On the Role of Partial Differentiation in Probabilistic Inference

Adnan Darwiche
Computer Science Department
University of California
Los Angeles, Ca 90095
darwiche@cs.ucla.edu

Abstract

We present in this paper one of the simplest, yet most comprehensive frameworks for inference in Bayesian networks. According to this framework, one compiles a Bayesian network into a polynomial — in which variables correspond to potential evidence and network parameters — and then computes the partial derivatives of this polynomial with respect to each variable. Once such derivatives are made available, one can compute in constant-time answers to a large class of probabilistic queries, which are central to classical inference, parameter estimation, model validation and sensitivity analysis. We show a number of key results relating to this framework. First, given a Bayesian network with n variables and an elimination order of width w, we present an elimination algorithm for compiling the polynomial in $O(n \exp(w))$ time and space. Next, given some evidence and parameter setting, we show that the compiled polynomial can be evaluated, and all its first partial derivatives computed simultaneously, in $O(n \exp(w))$ time and space. Once these derivatives are made available, we show that one can compute in constant-time: the posterior marginal of any network variable or family, the probability of evidence after having retracted the value of any evidence variable, and the sensitivity of evidence probability to change in any network parameter. Finally, we show that second partial derivatives can all be computed simultaneously in $O(n^2 \exp(w))$ time and space. Moreover, once such derivatives are made available, one can compute in constant-time a variety of sophisticated queries, such as context-specific independence between any pair of variables, and the amount of change to some network parameter which is needed to ensure a particular ranking on some hypotheses. The proposed framework provides new insights into the role of partial differentiation in probabilistic inference. Moreover, its combined simplicity, comprehensiveness and computational complexity appear to be unique among existing frameworks for inference in Bayesian networks.

Introduction

Consider the probability table on the left of Figure 1 and suppose that our goal is to compute probabilities of events with respect to this table, say $Pr(a)$, $Pr(a, \bar{b})$,

A	B	ϕ
true	true	.03
true	false	.27
false	true	.56
false	false	.14

A	B	ϕ
true	true	$\lambda_a \lambda_b .03$
true	false	$\lambda_a \lambda_{\bar{b}} .27$
false	true	$\lambda_{\bar{a}} \lambda_b .56$
false	false	$\lambda_{\bar{a}} \lambda_{\bar{b}} .14$

Figure 1: A probability table and its parameterization.

etc.[1] We can do this in the usual way. To compute $Pr(\mathbf{e})$ we simply identify the rows in the table which are consistent with $\mathbf{e}$ and sum up their probabilities. Alternatively, we can *compile* the probability table as follows. We *parameterize* the table as shown on the right of Figure 1, where we introduce what we call *evidence indicators* into each row. We then add up all rows to generate the multivariate polynomial:

$$\mathcal{F}(\lambda_{\bar{a}}, \lambda_{\bar{b}}, \lambda_b, \lambda_a) = .03\lambda_a\lambda_b + .27\lambda_a\lambda_{\bar{b}} + .56\lambda_{\bar{a}}\lambda_b + .14\lambda_{\bar{a}}\lambda_{\bar{b}},$$

which is depicted graphically in Figure 2.

Now, given any evidence $\mathbf{e}$, we can compute the probability of $\mathbf{e}$ by a simple evaluation of polynomial $\mathcal{F}$. We consider each indicator λ_x. If x is consistent with evidence $\mathbf{e}$, we set the value of variable λ_x to 1. Otherwise, we set its value to 0. We then evaluate the polynomial $\mathcal{F}$, where its value is guaranteed to be the probability of evidence $\mathbf{e}$. Basically, all we are doing is using the indicators to select which rows of the probability table to add together. Figure 3 depicts some examples of evidence and the corresponding values of indicators. Figure 4 shows an example evaluation to compute the probability of a.

We have promoted the compilation of Bayesian networks in earlier work, where we referred to functions

[1] We are using the standard notation: variables are denoted by upper–case letters (A) and their values by lower–case letters (a). Sets of variables are denoted by bold–face upper–case letters ($\mathbf{A}$) and their instantiations are denoted by bold–face lower–case letters ($\mathbf{a}$). For a variable A with values true and false, we use a to denote A= true and $\bar{a}$ to denote A= false. Finally, for a family $\mathbf{F} = \{X\} \cup \mathbf{U}$, we use $\theta_{\mathbf{f}}$ and $\theta_{x\mathbf{u}}$ to denote the network parameter corresponding to $Pr(x \mid \mathbf{u})$ where $\mathbf{f} = x\mathbf{u}$.

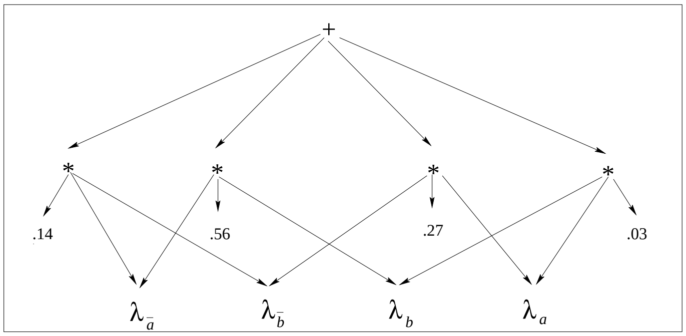


Figure 2: A probability-table compilation.

Evidence $\mathbf{e}$	λ_a	$\lambda_{\bar{a}}$	λ_b	$\lambda_{\bar{b}}$
$a \wedge b$	1	0	1	0
$a \wedge \bar{b}$	1	0	0	1
a	1	0	1	1
true	1	1	1	1

Figure 3: Evidence and corresponding indicators.

such as $\mathcal{F}$ as Query DAGs (Darwiche & Provan 1997). Our original motivation for compiling Bayesian networks was to simplify inference systems to the point where they could be implemented cost-effectively on a variety of software and hardware platforms. This paper is based on a number of new results, which make this notion of compilation much more interesting and useful than was originally conceived. Specifically, if the probability table is specified using a Bayesian network, then we can construct the polynomial $\mathcal{F}$ using a simple variable elimination algorithm in $O(n \exp(w))$ time and space, where n is the number of network variables and w is the width of some supplied elimination order. Moreover,

(a) We can compute all partial derivatives $\partial\mathcal{F}/\partial\lambda_x$ and $\partial\mathcal{F}/\partial\theta_{\mathbf{f}}$ simultaneously in time and space which are linear in the size of $\mathcal{F}$ (that is, in $O(n \exp(w))$) where $\theta_{\mathbf{f}}$ denotes a network parameter.

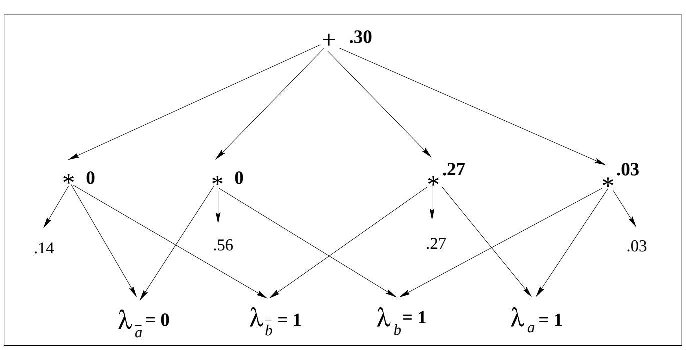


Figure 4: Evaluating a probability-table compilation under evidence $\mathbf{e} = a$.

(b) We can compute all second partial derivatives, such as $\partial\mathcal{F}^2/\partial\lambda_x\partial\lambda_y$, simultaneously in $O(n^2 \exp(w))$ time and space.[2]

Finally, once these derivatives are available, we can compute in constant time:

1. the posterior marginal of any network variable X, $Pr(x \mid \mathbf{e})$;
2. the posterior marginal of any network family $\{X\} \cup \mathbf{U}$, $Pr(x, \mathbf{u} \mid \mathbf{e})$;
3. the sensitivity of $Pr(\mathbf{e})$ to change in any network parameter $\theta_{\mathbf{f}}$;
4. the probability of evidence $\mathbf{e}$ after having flipped the value of any variable E, $Pr(e, \mathbf{e} - E)$;[3]
5. the posterior marginal of any variable E after having retracted evidence on E, $Pr(e \mid \mathbf{e} - E)$.
6. the posterior marginal of any pair of network variables X and Y, $Pr(x, y \mid \mathbf{e})$;
7. the impact of changing one network parameter $\theta_{\mathbf{f}_1}$ on the sensitivity of $Pr(\mathbf{e})$ to change in another parameter $\theta_{\mathbf{f}_2}$; and
8. the amount of change in parameter $\theta_{\mathbf{f}}$ needed to ensure that $Pr(y \mid \mathbf{e}) \leq Pr(\bar{y} \mid \mathbf{e})$ for some hypothesis variable Y.

The above queries represent the basis of inference, parameter estimation, model validation, and sensitivity analysis in Bayesian networks. Each of these queries can be answered in constant time after having computed first and second partial derivatives, which can be done in $O(n^2 \exp(w))$ time and space. Queries (1)-(5) require first partial derivatives only, which can be computed in only $O(n \exp(w))$ time and space. The remaining queries require second partial derivatives.

The computation of the above queries based on partial derivatives will be detailed in a following section. But to give a hint of how such a large number of queries can be answered based on partial derivatives, we provide a simple example next.

Consider the polynomial $\mathcal{F}(\lambda_{\bar{a}}, \lambda_{\bar{b}}, \lambda_b, \lambda_a)$ in Figure 4, where $\mathcal{F}(\lambda_{\bar{a}} = 0, 1, 1, 1) = .30$, meaning that the probability of evidence $\mathbf{e} = a$ is .30. To compute the probability $Pr(b, a)$, we need the value of $\mathcal{F}$ after having flipped the value of $\lambda_{\bar{b}}$ from 1 to 0: $\mathcal{F}(\lambda_{\bar{a}} = 0, \lambda_{\bar{b}} = 0, 1, 1)$. Since the polynomial is linear in $\lambda_{\bar{b}}$, we can compute this value from the current value of $\mathcal{F}$ and its partial derivative with respect to $\lambda_{\bar{b}}$:

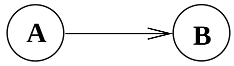
$$\mathcal{F}(0,0,1,1) = \mathcal{F}(0,1,1,1) + (-1)\partial\mathcal{F}(0,1,1,1)/\partial\lambda_{\bar{b}},$$

since $\lambda_{\bar{b}}$ has changed from 1 to 0. In fact, the partial derivative is .27 in this case and, hence,

$$\mathcal{F}(0,0,1,1) = .30 - .27 = .03 = Pr(a, b).$$

[2] This assumes that the size of each network family is bounded by a constant, in which case the total number of network parameters is $O(n)$.

[3] We define this notation formally later.



A	ϕ_A
true	.3
false	.7

A	ϕ_A
true	$\theta_a\lambda_a$
false	$\theta_{\bar{a}}\lambda_{\bar{a}}$

A	B	ϕ_B
true	true	.1
true	false	.9
false	true	.8
false	false	.2

A	B	ϕ_B
true	true	$\theta_{ab}\lambda_b$
true	false	$\theta_{a\bar{b}}\lambda_{\bar{b}}$
false	true	$\theta_{\bar{a}b}\lambda_b$
false	false	$\theta_{\bar{a}\bar{b}}\lambda_{\bar{b}}$

Figure 5: Parameterizing CPTs in a Bayesian network.

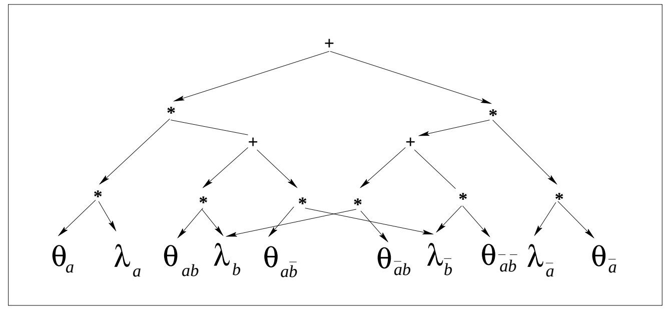


Figure 6: Compiling the Bayesian network in Figure 5.

In general, if we have the value of $\mathcal{F}$ at some input and we need to compute its value at another input which disagrees with the previous input on only one indicator λ_x, then we can compute the new value from the old value and $\partial\mathcal{F}/\partial\lambda_x$.

As it turns out, we can do much more once such partial derivatives are computed, a topic which we discuss in more detail later.

Compiling Bayesian Networks

Our goal in this section is to show how to compile a probability table which is specified using a Bayesian network $\mathcal{N}$ into a polynomial $\mathcal{F}$, and to bound the time and space complexity of such compilation. We have presented a proposal to this effect in (Darwiche & Provan 1997) based on jointrees. Here we present a much simpler approach based on a variable elimination scheme, where we also include network parameters as variables in the compiled polynomial $\mathcal{F}$. Specifically, given an order π of the variables in a Bayesian network $\mathcal{N}$, we do the following:

1. For every variable X with parents $\mathbf{U}$, *parameterize* the CPT of X as follows. Replace each entry in CPT ϕ_X which corresponds to instantiation $x\mathbf{u}$ by $\theta_{x\mathbf{u}}\lambda_x$. Variable $\theta_{x\mathbf{u}}$ is called a *network parameter* and variable λ_x is called a *network indicator*.
2. Consider every variable X according to order π:
   (a) Multiply all tables that contain variable X to yield table ϕ.
   (b) Replace the multiplied tables by the result of summing out X from ϕ, $\mathsf{sum\_out}(\phi, X)$.
3. Only one table ϕ will be left, with a single entry. Return the single entry of ϕ.

We will call this algorithm VE_COMPILE($\mathcal{N}, \pi$).

Following is an illustrative example of VE_COMPILE: we will use the elimination order $\pi =< B, A >$ to compile the network in Figure 5. The CPTs on the left of this figure are the ones provided with the network, while the ones on the right are the result of parameterization as given in Step 1 above. To eliminate variable B, we sum it out from table ϕ_B, since ϕ_B is the only table containing B, leading to:

A	$\mathsf{sum\_out}(\phi_B, B)$
true	$\theta_{ab}\lambda_b + \theta_{a\bar{b}}\lambda_{\bar{b}}$
false	$\theta_{\bar{a}b}\lambda_b + \theta_{\bar{a}\bar{b}}\lambda_{\bar{b}}$

To eliminate variable A, we must multiply the above table with table ϕ_A and then sum-out variable A from the result. Multiplying gives:

A	$\phi_A\mathsf{sum\_out}(\phi_B, B)$
true	$\theta_a\lambda_a(\theta_{ab}\lambda_b + \theta_{a\bar{b}}\lambda_{\bar{b}})$
false	$\theta_{\bar{a}}\lambda_{\bar{a}}(\theta_{\bar{a}b}\lambda_b + \theta_{\bar{a}\bar{b}}\lambda_{\bar{b}})$

Summing out, we get

$\mathsf{sum\_out}(\phi_A\mathsf{sum\_out}(\phi_B, B), A)$
$\theta_a\lambda_a(\theta_{ab}\lambda_b + \theta_{a\bar{b}}\lambda_{\bar{b}})+$ $\theta_{\bar{a}}\lambda_{\bar{a}}(\theta_{\bar{a}b}\lambda_b + \theta_{\bar{a}\bar{b}}\lambda_{\bar{b}})$

The polynomial,

$$\begin{aligned}&\mathcal{F}(\lambda_a, \lambda_{\bar{a}}, \lambda_b, \lambda_{\bar{b}}, \theta_a, \theta_{\bar{a}}, \theta_{ab}, \theta_{a\bar{b}}, \theta_{\bar{a}b}, \theta_{\bar{a}\bar{b}})\\ &\quad = \quad \theta_a\lambda_a(\theta_{ab}\lambda_b + \theta_{a\bar{b}}\lambda_{\bar{b}}) + \theta_{\bar{a}}\lambda_{\bar{a}}(\theta_{\bar{a}b}\lambda_b + \theta_{\bar{a}\bar{b}}\lambda_{\bar{b}})\end{aligned}$$

is then the result of this compilation and is depicted in Figure 6. Although such a polynomial is a special case of a Query-DAG (Darwiche & Provan 1997), we refer to it here as a *Bayesian network compilation*, or a *polynomial representation* of a Bayesian network. Before we can prove the soundness of our algorithm, we need some definitions.

Definition 1 *The value of an indicator λ_x at instantiation $\mathbf{e}$, denoted $\mathbf{e}(x)$, is 0 if X is instantiated to other than x in $\mathbf{e}$, and is 1 otherwise.*

Definition 2 *A quantification Θ of a Bayesian network is a function that assigns a value $\Theta(x\mathbf{u})$ to each instantiation $x\mathbf{u}$ of family $\mathbf{U} \cup \{X\}$, where $\Theta(x\mathbf{u})$ represents the conditional probability of x given $\mathbf{u}$.*

Definition 3 (Network Compilation) *Let $\mathcal{N}$ be a Bayesian network with variables $X, \ldots$ and families $\mathbf{F}, \ldots$. A network compilation for $\mathcal{N}$ is a function $\mathcal{F}(\lambda_x, \ldots, \theta_{\mathbf{f}}, \ldots)$ such that for every instantiation $\mathbf{e}$ and quantification Θ of $\mathcal{N}$, we have*

$$\mathcal{F}(\lambda_x = \mathbf{e}(x), \ldots, \theta_{\mathbf{f}} = \Theta(\mathbf{f}), \ldots) = Pr(\mathbf{e} \mid \Theta).$$

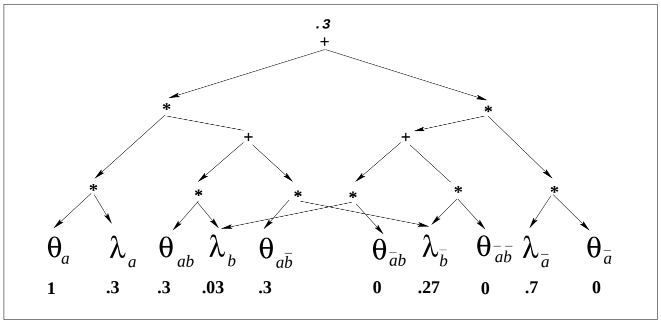


Figure 7: Partial derivatives are computed under $(\lambda_a, \lambda_{\bar{a}}, \lambda_b, \lambda_{\bar{b}}) = (1, 0, 1, 1)$ and $(\theta_a, \theta_{\bar{a}}, \theta_{ab}, \theta_{a\bar{b}}, \theta_{\bar{a}b}, \theta_{\bar{a}\bar{b}})$ $= (.3, .7, .1, .9, .8, .2)$.

A network compilation is then a function of many variables corresponding to the network indicators and parameters. For each instantiation $\mathbf{e}$ and quantification Θ, the compilation can be evaluated to compute the probability of $\mathbf{e}$ given Θ. We will often write $\mathcal{F}(\mathbf{e}, \Theta)$ or $\mathcal{F}(\mathbf{e})$ as a shorthand for $\mathcal{F}(\lambda_x = \mathbf{e}(x), \ldots, \theta_{\mathbf{f}} = \Theta(\mathbf{f}), \ldots)$ when no ambiguity is anticipated.

We can now state the soundness/complexity result:

Theorem 1 *The call* VE_COMPILE$(\mathcal{N}, \pi)$ *returns a network compilation for $\mathcal{N}$ of size $O(n \exp(w))$ in $O(n \exp(w))$ time, where n is the number of network variables and w is the width of elimination order π with respect to network $\mathcal{N}$.*

Therefore, the time complexity to generate, and space complexity to store, the compilation is similar to that of variable elimination algorithms. The win, however, is the large number of queries we can answer once we compute the partial derivatives of a network compilation, and the explicit relation between partial derivatives and various probabilistic queries. These relations seem to shed new light on probabilistic reasoning and provide a valuable tool for sensitivity analysis as we illustrate in the next section.

Differential Probabilistic Inference

Our goal in this section is to enumerate the class of queries that can be answered once the partial derivatives of a network compilation are computed.

Figure 7 depicts a network compilation, evaluated under evidence a. The partial derivative of $\mathcal{F}(a)$ with respect to each network indicator and parameter is shown below the indicator/parameter. For example, $\partial\mathcal{F}(a)/\partial\theta_a = 1$ and $\partial\mathcal{F}(a)/\theta_{\bar{a}} = 0$. Note that the probability of evidence a is unaffected by a change to parameter $\theta_{\bar{a}}$. However, for a parameter change to be probabilistically meaningful, a change δ to parameter $\theta_{\bar{a}}$ must be paralleled by a change $-\delta$ to parameter θ_a since $\theta_a + \theta_{\bar{a}} = 1$. Hence, for example, if we start with a network quantification Θ and increase parameter $\theta_{\bar{a}}$ by .1, we must also decrease parameter θ_a by .1, leading to quantification Θ'. The probability of evidence a will then change as follows:

$$\begin{aligned} &\mathcal{F}(a, \Theta') \\ &\quad = \mathcal{F}(a, \Theta) + (.1)\partial\mathcal{F}(a, \Theta)/\partial\theta_{\bar{a}} - (.1)\partial\mathcal{F}(a, \Theta)/\partial\theta_a \\ &\quad = .3 + (.1)0 - (.1)1 = .2 \end{aligned}$$

We have the following important property of network compilations returned by VE_COMPILE:

Theorem 2 *The compilation $\mathcal{F}(\lambda_x, \ldots, \theta_{\mathbf{f}}, \ldots)$ returned by* VE_COMPILE *is a multivariate polynomial of degree $2n$, in which each variable $\lambda_x, \theta_{\mathbf{f}}$ has degree one (n is number of Bayesian network variables). Moreover, for any two indicators λ_x and $\lambda_{x'}$ of variable X, we have $\partial\mathcal{F}/\partial\lambda_x\partial\lambda_{x'} = 0$. And for any two parameters $\theta_{\mathbf{f}}$ and $\theta_{\mathbf{f}'}$ of family $\mathbf{F}$, we have $\partial\mathcal{F}/\partial\theta_{\mathbf{f}}\partial\theta_{\mathbf{f}'} = 0$.*

This is a consequential theorem as it implies that:

1. $\mathcal{F}$ is a linear function in the indicators of any variable X: $\mathcal{F}(\mathbf{e}, \Theta) = \sum_x \beta_x \lambda_x$ for constants β_x.
2. $\mathcal{F}$ is a linear function in the parameters of any family $\mathbf{F}$: $\mathcal{F}(\mathbf{e}, \Theta) = \sum_{\mathbf{f}} \beta_{\mathbf{f}} \theta_{\mathbf{f}}$ for constants $\beta_{\mathbf{f}}$.

Moreover, such linear functions can be constructed in linear time from the compilation $\mathcal{F}$. All we have to do is evaluate $\mathcal{F}$ while not substituting the values of indicators/parameters of interest. For example, if we evaluate the compilation in Figure 2 (where evidence $\mathbf{e}$ is a) without substituting the values of B-indicators we get:

$$\mathcal{F}(\mathbf{e}, \Theta) = Pr(\mathbf{e} \mid \Theta) = .03\lambda_b + .27\lambda_{\bar{b}},$$

which tell us how findings on variable B will affect the probability of evidence. Similarly, we get

$$\mathcal{F}(\mathbf{e}, \Theta) = Pr(\mathbf{e} \mid \Theta) = .3\theta_{ab} + .3\theta_{a\bar{b}},$$

which tells us how changing the parameters of variable B will affect the probability of evidence—no effect in this case!

(Castillo, Gutiérrez, & Hadi 1996) have already observed that $Pr(\mathbf{e} \mid \Theta)$ is a polynomial of network parameters in which each parameter has degree one. We have therefore presented a variable-elimination algorithm for explicitly constructing this polynomial, with a structure-based guarantee on the time to construct it and the space to store it. In addition, our polynomial includes variables that represent evidence. As we see next, our major contribution to follow is based on computing the partial derivatives of this polynomial and on revealing the complexity and implications of such computation.

Posterior Marginals

We now turn to the computation of posterior marginals using the partial derivatives of a network compilation:

Theorem 3 (Marginals) *For all $X \notin \mathbf{E}$, we have:*

$$\begin{aligned} Pr(x, \mathbf{e}) &= \partial\mathcal{F}(\mathbf{e})/\partial\lambda_x; \\ Pr(x \mid \mathbf{e}) &= \frac{\partial\mathcal{F}(\mathbf{e})/\partial\lambda_x}{\mathcal{F}(\mathbf{e})}. \end{aligned}$$

Therefore, the partial derivatives will give us the posterior marginal of every variable in constant time. In Figure 7, where $\mathbf{e} = a$, we have $Pr(b, \mathbf{e}) = \partial F(\mathbf{e})/\partial\lambda_b = .03$ and $Pr(\bar{b}, \mathbf{e}) = \partial F(\mathbf{e})/\partial\lambda_{\bar{b}} = .27$. The ability to compute such posteriors efficiently is probably the key celebrated property of clustering algorithms (Huang & Darwiche 1996; Jensen, Lauritzen, & Olesen 1990), as compared to variable-elimination algorithms (Dechter 1996; Zhang & Poole 1996). The latter class of algorithms is much simpler except that they can only compute such posteriors by invoking themselves once for each network variable, leading to a complexity of $O(n^2 \exp(w))$. Clustering algorithms can do this in $O(n \exp(w))$, however, but at the expense of a more complicated algorithm. Our algorithm based on partial derivatives seems to be represent a middle ground in this efficiency/simplicity tradeoff.

Note also that once we have the partial derivatives $\partial\mathcal{F}/\partial\theta_{\mathbf{f}}$, we can implement the APN algorithm for learning network parameters (Russell *et al.* 1995). As for implementing the EM algorithm, we need the posterior marginals over network families (Lauritzen 1995). Such marginals, however, can be computed easily once we have the partial derivatives $\partial\mathcal{F}/\partial\theta_{\mathbf{f}}$—thanks to a result relating the two in (Russell *et al.* 1995):

Theorem 4 (Family Marginals) *Let $\mathcal{F}$ be the compilation of Bayesian network $\mathcal{N}$. For every family $\{X\} \cup \mathbf{U}$ in network $\mathcal{N}$, we have:*

$$Pr(x, \mathbf{u} \mid \mathbf{e}, \Theta) = \frac{\partial\mathcal{F}(\mathbf{e}, \Theta)/\partial\theta_{x\mathbf{u}}}{\mathcal{F}(\mathbf{e}, \Theta)} \Theta(x\mathbf{u}).$$

For example, in Figure 7, $Pr(b, a \mid a)$ is $\frac{\partial\mathcal{F}(a)/\partial\theta_{ab}}{\mathcal{F}(a)}\theta_{ab} = \frac{.3}{.3}.1 = .1$.

Evidence Retraction

One of the main complications in Bayesian network inference relates to the update of probabilities after having retracted evidence. This seems to pose no difficulties in the presented framework. For example, we can compute the posterior marginal of every instantiated variable, after the evidence on that variable has been retracted immediately. But we first need the following notation.

Let $\mathbf{e}$ be an instantiation and $\mathbf{X}$ be a set of variables. Then $\mathbf{e} - \mathbf{X}$ denotes the subset of instantiation $\mathbf{e}$ pertaining to variables not appearing in $\mathbf{X}$. For example, if $\mathbf{e} = ab\bar{c}$, then $\mathbf{e} - A = b\bar{c}$ and $\mathbf{e} - AC = b$.

Theorem 5 (Evidence Retraction) *For every variable X and evidence $\mathbf{e}$, we have:*

$$\begin{aligned} Pr(\mathbf{e} - X) &= \sum_x \frac{\partial\mathcal{F}(\mathbf{e})}{\partial\lambda_x}; \\ Pr(x \mid \mathbf{e} - X) &= \frac{\partial\mathcal{F}(\mathbf{e})/\partial\lambda_x}{\sum_x \partial\mathcal{F}(\mathbf{e})/\partial\lambda_x}. \end{aligned}$$

In Figure 7, where $\mathbf{e} = a$, we have $Pr(\mathbf{e} - A) = Pr(\mathsf{true}) = \partial\mathcal{F}(\mathbf{e})/\partial\lambda_a + \partial\mathcal{F}(\mathbf{e})/\partial\lambda_{\bar{a}} = 1$. Moreover, $Pr(\mathbf{e} - B) = Pr(\mathbf{e}) = \partial\mathcal{F}(\mathbf{e})/\partial\lambda_b + \partial\mathcal{F}(\mathbf{e})/\partial\lambda_{\bar{b}} = .3$.

The above computation is the basis of an investigation of model adequacy (Cowell *et al.* 1999, Chapter 10) and is typically implemented in the jointree algorithm using the technique of *fast retraction,* which requires a modification to the standard propagation method in jointrees (Cowell *et al.* 1999, Page 104). As given by the above theorem, we get this computation for free once we compute the partial derivatives with respect to network indicators.

The following theorem is the basis for performing "what-if" analysis, as it allows us to compute the conditional probability of some hypothesis in light of flipping each and every piece of evidence.

Theorem 6 (Evidence Flipping) *For every variable X and evidence $\mathbf{e}$, we have:*

$$Pr(x, \mathbf{e} - X) = \partial\mathcal{F}(\mathbf{e})/\partial\lambda_x.$$

In Figure 7, where $\mathbf{e} = a$, we have $Pr(\bar{a}, \mathbf{e} - A) = .7 = Pr(\bar{a})$.

Independence and Sensitivity

We now turn to the role of second partial derivatives in probabilistic inference. Here's the basic theorem:

Theorem 7 *For every pair of variables $X \neq Y \notin \mathbf{E}$, pair of families $\mathbf{F}_1 \neq \mathbf{F}_2$, and evidence $\mathbf{e}$, we have:*

$$\begin{aligned} \partial\mathcal{F}^2(\mathbf{e}, \Theta)/\partial\lambda_x\partial\lambda_y &= Pr(x, y, \mathbf{e} \mid \Theta); \\ \partial\mathcal{F}^2(\mathbf{e}, \Theta)/\partial\lambda_x\partial\theta_{\mathbf{f}_1} &= Pr(x, \mathbf{f}_1, \mathbf{e} \mid \Theta)/\Theta(\mathbf{f}_1); \\ \partial\mathcal{F}^2(\mathbf{e}, \Theta)/\partial\theta_{\mathbf{f}_1}\partial\theta_{\mathbf{f}_2} &= Pr(\mathbf{f}_1, \mathbf{f}_2, \mathbf{e} \mid \Theta)/\Theta(\mathbf{f}_1)\Theta(\mathbf{f}_2). \end{aligned}$$

Therefore, once we have second partial derivatives, we can compute marginals over pairs of variables, which is needed for identifying conditional independence among pairs of variables and for performing certain forms of sensitivity analysis (Laskey 1995; Castillo, Gutiérrez, & Hadi 1997; Coupé *et al.* 1999).

We start with independence first:

Theorem 8 (Independence) *Let $X, Y \notin \mathbf{E}$ be two distinct variables. Then $Pr(x \mid y, \mathbf{e}) = Pr(x \mid \mathbf{e})$ iff*

$$\frac{\partial\mathcal{F}^2(\mathbf{e})}{\partial\lambda_x\partial\lambda_y}\mathcal{F}(\mathbf{e}) = \frac{\partial\mathcal{F}(\mathbf{e})}{\partial\lambda_x}\frac{\partial\mathcal{F}(\mathbf{e})}{\partial\lambda_y}.$$

Therefore, once we have the second partial derivatives under some evidence $\mathbf{e}$, we can determine the context-specific independence of two events x and y given context $\mathbf{e}$ in constant time (Boutilier *et al.* 1996). Since we can compute all second partial derivatives simultaneously in $O(n^2 \exp(w))$ time and space, we can also determine the independence between all pairs of variables simultaneously in $O(n^2 \exp(w))$ time and space.[4]

Second partial derivatives also allow us to answer two new types of sensitivity queries:

1. To what extent is the sensitivity of $Pr(\mathbf{e})$ to parameter $\theta_{\mathbf{f}}$ impacted by further evidence on X?

[4] Two specialized techniques that can be for deciding such independence in the jointree algorithm are the *firing of variables* and the *propagation of variables* (Jensen 1996).

2. To what extent is the sensitivity of $Pr(\mathbf{e})$ to parameter $\theta_{\mathbf{f}_1}$ impacted by changing another parameter $\theta_{\mathbf{f}_2}$?

Theorem 7 reveals the classical probabilistic queries needed to answer such sensitivity questions. For example, it says that answering the second question requires the computation of marginals over pairs of network families, which is typically not supported directly by existing Bayesian network algorithms. Any such sensitivity query can be answered in constant time after having computed all second partial derivatives, which can be done in $O(n^2 \exp(w))$ time and space as we show in the following section.

As an example of results that are facilitated by the view we promote in this paper, we consider one of the key questions that arise in real–world diagnostic applications. Suppose we are given some evidence $\mathbf{e}$ and some hypothesis Y. We propagate evidence and find that $Pr(y \mid \mathbf{e}) > Pr(\bar{y} \mid \mathbf{e})$; that is, y is the fault given evidence $\mathbf{e}$. An expert may conclude differently, that $\bar{y}$ is more probable in this case, hence, indicating a problem in the Bayesian network model. Assuming that the network structure is correct, a key question is: which network parameters should we tweak in order to correct the problem — that is, leading to $Pr(y \mid \mathbf{e}) \leq Pr(\bar{y} \mid \mathbf{e})$. There is typically more than one parameter to tweak, but the following theorem provides key insights into answering this question:

Theorem 9 (Sensitivity) *Let Y and X be binary variables in a Bayesian network $\mathcal{N}$ with compilation $\mathcal{F}$. Let Θ and Θ' be two quantifications of $\mathcal{N}$ which agree on all parameters except those for $x\mathbf{u}$ and $\bar{x}\mathbf{u}$. If $Y \notin \mathbf{E}$, then $Pr(y \mid \mathbf{e}, \Theta') \leq Pr(\bar{y} \mid \mathbf{e}, \Theta')$ iff*

$$\begin{aligned} &\Theta(x\mathbf{u})\Theta(\bar{x}\mathbf{u})(G-H) \\ &\quad \leq \;\; [\Theta'(x\mathbf{u}) - \Theta(x\mathbf{u})][\Theta(x\mathbf{u})(I-J) + \Theta(\bar{x}\mathbf{u})(K-L)], \end{aligned}$$

where

$$\begin{aligned} G &= \partial\mathcal{F}(\mathbf{e},\Theta)/\partial\lambda_y \\ H &= \partial\mathcal{F}(\mathbf{e},\Theta)/\partial\lambda_{\bar{y}} \\ I &= \Theta(\bar{x}\mathbf{u})\partial\mathcal{F}^2(\mathbf{e},\Theta)/\partial\lambda_y\partial\theta_{\bar{x}\mathbf{u}} \\ J &= \Theta(\bar{x}\mathbf{u})\partial\mathcal{F}^2(\mathbf{e},\Theta)/\partial\lambda_{\bar{y}}\partial\theta_{\bar{x}\mathbf{u}} \\ K &= \Theta(x\mathbf{u})\partial\mathcal{F}^2(\mathbf{e},\Theta)/\partial\lambda_{\bar{y}}\partial\theta_{x\mathbf{u}} \\ L &= \Theta(x\mathbf{u})\partial\mathcal{F}^2(\mathbf{e},\Theta)/\partial\lambda_y\partial\theta_{x\mathbf{u}}. \end{aligned}$$

Let us call Θ the pre-tweak quantification and Θ' the post-tweak quantification. The above theorem is stating conditions on the pre-tweak quantification, and the amount of tweaking, which would ensure that a certain hypothesis ranking is achieved in the post-tweak quantification. That is, the theorem allows us to compute the change to parameters $\theta_{x\mathbf{u}}$ and $\theta_{\bar{x}\mathbf{u}}$, respectively, which ensure a particular ranking on the hypotheses y and $\bar{y}$. Note that parameter $\theta_{\bar{x}\mathbf{u}}$ must undergo an equal (but opposite) amount of change to that of parameter $\theta_{x\mathbf{u}}$, since $\theta_{x\mathbf{u}} + \theta_{\bar{x}\mathbf{u}} = 1$. As we shall see in the following section, we can compute the amount of change needed for each and every network parameter in time and space linear in the size of compilation $\mathcal{F}$. Therefore, the theorem allows us to build a tool in which one can list all network parameters, ranked according to their relevance to changing the ranking on hypotheses y and $\bar{y}$. If a very small change in a network parameter would ensure such ranking, it means that domain experts must pay close attention to assessing the exact value of that parameter.

Consider the network in Figure 5 where $Pr(b) = .59 > Pr(\bar{b}) = .41$. We want to compute the amount of change to parameter θ_a, initially set to $\Theta(a) = .3$, of root node A which is needed to ensure that $Pr(b) \leq Pr(\bar{b})$. Applying Theorem 9:

$$\Theta'(a) - \Theta(a) \geq \frac{(.3)(.7)(.59-.41)}{.3(.56-.14)+.7(.27-.03)} = .12857.$$

That is, $\Theta'(a) \geq .42857$ ensures that $Pr(b) \leq Pr(\bar{b})$.

For another example, note that $Pr(\bar{a}) = .7 > Pr(a) = .3$ in the same network. Let us then compute the amount of change needed in parameter θ_{ab} to ensure that $Pr(\bar{a}) \leq Pr(a)$. According to Theorem 9:

$$(.1)(.9)(.7-.3) \leq (\Theta'(ab)-\Theta(ab))(.1(0-.27)+.9(.03-0)) = 0.$$

Therefore, it is impossible to ensure that $Pr(\bar{a}) \leq Pr(a)$ by changing the parameter θ_{ab}, which is to be expected since $Pr(\bar{a}) = \theta_{\bar{a}}$ and $Pr(a) = \theta_a$ in this case — that is, $Pr(A)$ is independent of θ_{ab}.

Computing Partial Derivatives

We now turn to the process of evaluating a network compilation $\mathcal{F}$ under some evidence $\mathbf{e}$ and quantification Θ, and then computing all its partial derivatives. We will first describe the computation procedurally using a message–passing scheme and then provide the semantics of such computation.

As it turns out, the evaluation of $\mathcal{F}$ and computing its partial derivatives can be accomplished using a two-phase message passing scheme in which each message is simply a number. In the first phase, messages are sent from nodes to their parents in $\mathcal{F}$, leading to an evaluation of $\mathcal{F}$. In the second phase, messages are sent from nodes to their children, leading to the computation of all partial derivatives.

We will associate two attributes with each node i in the compilation $\mathcal{F}$: a value $\mathsf{val}(\mathsf{i})$ and a partial derivative $\mathsf{pd}(\mathsf{i})$. The goal of message passing is to set the value of each of these attributes under some evidence $\mathbf{e}$ and quantification Θ. We are mainly interested in the value of root node r, as it represents $\mathcal{F}(\mathbf{e},\Theta)$, and the partial derivative of each leaf node l, at it represents $\partial\mathcal{F}(\mathbf{e},\Theta)/\partial\mathsf{l}$. Message passing proceeds in two phases: the first phase sets the value of each node, and the second phase sets the partial derivatives.

Passing val-messages In the first phase, a val-message is passed from each node to all its parents according to the following rules:

1. *Initiation:* Each leaf node l sets its value:
   - $\mathsf{val}(\mathsf{l}) \leftarrow \mathbf{e}(x)$ when l is an indicator λ_x;

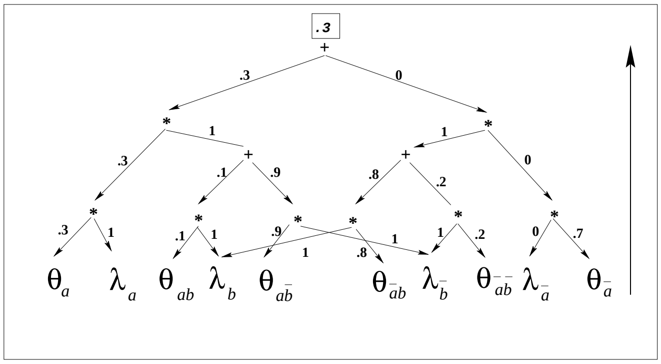


Figure 8: Passing val-messages under evidence $\mathbf{e} = a$ and quantification $\Theta : (\theta_a, \theta_{\bar{a}}, \theta_{ab}, \theta_{a\bar{b}}, \theta_{\bar{a}b}, \theta_{\bar{a}\bar{b}}) = (.3, .7, .1, .9, .8, .2)$.

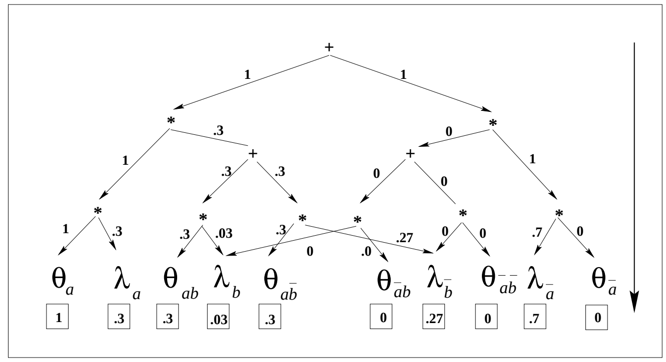


Figure 9: Passing pd-messages under evidence $\mathbf{e} = a$ and quantification $\Theta : (\theta_a, \theta_{\bar{a}}, \theta_{ab}, \theta_{a\bar{b}}, \theta_{\bar{a}b}, \theta_{\bar{a}\bar{b}}) = (.3, .7, .1, .9, .8, .2)$.

- $\mathsf{val}(\mathsf{l}) \leftarrow \Theta(x\mathbf{u})$ when l is a parameter $\theta_{x\mathbf{u}}$;

then sends a message $\mathsf{mes}(\mathsf{l}, \mathsf{k}) = \mathsf{val}(\mathsf{l})$ to each parent k.

2. *Iteration I:* When an <u>addition node</u> i receives messages from all its children j, it sets its value $\mathsf{val}(\mathsf{i}) \leftarrow \sum_{\mathsf{j}} \mathsf{mes}(\mathsf{j}, \mathsf{i})$ and then sends a message $\mathsf{mes}(\mathsf{i}, \mathsf{k}) = \mathsf{val}(\mathsf{i})$ to each parent k.
3. *Iteration II:* When a <u>multiplication node</u> i receives messages from all its children j, it sets its value $\mathsf{val}(\mathsf{i}) \leftarrow \prod_{\mathsf{j}} \mathsf{mes}(\mathsf{j}, \mathsf{i})$ and then sends a message $\mathsf{mes}(\mathsf{i}, \mathsf{k}) = \mathsf{val}(\mathsf{i})$ to each parent k.
4. *Termination:* The first phase terminates when the <u>root node</u> r receives messages from all its children. In that case, $\mathcal{F}(\mathbf{e}, \Theta) = \mathsf{val}(\mathsf{r})$.

This process is illustrated in Figure 8, where it leads to assigning $\mathsf{val}(\mathsf{r}) = .3$, indicating that the probability of evidence is .3.

Passing pd-messages In the second phase, a pd-message is passed from each node to all its children:

1. *Initiation:* The <u>root node</u> r sets $\mathsf{pd}(\mathsf{r}) \leftarrow 1$ and sends message $\mathsf{mes}(\mathsf{r}, \mathsf{j}) = 1$ to each child j.
2. *Iteration I:* When an <u>addition node</u> i receives messages from all its parents k, it sets $\mathsf{pd}(\mathsf{i}) \leftarrow \sum_{\mathsf{k}} \mathsf{mes}(\mathsf{k}, \mathsf{i})$ and then sends a message $\mathsf{mes}(\mathsf{i}, \mathsf{j}) = \mathsf{pd}(\mathsf{i})$ to each child j.
3. *Iteration II:* When a <u>multiplication node</u> i receives messages from all its parents k, it sets $\mathsf{pd}(\mathsf{i}) \leftarrow \sum_{\mathsf{k}} \mathsf{mes}(\mathsf{k}, \mathsf{i})$ and then sends a message $\mathsf{mes}(\mathsf{i}, \mathsf{j}) = \mathsf{pd}(\mathsf{i}) \prod_{\mathsf{j}' \neq \mathsf{j}} \mathsf{val}(\mathsf{j}')$ to each child j, where j' is a child of i.
4. *Termination:* The second phase terminates when each <u>leaf node</u> l receives messages from all its parents. In that case, $\mathsf{pd}(\mathsf{l})$ is guaranteed to be the partial derivative $\partial \mathcal{F}(\mathbf{e}, \Theta)/\partial \mathsf{l}$.

This process is illustrated in Figure 9, assuming the val-messages have been propagated as in Figure 8. The process leads to assigning $\mathsf{pd}(\mathsf{l})$ to each leaf node in the compilation, shown in a box below the leaf node.

Clearly, the number of messages passed in both phases is twice the number of edges in the compilation $\mathcal{F}$. The computation of $\mathsf{val}(\mathsf{i})$ is linear in the number of i's children, and the computation of $\mathsf{pd}(\mathsf{i})$ is linear in the number of its parents. Each val-message that i sends takes constant time to compute, and the same is true for each pd-message that an addition node sends. Finally, the computation of each pd-message sent by a multiplication node is linear in the number of its children. Therefore, both phases of the message passing scheme take time which is linear in the size of network compilation $\mathcal{F}$. We are currently working on counting the number of operations performed by our message passing scheme and comparing that to other frameworks for multiple-query inference, such as the HUGIN and Shenoy/Shafer architectures.

The Semantics of message passing We now turn to the semantics of our message-passing scheme. Specifically, any node i in function $\mathcal{F}$ can be viewed as a subfunction $\mathcal{F}_{\mathsf{i}}$ of $\mathcal{F}$. For any evidence $\mathbf{e}$, our message passing scheme guarantees that $\mathsf{val}(\mathsf{i}) = \mathcal{F}_{\mathsf{i}}(\mathbf{e}, \Theta)$. Alternatively, node i can be viewed as an intermediate variable V_{i} of function $\mathcal{F}$. Our message passing scheme guarantees that $\mathsf{pd}(\mathsf{i}) = \partial \mathcal{F}(\mathbf{e}, \Theta)/\partial V_{\mathsf{i}}$.

These results follow directly from the work of (Gunter Rote 1990), which addresses the more general problem of computing partial derivatives for a function that has an *acyclic computation graph.* Our network compilations $\mathcal{F}$ fall into that category. Specifically, let G be the DAG defined by network compilation $\mathcal{F}$. Let us assign a weight to each edge in G as follows. The weight of an edge $\mathsf{i} \rightarrow \mathsf{j}$ is 1 if i is an addition node, and is $\prod_{\mathsf{j}' \neq \mathsf{j}} \mathsf{val}(\mathsf{j}')$, where j' is a child of i, if i is a multiplication node. Define the weight of a directed path in G to be the multiplication of weights assigned to its edges.

It is then shown in (Gunter Rote 1990) that the partial derivative of function $\mathcal{F}$ with respect to a node i is the sum of all weights of paths from the root of G to node i. Our message passing scheme is simply implementing this computation.

We close this section by two remarks. The first regards the computation of rounding errors when evaluating $\mathcal{F}(\mathbf{e},\Theta)$ using a limited-precision computer. Specifically, let $\delta\mathsf{i}$ be the rounding error "generated" when computing the value of node i—that is, the difference between the value computed using limited-precision and infinite-precision. If $\mid \delta\mathsf{i} \mid \leq \epsilon \mid \mathsf{val}(\mathsf{i}) \mid$ for a machine–specific ϵ, then the rounding error in computing $\mathcal{F}(\mathbf{e},\Theta)$, $\Delta\mathcal{F}(\mathbf{e},\Theta)$, can be bound as follows (see (Masao 1984) for details):

$$\Delta\mathcal{F}(\mathbf{e},\Theta) \leq \epsilon \sum_{\text{non-leaf } \mathsf{i}} \mid \mathsf{pd}(i)\mathsf{val}(i) \mid .$$

Note that bounding the rounding error can be done simultaneously during the passage of pd-messages, and requires no extra space.

Our final remark is on the computation of second partial derivatives. The partial derivative $\mathsf{pd}(\mathsf{i})$ is by itself a function of many variables and can be differentiated again using the same approach. In particular, if the compilation $\mathcal{F}$ has $O(n)$ variables, then its second partial derivatives can be computed simultaneously in $O(n \mid \mathcal{F} \mid)$ time and space, where $\mid \mathcal{F} \mid$ is the size of $\mathcal{F}$. Again, this follows from a more general result in (Masao 1984).

Therefore, if the compilation $\mathcal{F}$ of a Bayesian network of size n is induced using an elimination order of width w, as suggested earlier, then the value of polynomial $\mathcal{F}$; all its first partial derivatives; and all its second partial derivatives can be computed simultaneously in $O(n^2 \exp(w))$ time and space. This is quite interesting given the number of queries that can be answered in constant time given such derivatives. We are unaware of any computational framework for Bayesian networks, which combines the simplicity, comprehensiveness and computational complexity of the presented framework based on partial derivatives.

Conclusion

We have presented one of the simplest, yet most comprehensive frameworks for inference in Bayesian networks. According to this framework, one compiles a Bayesian network into a multivariate polynomial $\mathcal{F}$ using a variable elimination scheme. For each piece of evidence $\mathbf{e}$, one then traverses the polynomial $\mathcal{F}$ twice. In the first pass, the value of $\mathcal{F}$ is computed. In the second pass, all its partial derivatives are computed. Once such derivatives are made available, one can compute in constant time answers to a very large class of probabilistic queries, relating to classical inference, parameter estimation, model validation and sensitivity analysis.

The proposed framework makes two key contributions to the state-of-the-art on probabilistic reasoning. First, it highlights a key and comprehensive role of partial differentiation in this form of reasoning, shedding new light on its utility in sensitivity analysis.[5] Second, it presents one of the simplest, yet most comprehensive frameworks for inference in Bayesian networks, which lends itself to cost-effective implementations on a variety of software and hardware platforms. Note that compiling a Bayesian network can be performed off-line. The only computation that is needed on-line is that of passing val-messages and pd-messages, which is quite simple. We are currently investigating the development of dedicated hardware for implementing the proposed message passing scheme. Similar dedicated hardware has been developed for other reasoning frameworks, such as genetic programming and neural networks, and can provide a valuable tool for computationally intensive tasks in Bayesian networks, such as the estimation of network parameters using APN/EM.

References

Boutilier, C.; Friedman, N.; Goldszmidt, M.; and Koller, D. 1996. Context–specific independence in bayesian networks. In *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence (UAI)*, 115–123.

Castillo, E.; Gutiérrez, J. M.; and Hadi, A. S. 1996. Goal oriented symbolic propagation in Bayesian networks. In *Proc. AAAI National Conference*.

Castillo, E.; Gutiérrez, J. M.; and Hadi, A. S. 1997. Sensitivity analysis in discrete Bayesian networks. *IEEE Transactions on Systems, Man, and Cybernetics* 27:412–423.

Coupé, V. M. H.; Peek, N.; Ottenkamp, J.; and Habbema, J. D. F. 1999. Using sensitivity analysis for efficient quantification of a belief network. *Artificial Intelligence in Medicine* 17:223–247.

Cowell, R.; Dawid, A.; Lauritzen, S.; and Spiegelhalter, D. 1999. *Probabilistic Networks and Expert Systems*. Springer.

Darwiche, A., and Provan, G. 1997. Query DAGs: A practical paradigm for implementing belief–network inference. *Journal of Artificial Intelligence Research* 6:147–176.

Darwiche, A. 2000. On the tractability of counting theory models and its application to belief revision and truth maintenance. In *Working notes of the 8^{th} international workshop on non-monotonic reasoning*.

Dechter, R. 1996. Bucket elimination: A unifying framework for probabilistic inference. In *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence (UAI)*, 211–219.

Gunter Rote, G. 1990. Path problems in graphs. *Computing Suppl.* 7:155–189.

[5] Partial differentiation turns out to also play a key role in symbolic truth maintenance and belief revision as shown in (Darwiche 2000).

Huang, C., and Darwiche, A. 1996. Inference in belief networks: A procedural guide. *International Journal of Approximate Reasoning* 15(3):225–263.

Jensen, F. V.; Lauritzen, S.; and Olesen, K. 1990. Bayesian updating in recursive graphical models by local computation. *Computational Statistics Quarterly* 4:269–282.

Jensen, F. V. 1996. *An Introduction to Bayesian Networks.* Springer Verlag, New York Inc.

Laskey, K. B. 1995. Sensitivity analysis for probability assessments in Bayesian networks. *IEEE Transactions on Systems, Man, and Cybernetics* 25:901–909.

Lauritzen, S. L. 1995. The EM algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis* 19:191–201.

Masao. 1984. Simultaneous computation of functions, partial derivatives and estimates of rounding error. *Japan J. Appl. Math.* 1:223–252.

Russell, S.; Binder, J.; Koller, D.; and Kanazawa, K. 1995. Local learning in probabilistic networks with hidden variables. In *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence (UAI)*, 1146–1152.

Zhang, N. L., and Poole, D. 1996. Exploiting causal independence in bayesian network inference. *Journal of Artificial Intelligence Research* 5:301–328.