Available online at www.sciencedirect.com
ScienceDirect
Journal of Computer and System Sciences 74 (2008) 84–96
JOURNAL OF COMPUTER AND SYSTEM SCIENCES
www.elsevier.com/locate/jcss

Case-factor diagrams for structured probabilistic modeling

David McAllester [a,*,1], Michael Collins [b,2], Fernando Pereira [c,3]

[a] *TTI at Chicago, 1427 East 60th Street, Chicago, IL 60637, USA*
[b] *CSAIL, Massachusetts Institute of Technology, USA*
[c] *CIS, University of Pennsylvania, USA*

Received 1 April 2005; received in revised form 1 August 2006

Available online 25 April 2007

Abstract

We introduce a probabilistic formalism handling both Markov random fields of bounded tree width and probabilistic context-free grammars. Our models are based on case-factor diagrams (CFDs) which are similar to binary decision diagrams (BDDs) but are more concise for circuits of bounded tree width. A probabilistic model consists of a CFD defining a feasible set of Boolean assignments and a weight (or cost) for each individual Boolean variable. We give versions of the inside–outside algorithm and the Viterbi algorithm for these models.

Keywords: Boolean decision diagrams; Markov random fields; Probabilistic context free grammars; Hidden Markov models; Conditional random fields; Probabilistic relational models; Structured labeling; Graphical models; Bayesian networks; Zero supression; Recursive conditioning; And/or graphs

1. Introduction

In this paper, we investigate efficient representations for structured probabilistic models. Informally, a structured model defines a distribution on structured objects such as sequences, parse trees, or assignments of values to variables. The number of possible structured objects typically grows exponentially in a natural measure of problem size. For example, the number of possible parse trees grows exponentially in the length of the string being parsed. Structured statistical models include Markov random fields (MRFs), probabilistic context-free grammars (PCFGs), hidden Markov models (HMMs), conditional random fields (CRFs) [1], dynamic Bayes nets [2], probabilistic Horn abduction [3], and probabilistic relational models (PRMs) [4].

For each of these model types one can define a corresponding structured classification problem. In HMMs, for example, the problem is to recover the hidden state sequence from the observable sequence. For PCFGs, the problem is to recover a parse tree from a given word string. In PRMs, the problem is to recover latent entity labels and relations

* Corresponding author.
E-mail addresses: mcallester@tti-c.org (D. McAllester), mcollins@ai.mit.edu (M. Collins), pereira@cis.upenn.edu (F. Pereira).
[1] Supported by National Science Foundation grant 0534820.
[2] Supported by National Science Foundation grant 0347631.
[3] Supported by National Science Foundation grant EIA-0205456

0022-0000/$ – see front matter
doi:10.1016/j.jcss.2007.04.015

for a given set of observed entities and relations. We follow an approach where the statistical model defines $P(y|x)$ and structured classification finds the most likely y for a given x. (Other approaches are possible—for example, maximum margin classifiers are discussed below.)

The structured statistical models discussed above are intuitively similar. They all involve local probability tables or local cost functions. It is widely believed that many, if not all, of the above modeling formalisms can be viewed as special cases of MRFs (undirected graphical models). More specifically, in a structured classification problem one should be able to represent $P(y|x)$ as an MRF. By assuming $P(y|x)$ is modeled as an MRF one can prove theorems and design algorithms and software at an abstract level which simultaneously applies to all of the modeling formalisms discussed above.

Unfortunately, for some of the above models the representation of $P(y|x)$ as an MRF is problematic. The most problematic case is perhaps PCFGs. It is fairly easy to construct an MRF representing $P(y|x)$ where y is a parse tree and x is a word string. Unfortunately, standard MRF algorithms take exponential time when applied to the natural MRF representation. This is a somewhat surprising outcome, given that there are well-known inference algorithms for PCFGs which run in cubic time in the length of the word string x.

This paper presents a modeling formalism which handles both MRFs of bounded tree width and PCFGs. First we define a *linear Boolean model* (LBM). An LBM consists of three parts: a set of boolean variables; a formula defining a set of possible assignments to these variables (a "feasible set"); and an assignment of a weight to each variable. The weight for a complete variable assignment is then the sum of weights for those variables in the assignment that are true. The weight associated with a truth assignment can be written as a linear function of the bits in the assignment—hence the term "linear." We show how to encode both standard MRFs and PCFGs as LBMs.

The main problem we solve is how to encode compactly the set of possible assignments to the variables in an LBM in a single formalism handling both MRFs of bounded tree width and PCFGs. The *case-factor diagrams* (CFDs) we introduce for that purpose are similar to binary-decision diagrams (BDDs) [5]. CFDs differ from BDDs in two ways. First, CFDs are similar to *zero-suppressed* BDDs (ZBDDs) [6]. ZBDDs are designed for sparse truth assignments—truth assignments where most of the Boolean variables are false. Sparseness is important for representing PCFGs. Second, CFDs, unlike BDDs, have "factor nodes" which allow a concise representation of problems that factor into independent subproblems. Factoring is important for representing MRFs of bounded tree width. We describe algorithms for CFDs that compute partition functions under Gibbs distributions for $P(y|x)$, that select the maximum likelihood (Viterbi) structure, and an inside–outside algorithm for computing the marginal distributions of all of the Boolean variables. These algorithms all run in time linear in the number of nodes in the CFD. We demonstrate that PCFG models can be encoded in a CFD which has $O(n^3)$ size where n is the length of the input string. We also show that MRFs of bounded tree width can be represented by a CFD with a polynomial number of nodes.

There are various lines of related work. A variant of BDDs for circuits of bounded tree width was introduced by McMillan [7]. Although McMillan's formalism is more elaborate, it turns out that simply extending BDDs with "and" nodes suffices for representing MRFs of bounded width. But representing PCFGs seems to require a zero-suppressed formalism. Case-factor diagrams combine zero suppression with factoring—a combination that seems essential to efficient representation of PCFG parsing problems.

CFDs are closely related to the recursive conditioning algorithm introduced by Darwiche [8,9]. Recursive conditioning cases on the value of a variable; factors the remaining problem into independent subproblems; and then solves the subproblems recursively. The nodes of a CFD correspond to the "subproblems" that arise in recursive conditioning. Darwiche has also defined a data structure for representing the subproblems of recursive conditioning based on arithmetic expressions [10]. The differences between CFGs and Darwiche's expressions involve the generalization to the problem of parsing PCFGs—a problem not addressed by Darwiche. CFDs and recursive conditioning can both exploit context-sensitive independence [11] where two variables are independent under some values of a third variable but not independent in general. Context-sensitive independence is particularly important for PCFGs where the tree width of the natural MRF representation is large. However, as is explained in Section 8, getting $O(n^3)$ time behavior rather than $O(n^6)$ time behavior seems to require handling context-sensitive *variable existence* as well as context-sensitive independence. In parsing, although there are $\Omega(n^2)$ possible substrings of the given input string, only $O(n)$ of those substrings represent phrases in any single parse and a parse is determined by the value of only $O(n)$ choices. This is very much unlike a Bayesian network where all variables are used in all assignments. In CFDs context-sensitive variable existence is handled with zero suppression. Boolean variables that are forced to be false need not be mentioned—in certain contexts these variables essentially cease to exist.

Rina Dechter has given an and/or graph data structure similar to CFDs and to Darwiche's expression graphs for representing Bayesian networks [12]. Another closely related formalism has been given by Jaeger [13] and related algorithms similar to recursive conditioning have been given by Bacchus, Dalmao and Pitassi [14,15]. As with Darwiche's formalism, however, these formalisms do not address the need for context-sensitive variable existence in the PCFG parsing problem.

It is also important to note that CFDs are similar to BDDs in that they carry a semantics independent of the way in which they were constructed. An independent semantics allows one to treat CFDs as first class models. One of the fundamental properties of BDDs is that one can define Boolean and projection operations on BDDs allowing them to be built up in a compositional way. CFDs, as a form of BDD, can also support compositional operations although we do not explore such operations here.

Developing a common language for structured modeling has potential applications to maximum-margin structured classification [16–18]. A maximum margin model is trained using an objective function stated in terms of margins rather than in terms of $P(y|x)$. However, the model parameters can still be viewed as defining a log-linear or maxent probabilistic representation. CFDs provide a formalism for structured modeling that allows these algorithms and others to be formulated at a level of generality that covers both MRFs of bounded tree width and weighted grammar formalisms like PCFGs.

2. Linear Boolean models

A *linear Boolean model* (LBM) is a triple $\langle V, F, \Psi\rangle$ where V is a set of Boolean variables with values in $\{0, 1\}$, F is a subset of the set of all assignments to V, and Ψ is an *energy function* $\Psi : V \to \mathbb{R}$. We call the elements of F the *feasible configurations* of the model. We extend Ψ to configurations $\rho \in F$ with the following "linear" definition:

$$\Psi(\rho) = \sum_{z\in V} \Psi(z)\rho(z). \tag{1}$$

If we view Ψ as a vector in $\mathbb{R}^{|V|}$ and ρ as a vector in $\{0, 1\}^{|V|}$ then $\Psi(\rho)$ is simply the inner product of Ψ and ρ. A LBM M defines a probability distribution $P(\cdot|F, \Psi)$ on feasible configurations $\rho \in F$ as follows:

$$P(\rho|F, \Psi) = \frac{1}{Z(F, \Psi)} e^{-\Psi(\rho)}, \tag{2}$$

$$Z(F, \Psi) = \sum_{\rho\in F} e^{-\Psi(\rho)}. \tag{3}$$

Given Eq. (2) we have that an LBM is really just a log-linear or maxent model [19] on a set F under the restrictions that all features are Boolean and that each element of F is uniquely determined by its feature values. A critical issue is how to represent the feasible set F. Before discussing the representation of F, however, we give two examples of representing structured models with LBMs.

3. Markov random fields

We first introduce some notation. We consider a set of *variables* V and *domains* $\mathcal{D}(x)$ for each $x \in V$. An *assignment* ρ maps $x \in V$ to $\rho(x) \in \mathcal{D}(x)$. A *partial* assignment σ maps a subset of the variables $\mathrm{dom}(\sigma) \subseteq V$ to appropriate values. We write $\rho' \sqsubseteq \rho$ if $\mathrm{dom}(\rho') \subseteq \mathrm{dom}(\rho)$ and $\rho'(x) = \rho(x)\ \forall x \in \mathrm{dom}(\rho')$. If ρ is a (possibly partial) assignment on V and $V' \subseteq V$, $\rho|_{V'}$ is the unique assignment such that $\rho|_{V'} \sqsubseteq \rho$ and $\mathrm{dom}(\rho|_{V'}) = \mathrm{dom}(\rho) \cap V'$. If all the variables are *Boolean*, that is $\mathcal{D}(x) = \{0, 1\}\ \forall x \in V$, the assignment is a *truth* assignment. If ρ is a (possibly partial) assignment, $x \in V$ a variable, and $v \in \mathcal{D}(x)$, $\rho[x := v]$ is the assignment identical to ρ except that $\rho[x := v](x) = v$.

A Markov random field (MRF) consists of variables and energy terms on configurations of those variables. More precisely, we assume a finite set of variables $y_1, \ldots, y_\ell$ with associated domains $\mathcal{D}(y_1), \ldots, \mathcal{D}(y_\ell)$. We take the domains $\mathcal{D}(y_j)$ to be finite sets with $|\mathcal{D}(y_i)| \geqslant 2$. We define a *configuration* to be an assignment ρ of values to the variables. An MRF is a set of such variables plus a set of energy terms $\Psi_1, \ldots, \Psi_N$ each of which maps a configuration to a real number. Any such set of energy terms defines a hypergraph on the variables. More specifically, we say that Ψ_k depends on variable y_j if there exists configurations ρ and ρ' which agree on all variables except y_j and such

that $\Psi_k(\rho) \neq \Psi_k(\rho')$. Let V_k denote the set of variables on which Ψ_k depends. The sets V_k define a hypergraph on the variables. If $|V_k| = 2$ for all k then these sets define a graph.

An MRF M defines a probability distribution over configurations $P(\rho|M)$ by the following equations:

$$P(\rho|M) = \frac{1}{Z(M)} e^{-\Psi(\rho)},$$
$$Z(M) = \sum_{\rho} e^{-\Psi(\rho)},$$
$$\Psi(\rho) = \sum_{k} \Psi_k(\rho).$$

To represent an MRF as a LBM we must represent a configuration of M as a truth assignment on Boolean variables and represent the energy terms by an energy function on Boolean variables. Given an MRF M we construct Boolean variables of the form "$y_i = v$" with y_i a variable of M and with $v \in \mathcal{D}(y_i)$. For each energy term Ψ_k with $V_k = \{y_1, \ldots, y_m\}$ and each tuple of values $v_1, \ldots, v_m$ with $v_i \in \mathcal{D}(y_i)$ we also introduce the Boolean variable "$k, y_1 = v_1 \wedge \cdots \wedge y_m = v_m$." Of course not all truth assignments to these Boolean variables correspond to configurations of the random field M. In order for a Boolean assignment to be feasible we must have that for each y exactly one of "$y = v_1$," "$\ldots$," "$y = v_n$" is true and furthermore "$k, y_1 = v_1 \wedge \cdots \wedge y_m = v_m$" is true if and only if each of "$y_1 = v_1$," "$\ldots$," "$y_m = v_m$" is true. Section 5 discusses a method for representing this feasible set of truth assignments. Finally we define the variable energy function as follows:

$$\Psi(\text{“}y = v\text{”}) = 0,$$
$$\Psi(\text{“}k, y_1 = v_1 \wedge \cdots \wedge y_m = v_m\text{”}) = \Psi_k(v_1, \ldots, v_m).$$

4. Parse distributions as LBMs

A CFG in Chomsky normal form is a set of productions of the following form where X, Y and Z are nonterminal symbols and a is a terminal symbol,

$$X \to YZ,$$
$$X \to a.$$

A parse tree is a tree each node of which is labeled by a production of the grammar in the standard way. In a weighted CFG each production $X \to \gamma$ is assigned an energy (weight) $\Psi(X \to \gamma)$. For any parse tree y we write yield(y) for the yield of y, i.e., the sequence of terminal symbols at the leaves of the parse tree. We write $\Psi(y)$ for the total energy of the parse tree y—$\Psi(y)$ is the sum over all nodes of y of the energy of the production used at that node. For a given string x of terminal symbols we have a probability distribution on parse trees y with yield$(y) = x$ defined as follows:

$$P(y|x) = \frac{1}{Z(x)} e^{-\Psi(y)}, \tag{4}$$
$$Z(x) = \sum_{y:\, \text{yield}(y)=x} e^{-\Psi(y)}. \tag{5}$$

To construct an LBM representation of $P(y|x)$ we first define a set of Boolean variables. Let n be the length of x. First we have a phrase variable "$X_{i,j}$" for each nonterminal X in the grammar and $1 \leqslant i < j \leqslant n+1$. This phrase variable represents the statement that the parse contains a phrase with nonterminal X spanning the string from i to $j-1$ inclusive. Second we have a branch variable "$X_{i,k} \to Y_{i,j} Z_{j,k}$" for each production $X \to YZ$ in the grammar and $1 \leqslant i < j < k \leqslant n+1$. A branch variable represents the statement that the parse contains a node labeled with the given production where the left child of the node spans the string from i to $j-1$ and the right child spans j to $k-1$. Finally, we have a terminal variable "$X_{i,i+1} \to a$" for each terminal production $X \to a$ and position i in the input string. A terminal variable represents the statement that the parse tree produces terminal symbol a from nonterminal X at position i. We take V to be the set of all such phrase, branch, and terminal variables. Each parse tree determines a truth assignment to the variables in V and we take F to be the set of assignments corresponding to parse trees.

Finally, we must define the energy of each Boolean variable. The variable energy function Ψ is given by the following equations:

$$\Psi(\text{“}X_{i,j}\text{”}) = 0,$$
$$\Psi(\text{“}X_{i,k} \to Y_{i,j}Z_{j,k}\text{”}) = \Psi(X \to YZ),$$
$$\Psi(\text{“}X_{i,i+1} \to a\text{”}) = \Psi(X \to a).$$

5. Case-factor diagrams (CFDs)

We first introduce some notation. If F is a set of assignments of values to variables then $F[x := v] = \{\rho[x := v]: \rho \in F\}$. If ρ and σ are truth assignments, $\rho \vee \sigma$ is the assignment such that $(\rho \vee \sigma)(x) = 1$ if and only if $\rho(x) = 1$ or $\sigma(x) = 1$. If F_1 and F_2 are sets of truth assignments, $F_1 \vee F_2 = \{\rho \vee \sigma\colon\ \rho \in F_1 \text{ and } \sigma \in F_2\}$. The *support* of a truth assignment is the set of variables set to 1 by the assignment.

A *case-factor diagram* represents the feasible set by a search tree over the set of possible truth assignments. The search tree cases on the value of individual variables and factors the feasible set into a product of independent feasible sets when possible. We represent this case-factor search tree by an expression.

Definition 1. A case-factor diagram (CFD) D is an expression generated by the following grammar where x is a Boolean variable; a case expression $\mathsf{case}(x, D_1, D_2)$ must satisfy the constraint that x does not appear in D_1 or D_2; and a factor expression $\mathsf{factor}(D_1, D_2)$ must satisfy the constraint that no variable occurs in both D_1 and D_2,

$$D ::= \mathsf{case}(x, D_1, D_2) | \mathsf{factor}(D_1, D_2) | \mathsf{unit} | \mathsf{empty}.$$

We denote by $V(D)$ the set of variables occurring in D.

To define the meaning of CFDs, it is convenient to see all CFD variables as members of a common countably infinite set of variables V. The interpretation $F(D)$ of a CFD D is then a finite set of finite support assignments to V. We use $\bar{0}$ for the totally false assignment (the zero vector). $F(D)$ is defined as follows:

$$F(\mathsf{unit}) = \{\bar{0}\},$$
$$F(\mathsf{empty}) = \emptyset,$$
$$F\big(\mathsf{case}(x, D_1, D_2)\big) = F(D_1)[x := 1] \cup F(D_2),$$
$$F\big(\mathsf{factor}(D_1, D_2)\big) = F(D_1) \vee F(D_2).$$

Note that, as with ZBDDs, variables that are false in all assignments in $F(D)$ need not be mentioned. In contrast, a BDD must mention any variable required to be false. In contrast to BDDS, ZBDDs give sparse representations of sparse assignments (assignments that are mostly false).

An example consider variables $x_1, x_2, \ldots$ and consider the CFD A_i defined as follows:

$$A_0 = \mathsf{unit},$$
$$A_{i+1} = \mathsf{case}(x_{i+1}, A_i, A_i).$$

Under the semantics stated above we have that $F(A_i)$ is the set of all the 2^i truth assignments ρ satisfying the constraint that $\rho(x_j) = 0$ for all $j > i$. As another example, consider B_i defined as follows:

$$B_0 = \mathsf{unit},$$
$$B_{i+1} = \mathsf{factor}\big(\mathsf{case}(x_{i+1}, \mathsf{unit}, \mathsf{unit}), B_i\big).$$

We leave it to the reader to verify that $F(B_i) = F(A_i)$. As a third example consider C_i defined as follows:

$$C_0 = \mathsf{unit},$$
$$C_{i+1} = \mathsf{case}(x_{i+1}, C_i, \mathsf{empty}).$$

We have that $F(C_i)$ contains only the single truth assignment ρ such that $\rho(x_j) = 1$ for $j \leqslant i$ and $\rho(x_j) = 0$ for $j > i$. In general this semantics has the property that if x does not occur in D then $\rho(x) = 0$ for any assignment $\rho \in F(D)$.

Because the two arguments of a factor expression cannot share variables, we have that the number of assignments in $F(\text{factor}(D_1, D_2))$ equals the number of assignments in $F(D_1)$ times the number of assignments in $F(D_2)$. We leave it to the reader to verify that any feasible set on any finite set of variables can be represented by a CFD.

The meaning of CFD expressions is independent of their representation as data structures. However, the running time of algorithms depends crucially on that representation. For all the algorithms we discuss, we assume that CFD expressions are represented as *diagrams*, which are DAGs with one node for each distinct subexpression, and edges from the node for an expression to the nodes for its immediate subexpressions. That is, common subexpressions are represented uniquely. For example, the CFD A_i defined above viewed as a tree has 2^i leaves. Viewed as a diagram, however, A_i has only $i+1$ nodes but 2^i different paths from the root node to the leaf node. The size of a CFD D, denoted $|D|$, is defined to be the number of distinct subexpressions of D (including D itself). In other words, $|D|$ is the number of nodes in the diagram view of D. We will often use the word "node" as a synonym for "expression." We will also use the standard DAG notions of parent, child, and (directed) path for CFDs.

6. CFDs for MRFs

Here we define a CFD representation of the feasible set for the LBM constructed in Section 3. Consider the problem of computing $Z(M)$ for an MRF M as defined in Section 3. We assume that the variables of M have been given in a fixed order $y_1, y_2, \ldots, y_n$. The assignments to these variables form a tree whose root has a branch for each value of y_1, the next level branches for each value of y_2 and so on. As variables are assigned, however, the residual hypergraph defined by the energy terms often factors into disjoint sets of terms on disjoint sets of variables. So one can compute $Z(M)$ by factoring the residual problem when possible and, if no factoring is possible, casing out on the value of the next variable (after which more factoring may be possible). This "case-factor process" determines a set of subproblems. The nodes (subexpressions) in the CFD representation of the MRF correspond to the subproblems that arise in this way. Each such subproblem is defined by a subset Σ of the energy terms and a partial assignment ρ to (some of) the variables occurring in Σ.

More formally, consider a subset Σ of the energy terms of M. Let $V(\Sigma)$ be the set of variables on which some energy term in Σ depends, i.e., $V(\Sigma) = \bigcup_{k \in \Sigma} V_k$. Let ρ be a partial assignment of values to (some of) the variables in $V(\Sigma)$. Following Section 3, the variables of M need not be Boolean—a given variable may have any finite number of values. For each pair of such a subset Σ and partial assignment ρ we now define a CFD $D(\Sigma, \rho)$ using Boolean variables of the form $y = v$. The CFD for the full feasible constraint is $D(\Sigma(M), \emptyset)$ where $\Sigma(M)$ is the set of all energy terms in M and $\emptyset$ is the empty partial assignment. For a given partial assignment ρ we define a graph structure on the energy terms in Σ by saying that there is an edge between two energy terms if there is a variable not assigned a value by ρ on which both terms depend. The key to concise representation is to factor the problem when Σ becomes disconnected. We use the notation $\text{case}(\langle z_1, D_1\rangle, \langle z_2, D_2\rangle, \ldots, \langle z_m, D_m\rangle)$ as an abbreviation for $\text{case}(z_1, D_1, \text{case}(\langle z_2, D_2\rangle, \ldots, \langle z_n, D_n\rangle))$ where $\text{case}(\langle z, D\rangle)$ is $\text{case}(z, D, \text{empty})$. The CFD $D(\Sigma, \rho)$ is defined as follows.

(i) If Σ is disconnected under partial assignment ρ we have

$$D(\Sigma, \rho) = \text{factor}\big(D(\Sigma_1, \rho|_{V(\Sigma_1)}), \ldots, D(\Sigma_n, \rho|_{V(\Sigma_n)})\big)$$

where $\Sigma_1, \ldots, \Sigma_n$ are the connected components of Σ and $\text{factor}(D_1, D_2, \ldots, D_n)$ abbreviates $\text{factor}(D_1, \text{factor}(D_2, \ldots, D_n))$.

(ii) Otherwise, if Σ consists of a single energy term Ψ_k and ρ assigns values to all of $V(\Sigma)$, we have the following where $V_k = \{y_1, \ldots, y_m\}$ and $v_i = \rho(y_i)$,

$$D(\Sigma, \rho) = \text{case}(\text{“}k, y_1 = v_1, \ldots, y_m = v_m\text{”}, \text{unit}, \text{empty}).$$

(iii) Otherwise, let y be the earliest variable (under the given variable order) in $V(\Sigma)$ that is not in $\text{dom}(\rho)$. In this case we have the following where $\mathcal{D}(y) = \{v_1, \ldots, v_n\}$:

$$D(\Sigma, \rho) = \text{case}\begin{pmatrix} \langle \text{“}y = v_1\text{”}, D(\Sigma, \rho[y := v_1])\rangle, \\ \vdots \\ \langle \text{“}y = v_n\text{”}, D(\Sigma, \rho[y := v_n])\rangle \end{pmatrix}.$$

We now show that MRFs of small tree width have concise CFD representations. First we define the notion of tree width.

Definition 2. We consider a fixed variable order $y_1 \ldots y_n$. Consider i with $1 \leqslant i \leqslant n+1$. We say that a variable y_j is past at i if $j < i$ and is future at i if $j \geqslant i$. Intuitively, past variables have been assigned values and future variables are yet to be assigned. We define G_i to be the graph whose nodes are the energy terms of M and where two energy terms are connected by an edge if they both depend on the same future variable (at i). A connected component of G_i (which is a set of energy terms) will be called active if it contains at least one future variable (at i). The tree width of M under the given variable ordering is the maximum over all i of the maximum over all active connected components of G_i of the number of past variables in that component (at i). The tree width of M is the minimum over all orderings of the tree width relative to that ordering.

Note that an energy term in which all variables are past is not connected to any other energy term. This implies that the inactive components of G_i are singletons in which all variables are past. It also implies that every energy term in an active component must contain at least one future variable.

The above definition of tree width matches standard definitions, see [20] and the references therein. Rather than state standard definitions and prove the equivalence we will simply show that under the above definition we have that the tree width of a tree is 1. Suppose that every energy term involves two variables and the graph formed by the energy terms is a tree. Pick a root of the tree and consider a variable ordering that orders parents before children. Now suppose that Σ is an active connected component of G_i. Every energy term in an active Σ must contain a future variable. So in this case the energy terms in an active Σ form a tree every edge of which contains a future variable. In such a tree only the root variable can be past in an ordering that orders parents before children. So the number of past variables in an active component is at most one. We now have the following theorem.

Theorem 1. *Let w be the tree width of M under the given variable ordering. Then $|D(\Sigma(M), \emptyset)|$ is $O(Nd^{w+1})$ where N is the number of energy terms in M and $d = \max_i |\mathcal{D}(y_i)|$.*

Proof. The definition of $D(\Sigma, \rho)$ can be viewed as a set of rules for generating pairs $\langle \Sigma, \rho \rangle$ such that the CFD contains $D(\Sigma, \rho)$. We will call a pair $\langle \Sigma, \rho \rangle$ an anchor pair if there exists an i such that Σ is an active connected component of G_i and ρ assigns values to the past variables of Σ. We assume that $\Sigma(M)$ is a connected component of G_1 (where all variables are future) and that $\Sigma(M)$ contains variables so that $\langle \Sigma(M), \emptyset \rangle$ is an anchor pair. The set of Σ such that there exists an i such that Σ is a connected component of G_i form a tree—as i increases connected components split ultimately terminating in inactive singleton sets. The number of nodes in a tree is not more than twice the number of leaves minus one. Therefore the number of Σ that can appear in anchor pairs is at most $2N - 1$. For a given anchor pair $\langle \Sigma, \rho \rangle$, let $i(\Sigma)$ be the greatest index such that Σ is a connected component of G_i. There are at most w variables in Σ that are past at time $i(\Sigma)$. In any anchor pair $\langle \Sigma, \rho \rangle$ we have that there exists a $j \leqslant i(\Sigma)$ such that ρ assigns values to the variables in Σ that are before j. The set of ρ satisfying this property forms a tree with at most d^w leaves. Again, since the number of nodes is no larger than twice the number of leaves we have that the number of such assignments ρ is $O(d^w)$. Therefore the total number of anchor pairs in the CFD is $O(Nd^w)$. But each anchor pair generates a certain set of intermediate nodes in the CFD before generating other anchor pairs. The number of intermediate nodes is bounded by the number of triples of the form $\langle \Sigma, \rho[y = v], \Sigma' \rangle$ where $\langle \Sigma, \rho \rangle$ is an anchor pair and Σ' is a component of Σ under $\rho[y = v]$. In each such triple we either have that $\Sigma' = \Sigma$ or we have that Σ and Σ' form an edge in the graph of possible sets Σ. The number of edges in a tree is no larger than twice the number of leaves. Hence we have that the number of pairs $\langle \Sigma, \Sigma' \rangle$ appearing in these triples is $O(N)$. For a given Σ, the number of assignments of the form $\rho[y = v]$ is $O(d^{w+1})$. So the number of such triples is $O(Nd^{w+1})$. □

7. CFDs for parsing

Here we construct a CFD for the feasible set of the LBM defined in Section 4 for a grammar G. We define the CFD $D(\text{“}X_{i,k}\text{”})$ such that the assignments in $F(D(\text{“}X_{i,k}\text{”}))$ are in one-to-one correspondence with the parse trees of

the span from i to $k-1$ with root nonterminal X. The CFD representing the full feasible set of parses is $D(\text{“}S_{1,n+1}\text{”})$. First we define $D(\text{“}X_{i,k}\text{”})$ as follows where $B(\text{“}X_{i,k}\text{”})$ represents the consequences of making “$X_{i,k}$” true:

$$D(\text{“}X_{i,k}\text{”}) = \mathsf{case}\big(\text{“}X_{i,k}\text{”}, B(\text{“}X_{i,k}\text{”}), \mathsf{empty}\big).$$

For $k > i+1$ we define the consequences $B(\text{“}X_{i,k}\text{”})$ as follows using the multi-branch case notation defined in Section 6:

$$B(\text{“}X_{i,k}\text{”}) = \mathsf{case}\big(\langle b_1, B(b_1)\rangle, \ldots, \langle b_n, B(b_n)\rangle\big)$$

where the variables b_p are all possible branch variables of the form “$X_{i,k} \to Y_{i,j} Z_{j,k}$,” and $B(\text{“}X_{i,k} \to Y_{i,j} Z_{j,k}\text{”}) = \mathsf{factor}(D(\text{“}Y_{i,j}\text{”}), D(\text{“}Z_{j,k}\text{”}))$.

Finally, if a_i is the ith input symbol, we have

$$B(\text{“}X_{i,i+1}\text{”}) = \begin{cases} \mathsf{case}(\text{“}X_{i,i+1} \to a_i\text{”}, \mathsf{unit}, \mathsf{empty}) & \text{if } X \to a_i \in G, \\ \mathsf{empty} & \text{otherwise.} \end{cases}$$

This construction has the property that $|D(\text{“}S_{1,n+1}\text{”})|$ is $O(|G|n^3)$ where $|G|$ is the number of productions in the grammar.

8. The importance of zero suppression

We can define the feasible set of parse trees for a given grammar and given input string directly as an MRF on the Boolean variables introduced in Section 4. In particular, we can define an MRF with hard constraint energy terms (energy terms that either have infinite energy or zero energy) expressing the following constraints.

(i) “$S_{1,n+1}$” is true.
(ii) If “$X_{i,k}$” is true for $k > i+1$ then there exists “$Y_{i,j}$” and “$Z_{j,k}$” where the grammar contains $X \to YZ$ and “$X_{i,k} \to Y_{i,j} Z_{j,k}$” is true.
(iii) If “$X_{i,k} \to Y_{i,j} Z_{j,k}$” is true then “$Y_{i,j}$” and “$Z_{j,k}$” are both true.
(iv) If “$X_{i,i+1}$” is true then the grammar must contain $X \to a$ where a is the nth symbol in the input string and “$X_{i,i+1} \to a$” is true.
(v) If “$X_{i,k} \to Y_{i,j} Z_{j,k}$” is true then no other variable of the form “$X_{i,k} \to W_{i,j'} U_{j',k}$” is true.
(vi) If “$Y_{i,j}$” is true, and is different from “$S_{1,n+1}$,” then either some variable of the form “$X_{i,k} \to Y_{i,j} Z_{j,k}$” is true or some variable of the form “$X_{k,j} \to Z_{k,i} Y_{i,j}$” is true.

Constraints 5 and 6 are not implied by 1, 2, 3 and 4. These constraints can be expressed with a SAT problem (a set of disjunctive clauses) where constraint 5 requires $O(G^2 n^4)$ clauses. We can then take the resulting MRF and compile it into an **and/or** graph [12] or an algebraic expression [10]. But in this approach both the MRF representation and the compiled form are too large. In the **and/or** graph representation we have **or** nodes representing the choice points corresponding to constraint 2 above. Each branch of an **or** node produces an **and** node. However, without zero suppression (without context-sensitive variable existence), each **and** node must list all the variables that become *false* at that node. For each of the $O(n^3)$ **and** nodes a cubic number of variables become false giving $O(n^6)$ edges in the **and/or** graph.

9. CFDs for edit distance

Pair HMMs [21] and weighted finite-state machines [22,23] have been used to represent trainable weighted edit distance models in text processing and computational biology. As another example of the expressive power of CFDs, we show here how to construct a CFD for the weighted edit distance problem. We will start with a simple context-independent edit cost model, and then indicate how to extend it for context-sensitive edit costs. Consider two strings $a = a_1 \cdots a_m$ and $b = b_1 \cdots b_n$ over a given alphabet V. We view b as being derived from a by insertions, deletions and substitutions. We use the Boolean variable $X_{i,j}$, with $0 \leqslant i \leqslant m$ and $0 \leqslant j \leqslant n$, to represent the statement that the j-long prefix of ${}_0b_j$ was derived by editing the i-long prefix of ${}_0a_i$. We require the level statement $X_{m,n}$ to be true—the string b is derived by editing the string a. In addition, we define the following edit variables:

- $A_{i,j,x}$ states that ${}_0b_j$ is derived from ${}_0a_i$ by first deleting the symbol $x \in V$ at position i in a and then deriving ${}_0b_j$ from ${}_0a_{i-1}$.
- $B_{i,j,x}$ states that ${}_0b_j$ is derived from ${}_0a_i$ by first inserting the symbol $x \in V$ at position j in b and then deriving ${}_0b_{j-1}$ from ${}_0a_i$.
- $S_{i,j,x,y}$ states that ${}_0b_j$ is derived from ${}_0a_i$ by substituting $x \in V$ at position j in b for $y \in V$ at position i in a and then deriving ${}_0b_{j-1}$ from ${}_0a_{i-1}$.

Then we define the CFD $D(\text{“}X_{i,j}\text{”})$ as follows:

$$D(\text{“}X_{i,j}\text{”}) = \begin{cases} \mathsf{case}(\text{“}X_{i,j}\text{”}, E(\text{“}X_{i,j}\text{”}), \mathsf{empty}) & \text{if } i+j>0, \\ \mathsf{unit} & \text{otherwise,} \end{cases}$$

$$E(\text{“}X_{i,j}\text{”}) = \mathsf{case}\Big(\big\langle e_x, E(e_x)\big\rangle_{x\in V}, \big\langle f_y, E(f_y)\big\rangle_{y\in V}, \big\langle s_{xy}, E(x_{xy})\big\rangle_{x\in V, y\in V}\Big)$$

where $e_x = \text{“}A_{i,j,x}\text{,”}$ $f_y = \text{“}B_{i,j,y}\text{,”}$ and $s_{xy} = \text{“}S_{i,j,x,y}\text{.”}$

Finally, we define

$$E(\text{“}A_{i,j,x}\text{”}) = \begin{cases} D(\text{“}X_{i-1,j}\text{”}) & \text{if } i>0, x=a_i, \\ \mathsf{empty} & \text{otherwise,} \end{cases}$$

$$E(\text{“}B_{i,j,y}\text{”}) = \begin{cases} D(\text{“}X_{i,j-1}\text{”}) & \text{if } j>0, y=b_j, \\ \mathsf{empty} & \text{otherwise,} \end{cases}$$

$$E(\text{“}S_{i,j,x,y}\text{”}) = \begin{cases} D(\text{“}X_{i-1,j-1}\text{”}) & \text{if } i>0, j>0, x=a_i, y=b_j, \\ \mathsf{empty} & \text{otherwise.} \end{cases}$$

It is easy to see that $|D(\text{“}X_{m,n}\text{”})| = O(|V|^2 mn)$, in agreement with the standard dynamic program for computing the best alignment. From this CFD, we can immediately build an LBM for weighted edit distance by setting

$$\Psi(\text{“}X_{i,j}\text{”}) = 0,$$

$$\Psi(\text{“}A_{i,j,x}\text{”}) = x \text{ deletion cost},$$

$$\Psi(\text{“}B_{i,j,y}\text{”}) = y \text{ insertion cost},$$

$$\Psi(\text{“}S_{i,j,x,y}\text{”}) = \text{cost of substituting } x \text{ for } y.$$

Context-dependent edits can be implemented by subscripting variables with different classes of edit contexts, and enforcing context-class constraints appropriate in the CFD.

10. Inference on CFD models

A *CFD model* $\langle D, \Psi\rangle$ is an LBM whose feasible set is defined by a CFD D and whose energy function Ψ assigns costs to the variables of D. We will now present the main inference algorithms on CFDs.

The inside algorithm. We first consider the problem of computing $Z(F(D), \Psi)$ as defined by Eq. (3). Here we write $Z(D, \Psi)$ as an abbreviated form of $Z(F(D), \Psi)$. It turns out that $Z(D, \Psi)$ can be computed by recursive descent on subexpressions of D using the following equations:

$$Z\big(\mathsf{case}(x, D_1, D_2), \Psi\big) = e^{-\Psi(x)} Z(D_1, \Psi) + Z(D_2, \Psi),$$

$$Z\big(\mathsf{factor}(D_1, D_2), \Psi\big) = Z(D_1, \Psi) Z(D_2, \Psi),$$

$$Z(\mathsf{unit}, \Psi) = 1,$$

$$Z(\mathsf{empty}, \Psi) = 0.$$

The correctness of these equations can be proved by induction on the size of D. By caching these computations for each subexpression of D, these equations give a way of computing $Z(D, \Psi)$ in time proportional to $|D|$. These equations are analogous to the inside algorithm used in statistical parsing.

The Viterbi Algorithm. Next we consider the problem of computing minimum energy over the elements of $F(D)$. In particular we define $\Psi^*(D,\Psi)$ as follows:

$$\Psi^*(D,\Psi) = \min_{\rho\in F(D)} \Psi(\rho).$$

We can compute $\Psi^*(D,\Psi)$ using the following equations:

$$\Psi^*\big(\mathsf{case}(z, D_1, D_2), \Psi\big) = \min\begin{pmatrix} \Psi(z) + \Psi^*(D_1,\Psi) \\ \Psi^*(D_2,\Psi)\end{pmatrix},$$
$$\Psi^*\big(\mathsf{factor}(D_1, D_2), \Psi\big) = \Psi^*(D_1,\Psi) + \Psi^*(D_2,\Psi),$$
$$\Psi^*(\mathsf{unit}, \Psi) = 0,$$
$$\Psi^*(\mathsf{empty}, \Psi) = +\infty.$$

Again the correctness of these equations can be proved by a direct induction on the size of D. These equations can easily be modified to also compute a truth assignment that achieves the minimum energy. This is a truth assignment of highest probability.

Marginals. Next we consider the problem of computing marginal probabilities of the form $P(z=1|D,\Psi,\sigma)$ where σ is a partial truth assignment that fixes the values of some of the CFD model variables. We will show that these marginals can be computed in time proportional to $|D||\mathrm{dom}(\sigma)|$.

The marginal $P(z=1|D,\Psi,\sigma)$ can be written as follows:

$$P(z|D,\Psi,\sigma) = \frac{Z(D,\Psi,\sigma[z:=1])}{Z(D,\Psi,\sigma)},$$
$$Z(D,\Psi,\sigma) = \sum_{\rho\in F(D):\,\sigma\sqsubseteq\rho} e^{-\Psi(\rho)}.$$

So it suffices to be able to compute $Z(D,\Psi,\sigma)$. We now define the auxiliary quantity $Z'(D,\Psi,\sigma) = Z(D,\Psi,\sigma|_{V(D)})$. Our procedure computes $Z(D,\Psi,\sigma)$ by computing $Z'(D',\Psi,\sigma)$ for all subnodes D' of D. Note that the number of such values is $|D|$. The Z' values satisfy the following equations for factor, unit and empty expressions:

$$Z'\big(\mathsf{factor}(D_1, D_2), \Psi, \sigma\big) = Z'(D_1,\Psi,\sigma)Z'(D_2,\Psi,\sigma),$$
$$Z'(\mathsf{unit}, \Psi, \sigma) = 1,$$
$$Z'(\mathsf{empty}, \Psi, \sigma) = 0.$$

Computing Z' on case expressions is more subtle. We now have the following equation where $Z(v, D, D', \Psi, \sigma)$ is defined below:

$$Z'\big(\mathsf{case}(z, D_1, D_2), \Psi, \sigma\big) = \begin{cases} e^{-\Psi(z)} Z(z, D, D_1, \Psi, \sigma) & \text{if } \sigma(z) = 1, \\ Z(z, D, D_2, \Psi, \sigma) & \text{if } \sigma(z) = 0, \\ e^{-\Psi(z)} Z(z, D, D_1, \Psi, \sigma) + Z(z, D, D_2, \Psi, \sigma) & \text{otherwise.}\end{cases}$$

$Z(z, D, D', \Psi, \sigma)$ expresses the constraint that omitted variables default to 0 in CFDs. If there exists $z' \neq z$ with $\sigma(z') = 1$ where z' occurs in D but not in D' then $Z(z, D, D', \Psi, \sigma) = 0$, otherwise $Z(z, D, D', \Psi, \sigma) = Z'(D', \Psi, \sigma)$.

To analyze the running time of computing $Z(D,\Psi,\sigma)$ we first note that there are a linear number of values needed of the form $Z(z, D', D'', \Psi, \sigma)$. Assuming unit time hash table operations, it is possible to cache the answer to all queries of the form $z \in D'$, for $z' \in \mathrm{dom}(\sigma)$ and D' a node in D, in $O(|D||\sigma|)$ time. Given this cache, each call to $Z(z, D, D', \Psi, \sigma)$ can be computed in time proportional to $|\sigma|$. So the overall computation takes time proportional to $|D||\sigma|$.

The inside–outside algorithm. Using the above conditional probability algorithm to compute $P(z = 1 \mid D, \Psi)$ for all variables z can take $\Omega(|D|^2)$ time. However, a generalization of the inside–outside algorithm can be used to simultaneously compute $P(z=1|D,\Psi)$ for all variables z in D in $O(|D|)$ time. The value $Z(D,\Psi)$ is the "inside"

value associated with D. Intuitively, the outside value of a node in a CFD is the total weight of the "contexts" in which that node appears. We write $D' \preccurlyeq D$ to state that node D' occurs in D where we take a node to occur in itself ($D \preccurlyeq D$). For a given top level CFD D_{top} and for $D \preccurlyeq D_{\text{top}}$ we define the *outside value* $O(D, D_{\text{top}}, \Psi)$ of D (in D_{top}) as follows. First define $O(D_{\text{top}}, D_{\text{top}}, \Psi) = 1$. For $D \neq D_{\text{top}}$ we define $O(D, D_{\text{top}}, \Psi)$ as follows:

$$\begin{aligned} O(D, D_{\text{top}}, \Psi) = &\sum_{\mathsf{case}(z, D, D') \preccurlyeq D_{\text{top}}} O\big(\mathsf{case}(z, D, D'), D_{\text{top}}, \Psi\big) e^{-\Psi(z)} \\ &+ \sum_{\mathsf{case}(z, D', D') \preccurlyeq D_{\text{top}}} O\big(\mathsf{case}(z, D', D), D_{\text{top}}, \Psi\big) \\ &+ \sum_{\mathsf{factor}(D', D') \preccurlyeq D_{\text{top}}} O\big(\mathsf{factor}(D, D'), D_{\text{top}}, \Psi\big) Z(D', \Psi) \\ &+ \sum_{\mathsf{factor}(D', D) \preccurlyeq D_{\text{top}}} O\big(\mathsf{factor}(D', D), D_{\text{top}}, \Psi\big) Z(D', \Psi). \end{aligned} \tag{6}$$

Once the inside value of every node has been computed, these equations allows the outside values to be computed from the top down, i.e., starting from $O(D_{\text{top}}, D_{\text{top}}, \Psi) = 1$. Note that in this recursion the top level CFD D_{top} does not change. We will write $O(D, \Psi)$ for $O(D, D_{\text{top}}, \Psi)$ when D_{top} is clear from context. Since D_{top} does not change, this top-down calculation can be done in time proportional to the number of nodes. Finally we can compute $P(z = 1 | D_{\text{top}}, \Psi)$ using the following theorem.

Theorem 2.

$$P(z = 1 | D_{\text{top}}, \Psi) = \frac{Z(D_{\text{top}}, \Psi, \emptyset[z := 1])}{Z(D_{\text{top}}, \Psi)},$$

$$Z\big(D_{\text{top}}, \Psi, \emptyset[z := 1]\big) = \sum_{\mathsf{case}(z, D, D') \preccurlyeq D_{\text{top}}} \begin{pmatrix} O(\mathsf{case}(z, D, D'), \Psi) \\ e^{-\Psi(z)} \\ Z(D, \Psi) \end{pmatrix}.$$

Proof. First we introduce a slight change of notation so as to put the equations in a more standard form for exponential models. Recall that the energy of an assignment ρ is defined as follows:

$$\Psi(\rho) = \sum_x \Psi(x) \rho(x).$$

We can think of ρ as a vector $\vec{x}$ with components $x_1, \ldots, x_n$ and we can rewrite $\Psi(\rho)$ as $\Psi(\vec{x})$ as follows:

$$\Psi(\vec{x}) = \sum_{i=1}^{n} \Psi_i x_i.$$

Here we should think of Ψ as a weight vector with components Ψ_i. We can write $Z(D_{\text{top}}, \Psi)$ as follows:

$$Z(D_{\text{top}}, \Psi) = \sum_{\vec{x} \in F(D_{\text{top}})} e^{-\sum_i \Psi_i x_i},$$

$$\frac{\partial Z(D_{\text{top}}, \Psi)}{\partial \Psi_i} = \sum_{\vec{x} \in F(D_{\text{top}})} -x_i e^{-\sum_i \Psi_i x_i} = -Z\big(D_{\text{top}}, \Psi, \emptyset[x_i = 1]\big).$$

So to compute $Z(D_{\text{top}}, \Psi, \emptyset[x_i = 1])$ it now suffices to compute $\partial Z / \partial \Psi_i$. We now compute $\partial Z / \partial \Psi_i$ by application of the chain rule to the rules for calculating $Z(D_{\text{top}}, \Psi)$. This is analogous to the use of the chain rule in computing partial derivatives in backpropagation for tuning the weights of a neural network. We can think of $Z(D_{\text{top}}, \Psi)$ as a function of $Z(D, \Psi)$—we expand the inside equation for $Z(D', \Psi)$ at every node $D' \neq D$ with $D \preccurlyeq D' \preccurlyeq D_{\text{top}}$. This function

from $Z(D, \Psi)$ to $Z(D_{\text{top}}, \Psi)$ can be differentiated yielding a well-defined value for $\partial Z(D_{\text{top}}, \Psi)/\partial Z(D, \Psi)$. We will now show the following:

$$O(D, D_{\text{top}}, \Psi) = \frac{\partial Z(D_{\text{top}}, \Psi)}{\partial Z(D, \Psi)}. \tag{7}$$

The proof is by induction on the maximum depth at which D occurs in D_{top}. For the base case we have $D = D_{\text{top}}$ and (7) follows from $\partial Z(D_{\text{top}}, \Psi)/\partial Z(D_{\text{top}}, \Psi) = 1$. For the induction case we can assume (7) for all shallower nodes. We then have the following where W ranges over all the parents of D

$$\frac{\partial Z(D_{\text{top}}, \Psi)}{\partial Z(D, \Psi)} = \sum_W \left(\frac{\partial Z(D_{\text{top}}, \Psi)}{\partial Z(W, \Psi)} \right) \left(\frac{\partial Z(W, \Psi)}{\partial Z(D, \Psi)} \right).$$

Equation (6) lists the four possible types of parents of D. We consider the case where W is $\mathsf{case}(z, D, D')$. In this case we have the following:

$$\begin{aligned}
&\frac{\partial Z(D_{\text{top}}, \Psi)}{\partial Z(W, \Psi)} = O\big(\mathsf{case}(z, D, D'), \Psi\big), \\
&Z(W, \Psi) = e^{-\Psi(z)} Z(D, \Psi) + Z(D', \Psi), \\
&\frac{\partial Z(W, \Psi)}{\partial Z(D, \Psi)} = e^{-\Psi(z)}.
\end{aligned}$$

These equations imply the first line in (6) covers this form of parent W. The second line (6) similarly covers parents of the form $\mathsf{case}(z, D'', D')$. Now suppose that the parent W has the form $\mathsf{factor}(D, D')$. In this case we have the following:

$$\begin{aligned}
&\frac{\partial Z(D_{\text{top}}, \Psi)}{\partial Z(W, \Psi)} = O\big(\mathsf{factor}(D, D'), D\big), \\
&Z(W, \Psi) = Z(D, \Psi) Z(D', \Psi), \\
&\frac{\partial Z(W, \Psi)}{\partial Z(D, \Psi)} = Z(D', \Psi).
\end{aligned}$$

These equations imply that the third line in (6) properly handles parents of the form $\mathsf{factor}(D, D')$. The analysis of the last line in (6) is similar.

Finally, it suffices to show that the right-hand side of the second equation in Theorem 2 equals $-\partial Z/\partial \Psi(z)$. We have the following where W now ranges over all parents of the variable z:

$$\frac{\partial Z(D_{\text{top}}, \Psi)}{\partial \Psi(z)} = \sum_W \left(\frac{\partial Z(D_{\text{top}}, \Psi)}{\partial Z(W, \Psi)} \right) \left(\frac{\partial Z(W, \Psi)}{\partial \Psi(z)} \right).$$

The parents of the variable z are exactly the nodes of the form $\mathsf{case}(z, D, D')$ for which we have the following:

$$\begin{aligned}
&Z\big(\mathsf{case}(z, D, D'), \Psi\big) = e^{-\Psi(z)} Z(D, \Psi) + Z(D', \Psi), \\
&\left(\frac{\partial Z(\mathsf{case}(z, D, D'), \Psi)}{\partial \Psi(z)} \right) = -e^{-\Psi(z)} Z(D, \Psi).
\end{aligned}$$

These equations imply that the right-hand side of the second equation in Theorem 2 equals $-\partial Z/\partial \Psi(z)$ as desired. □

11. Conclusions

We have described a class of structured probabilistic models based on case-factor diagrams. We have also shown that for a given a weighted context-free grammar G and input string x the conditional probability $P(y|x)$ can be represented by a CFD model with $O(|G|n^3)$ nodes. We have also shown that any MRF with tree width w in which variables have V possible values and with N energy terms can be represented by a CFD model with $O(NV^w)$ nodes. We have shown that for an arbitrary CFD model, computing the partition function, most likely variable assignment,

and the probability of each Boolean variable, can all be done in time linear in the number of nodes. We believe that CFD models will provide a common language for specifying algorithms and stating theorems that can play for structured probabilistic models a similar role to that of BDDs in Boolean inference problems.

References

[1] J. Lafferty, A. McCallum, F. Pereira, Conditional random fields: Probabilistic models for segmenting and labeling sequence data, in: Proc. 18th International Conf. on Machine Learning, ICML, Morgan Kaufmann, San Francisco, CA, 2001, pp. 282–289.
[2] K. Kanazawa, D. Koller, S. Russell, Stochastic simulation algorithms for dynamic probabilistic networks, in: Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence, UAI, 1999, pp. 346–351.
[3] D. Poole, Probabilistic horn abduction and Bayesian networks, Artificial Intelligence 64 (1) (1993) 81–129.
[4] N. Friedman, L. Getoor, D. Koller, A. Pfeffer, Learning probabilistic relational models, in: Proceedings of the 16th International Joint Conference on Artificial Intelligence, IJCAI, 1999, pp. 1300–1309.
[5] R.E. Bryant, Graph-based algorithms for boolean function manipulation, IEEE Trans. Comput. C 35 (8) (1986) 677–691.
[6] S. Minato, Zero-suppressed BDDs for set manipulation in combinatorial problems, in: Proc. of 30th ACM/IEEE Design Automation Conference, DAC '93, ACM Press, 1993, pp. 272–277.
[7] K.L. McMillan, Hierarchical representation of discrete functions, with application to model checking, in: Computer Aided Verification, CAV, 6th International Conference, 1994, pp. 41–54.
[8] A. Darwiche, Recursive conditioning, Artificial Intelligence 125 (1–2) (2001) 5–41.
[9] D. Allen, A. Darwiche, New advances in inference by recursive conditioning, in: Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence, UAI, 2003, pp. 2–10.
[10] A. Darwiche, A differential approach to inference in Bayesian networks, J. ACM (2003) 280–305.
[11] C. Boutilier, N. Friedman, M. Goldszmidt, D. Koller, Context-specific independence in Bayesian networks, in: Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence, UAI, 1996, pp. 115–123.
[12] R. Dechter, And/or search spaces for graphical models, ICS Technical Report, Artificial Intelligence (March 2004), in press.
[13] M. Jaeger, Probabilistic decision graphs: Combining verification and AI techniques for probabilistic inference, Internat. J. Uncertain. Fuzzyness Knowledge-Based Systems 12 (2004) 19–42.
[14] F. Bacchus, S. Dalmao, T. Pitassi, Algorithms and complexity results for #sat and Bayesian inference, in: Proceedings of the 44th Annual IEEE Symposium on the Foundations of Computer Science, FOCS, 2003, pp. 340–351.
[15] F. Bacchus, S. Dalmao, T. Pitassi, Value elimination: Bayesian inference via backtracking search, in: Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence, UAI, 2003, pp. 20–28.
[16] B. Taskar, C. Guestrin, D. Koller, Max-margin Markov networks, in: Proceedings of the 17th Annual Conference on Neural Information Processing Systems Conference, NIPS, 2003, a version will appear in J. Mach. Learn. Res.
[17] M. Collins, Parameter estimation for statistical parsing models: Theory and practice of distribution-free methods, in: H. Bunt, J. Carroll, G. Satta (Eds.), New Developments in Parsing Technology, Kluwer, 2004, revised version of the paper that appeared at IWPT 2001.
[18] Y. Altun, T. Hofmann, Large margin methods for label sequence learning, in: 8th European Conference on Speech Communication and Technology, EuroSpeech, 2003.
[19] S. Della Pietra, V. Della Pietra, J. Lafferty, Inducing features of random fields, IEEE Trans. Pattern Anal. Mach. Intell. 19 (4) (1997) 380–393.
[20] H.L. Bodlaender, A linear time algorithm for finding tree-decompositions of small treewidth, SIAM J. Comput. 25 (1996) 1305–1317.
[21] R. Durbin, S. Eddy, A. Krogh, G. Mitchison, Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids, Cambridge University Press, 1998.
[22] E.S. Ristad, P.N. Yianilos, Learning string edit distance, IEEE Trans. Pattern Anal. Mach. Intell. 20 (5) (1998) 522–532.
[23] J. Eisner, Parameter estimation for probabilistic finite-state transducers, in: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, ACL, 2002, pp. 1–8.