$(\mathbb{N}[\mathbf{X}], +, \times, 0, 1)$	Discrete	K-relation tuples	-

by decomposing the NNF or by expanding it to a CNF and using a SAT solver. DPLL (Davis et al., 1962), the standard algorithm for solving SAT, is an instance of SUMSPF (see also Huang & Darwiche (2007)). Specifically, DPLL is a recursive algorithm that at each level chooses a variable $X \in \mathbf{X}$ for CNF $F(\mathbf{X})$ and computes $F = F|_{X=0} \vee F|_{X=1}$ by recursing on each disjunct, where $F|_{X=x}$ is F with X assigned value x. Thus, each level of recursion of DPLL corresponds to a call to DECOMPOSE.

Learning in the Boolean semiring is a well-studied area, which includes problems from learning Boolean circuits (Jukna, 2012) (of which decomposable SPFs are a restricted subclass, known as syntactically multilinear circuits) to learning sets of rules (Rivest, 1987). However, learned rule sets are typically encoded in large CNF knowledge bases, making reasoning over them intractable. In contrast, decomposable NNF is a tractable but expressive formalism for knowledge representation that supports a rich class of polynomial-time logical operations, including SAT (Darwiche, 2001). Thus, LEARNSPF in this semiring provides a method for learning large, complex knowledge bases that are encoded in decomposable NNF and therefore support efficient querying, which could greatly benefit existing rule learning systems.

5.2. Constraint satisfaction.

A constraint satisfaction problem (CSP) consists of a set of constraints $\{C_i\}$ on variables $\mathbf{X}$, where each constraint $C_i(\mathbf{X}_i)$ specifies the satisfying assignments to its variables. Solving a CSP consists of finding an assignment to $\mathbf{X}$ that satisfies each constraint. When constraints are functions $C_i : \mathcal{X}_i \to \mathbb{B}$ that are 1 when C_i is satisfied and 0 otherwise, then $F(\mathbf{X}) = \bigwedge_i C_i(\mathbf{X}_i) = \bigwedge_i \bigvee_{\mathbf{x}_i \in \mathcal{X}_i} \left(C_i(\mathbf{x}_i) \wedge [\mathbf{X}_i = \mathbf{x}_i]\right) = \bigwedge_i \bigvee_{\mathbf{x}_i \in \mathcal{X}_i} \left(C_i(\mathbf{x}_i) \wedge \bigwedge_{X_t \in \mathbf{X}_i} [X_t = \mathbf{x}_{it}]\right)$ is a CSP and F is an SPF on the Boolean semiring $\mathcal{B}$, i.e., an OR-AND network (OAN), a generalization of NNF, and a decomposable CSP is one with a decomposable OAN. Solving F corresponds to computing $\bigvee_{\mathcal{X}} F(\mathbf{X})$, which is summation on $\mathcal{B}$ (see also Bistarelli et al. (1997); Chang & Mackworth (2005); Rollon et al. (2013)). The solution for F can be recovered with a downward pass that recursively selects the (or a) non-zero child of an OR node, and all children of an AND node. Corollary 4 follows immediately.

Corollary 4. *Every decomposable CSP can be solved in time linear in its size.*

Thus, for inference to be efficient it suffices that the CSP be expressible by a tractably-sized decomposable OAN; a much weaker condition than that of low treewidth. Like DPLL, backtracking-based search algorithms (Kumar, 1992) for CSPs are also instances of SUMSPF (see also Mateescu & Dechter (2005)). Further, SPFs on a number of other semirings correspond to various extensions of CSPs, including fuzzy, probabilistic, and weighted CSPs (see Table 1 and Bistarelli et al. (1997)).

LEARNSPF for CSPs addresses a variant of structured prediction (Taskar et al., 2005); specifically, learning a function $F : \mathbf{X} \to \mathbb{B}$ such that $\arg \bigvee_{\mathbf{y}} F(\mathbf{x}^{(i)}, \mathbf{y}) \approx \mathbf{y}^{(i)}$ for training data $\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}$, where $\mathbf{x}^{(i)}$ is a structured object representing a CSP and $\mathbf{y}^{(i)}$ is its solution. LEARNSPF solves this problem while guaranteeing that the learned CSP remains tractable. This is a much simpler and more attractive approach than existing constraint learning methods such as Lallouet et al. (2010), which uses inductive logic programming and has no tractability guarantees.

5.3. Probabilistic inference

Many probability distributions can be compactly represented as graphical models: $P(\mathbf{X}) = \frac{1}{Z}\prod_i \psi_i(\mathbf{X}_i)$, where ψ_i is a potential over variables $\mathbf{X}_i \subseteq \mathbf{X}$ and Z is known as the partition function (Pearl, 1988). One of the main inference problems in graphical models is to compute the probability of evidence $\mathbf{e} \in \mathcal{X}_E$ for variables $\mathbf{X}_E \subseteq \mathbf{X}$, $P(\mathbf{e}) = \sum_{\mathcal{X}_{\overline{E}}} P(\mathbf{e}, \mathbf{X}_{\overline{E}})$, where $\mathbf{X}_{\overline{E}} = \mathbf{X} \backslash \mathbf{X}_E$. The partition function Z is the unnormalized probability of empty evidence ($\mathbf{X}_E = \varnothing$). Unfortunately, computing Z or $P(\mathbf{e})$ is generally intractable. Building on a number of earlier works (Darwiche, 2003; Dechter & Mateescu, 2007; Bacchus et al., 2009), Poon & Domingos (2011) introduced sum-product networks (SPNs), a class of distributions in which inference is guaranteed to be tractable. An SPN is an SPF on the non-negative real sum-product semiring $(\mathbb{R}_+, +, \times, 0, 1)$. A graphical model is a flat SPN, in the same way that a CNF is a flat NNF (Darwiche & Marquis, 2002). For an SPN S, the unnormalized probability of evidence $\mathbf{e} \in \mathcal{X}_E$ for variables $\mathbf{X}_E$ is computed by replacing each leaf function $\phi_l \in \{\phi_l(X_j) \in S | X_j \in \mathbf{X}_E\}$ with the constant $\phi_l(\mathbf{e}_j)$ and summing the SPN. The corollary below follows immediately from the sum-product theorem.

Corollary 5. *The probability of evidence in a decomposable SPN can be computed in time linear in its size.*

A similar result (shown in the supplement) for finding the most probable state of the non-evidence variables also follows from the sum-product theorem. One important consequence of the sum-product theorem is that decomposability is the sole condition required for an SPN to be tractable; previously, completeness was also required (Poon & Domingos, 2011; Gens & Domingos, 2013). This expands the range of tractable SPNs and simplifies the design of tractable representations based on them, such as tractable probabilistic knowledge bases (Domingos & Webb, 2012).

Most existing algorithms for inference in graphical models correspond to different methods of decomposing a flat SPN, and can be loosely clustered into tree-based, conditioning, and compilation methods, all of which SUMSPF generalizes. Details are provided in the supplement.

LEARNSPF for SPNs corresponds to learning a probability distribution from a set of samples $\{(\mathbf{x}^{(i)}, y^{(i)})\}$. Note that $y^{(i)}$ in this case is defined implicitly by the empirical frequency of $\mathbf{x}^{(i)}$ in the dataset. Learning the parameters and structure of SPNs is a fast-growing area of research (e.g., Gens & Domingos (2013); Rooshenas & Lowd (2014); Peharz et al. (2014); Adel et al. (2015)), and we refer readers to these references for more details.

5.4. Integration and optimization

SPFs can be generalized to continuous (real) domains, where each variable X_i has domain $\mathcal{X}_i \subseteq \mathbb{R}$ and the semiring set is a subset of $\mathbb{R}_\infty$. For the sum-product theorem to hold, the only additional conditions are that (C1) $\bigoplus_{X_j} \phi_l(X_j)$ is computable in constant time for all leaf functions, and (C2) $\bigoplus_{\mathcal{X}_{v\backslash c}} 1 \neq \infty$ for all sum nodes $v \in S$ and all children $c \in \text{Ch}(v)$, where $\mathcal{X}_{v\backslash c}$ is the domain of $\mathbf{X}_{v\backslash c} = \mathbf{X}_v \backslash \mathbf{X}_c$.

Integration. In the non-negative real sum-product semiring $(\mathbb{R}_+, +, \times, 0, 1)$, summation of an SPF with continuous variables corresponds to integration over $\mathcal{X}$. Accordingly, we generalize SPFs as follows. Let $\mu_1, \ldots, \mu_n$ be measures over $\mathcal{X}_1, \ldots, \mathcal{X}_n$, respectively, where each leaf function $\phi_l : \mathcal{X}_j \to \mathbb{R}_+$ is integrable with respect to μ_j, which satisfies (C1). Summation (integration) of an SPF $S(\mathbf{X})$ then corresponds to computing $\int_\mathcal{X} S(\mathbf{X}) d\mu = \int_{\mathcal{X}_1} \cdots \int_{\mathcal{X}_n} S(\mathbf{X}) d\mu_1 \cdots d\mu_n$. For (C2), $\bigoplus_{\mathcal{X}_{v\backslash c}} 1 = \int_{\mathcal{X}_{v\backslash c}} 1 \, d\mu_{v\backslash c}$ must be integrable for all sum nodes $v \in S$ and all children $c \in \text{Ch}(v)$, where $d\mu_{v\backslash c} = \prod_{\{j : X_j \in \mathbf{X}_{v\backslash c}\}} d\mu_j$. We thus assume that either $\mu_{v\backslash c}$ has finite support over $\mathcal{X}_{v\backslash c}$ or that $\mathbf{X}_{v\backslash c} = \varnothing$. Corollary 6 follows immediately.

Corollary 6. *Every decomposable SPF of real variables can be integrated in time linear in its size.*

Thus, decomposable SPFs define a class of functions for which exact integration is tractable. SUMSPF defines a novel algorithm for (approximate) integration that is based on recursive problem decomposition, and can be exponentially more efficient than standard integration algorithms such as trapezoidal or Monte Carlo methods (Press et al., 2007) because it dynamically decomposes the problem at each recursion level and caches intermediate computations. More detail is provided in the supplement.

In this semiring, LEARNSPF learns a decomposable continuous SPF $S : \mathcal{X} \to \mathbb{R}_+$ on samples $\{(\mathbf{x}^{(i)}, y^{(i)} = F(\mathbf{x}^{(i)}))\}$ from an SPF $F : \mathcal{X} \to \mathbb{R}_+$, where S can be integrated efficiently over the domain $\mathcal{X}$. Thus, LEARNSPF provides a novel method for learning and integrating complex functions, such as the partition function of continuous probability distributions.

Nonconvex optimization. Summing a continuous SPF in one of the min-sum, min-product, max-sum, or max-product semirings corresponds to optimizing a (potentially nonconvex) continuous objective function. Our results hold for all of these, but we focus here on the real min-sum semiring $(\mathbb{R}_\infty, \min, +, \infty, 0)$, where summation of a min-sum function (MSF) $F(\mathbf{X})$ corresponds to computing $\min_\mathcal{X} F(\mathbf{X})$. A flat MSF is simply a sum of terms. To satisfy (C1), we assume that $\min_{x_j \in \mathcal{X}_j} \phi_l(x_j)$ is computable in constant time for all $\phi_l \in F$. (C2) is trivially satisfied for $\min$. The corollary below follows immediately.

Corollary 7. *The global minimum of a decomposable MSF can be found in time linear in its size.*

SUMSPF provides an outline for a general nonconvex optimization algorithm for sum-of-terms (or product-of-factors) functions. The recent RDIS algorithm for nonconvex optimization (Friesen & Domingos, 2015), which achieves exponential speedups compared to other algorithms, is an instance of SUMSPF where values are chosen via multi-start gradient descent and variables in DECOMPOSE are chosen by graph partitioning. Friesen & Domingos (2015), however, do not specify tractability conditions for the optimization; thus, Corollary 7 defines a novel class of functions that can be efficiently globally optimized.

For nonconvex optimization, LEARNSPF solves a variant of structured prediction (Taskar et al., 2005), in which the variables to predict are continuous instead of discrete (e.g., protein folding, structure from motion (Friesen & Domingos, 2015)). The training data is a set $\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}$, where $\mathbf{x}^{(i)}$ is a structured object representing a nonconvex function and $\mathbf{y}^{(i)}$ is a vector of values specifying the global minimum of that function. LEARNSPF learns a function $S : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}_\infty$ such that $\arg\min_{\mathbf{y}\in\mathcal{Y}} S(\mathbf{x}^{(i)}, \mathbf{y}) \approx \mathbf{y}^{(i)}$, where the $\arg\min$ can be computed efficiently because S is decomposable. More detail is provided in Section 6.

6. Experiments

We evaluated LEARNSPF on the task of learning a nonconvex decomposable min-sum function (MSF) from a training set of solutions of instances of a highly-multimodal test function consisting of a sum of terms. By learning an MSF, instead of just a sum of terms, we learn the general mathematical form of the optimization problem in such a way that the resulting learned problem is tractable, whereas the original sum of terms is not. The test function we learn from is a variant of the Rastrigin function (Torn & Zilinskas, 1989), a standard highly-multimodal test function for global optimization consisting of a sum of multi-dimensional sinusoids in quadratic basins. The function, $F_{\mathbf{X}}(\mathbf{Y}) = F(\mathbf{Y}; \mathbf{X})$, has parameters $\mathbf{X}$, which determine the dependencies between the variables $\mathbf{Y}$ and the location of the minima. To test LEARNSPF, we sampled a dataset of function instances $T = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^m$ from a distribution over $\mathcal{X} \times \mathcal{Y}$, where $\mathbf{y}^{(i)} = \arg\min_{\mathbf{y}\in\mathcal{Y}} F_{\mathbf{x}^{(i)}}(\mathbf{y})$.

LEARNSPF partitioned variables $\mathbf{Y}$ based on the connected components of a graph containing a node for each $Y_i \in \mathbf{Y}$ and an edge between two nodes only if Y_i and Y_j were correlated, as measured by Spearman rank correlation. Instances were clustered by running k-means on the values $\mathbf{y}^{(i)}$. For this preliminary test, LEARNSPF did not learn the leaf functions of the learned min-sum function (MSF) $M(\mathbf{Y})$; instead, when evaluating or minimizing a leaf node in M, we evaluated or minimized the test function with all variables not in the scope of the leaf node fixed to 0 (none of the optima were positioned at 0). This corresponds to having perfectly learned leaf nodes if the scopes of the leaf nodes accurately reflect the decomposability of F, otherwise a large error is incurred. We did this to study

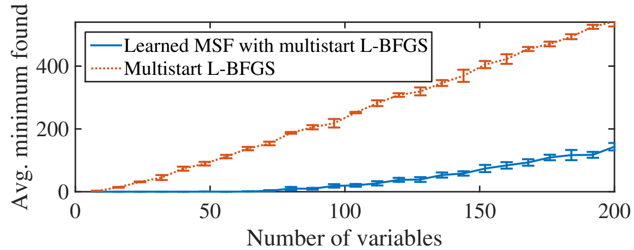

Figure 1. The average minimum found over 20 samples of the test function versus the number of variables, with standard error bars. Each function was optimized for the same amount of time.

the effect of learning the decomposability structure in isolation from the error due to learning leaf nodes. The function used for comparison is also perfectly learned. Thresholds t and v were set to 30 and 2, respectively.

The dataset was split into 300 training samples and 50 test samples, where $\min_{\mathcal{Y}} F_{\mathbf{x}^{(i)}}(\mathbf{Y}) = 0$ for all i for comparison purposes. After training, we computed $\mathbf{y}_M = \arg\min_{\mathcal{Y}} M(\mathbf{Y})$ for each function in the test set by first minimizing each leaf function (with respect to only those variables in the scope of the leaf function) with multi-start L-BFGS (Liu & Nocedal, 1989) and then performing an upward and a downward pass in M. Figure 1 shows the result of minimizing the learned MSF M and evaluating the test function at $\mathbf{y}_M$ (blue line) compared to running multi-start L-BFGS directly on the test function and reporting the minimum found (red line), where both optimizations are run for the same fixed amount of time (one minute per test sample). LEARNSPF accurately learned the decomposition structure of the test function, allowing it to find much better minima when optimized, since optimizing many small functions at the leaves requires exploring exponentially fewer modes than optimizing the full function. Additional experimental details are provided in the supplement.

7. Conclusion

This paper developed a novel foundation for learning tractable representations in any semiring based on the sum-product theorem, a simple tractability condition for all inference problems that reduce to summation on a semiring. With it, we developed a general inference algorithm and an algorithm for learning tractable representations in any semiring. We demonstrated the power and generality of our approach by applying it to learning a nonconvex function that can be optimized in polynomial time, a new type of structured prediction problem. We showed empirically that our learned function greatly outperforms a continuous function learned without regard to the cost of optimizing it. We also showed that the sum-product theorem specifies an exponentially weaker condition for tractability than low treewidth and that its corollaries include many previous results in the literature, as well as a number of novel results.

Acknowledgments

This research was partly funded by ONR grants N00014-13-1-0720 and N00014-12-1-0312, and AFRL contract FA8750-13-2-0019. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ONR, AFRL, or the United States Government.

References

Adel, T., Balduzzi, D., and Ghodsi, A. Learning the structure of sum-product networks via an SVD-based algorithm. In *Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence*, pp. 32–41, 2015.

Aji, S. M. and McEliece, R. J. The generalized distributive law. *IEEE Transactions on Information Theory*, 46:325–343, 2000.

Bacchus, F., Dalmao, S., and Pitassi, T. Solving #SAT and Bayesian inference with backtracking search. *Journal of Artificial Intelligence Research*, 34:391–442, 2009.

Bach, F. and Jordan, M. I. Thin junction trees. In *Advances in Neural Information Processing Systems*, pp. 569–576, 2001.

Barwise, J. *Handbook of mathematical logic*. Elsevier, 1982.

Bistarelli, S., Montanari, U., and Rossi, F. Semiring-based constraint satisfaction and optimization. *Journal of the ACM*, 44:201–236, 1997.

Chandrasekaran, V., Srebro, N., and Harsha, P. Complexity of inference in graphical models. In *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence*, pp. 70–78, 2008.

Chang, L. and Mackworth, A. K. A generalization of generalized arc consistency: From constraint satisfaction to constraint-based inference. In *Proceedings of the IJCAI-05 Workshop on Modeling and Solving Problems with Constraints*, 2005.

Chechetka, A. and Guestrin, C. Efficient principled learning of thin junction trees. In *Advances in Neural Information Processing Systems*, pp. 273–280, 2007.

Darwiche, A. and Marquis, P. A knowledge compilation map. *Journal of Artificial Intelligence Research*, 17: 229–264, 2002.

Darwiche, A. Decomposable negation normal form. *Journal of the ACM*, 48:608–647, 2001.

Darwiche, A. A differential approach to inference in Bayesian networks. *Journal of the ACM*, 50:280–305, 2003.

Davis, M., Logemann, G., and Loveland, D. A machine program for theorem-proving. *Communications of the ACM*, 5:394–397, 1962.

Dechter, R. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113:41–85, 1999.

Dechter, R. and Mateescu, R. AND/OR search spaces for graphical models. *Artificial intelligence*, 171:73–106, 2007.

Domingos, P. and Webb, W. A. A tractable first-order probabilistic logic. In *Proceedings of the 26th Conference on Artificial Intelligence*, pp. 1902–1909, 2012.

Domingos, P., Niepert, M., and Lowd, D. (eds.). *Proceedings of the ICML-14 Workshop on Learning Tractable Probabilistic Models*. ACM, 2014.

Friesen, A. L. and Domingos, P. Recursive Decomposition for Nonconvex Optimization. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, pp. 253–259, 2015.

Gens, R. and Domingos, P. Discriminative learning of sum-product networks. In *Advances in Neural Information Processing Systems*, pp. 3239–3247, 2012.

Gens, R. and Domingos, P. Learning the structure of sum-product networks. In *Proceedings of the 30th International Conference on Machine Learning*, pp. 873–880, 2013.

Green, T., Karvounarakis, G., and Tannen, V. Provenance semirings. In *Proceedings of the 26th ACM SIGMOD-SIGACT-SIGART symposium on Principles of Database Systems*, pp. 31–40. ACM Press, 2007.

Huang, J. and Darwiche, A. The language of search. *Journal of Artificial Intelligence Research*, 29:191–219, 2007.

Jukna, S. *Boolean Function Complexity*. Springer Berlin Heidelberg, 2012.

Kask, K., Dechter, R., Larrosa, J., and Dechter, A. Unifying tree decompositions for reasoning in graphical models. *Artificial Intelligence*, 166:165–193, 2005.

Kimmig, A., Van Den Broeck, G., and De Raedt, L. Algebraic model counting. *arXiv preprint arXiv:1211.4475*, 2012.

Kumar, V. Algorithms for constraint-satisfaction problems: A survey. *AI Magazine*, 13:32–44, 1992.

Lallouet, A., Lopez, M., Martin, L., and Vrain, C. On learning constraint problems. *Proceedings of the International Conference on Tools with Artificial Intelligence*, 1:45–52, 2010.

Lauritzen, S. L. and Spiegelhalter, D. J. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society*, 50(2):157–224, 1988.

Liu, D. C. and Nocedal, J. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45:503–528, 1989.

Mateescu, R. and Dechter, R. The relationship between AND/OR search spaces and variable elimination. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*, pp. 380–387, 2005.

Ngo, H. Q., Ré, C., and Rudra, A. Skew strikes back. *ACM SIGMOD Record*, 42(4):5–16, 2014.

Pearl, J. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kauffmann, San Mateo, CA, 1988.

Peharz, R., Gens, R., and Domingos, P. Learning selective sum-product networks. In *Proceedings of the ICML-14 Workshop on Learning Tractable Probabilistic Models*, 2014.

Poon, H. and Domingos, P. Sum-product networks: A new deep architecture. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence*, pp. 337–346. AUAI Press, 2011.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. *Numerical Recipes: The Art of Scientific Computing (3rd ed.)*. Cambridge University Press, 2007.

Rivest, R. L. Learning decision lists. *Machine Learning*, 2: 229–246, 1987.

Rollon, E., Larrosa, J., and Dechter, R. Semiring-based mini-bucket partitioning schemes. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, pp. 644–650, 2013.

Rooshenas, A. and Lowd, D. Learning sum-product networks with direct and indirect variable interactions. In *Proceedings of the 31st International Conference on Machine Learning*, pp. 710–718, 2014.

Sang, T., Beame, P., and Kautz, H. A dynamic approach to MPE and weighted MAX-SAT. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pp. 173–179, 2007.

Shpilka, A. and Yehudayoff, A. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5: 207–388, 2010.

Taskar, B., Chatalbashev, V., Koller, D., and Guestrin, C. Learning structured prediction models: A large margin approach. In *Proceedings of the 22nd International Conference on Machine Learning*, pp. 896–903, 2005.

Torn, A. and Zilinskas, A. *Global Optimization*. Springer-Verlag, 1989.

Wilson, N. Decision diagrams for the computation of semiring valuations. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pp. 331–336, 2005.