Available online at www.sciencedirect.com

ScienceDirect

Artificial Intelligence 172 (2008) 483–513

Artificial Intelligence

www.elsevier.com/locate/artint

Quantifying the uncertainty of a belief net response: Bayesian error-bars for belief net inference

Tim Van Allen[a], Ajit Singh[b], Russell Greiner[c,*], Peter Hooper[d]

[a] *Apollo Data Technologies, 12729 N.E. 20th Suite 7, Bellevue, WA 98005, USA*
[b] *Center for Automated Learning and Discovery, Carnegie Mellon University, Pittsburgh, PA 15213, USA*
[c] *Department of Computing Science, University of Alberta, Edmonton, Alberta T6G 2E8, Canada*
[d] *Department of Mathematical and Statistical Sciences, University of Alberta, Edmonton, Alberta T6G 2G1, Canada*

Received 22 June 2006; received in revised form 8 September 2007; accepted 10 September 2007

Available online 26 September 2007

Abstract

A Bayesian belief network models a joint distribution over variables using a DAG to represent variable dependencies and network parameters to represent the conditional probability of each variable given an assignment to its immediate parents. Existing algorithms assume each network parameter is fixed. From a Bayesian perspective, however, these network parameters can be random variables that reflect uncertainty in parameter estimates, arising because the parameters are learned from data, or because they are elicited from uncertain experts.

Belief networks are commonly used to compute responses to queries—i.e., return a number for $\mathrm{P}(\mathtt{H} = \mathtt{h} \mid \mathbf{E} = \mathbf{e})$. Parameter uncertainty induces uncertainty in query responses, which are thus themselves random variables. This paper investigates this query response distribution, and shows how to accurately model this distribution for any query and any network structure. In particular, we prove that the query response is asymptotically Gaussian and provide its mean value and asymptotic variance. Moreover, we present an algorithm for computing these quantities that has the same worst-case complexity as inference in general, and also describe straight-line code when the query includes all n variables. We provide empirical evidence that (1) our approximation of the variance is very accurate, and (2) a Beta distribution with these moments provides a very accurate model of the observed query response distribution. We also show how to use this to produce accurate error bars around these responses—i.e., to determine that the response to $\mathrm{P}(\mathtt{H} = \mathtt{h} \mid \mathbf{E} = \mathbf{e})$ is $x \pm y$ with confidence $1 - \delta$.

Keywords: Bayesian belief network; Variance; Bucket elimination; Credible interval; Error bars

1. Introduction

Bayesian belief nets (BNs), which provide a succinct model of a joint probability distribution, are used in an ever increasing range of applications [13]. They are typically built by first finding an appropriate structure (either by interviewing an expert, or by selecting a good model from training data), and then using a training sample to estimate

* Corresponding author.
E-mail addresses: tim@apollodatatech.com (T. Van Allen), ajit+@cs.cmu.edu (A. Singh), greiner@cs.ualberta.ca (R. Greiner), hooper@stat.ualberta.ca (P. Hooper).

0004-3702/$ – see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.artint.2007.09.004

the parameters [22]. The resulting belief net is then used to answer queries—e.g., compute the conditional probability $\texttt{P(Cancer=true | Smoke=true, Gender=male)}$. These responses clearly depend on the training sample used to instantiate the parameters, in that different training samples will produce different parameters, which will lead to different responses.

This paper investigates how variability within a sample induces variance in a query response, and presents a technique for estimating the posterior distribution of the query responses produced by a belief net. Stated informally, our goal is an algorithm that takes

- a belief net structure that we assume is correct (i.e., an I-map of the true distribution $\mathcal{D}$ [34])
- a prior distribution over the network parameters Θ
- a data sample S generated from $\mathcal{D}$
- a query of the form "What is $q(\Theta) = \mathrm{P}(\mathtt{H} = \mathtt{h} \mid \mathbf{E} = \mathbf{e}, \Theta)$?"

and returns both the expected value and the approximate variance of the query response $q(\Theta)$, based on the posterior distribution of parameters given the sample. By using these moments and an appropriate distributional form, we approximate the density of $q(\Theta)$, from which we can produce explicit $[L, U] \subseteq [0, 1]$ bounds such that $100(1-\delta)\%$ of the posterior density is within the interval. In other words, the algorithm returns both a point-estimate of the answer and *error bars* around it.

There are many other ways to use this variance around a query response. (1) In general, a good classifier should minimize Mean Squared Error, which can be expressed as "Bias^2 + Variance" [36]. The results in this paper provide a way to compute variance, which we then used as part of the Bias^2 + Variance measure to estimate the quality of different belief net structures, when seeking the best classifier. Empirical evidence [18] suggests that this measure is, in fact, one of the most effective discriminative model selection criteria. (2) The maximum likelihood approach to combining the responses of various independent belief net classifiers P_j involves weighting their respective (mean) probabilities by 1/variance; i.e., $P^*(\mathrm{h}_i|\mathbf{e}) \propto \sum_j P_j(\mathrm{h}_i \mid \mathbf{e})/\mathrm{var}_j(\mathrm{h}_i|\mathbf{e})$. We [32] show that this works very well in practice. (3) We could use query variance to detect outliers, as it could help differentiate sampling variation from true outliers [33]. (4) In classification, high probability is frequently taken as a proxy for confidence. Error bars provide a more statistically rigorous measure of confidence. For example, imagine we have determined that *action1* (e.g., "apply treatmentX") is optimal if $\mathrm{P}(+c \mid \mathbf{e}) > 0.5$, as this condition means *action1* maximizes expected utility (MEU) [27]. In the "traditional view", we would be more comfortable taking *action1* for larger values of $\mathrm{P}(+c \mid \mathbf{e})$—e.g., more confident given $\mathbf{e}'$ if $\mathrm{P}(+c \mid \mathbf{e}') = 0.7$, versus $\mathbf{e}''$ when $\mathrm{P}(+c \mid \mathbf{e}'') = 0.6$. Now imagine we know that $\mathrm{P}(+c \mid \mathbf{e}') = 0.7 \pm 0.5$ and $\mathrm{P}(+c \mid \mathbf{e}'') = 0.6 \pm 0.001$. In either case, the MEU response is still to take *action1*, as the *expected* utility depends only on the expected value of the response, but not on other characteristics of the posterior distribution. However, we should be more confident in taking this action in the second situation, $\mathbf{e}''$, as there is less "probability mass" on the other side of the 0.5 decision boundary. This could also be useful when making decisions in safety-critical scenarios, as this analysis can help to quantify the chance of a bad outcome, after taking the appropriate action—"good decision, bad outcome". (5) Finally, if an expert is available to also provide the query response, error bars can be used to validate a given belief net structure. For example, if an expert claims that $\mathrm{P}(B = 1 \mid D = 0, C = 1) = 0.5$ but our technique asserts that this response is in [0.26, 0.34] with 99% confidence, then we may question the validity of the network. However, if our technique instead asserts that this response is in [0.17, 0.58] with 99% confidence, we may not need to question the network structure.

This paper provides a way to quantify this variance of a response. In particular, we show how to approximate the posterior distribution of the query response, from which credible intervals ("Bayesian error-bars") can be readily computed. The overall process is shown in Fig. 1. We begin with a network structure, with a prior on its parameters. The ComputePosterior routine uses a data sample to perform a Bayesian update of this prior, yielding a posterior distribution over the network parameters. Next, the MeanVar algorithm calculates the mean and approximate variance of the query response. Finally, the ComputeErrorBar routine uses these moments to produce a model of the posterior distribution of the query response, from which it computes a $1-\delta$ credible interval. The ComputePosterior and ComputeErrorBar subroutines are well understood; our main contribution is defining and implementing MeanVar.

Section 2 provides the required background: defining belief nets and describing the Bayesian framework we use. This section also describes the ComputePosterior and ComputeErrorBars steps. Section 3 presents the the-

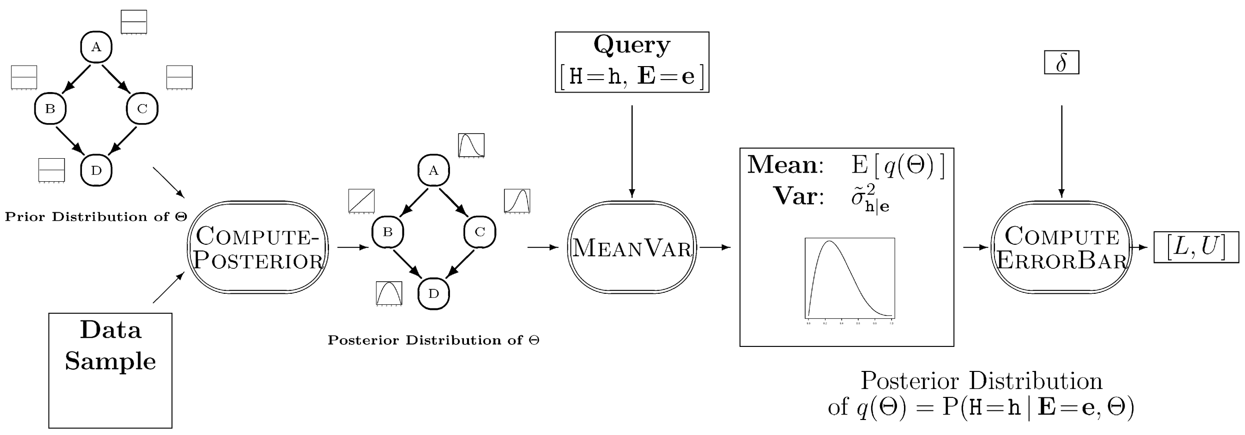


Fig. 1. Overview of the overall process for computing error bar around the query response.

oretical justification for MEANVAR. Given a set of explicitly specified assumptions, we prove that the query response distribution is asymptotically univariate Gaussian, and provide a closed-form formula for its asymptotic variance.[1] This theorem is the basis for our approximation of the posterior distribution.

Section 4 presents an algorithm, MEANVAR, for computing these variances: In particular, it first describes the BUCKELIM$^+$ algorithm (an extension of the well-known Bucket Elimination algorithm [14]) that computes both the response and also the derivative of the response wrt each of the CPtable entries; MEANVAR uses these derivatives for computing our approximation to the variance. This section provides theorems that state that the algorithm is correct, and that its asymptotic computational cost is the same as inference. Appendix A analyzes the special case where the query $\mathrm{P}(\mathtt{H} = \mathtt{h} \,|\, \mathbf{E} = \mathbf{e})$ is "complete", in that **E** specifies a value for every variable except H. It also provides a straightforward linear-time algorithm for this case.

While our variance approximation is asymptotically accurate, it is not clear whether our error bar algorithm will work well in practice, especially for a small sample, as our approximation is only first-order, and COMPUTEERROR-BAR inherits the assumption that the posterior query response will be Gaussian. We therefore perform experiments, based on Monte Carlo simulations, over a range of belief net structures and queries. The results, described in Section 5, suggest that (1) our approximation to the variance is acceptable, but (2) the resulting distribution is not well-modeled under the Gaussian assumption. This negative result is not surprising: The response is a probability and so must be in the [0, 1] interval; however a Gaussian distribution can have significant mass outside [0, 1]. We therefore considered approximating the query response with a Beta distribution, and found that this produces more accurate error bars.

Section 6 surveys relevant work, to place our results in context. Appendix B provides proofs for the claims made in this paper. Given the abundance of notation, we provide a list of symbols in Fig. 3. Further details, along with complete results, inputs, experimental parameters, and raw data, are available in the webpage [20].

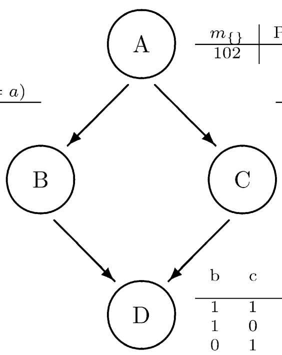
2. Preliminaries

2.1. *Belief nets*

We assume there is a fixed underlying distribution over n discrete random variables $\{X_1, \ldots, X_n\}$, which we denote as the *underlying distribution* or the *event distribution*.

We encode the event distribution as a belief net[2] $\langle \mathcal{V}, \mathcal{A}, \Theta \rangle$ that consists of a directed acyclic graph (DAG) whose nodes $\mathcal{V}$ represent variables and whose directed arcs $\mathcal{A}$ represent dependencies between variables. The $\langle \mathcal{V}, \mathcal{A} \rangle$ network structure encodes the independency relationships between variables in the underlying distribution—i.e., a node is independent of its non-descendants, given an assignment to its immediate parents. Each node $\mathtt{C} \in \mathcal{V}$ is also associated

[1] Essentially, "asymptotic" refers to a sufficiently large sample; see Theorem 2.
[2] Also known as Bayesian network, Bayesian belief network, probability net.



$m_{\{\}}$	$\mathrm{P}(A=1)$	$\mathrm{P}(A=0)$
102	0.343	0.657

a	$m_{A=a}$	$\mathrm{P}(B=1\,\vert\,A=a)$	$\mathrm{P}(B=0\,\vert\,A=a)$
1	36	0.194	0.806
0	68	0.412	0.588

a	$m_{A=a}$	$\mathrm{P}(C=1\,\vert\,A=a)$	$\mathrm{P}(C=0\,\vert\,A=a)$
1	36	0.250	0.750
0	68	0.353	0.647

b	c	$m_{B=b,C=c}$	$P\left(D=1\,\middle\vert\,\begin{matrix}B=b\\C=c\end{matrix}\right)$	$P\left(D=0\,\middle\vert\,\begin{matrix}B=b\\C=c\end{matrix}\right)$
1	1	10	0.100	0.900
1	0	25	0.360	0.640
0	1	27	0.259	0.741
0	0	46	0.457	0.543

Fig. 2. Diamond network: A simple example of a belief network, with CPtable distributions.

with a conditional probability table, called its "CPtable", that specifies $\Theta_{\mathtt{C}|\mathbf{f}} = \mathrm{P}(\mathtt{C} = \mathtt{c}\,|\,\mathbf{F} = \mathbf{f})$ for each value $c \in \mathtt{C}$ and each assignment $\mathbf{f}$ to its set of parents $\mathbf{F}$ [34]. That is, a belief network factors the underlying distribution into the product of these conditional probabilities. While we will typically view each CPtable as a table, it is useful in Section 4 to view it as a function that maps each assignment of the variables $\{\mathtt{C}\} \cup \mathbf{F}$ to the associated probability value $\mathrm{P}(\mathtt{C} = \mathtt{c}\,|\,\mathbf{F} = \mathbf{f}) \in [0, 1]$.

Fig. 2 provides an example of a belief network. As a function, the CPtable for B is denoted $f_B(a, b)$; in the row view, as $\Theta_{B=1|A=0} = 0.412$ and $\Theta_{B=0|A=0} = 0.588$. As $\Theta_{B|A=0}$ is a distribution, the row entries must sum to $\sum_b \Theta_{B=b|A=0} = 1$, and so $\Theta_{B=0|A=0}$ is implicitly defined by $\Theta_{B=1|A=0}$. We let $\Theta = \{\Theta_{\mathtt{c}|\mathbf{f}}\}$ denote the set of all CPtable rows.

In addition to providing a compact representation of a joint distribution, belief nets are used to effectively compute marginal and conditional probabilities. A query is a question of the form "What is $\mathrm{P}(\mathtt{H} = \mathtt{h}\,|\,\mathbf{E} = \mathbf{e})$?" and the answer (a real number in [0, 1]) is known as a query response.[3] In this paper we assume the network structure is fixed, and so for a particular query the answer depends only on the network parameters Θ; *cf.,* Eq. (16) in Section 5.1, associated with the Diamond network on Fig. 2. To emphasize this relationship, we will use $q_{\mathtt{h}|\mathbf{e}}(\Theta) = q(\Theta)$ to denote the response to the query $\mathrm{P}(\mathtt{h}\,|\,\mathbf{e}, \Theta)$, as a function of the parameters Θ.

2.2. ComputePosterior: *Learning network parameters*

We view Θ as a random vector, and follow the Bayesian view of parameter learning by placing a prior on Θ, and then integrating the data to yield the posterior distribution of Θ; see Fig. 4. We will assume the rows are independent (see Definition 1 (part 2) in Section 3), in that each row $\Theta_{\mathtt{C}|\mathbf{f}}$ is independent of every other row. Hence, the prior over Θ can be decomposed into priors on each row. Following standard practice [22], we will assume that each row is Dirichlet distributed: $\Theta_{C|\mathbf{f}} \sim \mathrm{Dir}(m_{\mathtt{C}=\mathtt{c}_1|\mathbf{f}}, \ldots, m_{\mathtt{C}=\mathtt{c}_r|\mathbf{f}})$ where each $m_{\mathtt{C}=\mathtt{c}_i|\mathbf{f}} > 0$.[4] An alternative notation for the same Dirichlet distribution is

$$\Theta_{\mathtt{C}|\mathbf{f}} = \langle \Theta_{\mathtt{C}=\mathtt{c}_1|\mathbf{f}}, \ldots, \Theta_{\mathtt{C}=\mathtt{c}_r|\mathbf{f}} \rangle \sim \mathrm{Dir}(m_{\mathtt{C}|\mathbf{F}=\mathbf{f}}; \hat{\Theta}_{\mathtt{C}=\mathtt{c}_1|\mathbf{f}}, \ldots, \hat{\Theta}_{\mathtt{C}=\mathtt{c}_r|\mathbf{f}}) \tag{1}$$

where

$$m_{\mathtt{C}|\mathbf{f}} = \sum_i m_{\mathtt{C}=\mathtt{c}_i|\mathbf{f}} \quad \text{and} \quad \hat{\Theta}_{\mathtt{C}=\mathtt{c}_i|\mathbf{f}} = \frac{m_{\mathtt{C}=\mathtt{c}_i|\mathbf{f}}}{m_{\mathtt{C}|\mathbf{f}}} \tag{2}$$

$m_{\mathtt{C}|\mathbf{f}}$ is called the *effective sample size* of the distribution and each $\hat{\Theta}_{\mathtt{C}=\mathtt{c}|\mathbf{f}} = \mathrm{E}[\Theta_{\mathtt{C}=\mathtt{c}|\mathbf{f}}]$ is the expected value of $\Theta_{\mathtt{C}=\mathtt{c}|\mathbf{f}}$ [9].

[3] We allow for the conditioning event, $\mathbf{E} = \mathbf{e}$, to involve no variables to allow unconditional queries of the form $\mathrm{P}(\mathtt{H} = \mathtt{h})$. Also, while our notation "$\mathtt{H} = \mathtt{h}$" suggests that our approach works only when there is a single query variable, note that everything holds when dealing with a *set* of query variables—i.e., $\mathrm{P}(H_1 = h_1, \ldots, H_r = h_r\,|\,\mathbf{E} = \mathbf{e})$.

[4] A Dirichlet distribution with only two parameters is also known as a Beta distribution, which we will denote as $\mathrm{Be}(a, b)$.

Top level algorithms	
BUCKELIM	Bucket elimination algorithm for computing the expected value of a query response; Section 4.1
BUCKELIM$^+$	Variant of the bucket elimination algorithm that computes the expected value of a query response and the derivatives $\{q^{(\mathtt{c}\|\mathbf{f})}_{\mathtt{h}\|\mathbf{e}}(\hat{\Theta})\}_{\mathtt{c},\mathbf{f}}$; Section 4.2
COMPUTEPOSTERIOR	Computes the posterior distribution over the network parameters; Section 2.2
COMPUTEERRORBARS	Computes the actual (Bayesian) error bars, based on the distribution; Section 2.3
MEANVAR	Computes the mean and variance of the response; Section 4

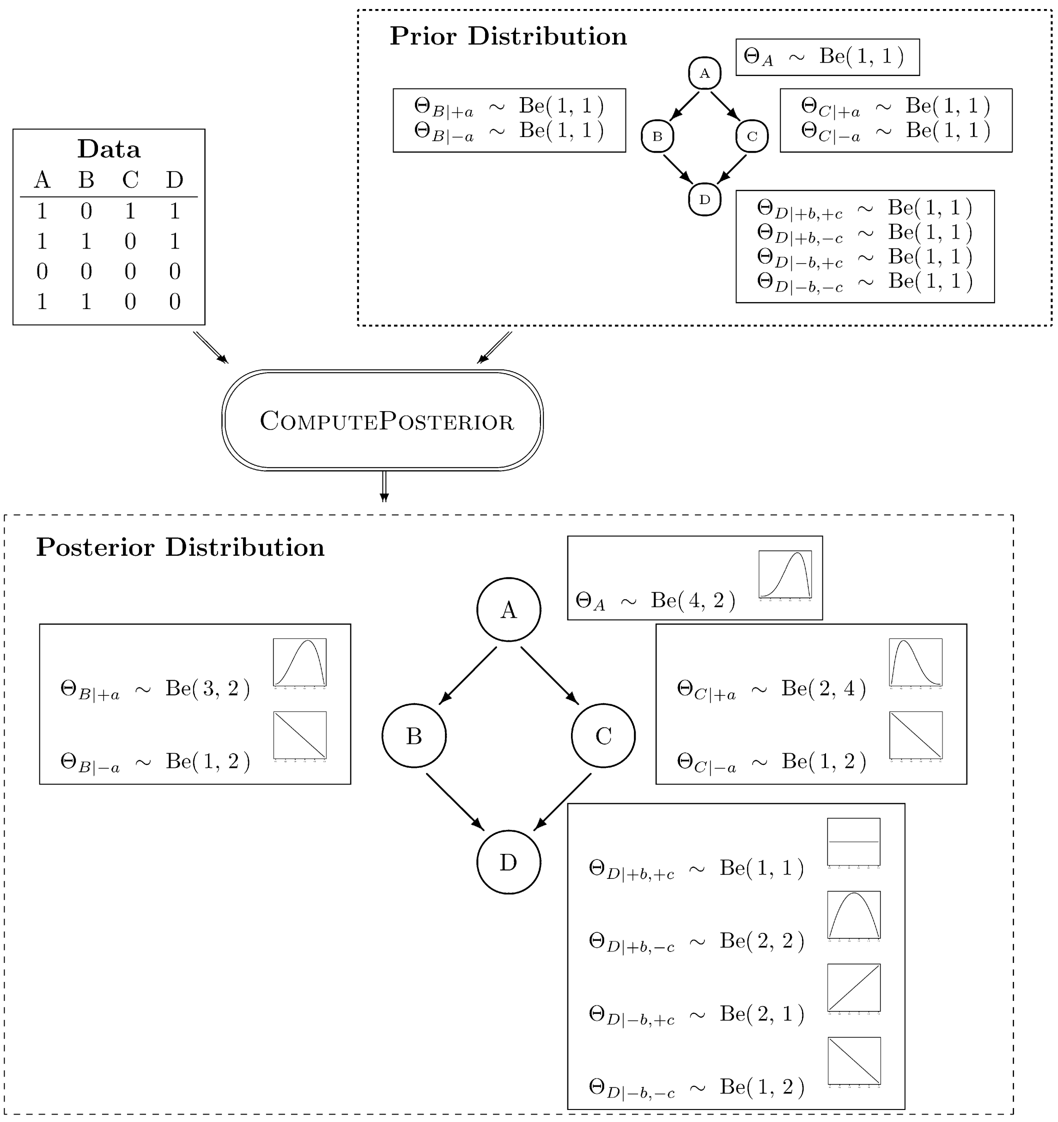
Belief nets	
$\mathcal{D}$	Underlying distribution from which the data was drawn
n	Number of nodes/variables in the belief net
k	Total number of CPtable rows (over entire network)
$\mathtt{C}$	Capital letters denote nodes in the belief net (or equivalently the variables they represent). **Bold** denotes a set of variables—e.g., **F**
$\mathtt{C} = \mathtt{c}$	Lowercase letters denote an assignment to a variable. If the assignment is to a single binary variable, then $\{+\mathtt{c}, -\mathtt{c}\}$ denotes the two values
$(\mathtt{C}, \mathbf{f})$	Refers to the CPtable row of variable C, corresponding to parental assignment $\mathbf{F} = \mathbf{f}$
$\Theta = \{\Theta_{\mathtt{C}\|\mathbf{f}}\}_{\mathtt{C},\mathbf{f}}$	Set of all network parameters, over the entire network
$\Theta_{\mathtt{C}\|\mathbf{f}} = \Theta_{\mathtt{C}\|\mathbf{F}=\mathbf{f}}$	CPtable row $\Theta_{\mathtt{C}\|\mathbf{f}} = \langle\Theta_{\mathtt{c}_1\|\mathbf{f}}, \ldots, \Theta_{\mathtt{c}_\ell\|\mathbf{f}}\rangle$
$\Theta_{\mathtt{c}\|\mathbf{f}} = \Theta_{\mathtt{C}=\mathtt{c}\|\mathbf{F}=\mathbf{f}}$	Single network parameter, corresponding to $\mathrm{P}(\mathtt{C} = \mathtt{c} \mid \mathbf{F} = \mathbf{f})$
$\hat{\Theta} = \mathrm{E}[\Theta]$	Expected values of network parameters
$m_{\mathtt{C}\|\mathbf{F}=\mathbf{f}}$	Effective sample size, associated with the CPtable row $\Theta_{\mathtt{C}\|\mathbf{F}=\mathbf{f}}$ (Eq. (2))
$q_{\mathtt{h}\|\mathbf{e}}(\Theta) = q(\Theta)$	Query response as function of parameters, for fixed implicit belief net structure, and fixed implicit query $\mathrm{P}(\mathtt{H} = \mathtt{h} \mid \mathbf{E} = \mathbf{e}, \Theta)$
$q^{(\mathtt{c}\|\mathbf{f})}(\Theta) = \frac{\partial q(\Theta)}{\partial \Theta_{\mathtt{c}\|\mathbf{f}}}$	Derivative of query response with respect to formal parameter $\Theta_{\mathtt{c}\|\mathbf{f}}$ (Eq. (6))
$\sigma^2_{\mathtt{h}\|\mathbf{e}} = \sigma^2_{\mathtt{h}\|\mathbf{e}}(\Theta)$	Variance of the query response
$\tilde{\sigma}^2_{\mathtt{h}\|\mathbf{e}} = \tilde{\sigma}^2_{\mathtt{h}\|\mathbf{e}}(\Theta)$	Approximate variance of the query response (Eq. (7))
$v_{\mathtt{h}\|\mathbf{e}}(\mathtt{C}\|\mathbf{f})$	Contribution to the variance of query $\mathrm{P}(\mathtt{h} \mid \mathbf{e})$ due to CPtable row $\Theta_{\mathtt{C}\|\mathbf{F}=\mathbf{f}}$ (Eq. (8))

BUCKELIM and BUCKELIM$^+$ (Section 4)	
$b_{X_i}, b_\emptyset$	"Buckets" for holding tables, including the CPtables
$f_{i,j}(\cdot)$	Function belonging to bucket b_{X_i}
$Join(\mathcal{F})$	Join of the functions $f \in \mathcal{F}$
$Elim(X, f)$	Marginalizes the variable X from table f
$maxIndex(f)$	Largest index of the variables within $Scheme(f)$ (Eq. (11))
y	Query response $y = q(\Theta) = \mathrm{P}(\mathtt{H} = \mathtt{h} \mid \mathbf{E} = \mathbf{e}, \Theta)$

Fig. 3. Notation.

Consider the density of $\Theta_{C|A=1}$ in Fig. 2. Assume the prior density is Dir(1, 1) and we have a complete data set—i.e., every instance in the set specifies the value of all variables. Only instances with $A = 1$ can contribute to $\Theta_{C|A=1}$'s posterior. Moreover, the values of C are multinomially distributed within the selected sample. If there are 34 instances where $A = 1$, of which 8 have $C = 1$ and 26 have $C = 0$, then the posterior distribution of $\Theta_{C|A=1}$ is $\mathrm{Dir}(1 + 8, 1 + 26) = \mathrm{Dir}(9, 27)$, which corresponds to the first row of C's CPtable shown in Fig. 2. This is because a Bayesian update of a Dirichlet prior with multinomial data yields a Dirichlet posterior—i.e., the Dirichlet distribution is the *conjugate prior* for multinomial distributions [43].

Bayesian parameter learning in belief networks consists of updating each row in this fashion. In the absence of prior knowledge a uniform prior, $\mathrm{Dir}(1, \ldots, 1)$, over each row is often used. (Note this distribution is "flat"; that is, every assignment is equally likely.) Frequently, the posterior Dirichlet row distributions are replaced by their expectations. In the above example, $\Theta_{C|A=1} \sim \mathrm{Dir}(9, 27)$ would be reduced to $\hat{\Theta}_{C|A=1} = \langle 0.25, 0.75\rangle$. Replacing the posterior row distributions with their expectations effectively ignores parameter uncertainty. When the network parameters are fixed real numbers, the query response will be a real number. However, uncertain network parameters



Data

A	B	C	D
1	0	1	1
1	1	0	1
0	0	0	0
1	1	0	0

Prior Distribution
$\Theta_A \sim \mathrm{Be}(1,1)$
$\Theta_{B|+a} \sim \mathrm{Be}(1,1)$
$\Theta_{B|-a} \sim \mathrm{Be}(1,1)$
$\Theta_{C|+a} \sim \mathrm{Be}(1,1)$
$\Theta_{C|-a} \sim \mathrm{Be}(1,1)$
$\Theta_{D|+b,+c} \sim \mathrm{Be}(1,1)$
$\Theta_{D|+b,-c} \sim \mathrm{Be}(1,1)$
$\Theta_{D|-b,+c} \sim \mathrm{Be}(1,1)$
$\Theta_{D|-b,-c} \sim \mathrm{Be}(1,1)$
A
B
C
D
COMPUTEPOSTERIOR
Posterior Distribution
$\Theta_A \sim \mathrm{Be}(4,2)$
$\Theta_{B|+a} \sim \mathrm{Be}(3,2)$
$\Theta_{B|-a} \sim \mathrm{Be}(1,2)$
$\Theta_{C|+a} \sim \mathrm{Be}(2,4)$
$\Theta_{C|-a} \sim \mathrm{Be}(1,2)$
$\Theta_{D|+b,+c} \sim \mathrm{Be}(1,1)$
$\Theta_{D|+b,-c} \sim \mathrm{Be}(2,2)$
$\Theta_{D|-b,+c} \sim \mathrm{Be}(2,1)$
$\Theta_{D|-b,-c} \sim \mathrm{Be}(1,2)$

Fig. 4. Example of COMPUTEPOSTERIOR.

will induce a distribution over the query response; that is, if Θ is a random variable, then $q(\Theta)$ is also a random variable. Furthermore, Cooper and Herskovits [9] have shown that, under our assumptions (Definition 1),

$$\mathrm{E}\big[q(\Theta)\big] = q\big(\mathrm{E}[\Theta]\big) = q(\hat{\Theta}) \tag{3}$$

That is, the posterior expectation of $q(\Theta)$ is simply the query response on the network with mean parameters. Recall that $\hat{\Theta} = \langle \hat{\Theta}_1, \ldots, \hat{\Theta}_r \rangle$ is encoded in the posterior row distributions, $\mathrm{Dir}(m_{\mathrm{C}|\mathbf{f}}; \hat{\Theta}_1, \ldots, \hat{\Theta}_r)$ in Eq. (1).

There are a number of algorithms for computing the query response when the network parameters are fixed values. Section 4 will describe the one we will use below, Bucket Elimination. Such algorithms calculate the expected value of the response $q_{\mathrm{h}|\mathbf{e}}(\hat{\Theta})$ but do not provide the variance:

$$\sigma^2_{\mathrm{h}|\mathbf{e}} = \mathrm{E}\big[\big(q_{\mathrm{h}|\mathbf{e}}(\Theta) - q_{\mathrm{h}|\mathbf{e}}(\hat{\Theta})\big)^2\big]$$

Theorem 2 provides an asymptotic form for this variance, and Section 4 describes an algorithm, MEANVAR, that can effectively compute this quantity in general.

2.3. ComputeErrorBars: *Error bars for query response*

ComputeErrorBars uses the mean and variance of a query response to estimate the posterior distribution of $q(\Theta)$, which is then used to construct approximate Bayesian credible regions (error bars). This depends on the actual form of the distribution; here we consider two classes, Normal and Beta. In the latter case, we need to find the Beta parameters α and β that correspond to a given mean μ and variance σ^2, namely

$$\alpha = \mu \frac{\mu(1-\mu) - \sigma^2}{\sigma^2} \quad \text{and} \quad \beta = (1-\mu)\frac{\mu(1-\mu) - \sigma^2}{\sigma^2} \tag{4}$$

(If either α or β is non-positive, then the associated density function is not well-defined. Also, if either parameter is less than 1, associated density has mode(s) at 0 and/or 1.)

For any distribution $\mathcal{D}$ and any $\delta \in (0, 1)$, a $1 - \delta$ credible region for the query response is a set $\omega(\mathcal{D})$ such that $\mathrm{P}(q(\Theta) \in \omega(\mathcal{D})) = 1 - \delta$. While there are infinitely many credible intervals on a continuous distribution, our ComputeErrorBars system chooses contiguous intervals of the form $[L, U]$ such that the mass below the lower bound L equals the mass above the upper bound U. Letting $\mathrm{CDF}^{-1}(\cdot; q_{\mathrm{h}|\mathbf{e}}(\Theta))$ denote the inverse cumulative density function of our model of $q_{\mathrm{h}|\mathbf{e}}(\Theta)$, then

$$L = \mathrm{CDF}^{-1}\left(\frac{\delta}{2};\ q_{\mathrm{h}|\mathbf{e}}(\Theta)\right) \qquad U = \mathrm{CDF}^{-1}\left(1 - \frac{\delta}{2};\ q_{\mathrm{h}|\mathbf{e}}(\Theta)\right)$$

Note that nothing in our framework precludes the use of other criteria for interval selection (e.g., shortest contiguous $1 - \delta$ interval).

3. The posterior distribution of a response

The MeanVar subroutine requires an effective way to compute the variance of the response $\tilde{\sigma}^2_{\mathrm{h}|\mathbf{e}} = \tilde{\sigma}^2_{\mathrm{h}|\mathbf{e}}(\Theta)$. Our main theorem provides a closed-form expression for (an asymptotic approximation of) this quantity, given the following assumptions:

Definition 1. The independent Dirichlet property with respect to a DAG $\langle \mathcal{V}, \mathcal{A} \rangle$ holds under the following conditions:

(1) True Structure: $\langle \mathcal{V}, \mathcal{A} \rangle$ is an I-map of the underlying joint distribution [34]—i.e., every independence claim implied by the graph holds in the underlying distribution.
(2) Parameter Independence: The distribution of a variable conditioned on one assignment to its parents is independent of the distribution conditioned on any other assignment to its parents. That is, if $\mathbf{f}_1 \neq \mathbf{f}_2$, then $\Theta_{C|\mathbf{f}_1}$ and $\Theta_{C|\mathbf{f}_2}$ are independent (local parameter independence). Furthermore, the local conditional probability tables are independent of one another (global parameter independence); see [39].
(3) Dirichlet Assumption: The distribution of a variable given an assignment to its parents is Dirichlet distributed: $\Theta_{\mathrm{C}|\mathbf{F}=\mathbf{f}} \sim \mathrm{Dir}(m_{\mathrm{C}=\mathrm{c}_1|\mathbf{f}}, \ldots, m_{\mathrm{C}=\mathrm{c}_r|\mathbf{f}})$.
(4) Non-degenerate Posterior Means: The expected posterior parameters $\hat{\Theta}_{\mathrm{c}|\mathbf{f}}$ are strictly between 0 and 1.[5]

Note these assumptions conform with those made in *maximum a posteriori* learning of belief network parameters [22].

As we are assuming that each network parameter $\Theta_{\mathrm{C}=\mathrm{c}|\mathbf{F}=\mathbf{f}} = \Theta_{\mathrm{c}|\mathbf{f}}$ is a random variable, the query response is a function of the network parameters, and is thus itself a random variable. We will continue to use $\hat{\Theta} = \{\hat{\Theta}_{\mathrm{c}|\mathbf{f}}\} = \mathrm{E}[\Theta]$ to refer the expected value of the parameters, under the posterior distribution, conditioned implicitly on observed data and/or expert opinion.

[5] We discuss this constraint below—both after the statement of Theorem 2 and also following its proof in Appendix B.

As mentioned above, we denote the query response as $q(\Theta)$ to emphasize that it is simply a function from an instantiation of the network parameters to a real number in [0, 1]. For example, the query $q_{+a|+b}(\Theta) = \mathrm{P}(+a \mid +b, \Theta)$ from Fig. 2[6] can be viewed as

$$\begin{aligned} q_{+a|+b}(\Theta) &= q_{+a|+b}(\Theta_{+a|\{\}}, \Theta_{-a|\{\}}, \Theta_{+b|+a}, \ldots) \\ &= \frac{\Theta_{+b|+a} \times \Theta_{+a|\{\}}}{\Theta_{+b|+a} \times \Theta_{+a|\{\}} + \Theta_{+b|-a} \times \Theta_{-a|\{\}}} \end{aligned} \tag{5}$$

For each network parameter $\Theta_{\mathtt{c}|\mathbf{f}} \in \Theta$, we can consider the partial derivative of the query with respect to that parameter:

$$q^{(\mathtt{c}|\mathbf{f})}(\theta) = \left.\frac{\partial q(\Theta)}{\partial \Theta_{\mathtt{c}|\mathbf{f}}}\right|_{\theta} \tag{6}$$

which is functional in θ and so can be evaluated at any specific instantiation of the network parameters. Using the $q_{+a|+b}(\Theta)$ function from Eq. (5), observe that

$$q^{(+b|+a)}(\Theta) = \Theta_{+a|\{\}} \times \frac{(\Theta_{+b|+a}\Theta_{+a|\{\}} + \Theta_{+b|-a}\Theta_{-a|\{\}}) - (\Theta_{+b|+a}\Theta_{+a|\{\}})}{(\Theta_{+b|+a}\Theta_{+a|\{\}} + \Theta_{+b|-a}\Theta_{-a|\{\}})^2}$$

We can now state our main technical result:

Theorem 2. *Given the "independent Dirichlet property wrt a given DAG" assumption, the posterior mean of* $q_{\mathtt{h}|\mathbf{e}}(\Theta)$ *is* $\mathrm{E}[q_{\mathtt{h}|\mathbf{e}}(\Theta)] = q_{\mathtt{h}|\mathbf{e}}(\hat{\Theta})$. *Now consider an asymptotic framework where* $\min\{m_{\mathtt{C}|\mathbf{f}}\} \to \infty$ *while the posterior means* $\hat{\Theta}_{\mathtt{c}|\mathbf{f}}$ *remain fixed at values strictly between* 0 *and* 1, *and let*

$$\tilde{\sigma}^2_{\mathtt{h}|\mathbf{e}} = \sum_{(\mathtt{C},\mathbf{F}=\mathbf{f})} \frac{1}{1+m_{\mathtt{C}|\mathbf{f}}} v_{\mathtt{h}|\mathbf{e}}(\mathtt{C}|\mathbf{f}) \tag{7}$$

$$v_{\mathtt{h}|\mathbf{e}}(\mathtt{C}|\mathbf{f}) = \left(\sum_{\mathtt{c}\in\mathtt{C}} q_{\mathtt{h}|\mathbf{e}}^{(\mathtt{c}|\mathbf{F}=\mathbf{f})}(\hat{\Theta})^2\, \hat{\Theta}_{\mathtt{c}|\mathbf{F}=\mathbf{f}}\right) - \left(\sum_{\mathtt{c}\in\mathtt{C}} q_{\mathtt{h}|\mathbf{e}}^{(\mathtt{c}|\mathbf{F}=\mathbf{f})}(\hat{\Theta})\, \hat{\Theta}_{\mathtt{c}|\mathbf{F}=\mathbf{f}}\right)^2 \tag{8}$$

Under this framework, the standardized random variable

$$\frac{q_{\mathtt{h}|\mathbf{e}}(\Theta) - q_{\mathtt{h}|\mathbf{e}}(\hat{\Theta})}{\tilde{\sigma}_{\mathtt{h}|\mathbf{e}}} \tag{9}$$

converges in distribution to the standard Normal distribution $\mathcal{N}(0, 1)$.

Given the form of Eq. (9), we see that $\tilde{\sigma}^2_{\mathtt{h}|\mathbf{e}}$ (Eq. (7)) corresponds to the asymptotic variance. (Recall from Eq. (3) that $q(\hat{\Theta}) = \mathrm{E}[q(\Theta)]$ is the mean.)

Each summation in Eq. (8) is over values of a particular variable $\mathtt{C}$—so if this variable is binary, each sum will involve one term for $+\mathtt{c}$ and another for $-\mathtt{c}$.

The summation in Eq. (7) is over all CPtable rows; hence for the Diamond network in Fig. 2, this would involve 9 terms, corresponding to 9 different $v_{\mathtt{h}|\mathbf{e}}(\mathtt{C}|\mathbf{f})$ values: $\{v_{\mathtt{h}|\mathbf{e}}(A|\{\}), v_{\mathtt{h}|\mathbf{e}}(B|+a), v_{\mathtt{h}|\mathbf{e}}(B|-a), \ldots, v_{\mathtt{h}|\mathbf{e}}(D|-b,-c)\}$. Moreover, this $v_{\mathtt{h}|\mathbf{e}}(\mathtt{C}|\mathbf{f})$ is 0 whenever $q_{\mathtt{h}|\mathbf{e}}^{(\mathtt{c}|\mathbf{f})}(\hat{\Theta}) = 0$ for each $\mathtt{c} \in \mathtt{C}$, which happens whenever this CPtable row is not involved in computing $q_{\mathtt{h}|\mathbf{e}}(\Theta) = \mathrm{P}(\mathtt{h} \mid \mathbf{e}, \Theta)$—e.g., when this $\mathtt{C}$ is d-separated from the query (i.e., $\mathtt{H} \perp \mathtt{C}, \mathbf{F}|\mathbf{E}$) or when $\mathtt{C}$ is a barren node (i.e., neither it nor any of its descendants are instantiated in this query [31]); see Appendix A. As an extreme case, imagine the query corresponded to a single CPtable entry; e.g., in Fig. 2, the query was $q_{+b|+a} = \mathrm{P}(+b \mid +a)$. Here only $v_{+b|+a}(B|+a)$ is relevant; the value of $v_{+b|+a}(\mathtt{C}|\mathbf{f})$ is 0 for the terms corresponding to the other 8 CPtable rows $(\mathtt{C}, \mathbf{f})$. As $q_{+b|+a}^{(+b|+a)}(\hat{\Theta}) = 1$ and $q_{+b|+a}^{(-b|+a)}(\hat{\Theta}) = 0$ for this query,

[6] This corresponds to $Q2$ from Eq. (16).

$$\begin{aligned}v_{+b|+a}(B\,|+a) &= \left(\sum_{b\in B} q_{+b|+a}^{(b|+a)}(\hat{\Theta})^2\ \hat{\Theta}_{b|+a}\right) - \left(\sum_{b\in B} q_{+b|+a}^{(b|+a)}(\hat{\Theta})\ \hat{\Theta}_{b|+a}\right)^2\\ &= (1^2\cdot\hat{\Theta}_{+b|+a}\ +\ 0^2\cdot\hat{\Theta}_{-b|+a}) - (1\cdot\hat{\Theta}_{+b|+a}\ +\ 0\cdot\hat{\Theta}_{-b|+a})^2\\ &= (\hat{\Theta}_{+b|+a}) - (\hat{\Theta}_{+b|+a})^2 = \hat{\Theta}_{+b|+a}\times\hat{\Theta}_{-b|+a}\end{aligned}$$

which means the variance associated for this $q_{+b|+a}$ query is

$$\tilde{\sigma}^2_{+b|+a} = \frac{1}{1+m_{B|+a}}[\hat{\Theta}_{+b|+a}\times\hat{\Theta}_{-b|+a}]$$

which as expected is exactly the variance associated with a single Dirichlet distributed variable Dir($m_{B|+a}$; $\hat{\Theta}_{+b|+a}$, $\hat{\Theta}_{-b|+a}$)—i.e., here Eq. (7) is exact and not just an approximation. Hooper [24] generalizes this to other situations where Eq. (7) is exact. (Appendix A provides another class of situations where it is easy to compute $\tilde{\sigma}^2_{\mathtt{h}|\mathbf{e}}$.)

Eqs. (7) and (8) show that $\tilde{\sigma}^2_{\mathtt{h}|\mathbf{e}}$ adds up the influence of each $\Theta_{\mathtt{C}|\mathbf{f}}$ on the query P(h | **e**), which is based on the derivative $q_{\mathtt{h}|\mathbf{e}}^{(\mathtt{c}|\mathbf{f})}(\hat{\Theta})$, weighted by $(1+m_{\mathtt{C}|\mathbf{f}})^{-1}$. So while $m_{\mathtt{C}|\mathbf{f}}$ is the same for each query, this variance will be different for different queries, as these derivatives will differ.

While Appendix B provides the full proof, we note here that both the asymptotic normality and the derivation of approximate variance (Eq. (7)) employ a first-order Taylor expansion of the function $q(\Theta)$ about the posterior mean $\hat{\Theta}$. The conditions on $\hat{\Theta}_{\mathtt{c}|\mathbf{f}}$ and $m_{\mathtt{C}|\mathbf{f}}$ ensure that this first-order approximation is asymptotically valid. Comment#1 after the proof in Appendix B discusses possible violations of these assumptions.

We can also use Eq. (7) to understand how to deal with “fixed” CPtable rows—i.e., specific $\Theta_{\mathtt{c}|\mathbf{f}}$ entries that correspond to definitions, and so can be viewed as constants rather than variables. Here, this corresponds to using an effectively infinite $m_{\mathtt{C}|\mathbf{f}}$ value, which means this row will contribute $\frac{1}{1+m_{\mathtt{C}|\mathbf{f}}}v_{\mathtt{h}|\mathbf{e}}(\mathtt{C}|\mathbf{f})\approx 0$ to the variance approximation—i.e., Eq. (7)’s summation can simply ignore these CPtable rows. Here, we can of course allow a CPtable entry to be deterministic—i.e., 0 or 1.

Section 4 presents an algorithm that can compute both the mean and variance of the query response in $O(n\cdot\exp(w))$, where n is the number of variables in the network and w is the induced tree width [14]. This is the same worst-case complexity as inference alone.

3.1. The Beta approximation to the posterior

Theorem 2 suggests using a Normal distribution to approximate the response posterior distribution; however, the response is confined to [0, 1], whereas a Normal distribution has positive density on the entire real line. When the variance is large, the Normal approximation may deviate considerably from the true posterior. Furthermore, we expect the response distribution may often be skewed when its mean is near 0 or 1. It may therefore make sense to use a Beta distribution instead, as it is confined to [0, 1], can model skewed distributions, and when standardized, it converges in

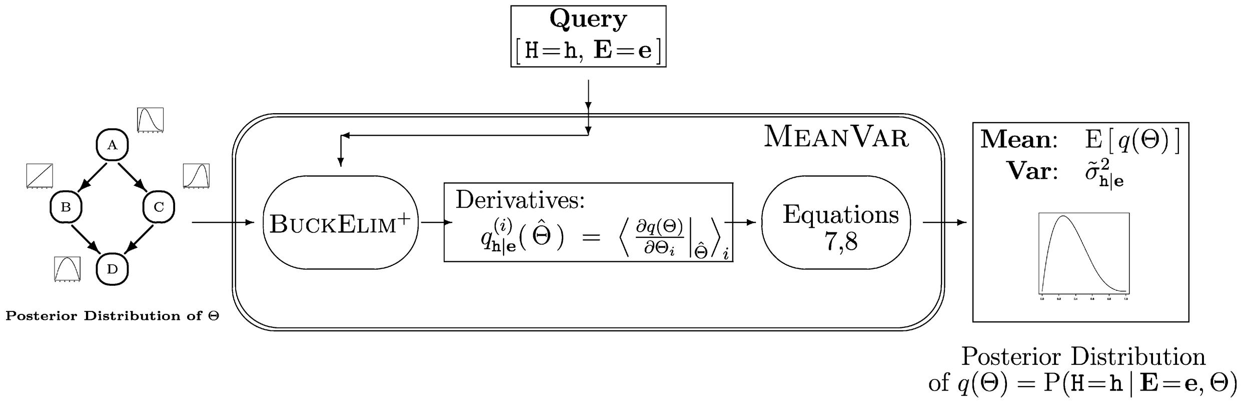

Query
[H=h, E=e]
MEANVAR
A
B
C
D
Posterior Distribution of Θ
BUCKELIM+
Derivatives:
$q^{(i)}_{\mathtt{h}|\mathbf{e}}(\hat{\Theta}) = \left\langle \frac{\partial q(\Theta)}{\partial \Theta_i}\Big|_{\hat{\Theta}} \right\rangle_i$
Equations 7,8
Mean: E[q(Θ)]
Var: $\tilde{\sigma}^2_{\mathtt{h}|\mathbf{e}}$
Posterior Distribution of q(Θ) = P(H=h | E=e, Θ)

Fig. 5. MEANVAR Process (including BUCKELIM+).

distribution to a standard Normal random variable [3]. Eq. (4) shows how to map the computed mean μ and variance quantities σ^2 to the Beta distribution's α and β parameters.

4. Computing the variance: BUCKELIM$^+$ and MEANVAR

Eqs. (7) and (8) within Theorem 2 approximate the variance of the response. This section sketches the MEANVAR algorithm (Fig. 5) for computing this approximate variance. In particular, these equations involve the derivative of the response with respect to each parameter. The BUCKELIM$^+$ algorithm, appearing in Section 4.2, computes these values. That section also presents theorems showing the correctness and efficiency of this algorithm. As BUCKELIM$^+$ is basically an extension of the Bucket Elimination algorithm, we first review that algorithm in Section 4.1. Finally, Section 4.3 presents some remaining issues; in particular, it explicitly describes how the results of the BUCKELIM$^+$ algorithm can be used to compute the variance.

There are many ways to optimize our overall MEANVAR system. First, note that MEANVAR uses BUCKELIM$^+$ which in turn is based on BUCKELIM. As BUCKELIM is a standard, well-studied algorithm, there are now a number of known effective optimizations, including graph reduction, heuristics for finding a good ordering, etc.; each of these translates directly to an improvement to BUCKELIM$^+$ and hence to MEANVAR. Second, we can also find other insights by squinting at Eq. (8). In particular, Appendix A provides straight-line code that applies to the special case when the query is complete (i.e., when the evidence includes all variables except the query variable).

4.1. *Bucket elimination algorithm,* BUCKELIM

The bucket elimination algorithm BUCKELIM [14] is an elegant framework for belief net inference in general, which uses non-serial dynamic programming to iteratively eliminate variables by marginalization.

Notation. We must first introduce the notation of a *table*, which is a function mapping a set of *named* variables to $\Re$. For example,

$f(A, B) =$

a	b	$f(A=a, B=b)$
1	1	0.30
1	0	0.70
0	1	0.91
0	0	0.09

Notice this table corresponds to the $\Theta_{B|A}$ function shown in Fig. 2. A table's input variables are called its *scheme*; here *Scheme*$(f) = \{A, B\}$.

We next define two operations on tables, *Join*$(\cdot)$ and *Elim*$(\cdot,\cdot)$. The *join* operation combines a set of tables into a new table: Letting $h = \mathit{Join}(\mathcal{F})$, then $\mathit{Scheme}(h) = \bigcup_{f\in\mathcal{F}} \mathit{Scheme}(f)$ is the union of the variables of the tables, and h's values are

$$h(\mathbf{x}) = \prod_{f\in\mathcal{F}} f(\mathbf{x}|_{\mathit{Scheme}(f)})$$

where "$\mathbf{x}|_\mathbf{S}$" denotes the projection of an assignment $\mathbf{x} \in \mathbf{X}$ onto a subset of its variables $\mathbf{S} \subseteq \mathbf{X}$; e.g., $\langle A=1, B=0, C=0\rangle|_{\{A,C\}} = \langle A=1, C=0\rangle$. Thus, the value of h on $\mathbf{x}$ is the product of the $f \in \mathcal{F}$s evaluated on the projections of $\mathbf{x}$ onto the scheme for that f. We will often simplify our notation by using *Join*(f, g) as an abbreviation for *Join*$(\{f, g\})$, etc. Note *Join*$(\{f\}) = f$ and *Join*$(\{\})$ is undefined. To make this *Join* more concrete, consider the table

$g(A, C) =$

a	c	$g(A=a, C=c)$
1	1	0.22
1	0	0.78
0	1	0.99
0	0	0.01

Then $h = \mathit{Join}(f, g)$ is

$h(A, B, C) =$

a	b	c	$h(A = a, B = b, C = c)$
1	1	1	0.3×0.22
1	1	0	0.3×0.78
1	0	1	0.7×0.22
1	0	0	0.7×0.78
0	1	1	0.91×0.99
0	1	0	0.91×0.01
0	0	1	0.09×0.99
0	0	0	0.09×0.01

This operation is similar to the relational join used in database theory [28].

The *elimination* operation reduces a table f by marginalizing over one of the variables in f's scheme, producing a function $f_{-X} = \mathit{Elim}(X, f)$ using the variables $\mathit{Scheme}(f) - \{X\}$. Assuming $\mathit{Scheme}(f) = \{X, Y_1, \dots, Y_k\}$,

$$f_{-X}(y_1, \dots, y_k) = \sum_x f(x, y_1, \dots, y_k)$$

Hence, using the f defined above,

$\mathit{Elim}(A, f) = f_{-A}(B) =$

b	$f_{-A}(B = b)$
1	$0.3 + 0.91$
0	$0.7 + 0.09$

We can eliminate a *set* of variables, by repeating this marginalization process iteratively, for each variable; here, we would write $\mathit{Elim}(\{X_i, \dots, X_j\}, f)$. (Observe we obtain the same result independent of the order in which we eliminate variables $\{X_i, \dots, X_j\}$.) Notice $\mathit{Elim}(\{\}, f) \equiv f$.

BUCKELIM **Algorithm.** The bucket elimination algorithm BUCKELIM takes as input a belief net, encoded as its set of associated CPtables $\mathcal{F}$ over the variables $\mathcal{V}$, a partial assignment to a subset of the variables $\mathbf{E} = \mathbf{e}$, and an ordering of the variables—e.g., $\pi_0 = \langle A, B, C, D\rangle$. Below, we will assume the variables are numbered in order: $\langle X_1, X_2, \dots, X_n\rangle$. BUCKELIM then computes the value:

$$\mathrm{P}(\mathbf{E} = \mathbf{e}) = y = \sum_{\{\mathbf{x}:\, \mathbf{x}|_{\mathbf{E}} = \mathbf{e}\}} \prod_{f \in \mathcal{F}} f(\mathbf{x}|_{\mathit{Scheme}(f)}) \tag{10}$$

where the sum is over all assignments to $\mathcal{V}$ that include the partial assignment $\mathbf{E} = \mathbf{e}$. Of course, we could compute y by simply joining all the functions in $\mathcal{F}$ and eliminating the variables in $\mathcal{V}$ not in $\mathbf{E}$. However, BUCKELIM achieves better efficiency than this brute force approach by exploiting the fact that each function depends only on the variables in its scheme.

To do this, BUCKELIM builds and uses a set of *buckets* $\{b_{X_i}\}_i$ associated with each variable $X_i \in \mathcal{V}$, and also one additional bucket $b_\emptyset$. Each bucket will hold a set of functions. The buckets are ordered according to π, with $b_\emptyset$ at the beginning. BUCKELIM initially loads the input functions from $\mathcal{F}$ (that is, the belief net's CPtables) into the buckets, as follows: Recalling that each table f uses a set of variables $\mathit{Scheme}(f) = \{X_i\}_i$, let

$$\mathit{maxIndex}(f) = \operatorname*{argmax}_i \bigl\{X_i \in \mathit{Scheme}(f)\bigr\} \tag{11}$$

be the largest index of f's variables, based on the π order. We then assign f to the $\mathit{maxIndex}(f)$ bucket. (If $\mathit{Scheme}(f) = \emptyset$, we define $\mathit{maxIndex}(f) = \emptyset$.) Hence, using π_0, we would assign $\Theta_{A|\{\}} = \Theta_A$ to bucket b_A, $\Theta_{B|A}$ to bucket b_B, $\Theta_{C|A}$ to bucket b_C and $\Theta_{D|B,C}$ to bucket b_D. (Here, there happened to be a one-to-one mapping between CPtable functions and buckets; that is not true in general.[7])

[7] For example, in the ordering $\langle D, C, B, A\rangle$, the b_D and b_C buckets would both be empty, and b_B bucket would include the function $\Theta_{D|B,C}$ and the b_A bucket would include the three functions $\Theta_{A|\{\}}$, $\Theta_{B|A}$ and $\Theta_{C|A}$. The rest of the text will consider only the π_0 ordering.

Step	$b_\emptyset$	b_A	b_B	b_C	b_D
0	–	$f_{A,1}(a) = \Theta_{A=a\|\{\}}$	$f_{B,1}(b,a) = \Theta_{B=b\|A=a}$	$f_{C,1}(c,a) = \Theta_{C=c\|A=a}$ $f_{C,2}(b,c) = \Theta_{D=1\|B=b,C=c}$	–
1	–	$f_{A,1}(a) = \Theta_{A=a\|\{\}}$	$f_{B,1}(b,a) = \Theta_{B=b\|A=a}$ $\boxed{f_{B,2}(a,b) = g_C(a,b)}$		
2	–	$f_{A,1}(a) = g_B(a)$ $\boxed{f_{A,2}(a) = g_B(a)}$			
3	$\boxed{f_{\emptyset,1}() = g_A()}$				

Fig. 6. Trace of BUCKELIM algorithm, on P($D = 1$).

BUCKELIM uses these buckets to answer queries. It traverses these buckets, in *reverse* order (here, this means processing b_D first, then b_C, b_B, b_A and $b_\emptyset$). To process each bucket b_{X_j}: If the bucket is empty, BUCKELIM will skip it. Otherwise, BUCKELIM joins the functions in the bucket, then eliminates the variable X_j, producing

$$g_j = Elim\big(X_j, Join(b_{X_j})\big). \tag{12}$$

We then store g_j in the bucket b_{X_i}, where $i = maxIndex(g_j)$. That is, if $Scheme(g_j) = \emptyset$, we store g_j in the $b_\emptyset$ bucket. Otherwise, $Scheme(g_j)$ has a variable with the highest index, say X_i. (Note that $i < j$, as j had been the largest index of each of the functions in b_{X_j}, and it is now eliminated.) BUCKELIM then stores g_j in the b_{X_i} bucket, and continues down the ordering, to process $b_{X_{j-1}}$. At the end, it processes the $b_\emptyset$ bucket—which involves $Join(\cdot)$-ing constant functions, which corresponds to a simple scalar multiplication. BUCKELIM returns the resulting scalar.

The algorithm has one final complication: The process suggested above would compute P({}), which is 1. In general we want to compute an expression of the form P($\mathbf{E} = \mathbf{e}$). To do this, we first initialize the relevant CPtable functions to correspond to those instantiations, and use only those "restricted" functions.

Example. To illustrate this, consider the P($D = 1$) query, using the π_0 ordering. Observe first that we do not need all of D's CPtable entries, but only $f_{D=1,B,C}(b,c) = \Theta_{D=1|b,c}(b,c)$. As this is a function of B and C, but not D—that is $maxIndex(f_{D=1,B,C}) = C$—we store this function in bucket b_C. Hence, when BUCKELIM starts, its 5 buckets are configured as shown in Step#0 of Fig. 6.

BUCKELIM then processes the buckets in reverse order. As b_D is empty, BUCKELIM's first non-trivial operation deals with b_C; here it joins the two functions in that bucket, and eliminates the variable C; this produces[8]

$$\begin{aligned} g_C(a,b) &= Elim\big(C, Join(f_{C,1}, f_{C,2})\big) = \sum_c \Theta_{C=c|A=a}\,\Theta_{D=1|B=b,C=c} \\ &= \sum_c \mathrm{P}(D=1, C=c \mid B=b, A=a) = \mathrm{P}(D=1 \mid B=b, A=a) \end{aligned}$$

As this function has the scheme $Scheme(g_C) = \{A, B\}$, it is stored in the bucket b_B, where it is called $f_{B,2}$; see the Step#1 entry in Fig. 6. BUCKELIM then sums out b_B, computing

$$\begin{aligned} g_B(a) &= Elim\big(B, Join(f_{B,1}, f_{B,2})\big) = \sum_b \Theta_{B=b|A=a}\,\mathrm{P}(D=1 \mid B=b, A=a) \\ &= \sum_b \mathrm{P}(D=1, B=b \mid A=a) = \mathrm{P}(D=1 \mid A=a) \end{aligned}$$

[8] In some cases, the result of the computation has an easy interpretation; such as P($D = 1 \mid B = b, A = a$) shown here. This is not always the case.

As this function involves only the variable A, it is stored in b_A and called $f_{A,2}$. It next sums out b_A to produce a constant

$$g_A() = Elim\big(A, Join(f_{A,1}, f_{A,2})\big) = \sum_a \mathrm{P}(A=a)\,\mathrm{P}(D=1 \mid A=a) = \mathrm{P}(D=1)$$

which is stored in $b_\emptyset$; see Step#2 and Step#3 of Fig. 6. BUCKELIM then returns $f_{\emptyset,1}()$, as the correct response to this query.

4.2. *Extension to compute derivatives,* BUCKELIM$^+$

BUCKELIM$^+$ takes the same inputs as BUCKELIM: a belief net (represented by a set of CPtable functions $\mathcal{F}$ over the variables $\mathcal{V}$), an ordering of the variables π, and a partial assignment to the variables $\mathbf{E}=\mathbf{e}$. It returns both the expected response (the $y = \mathrm{P}(\mathbf{E}=\mathbf{e}) = E_\Theta[P(E=e|\Theta)]$ shown in Eq. (10)) and also, for each CPtable $f_i \equiv \Theta_i$, the partial derivative $\frac{\partial y}{\partial f_i} = q^{(i)}(\Theta) = \partial q(\Theta)/\partial\Theta_i$. For notation, we will continue to name the functions appearing in bucket b_{X_i} as $\{f_{i,j}\}_j = \{f_{i,1}, f_{i,2}, \ldots\}$; e.g., the functions within b_A are $\{f_{A,1},\ f_{A,2}\}$. Notice every such function is either an original CPtable (or perhaps a subset of such a table; e.g., $f_{C,2} = \Theta_{D=1|B,C}$) or is a function produced by first joining a set of previously produced functions in a common bucket, then eliminating a variable from the resulting function—e.g.,

$$f_{X,2}(\cdot) = g_Y(\cdot) = Elim\big(Y, Join(f_{Y,1},\ f_{Y,2},\ f_{Y,3})\big) \tag{13}$$

We will continue to let $g_i(\cdot)$ refer to the "output" of bucket b_{X_i}.

BUCKELIM$^+$ uses this information to compute the partial derivative $\partial q(\Theta)/\partial f_{i,j}$ of each function appearing in each bucket (which includes each Θ_i CPtable function). It does this using the chain rule: e.g., using Eq. (13),

$$\frac{\partial q(\Theta)}{\partial f_{Y,1}} = \frac{\partial q(\Theta)}{\partial f_{X,2}} \times \frac{\partial f_{X,2}}{\partial f_{Y,1}}$$

Moreover, given the form of Eq. (13), we see that $\frac{\partial f_{X,2}}{\partial f_{Y,1}}$ corresponds roughly to $Join(f_{Y,2},\ f_{Y,3})$, after marginalizing out some variables; see Eq. (14) below. The only remaining trick is to process the buckets in the *forward* direction, to make sure we have computed $\frac{\partial q(\Theta)}{\partial f_{X,2}}$ when we are asked to compute $\frac{\partial q(\Theta)}{\partial f_{Y,1}}$.

The overall algorithm is then...

Step A Run BUCKELIM on its inputs to compute y. The intermediate results (the functions $g_i(\cdot)$ created as the output of each bucket) are stored for use in Step B.

Step B Compute the derivative function for each bucket function f_{ij} as follows.

(1) Compute $\frac{\partial y}{\partial g_\emptyset}$ for the output of $b_\emptyset$: $g_\emptyset = Elim(\{\}, Join(b_\emptyset)) = \prod_i f_{\emptyset,i}$ (Eq. (12)). This is trivial, as $g_\emptyset$ is a constant $y = g_\emptyset$; thus $\frac{\partial y}{\partial g_\emptyset} = 1$.

(2) Iterate over the remaining buckets in the order *reversed* from BUCKELIM—i.e., in the original order π, starting with $b_\emptyset$.

Let b_{X_i} be the current bucket, whose output is $g_i = Elim(X_i, Join(b_{X_i}))$. Now recall that $maxIndex(g_i) < i$, as g_i was formed by joining functions from the X_i bucket (whose variables had indices at most i), and then eliminating X_i. As we are processing buckets *in order*, BUCKELIM$^+$ will have computed $\frac{\partial y}{\partial g_i}$ before reaching b_{X_i}.

For each $f_{ij} \in b_{X_i}$, compute $\frac{\partial y}{\partial f_{ij}}$ using:

$$\mathcal{J} = Join\left(\left\{\frac{\partial y}{\partial g_i}\right\} \cup \{h \in b_{X_i}\colon\ h \neq f_{ij}\}\right)$$

$$\frac{\partial y}{\partial f_{ij}} = Elim\big(Scheme(\mathcal{J}) - Scheme(f_{ij}), \mathcal{J}\big) \tag{14}$$

(The first argument of *Elim* is $Scheme(\mathcal{J}) - Scheme(f_{ij}) = \{X \in Scheme(\mathcal{J}) :\ X \notin Scheme(f_{ij})\}$.) That is, we join $\frac{\partial y}{\partial g_i}$ and all the functions of the bucket *other than* f_{ij}, then eliminate the variables that do not appear in $Scheme(f_{ij})$ to produce a function $\frac{\partial y}{\partial f_{ij}}$ whose scheme $Scheme(\frac{\partial y}{\partial f_{ij}})$ equals $Scheme(f_{ij})$, and whose values are the partial derivatives of y with respect to f_{ij}. (Note that $q^{(C|\mathbf{f})}(\cdot) = \frac{\partial y}{\partial f_{C,1}}$ in general.)

Example. We already saw that, in $b_\emptyset$, $\frac{\partial y}{\partial g_\emptyset} = 1$, which means $\frac{\partial y}{\partial f_{\emptyset,1}} = 1$. In b_A, as $f_{\emptyset,1}$ was formed from the $f_{A,i}$'s,

$$\begin{aligned} q^{(A|\{\})}(a) &= \frac{\partial y}{\partial f_{A,1}} = Elim\left(\{\}, Join\left(\frac{\partial y}{\partial f_{\emptyset,1}}, \frac{\partial f_{\emptyset,1}}{\partial f_{A,1}}\right)\right) \\ &= \frac{\partial y}{\partial f_{\emptyset,1}} \times \frac{\partial f_{\emptyset,1}}{\partial f_{A,1}} = 1 \times \prod_{j\neq 1} f_{A,j} = f_{A,2} = \mathrm{P}(D=1 \mid A=a) \end{aligned}$$

That is, as $f_{A,1} = \Theta_{A|\{\}}$, $\frac{\partial y}{\partial \Theta_{A|\{\}}} = \mathrm{P}(D=1 \mid A=a)$. This makes sense, as $y = \mathrm{P}(D=1) = \sum_a \mathrm{P}(D=1 \mid A=a) \times \mathrm{P}(A=a)$ and $f_{A,1}(a) = \mathrm{P}(A=a)$. Notice $Scheme(\frac{\partial y}{\partial f_{A,1}}) = \{A\} = Scheme(f_{A,1})$. Similarly

$$\frac{\partial y}{\partial f_{A,2}} = f_{A,1}(a) = \mathrm{P}(A=a)$$

BUCKELIM$^+$ continues working forward in the order—next processing the functions within b_B: Here, as $f_{B,1}$ was used to produce $f_{A,2}$,

$$\begin{aligned} q^{(B|A)}(a,b) &= \frac{\partial y}{\partial f_{B,1}} = Elim\left(\{\}, \frac{\partial y}{\partial f_{A,2}} \times \frac{\partial f_{A,2}}{\partial f_{B,1}}\right) = \mathrm{P}(A=a) \times \prod_{j\neq 1} f_{B,j} \\ &= \mathrm{P}(A=a) \times f_{B,2} = \mathrm{P}(A=a) \times \mathrm{P}(D=1 \mid B=b, A=a) \end{aligned}$$

and $\frac{\partial y}{\partial f_{B,2}} = Elim(\{\}, \mathrm{P}(A=a) \times \mathrm{P}(B=b \mid A=a)) = \mathrm{P}(A=a, B=b)$. Finally, as $f_{C,1}$ and $f_{C,2}$ were used to produce $f_{B,2}$,

$$\begin{aligned} q^{(C|A)}(a,c) &= \frac{\partial y}{\partial f_{C,1}} = Elim\left(B, \frac{\partial y}{\partial f_{B,2}} \times \frac{\partial f_{B,2}}{\partial f_{C,1}}\right) = \sum_b \mathrm{P}(A=a, B=b) \times \prod_{j\neq 1} f_{C,j} \\ &= \sum_b \mathrm{P}(A=a, B=b) \times f_{C,2} \\ &= \sum_b \mathrm{P}(A=a, B=b) \times \mathrm{P}(D=1 \mid B=b, C=c) \end{aligned}$$

and $q^{(D=1|B,C)}(b,c) = \frac{\partial y}{\partial f_{C,2}} = Elim(A, \frac{\partial y}{\partial f_{B,2}} \times \frac{\partial f_{B,2}}{\partial f_{C,1}}) = \sum_a \mathrm{P}(A=a, B=b) \times \mathrm{P}(C=c \mid A=a)$.

Appendix B provides the proofs for following two theorems, which respectively prove that this algorithm is correct, and bound its computational complexity.

Theorem 3. BUCKELIM$^+$ *correctly computes the partial derivatives of the response wrt each CPtable* $q^{C|\mathbf{F}}(\Theta) = \partial y / \partial \Theta_{C|\mathbf{F}}$.

Theorem 4. *The worst case complexity of* BUCKELIM$^+$ *is* $O(n \cdot r^w)$ *time, where* n *is the number of nodes,* r *is the* (*fixed*) *size of largest domain over the variables, and* w *is the induced tree width, given the ordering.*

4.3. How MEANVAR *uses* BUCKELIM$^+$

The basic BUCKELIM algorithm deals only with the unconditional probabilities. To compute *conditional* probabilities requires a summation and a division: That is, for $\mathrm{P}(\mathtt{H}=\mathtt{h} \mid \mathbf{E}=\mathbf{e}) = \frac{\mathrm{P}(\mathtt{H}=\mathtt{h}, \mathbf{E}=\mathbf{e})}{\mathrm{P}(\mathbf{E}=\mathbf{e})}$, these inference algorithms will first compute $\mathrm{P}(\mathtt{H}=\mathtt{h}_i, \mathbf{E}=\mathbf{e})$ for each i, then sum these values to compute $\mathrm{P}(\mathbf{E}=\mathbf{e}) = \sum_i \mathrm{P}(\mathtt{H}=\mathtt{h}_i, \mathbf{E}=\mathbf{e})$; finally it returns $\mathrm{P}(\mathtt{H}=\mathtt{h} \mid \mathbf{E}=\mathbf{e}) = \mathrm{P}(\mathtt{H}=\mathtt{h}, \mathbf{E}=\mathbf{e}) / \mathrm{P}(\mathbf{E}=\mathbf{e})$.

Our MEANVAR will therefore first call BUCKELIM$^+$ on $\text{P}(\mathtt{H} = \mathtt{h}, \mathbf{E} = \mathbf{e})$, then on $\text{P}(\mathbf{E} = \mathbf{e})$. In each case, after computing the response, BUCKELIM$^+$ will also compute the derivatives $\partial\text{P}(\mathtt{H} = \mathtt{h}, \mathbf{E} = \mathbf{e})/\partial\Theta_i$ (resp., $\partial\text{P}(\mathbf{E} = \mathbf{e})/\partial\Theta_i$) for *all* of the parameters Θ_i's, evaluated at $\hat{\Theta}$. MEANVAR then uses the quotient rule for derivatives,

$$\frac{\partial\text{P}(\mathtt{H} = \mathtt{h} \mid \mathbf{E} = \mathbf{e})}{\partial\Theta_i} = \frac{1}{\text{P}(\mathbf{E} = \mathbf{e})}\left[\frac{\partial\text{P}(\mathtt{H} = \mathtt{h}, \mathbf{E} = \mathbf{e})}{\partial\Theta_i} - q(\Theta)\frac{\partial\text{P}(\mathbf{E} = \mathbf{e})}{\partial\Theta_i}\right]$$

to compute the actual derivatives we need. Given the complexity results in Theorem 4, it is trivial to observe that MEANVAR is also $\text{O}(n \cdot \exp(w))$ time.

5. Experiments

Theorem 2 provides an approximation of the variance of a query response, $\tilde{\sigma}^2_{\mathtt{h}|\mathbf{e}}$ that is asymptotically accurate. When network parameters are estimated from small or even moderately sized data sets, there is no guarantee on the accuracy of $\tilde{\sigma}^2_{\mathtt{h}|\mathbf{e}}$. However, Section 5.2 shows that the approximation works well in practice.

Given the true mean and approximate variance of a query response, we considered two models for the distribution of $q(\Theta)$: Normal and Beta, based respectively on asymptotic behavior, and observations of issues like support. Section 5.3 compares the fit of each model to samples from $q(\Theta)$. An important application of the query response model is the estimation of error-bars, which is explored in Section 5.4. Finally, some researchers have suggested that the variance is well approximated as a simple binomial expression. Section 5.5 presents empirical evidence to demonstrate that this not a good approximation.

5.1. Experimental setup

To explore these claims, we require the ability to control the parameter uncertainty of Θ and the ability to sample from $q(\Theta)$. Our experiments will use common benchmark networks, whose parameters are usually given as fixed quantities, e.g., $\Theta_{B|a+} = (0.3, 0.7)$. We require parameter uncertainty in the form of Dirichlet row distribution, e.g., $\Theta_{B|a+} \sim \text{Beta}(3, 7)$. To resolve this problem we treat the network with fixed parameters as the underlying distribution, from which we produce a sample D by drawing m tuples from the network. Using the same structure as the initial network, we learn new parameters using this D. Assuming a uniform Dirichlet prior over the parameters

```
Input: Belief Net $B$ (structure + parameters)
       Set of queries $\{q_i(\Theta) = \text{P}(\mathtt{h}_i \mid \mathbf{e}_i, \Theta)\}$
       $m$ = "effective size" used to compute parameters
       $r$ = number of "trials"—each involves one specific set of parameter values
1. Generate posterior over parameters
   For $k = 1..m$
       $d^{(k)}$ := RandomInstanceFrom($B$)
   Compute posterior $\Theta|D$ from initial distribution $\Theta_{\mathtt{c}|\mathbf{f}} \sim \text{Dir}(1, 1, \ldots, 1)$ and data $D = \{d^{(k)}\}$
2. Compute replicate parameters, and query responses
   For $j = 1..r$
       Draw $\Theta^{(j)}$ from $\Theta|D$ posterior distribution
       For each query $i$:     $Q_i^j := q_i(\Theta^{(j)})$
3. Actual Analysis of $\{Q_1^j\}$, $\{Q_2^j\}$, ...
   For each query $i$, based on $\{Q_i^j\}_j$ samples...
       a. Compute and compare $a_i$ to $e_i$     (Section 5.2)
       b. Perform relevant statistical tests—Normal vs Beta
          Produce Quantile-Quantile plots     (Section 5.3)
       c. Perform coverage experiments "error bars"     (Section 5.4)
       d. Compare to binomial distribution     (Section 5.5)
```

Fig. 7. Experimental setup.

$\Theta_i \sim \text{Dir}(1, \ldots, 1)$, the data is integrated in to produce posterior parameters $\Theta|D$, as discussed above; see COMPUTEPOSTERIOR in Section 2.2.

Given such a belief net and posterior distribution, we use a Monte Carlo strategy to sample from $q(\Theta)$. We generate r replicates from Θ, denoted $\{\Theta^{(j)}\}_{j=1}^{r}$, where each replicate is formed by replacing each Dirichlet row distribution by a sample from it. That is, each replicate instantiates a belief net with fixed values for parameters Θ.

To illustrate, imagine we started with the Diamond network, whose parameters were each initially uniform—i.e., each $\Theta_i \sim \text{Beta}(1, 1)$—then used a $m = 100$ data samples to produce the posterior $\Theta|D$ distribution shown in Fig. 2, with $\Theta_{A|\{\}} \sim \text{Beta}(35, 67)$ and $\Theta_{B|+a} \sim \text{Beta}(7, 29)$. We then drew samples of $\Theta|D$ of the form

$$\begin{array}{l} \langle \Theta_{+a}^{(1)} = 0.35, \quad \Theta_{+b|+a}^{(1)} = 0.18, \quad \ldots \rangle \\ \langle \Theta_{+a}^{(2)} = 0.31, \quad \Theta_{+b|+a}^{(2)} = 0.21, \quad \ldots \rangle \\ \langle \Theta_{+a}^{(3)} = 0.33, \quad \Theta_{+b|+a}^{(3)} = 0.17, \quad \ldots \rangle \\ \langle \qquad \vdots \qquad\qquad \vdots \qquad\quad \ldots \rangle \end{array} \tag{15}$$

Notice this is *not* the same as drawing replicates of values for the domain variables $A, B, \ldots$. For each instantiation of the parameters $\Theta^{(j)}$ and fixed query $q_i(.)$, we calculate $Q_i^j = q_i(\Theta^{(j)})$ using an exact algorithm for belief net inference.

Our experiments are based on $r = 1000$ samples. This allows us to empirically evaluate the quality of the approximations implicit in Theorem 2, which deals with the distribution over parameters Θ (Eq. (15)), not over the basic tuples. We then performed various tests on these $\{Q_i^j\}_j$ values; see next subsections. Fig. 7 summarizes these steps.

We use four network structures in our experiments: Diamond, Alarm, Insurance and Hailfinder. Diamond is the four variable network illustrated in Fig. 2, that allows for a variety of inferential patterns. We considered the following queries:

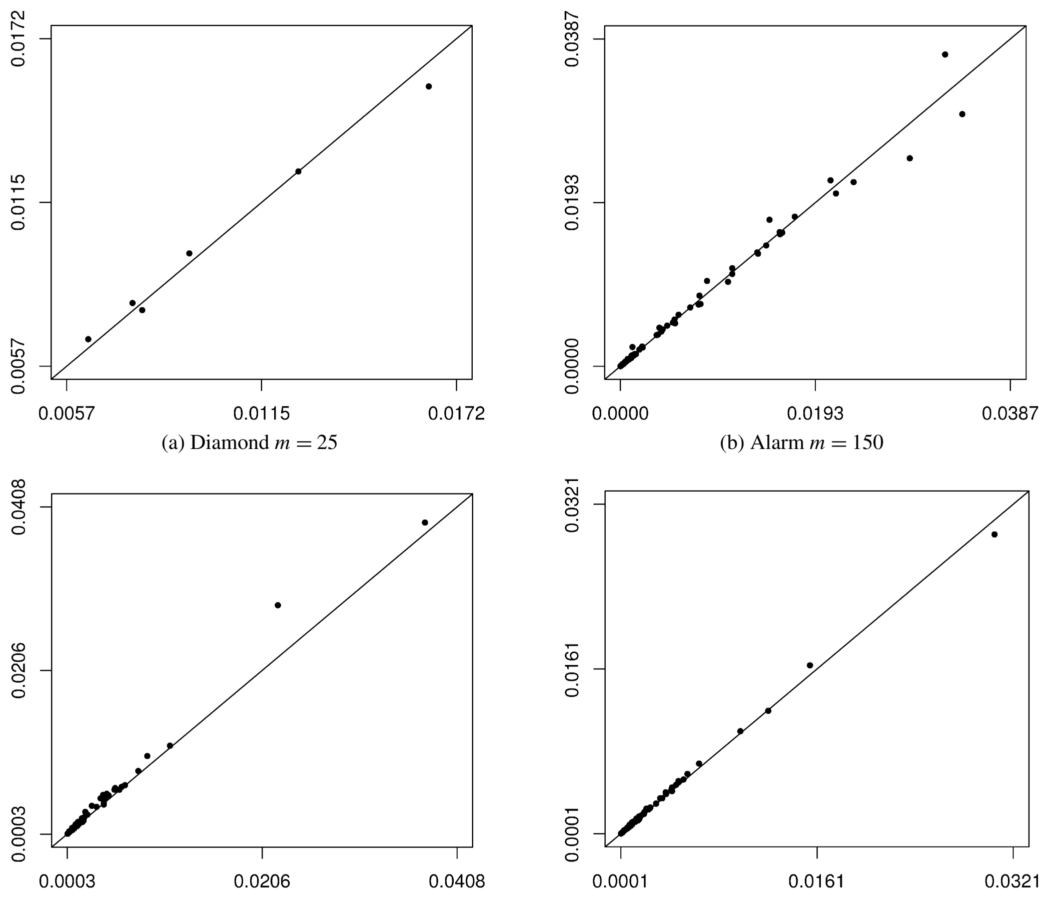
$$\begin{array}{l} Q1: \text{P}(+a) = \Theta_{+a|\{\}} \\ Q2: \text{P}(+a\,|\,+b) = \frac{\Theta_{+b|+a} \times \Theta_{+a|\{\}}}{\Theta_{+b|+a} \times \Theta_{+a|\{\}} + \Theta_{+b|-a} \times \Theta_{-a|\{\}}} \\ Q3: \text{P}(+a\,|\,+b, +c) = \frac{\Theta_{+b|+a} \times \Theta_{+c|+a} \times \Theta_{+a|\{\}}}{\Theta_{+b|+a} \times \Theta_{+c|+a} \times \Theta_{+a|\{\}} + \Theta_{+b|-a} \times \Theta_{+c|-a} \times \Theta_{-a|\{\}}} \\ Q4: \text{P}(+b, +c\,|\,+a) = \Theta_{+b|+a} \times \Theta_{+c|+a} \\ Q5: \text{P}(+a\,|\,+d) = \frac{\sum_{b,c} \Theta_{+d|B=b,C=c} \times \Theta_{B=b|+a} \times \Theta_{C=c|+a} \times \Theta_{+a|\{\}}}{\sum_{a,b,c} \Theta_{+d|B=b,C=c} \times \Theta_{B=b|A=a} \times \Theta_{C=c|A=a} \times \Theta_{A=a|\{\}}} \\ Q6: \text{P}(+d\,|\,+a) = \sum_{b,c} \Theta_{+d|B=b,C=c} \times \Theta_{B=b|+a} \times \Theta_{C=c|+a} \end{array} \tag{16}$$

The Alarm network, a 37 variable network described in [21], was designed by medical experts for monitoring intensive care patients. Alarm has become a standard for evaluating belief net learning and inference algorithms. Here and below, we then used $m = 150$ samples to produce Dirichlet parameters. We generated 100 queries by choosing a single query variable and three to five evidence assignments, using [21] to determine which variables could be query variables. (Here, and below, the value of each query and evidence variable is assigned uniformly at random.) The mean response of these queries tends to cover the [0, 1] interval well, including many queries whose mean response was near 0 or 1.

The Insurance network [6] models car insurance risk, using 27 variables. We generated 100 queries by randomly sampling one query variable and either zero, one, or two evidence variables. To ensure that the mean response of the queries covered the [0,1] interval, we used a rejection sampling procedure: We divided [0,1] into 5 ranges [0,0.2), [0.2,0.4), ..., [0.8,1.0], and generated queries until an equal number of responses fell into each bin.[9]

The Hailfinder network [1] is a 56 variable network for forecasting summer hailstorms. Again we generated 100 queries using the same rejection sampling procedure. (More details about these networks, queries and parameters, as well as more extensive experimental results, are available in [20].)

[9] This was required because we found that the naive approach of uniformly sampling queries with 1 query variable and 0–2 evidence variables produced queries whose responses were tightly clustered around 0.2–0.3.



(a) Diamond $m = 25$

(b) Alarm $m = 150$

(c) Insurance $m = 150$

(d) Hailfinder $m = 150$

Fig. 8. Each $\langle x, y\rangle$ point represents a query whose x-value is the approximate variance of a query response computed using MeanVar, a_i, and whose y-value is the sample variance of the response based on $r = 1000$ instances drawn from the true response distribution, e_i from Eq. (17). Subfigures: (a) six queries on Diamond; (b) one hundred queries on Alarm; (c) one hundred queries on Insurance; (d) one hundred queries on Hailfinder.

5.2. Accuracy of variance approximation

The simplest measure of the accuracy of $\tilde{\sigma}^2_{\mathrm{h}|\mathbf{e}}$ is how close it is to the true variance of a query response. Even though we do not know the true variance of a query response, using $r = 1000$ samples from the query response distribution, $\{Q^i\}_{i=1}^r$, provides a good estimate. This is especially true since $q(\Theta)$ is unidimensional and often unimodal. That is, over a wide range of queries q_i, we want to see how our analytic approximation a_i (i.e., the $\tilde{\sigma}^2_{\mathrm{h}|\mathbf{e}}$ from Eq. (7) in Theorem 2) compares to the large-sample estimate.[10]

$$e_i = \frac{1}{r}\sum_{j=1}^{r}\left(Q_i^j - E[Q_i]\right)^2 \tag{17}$$

As we know that our approximate variance is *asymptotically* correct as $m \to \infty$, we are most interested in seeing its behavior for relatively small values of m. Fig. 8 shows our results on the four networks. We used $m = 150$ for the three larger networks, but $m = 25$ for Diamond. (Using $m = 150$ here would reduce the variance in the query responses to nearly zero.) The fact that essentially all of the values are extremely near the "$y = x$" diagonal line shows a very tight fit over almost all of the queries. We quantify this below. While the absolute scale of the variances appears

[10] In our experiments, we use a nonparametric bootstrap to improve on the large-sample estimate.

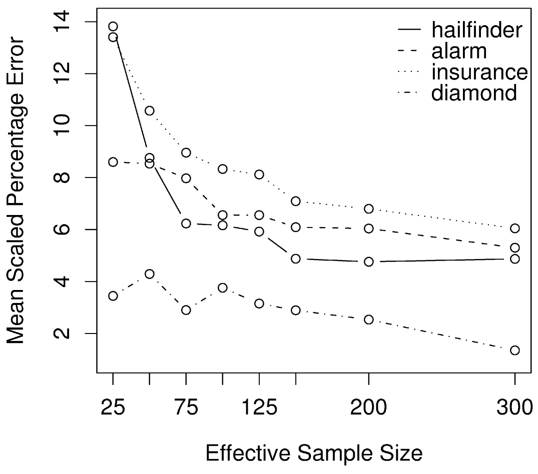


Fig. 9. Mean scaled percentage error vs. effective sample size m.

small, note that the variance is usually less than $\frac{1}{12}$, which is the variance of the uniform distribution on [0,1]. Of course, query distributions typically have much smaller variances. E.g., if a query has mean 0.4 and variance 0.01, then it will usually fall within two standard deviations of its mean—i.e., between 0.2 and 0.6.[11]

While all of the deviances (between a_i and e_i) are relatively small, many of the larger deviance value are associated with the queries with the largest variance values. This is not surprising, and follows from the fact that we are using the Delta method, which is based on a first-order Taylor expansion. We elaborate in Comment#2 after the proof of Theorem 2 (in Appendix B).

We use Mean Scaled Percentage Error

$$\text{MSPE} = \frac{100}{n} \sum_i \frac{|a_i - e_i|}{e_i}$$

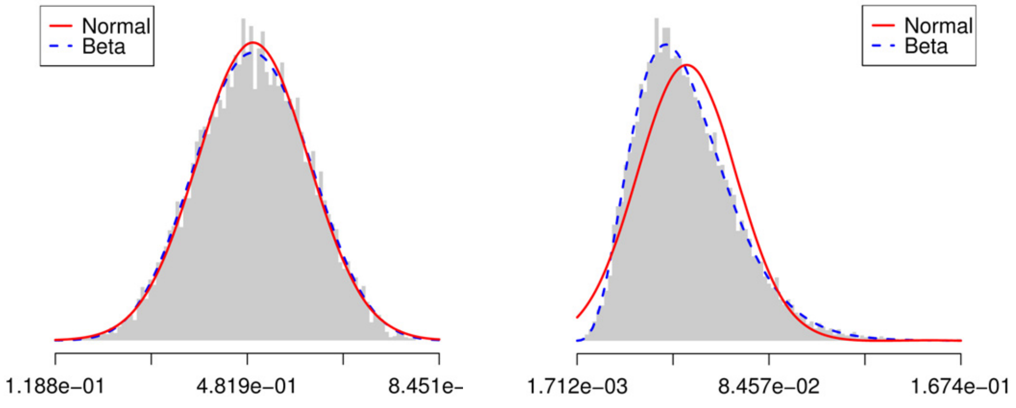
(i.e., the average percentage that a_i deviates from e_i over n queries) to evaluate the quality of our approximation. Fig. 9 shows that our MEANVAR approximations are accurate across a range of sample sizes. For example, even with only $m = 25$ samples, our worst average over any structure was under 14%, and after 200 examples, the worst was around 7%. The rate of convergence of $\tilde{\sigma}^2_{\mathtt{h}|\mathbf{e}}$ differs with each query; see Comment#1 after the proof of Theorem 2, in Appendix B.

5.3. Normal vs Beta distribution

Some tasks, such as the ones mentioned in [18] and [32], require only (an approximation of) variance; the previous section showed that our analytic approximation of variance fits the empirical evidence very closely. However, for many other tasks (such as computing error bars), we also need to know the form of the underlying distribution. While this parametric form is unknown in general [24], we can still determine whether we would get appropriate answers if we used some plausible form. Section 3.1 argued for fitting the mean and approximate variance to either a Normal or Beta distribution. Here we compare samples from the true distribution, $\{Q^i\}_{i=1}^r$, against each model.

Fig. 10 presents two histograms of these $\{Q_i^j\}_j$ values for two different queries Q_i, with expected means near 0.5 and 0.05 respectively. On each, we have superimposed the pdf (probability distribution function) for the best fit Normal and best fit Beta distributions. The left figure shows that both distributions can have good fits, when the mean is far from 0 and from 1, and the distribution is fairly symmetric. The right figure shows that even the best-fitting Normal distribution has problems when the distribution is skewed.

[11] Each point in Fig. 8 is based on a single Θ vector produced from a single data set. We also considered other Θ parameter values, by estimating new parameters from other data sets generated from the underlying distribution—i.e., re-running process in Fig. 7, starting from Step 1. We found that the quality of the variance approximation is similar.



(a) Low skew (b) High skew

Fig. 10. Two query response distributions from Alarm: Each is a histogram of the $\{Q^i\}_{i=1}^r$ instances, overlayed with the fitted Beta and Normal models. When the skew is low, the models tend to be similar, but when the skew is high, the Beta model is a much better fit.

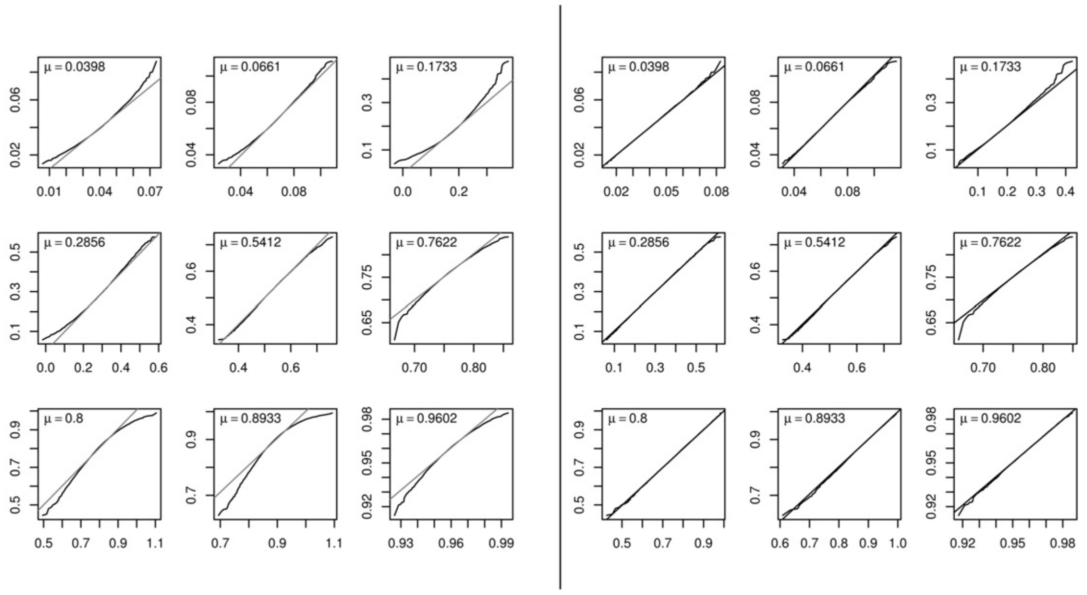


Fig. 11. Quantile–quantile plots for nine query instances (Alarm network), each comparing query sample distribution against our approximation of the $q(\Theta)$ distribution using calculated mean and variance. The left-hand side assumes $q(\Theta)$ is Gaussian; the right-hand side assumes $q(\Theta)$ is Beta distributed. The μ value is the expected query response.

A "quantile–quantile plot" provides a way to visualize this fit, by plotting sample quantiles against the theoretical quantiles of the fitted (Normal or Beta) distributions; see [4]. Here, a straight diagonal line indicates that, for each $k = 1..r$, the kth instance (of r) appears where the distribution predicts k/r of the data should be. Fig. 11 shows nine such quantile–quantile plots, for the queries numbered $\{15, 25, \ldots, 95\}$, where the numbers are based on the query's mean response.[12] We see that many of the queries are better approximated by the Beta than the Normal, especially when the true distribution is skewed (i.e., the ones with mean response μ near 0 or near 1).

One way to quantify the difference between models is to compare the (log)likelihood of the query instances $Q_i^j = q_i(\Theta^j)$ computed based on the Normal (resp., Beta) parameters $\sum_j \log \mathrm{P}_{\mathrm{Normal}(\mu_i,\sigma_i^2)}(Q_i^j)$ (resp.,

[12] These are only a representative sample of the 600+ quantile–quantile plots for Alarm. All of the plots are available in [20].

Table 1
Fail-Beta is the fraction of queries that did not fit the Beta hypothesis at 0.05 significance; Fail-Normal is the fraction of queries that did not fit the Normal hypothesis at 0.05 significance

Domain	Fail-Beta	Fail-Normal
Diamond ($m = 25$)	0/6	1/6
Alarm ($m = 300$)	16/100	50/100
Insurance ($m = 300$)	13/100	36/100
Hailfinder ($m = 300$)	10/100	31/100

$\sum_j \log \mathrm{P}_{\mathrm{Beta}(a_i,b_i)}(Q_i^j)$). For Diamond, constructed using $m = 25$ samples, the likelihood of the Beta model is higher than Normal model for each of the six queries. For Alarm ($m = 300$), the likelihood of the Beta model is higher than the Normal model for 92 of the 100 queries. For Insurance ($m = 300$), the Beta model is better for 89 of the 100 queries. For Hailfinder ($m = 300$), the Beta model is better for 89 of the 100 queries.

The fact that Beta has higher likelihood is even more impressive, given that this log-likelihood measure unnaturally favors the Normal model, especially when the response is near 0 or near 1: As $\Theta \to 0$ (or $\Theta \to 1$), the log-likelihood of the Beta distribution usually tends to $-\infty$, but this does not occur with the Normal distribution. This is because the log-likelihood for the $\mathcal{N}(\mu, \sigma^2)$ model is a constant plus a sum of terms $-(1/2) \times [(x - \mu)/\sigma]^2$ while the log-likelihood for $\mathrm{Be}(a, b)$ is a constant plus a sum of terms $(a - 1) \times \log(x) + (b - 1) \times \log(1 - x)$, which can blow up (large negative) if $a > 1$ and x is near 0, or if $b > 1$ and x is near 1. Hence, the Beta distribution is strongly affected by values near the boundary of the unit interval, while the Normal log likelihood is not.

Another approach is to use a goodness-of-fit test where, formally, the null hypothesis is that the samples $\{Q^j\}_{j=1}^r$ were drawn from the Normal model (resp., Beta model) and the alternate is the negation of the null hypothesis. Such tests do *not* allow us to state that the data was drawn from the null model, instead the focus "is on the measure of agreement of the data with the null hypothesis" [16]. Here, we are asking how close the observed distribution is to the best-fit Normal (resp., the best-fit Beta) distribution.

These goodness-of-fit procedures suggest that the Normal model is problematic. Using an Anderson–Darling normality test [2], for example, 80 of the 100 Alarm queries showed evidence of non-normality at a significance level of 0.05. Due to a lack of critical values for the Beta distribution, we resort to a lower power Kolmogorov–Smirnov test to compare model [11]. At the same 0.05 significance level, we get the results in Table 1. In all four domains, the Beta distribution tends to fit queries better than the Normal distribution.

5.4. Accuracy of Bayesian error bars

Another way to compare the Normal and Beta models is based on their respective performance on some task. If a model is accurate then functionals of that model, such as credible regions of the posterior response, should also be accurate.

As mentioned above, a $(1 - \delta)$ credible set is a region $\omega \subset \Re$ such that $\mathrm{P}(q(\Theta) \in \omega(\mathcal{D})) = (1 - \delta)$. If our model of the posterior response distribution is good, then the expected fraction of query samples Q^j in ω should be $(1 - \delta)$.

Here, for each query q, we first used MEANVAR to compute the mean and variance of the response, then used these values to compute a 90% credible region for the Normal distribution $[L_N, U_N]$, and for the Beta distribution $[L_B, U_B]$. We then drew $r = 1000$ instances from $q(\Theta)$, using the procedure described in Section 5.1, and recorded what fraction of instances appeared in $[L_N, U_N]$ (resp., $[L_B, U_B]$). If the distributional assumption is correct, we expect this ratio to be close to 0.9. The results are presented in Fig. 12.

While both approximations had problematic cases, the Beta distribution produces more accurate intervals for most queries. The problems with the Normal model are conspicuous when the mean query response is near 0 or 1, which is when $q(\Theta)$ exhibits significant skew. Repeating this experiment with an 80% credible interval produces similar results.[13]

[13] We considered yet other intervals. However, larger intervals (e.g., 95% or even 99%) are contrary to the goal of verifying whether the Normal or Beta distributions are good models of the posterior, as there are more distributions that will satisfy a given 0.99 interval by chance than there are distributions that will satisfy a 0.90 interval by chance. By contrast, a really small interval (like 50% or 10%) would reflect only the model quality immediately around $\mathrm{E}[q(\Theta)]$.

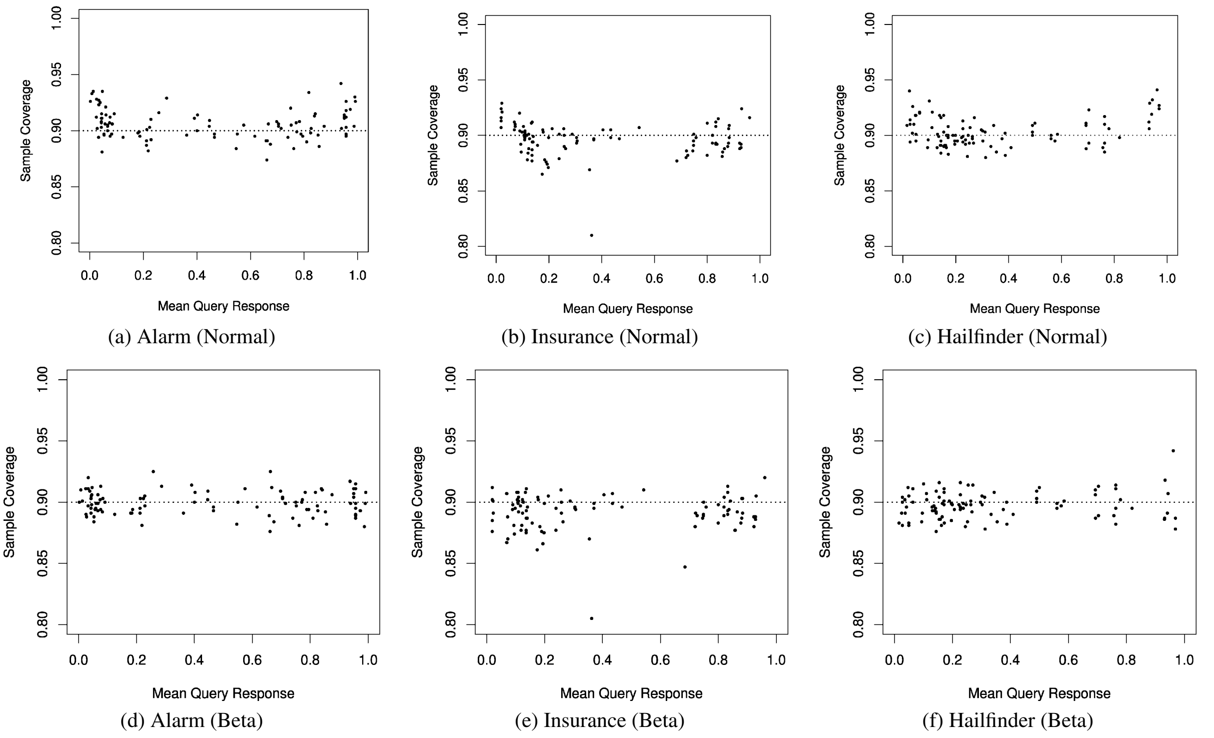
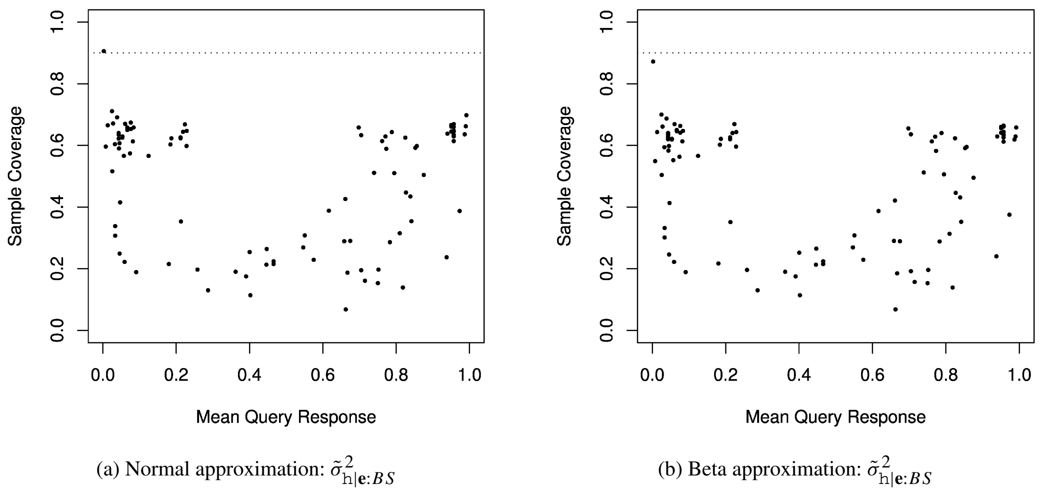


Fig. 12. Query sample coverage of computed 90% Bayesian credible intervals. (a)–(c) assumes $q(\Theta)$ is Normally distributed; (d)–(f) assumes $q(\Theta)$ is Beta distributed. Each point represents a query, sorted in order of expected response. The horizontal dotted line represents the desired result: 90% coverage.



(a) Normal approximation: $\tilde{\sigma}^2_{\mathrm{h}|\mathbf{e}:BS}$ (b) Beta approximation: $\tilde{\sigma}^2_{\mathrm{h}|\mathbf{e}:BS}$

Fig. 13. Query sample coverage of computed 90% Bayesian credible intervals. These plots were generated using the same queries and samples as Figs. 12(a) and 12(d), except $\tilde{\sigma}^2_{\mathrm{h}|\mathbf{e}:BS}$ (Eq. (18)) was used instead of $\tilde{\sigma}^2_{\mathrm{h}|\mathbf{e}}$ (Eq. (7)).

5.5. Binomial variance

Several researchers have suggested (personal communication) that the variance of the response may be much simpler, perhaps as trivial as the binomial variance $\tilde{\sigma}^2_{\mathrm{h}|\mathbf{e}:BS} = \frac{q(\hat{\Theta})\ (1-q(\hat{\Theta}))}{m}$ where m is the total number of tuples, or perhaps

$$\tilde{\sigma}^2_{\mathrm{h}|\mathbf{e}:BS} = \frac{q(\hat{\Theta})\,(1-q(\hat{\Theta}))}{m+k+1} \tag{18}$$

in the Bayesian framework when we assume uniform priors and the query variable has k values. To investigate this claim, we repeated the coverage test on Alarm, but used this $\tilde{\sigma}^2_{\mathrm{h}|\mathbf{e}:BS}$ approximation in place of $\tilde{\sigma}^2_{\mathrm{h}|\mathbf{e}}$ (Eq. (7)). Fig. 13(a) (resp., Fig. 13(b)) shows the results, assuming the distribution is Normal (resp., Beta). Clearly this approximation performs poorly in both instances. This is expected: Eq. (18) matches Eq. (7) essentially only if the query variable is not connected to any other nodes:

Observation 5. *If the query variable* H *is k-ary, is given uniform priors, and is not connected to any other nodes, then for any evidence* $\mathbf{E}=\mathbf{e}$, $\tilde{\sigma}^2_{\mathrm{h}|\mathbf{e}} = q(\hat{\Theta})\,(1-q(\hat{\Theta}))/(1+k+m)$.

In general, when there are connections, this variance will be larger, as the sum will include other positive quantities.

5.6. Timing information

Both BUCKELIM and BUCKELIM$^+$ have $O(n \cdot \exp(w))$ time complexity, where w is the induced treewidth of the ordering. While BUCKELIM$^+$ is obviously slower, the additional cost of running BUCKELIM$^+$ depends on the average size of the functions after running BUCKELIM. This is because BUCKELIM$^+$ must compute derivatives with respect to all of the functions in the buckets after BUCKELIM. For example, on the 100 Alarm queries, the average cost of our MEANVAR algorithm was 3.3 times that of BUCKELIM alone.

6. Related work

Our results provide a way to compute the variance of a belief net's response to a query, based on the posterior distribution over Θ, which is determined by the data sample. This is done using the "Delta method", which basically propagates the variance of each parameter, based on the partial derivatives.

Kleiter [26] similarly uses this method within a stochastic simulation technique to approximate the mean and variance of a query. His methodology appears to be more general than ours, allowing incomplete data, but it is also more computationally intensive. His system calculates an approximate variance for a complete query at each iteration of the simulation using an expression derived by the Delta method. While his derivation is somewhat similar to the one given in our Appendix B, we obtain a different expression for the approximate variance. We have been unable to verify whether the two expressions are equivalent. Kleiter also presents an exact result for a network with two nodes, $A \to B$. He shows that, given complete data and an appropriate prior distribution, the query $\mathrm{P}(A \mid B)$ has a Beta distribution. In recent work, Hooper [24] generalizes this result using ideas related to likelihood equivalence of the BDe metric [23]; i.e., given a BDe prior and complete data, a query has a Beta distribution if can be represented as a CPtable parameter for an equivalent DAG structure inducing the same dependence model. In addition, our paper differs from Kleiter's by also providing an effective way to compute the variance information BUCKELIM$^+$, as well as empirical evidence that our approach typically works effectively, despite its approximations.

Several other researchers also consider the *posterior distribution over CPtables*, but for different purposes. For example, Cooper and Herskovits [9] use it to compute the expected response to a query (see Eq. (3)). We extend their foundational result by computing the *variance* of the response as well. Similarly, while many BN-learning algorithms compute the posterior distribution over CPtables [22], most seek a single set of parameters that maximizes the likelihood, which again is different from our task.

Others, including [42], use this response variance for various tasks, including a bias-variance analysis for various probabilistic classifiers (similar to [18]). These earlier systems, however, will estimate this quantity *empirically*—sometimes based on replicates of the Θ parameters, but more often based on a set of training instances from the base domain D (each used to instantiate the parameters, and then to compute the response to a fixed question, etc.). We provide empirical evidence that our analytic approximations are as accurate, but much faster to compute: Most of their empirical estimates require hundreds or thousands of replicates, which means they require a computation that is hundreds or thousands of times more than just computing a single response. By contrast, we noted in Section 5.6 that our approach can require a total computation only a small factor slower than just computing the response itself. For complete queries (Appendix A), the additional cost needed to compute the variance is negligible.

Our main theoretical result (Theorem 2) uses the *sensitivity* of the query to each CPtable entry. Many other projects consider such sensitivity analyses, providing mechanisms for propagating ranges of CPtable values to produce a range in the response; cf., [8,10,12,29,30]. Those papers do not consider variance, but typically assume that parameters are intervals to be propagated throughout the network. While they require the user to explicitly specify the range of a local CPtable entry, our work uses a data sample as the source of these "intervals", in that we consider intervals based on the posterior distribution of the CPtable parameters $\{\Theta_{\mathrm{c}|\mathbf{f}}\}$, which in turn is based on the observed data. This assumes the CPtable parameters are independent Dirichlet rows, which is in accord with common models of learning belief network parameters.

While our system must propagate *all* of the "ranges", most of the other systems only propagate a single range, based on the one associated derivative. One exception is the Darwiche [12] system, which can simultaneously produce all of the derivatives. This system compiles a belief network into a polynomial representation, which can then be symbolically differentiated. There appears to be a close similarity between his approach and our (independently developed) BUCKELIM$^+$ algorithm (Section 4.2), as his polynomial can be viewed as a symbolic representation of the variable elimination process, and his differentiation proceeds from the leaves of the tree-representation of the polynomial to its root, tracing out the same pattern in space that our BUCKELIM$^+$ traces over time. However, Darwiche does not consider our error-bar application, and so does not include the additional optimizations we could incorporate.

Excluding the [12] result, none of the other projects provides an efficient way to compute those partial derivatively. Also, some of those other papers focus on properties of this derivative—e.g., when it is 0 for some specific CPtable entry. Note this information can be derived from Eq. (8). Finally, while some other results deal only with singly connected networks [10], our analysis holds for arbitrary structures.

Our analysis also connects to work on abstractions, which also involves determining how influential a CPtable entry is, with respect to a query, towards deciding whether to include a specific node or arc [17]. Their goal is typically computational efficiency in computing that response, either exactly or approximately. By contrast, our focus is in computing the error-bars around the response, independent of the time required to determine that results.

Lastly, our experiments require instances drawn from $q(\Theta)$, which can themselves be used to estimate approximate credible intervals. There are applications of sampling methods to other problems in belief nets, which can possibly be confused with our concerns. For example, in stochastic sampling inference algorithms, confidence intervals on the posterior [7] refer to the distribution induced by sampling. However, the underlying network has no parameter uncertainty in those applications.

7. Conclusion

Further extensions

Our current system has been implemented, and our data indicates that reasonably accurate error-bars can be produced. There are several possible extensions. One class of extensions involves discharging the assumptions listed in Definition 1. It may be possible to deal with situations where a single correct *structure* is not provided. Instead, we may be given a distribution over structures or be forced to learn both structure and parameters from a data sample. (However, our empirical evidence on tasks that use these approximations [18,32] suggest that Eq. (7) works fairly well even if the model structure is wrong.) We assume that the network parameters, Θ, can be decomposed into Dirichlet row distributions. It would be useful to apply a similar analysis to alternate CPtable encodings, such as Noisy-OR [34] or CP-Trees [5].

The normative process, depicted in Fig. 1, starts with completely specified training instances. It is not clear how to use an *incomplete* training sample. First, the posterior distribution over Θ is not guaranteed to be Dirichlet. Second, it is not clear how to compute quantities like effective sample size. The naïve EM algorithm for learning parameters [22] will produce an effective sample size that is too large. Note, however, that our technique is independent of the source of the Θ distributions; i.e., they apply whenever we are given a product of Dirichlet distributions, however they are obtained.

While our system takes advantage of optimizations available to Bucket Elimination algorithms (e.g., optimized variable orderings), the bucket data structures must be initialized for each query. It would be desirable to use a multiple query algorithm for inference and calculation of the partial derivatives, as this would amortize the cost of initializing tables with evidence, and would also us to reuse intermediate computations shared by different queries.

Query-DAGs [15], given its similarity to Bucket Elimination, is an obvious candidate; another option is Junction Trees [25].

Appendix A provided a very simple algorithm for computing our variance approximation $\tilde{\sigma}^2_{\mathtt{h}|\mathbf{e}}$ for the special class of "complete data" queries, and then showed that we could use this to quickly deal with any query to a Naïve Bayes structure (provided only the query node is the root). It would be useful to develop other specialized algorithms (that similarly bypass BUCKELIM$^+$) for other specific network topologies.

Contributions

Many real-world systems work by reasoning probabilistically, based on a given belief net model. When belief net parameters are based on a finite random sample, these parameters can be viewed as random variables, whose uncertainty induces uncertainty in the response to a given query. This paper addresses the challenge of computing this posterior distribution of a belief net's response to a query.

We define the task, then prove that, given standard assumptions about the parameters, this response distribution is asymptotically Normal, and provide its mean and asymptotic variance (Theorem 2), from which we can estimate the associated credible regions ("Bayesian error bars") around the expected response. We also connect this task to the well-understood problem of learning belief networks parameters from complete data, and provide an algorithm MEANVAR for computing this asymptotic variance for *any* query on *any* belief net. We prove both that MEANVAR is correct, and that it has the same asymptotic complexity as belief net inference. This procedure, in effect, propagates the uncertainty of each parameter based on its partial derivative. The subroutine used to compute all of these derivatives, BUCKELIM$^+$, is of independent interest as these derivatives have many uses outside the scope of this paper [15, 38]. We also provide a much simpler "straight-line" algorithm for efficiently computing the variance in two common situations: "complete data" queries, and Naïve Bayes inference.

Our underlying theoretical claims, that the distribution is Normal and has variance $\tilde{\sigma}^2_{\mathtt{h}|\mathbf{e}}$ (Eq. (7)), are only asymptotic; they say nothing about the properties of this distribution given only a finite sample. We therefore ran a body of empirical tests to investigate the performance. Our results show that our approximate variance $\tilde{\sigma}^2_{\mathtt{h}|\mathbf{e}}$ is extremely close to correct, given even small sample sizes. However, the associated distribution is often not close to Normal. Our experiments did show that this distribution is often well-approximated by the Beta distribution, and in particular, the associated error-bars are typically fairly accurate.

In earlier works, we have already identified two tasks that can use these variance quantities: discriminative model selection, and as a way to combine responses from different Bayesian classifiers. These existing tasks are enabled by our (relatively) efficient methods for approximating variance. We eagerly anticipate the emergence of many other applications.

Acknowledgements

We are grateful for the many comments and suggestions received from Adnan Darwiche and the three anonymous reviewers. All authors gratefully acknowledge the generous support of NSERC. T.v.A. and A.S. also acknowledge support from iCORE, A.S. acknowledges further support in part by the National Science Foundation (grant IIS-0325581), and R.G. from the Alberta Ingenuity Center for Machine Learning. Most of this work was done while T.v.A and A.S. were students at the University of Alberta.

Appendix A. "Complete query" case

This appendix considers the challenge of computing the variance of the response to a query $\mathrm{P}(\mathtt{H} = \mathtt{h} \,|\, \mathbf{E} = \mathbf{e})$ in simple case where the query is "complete", in that the evidence $\mathbf{E}$ includes every variable except the query $\mathtt{H}$. Here we show straight-line code that can compute the variance (as well as expected value) of the response.

Using the identity [12,19]

$$q^{(\mathtt{c}|\mathbf{f})}_{\mathtt{h}|\mathbf{e}}(\Theta) = \frac{\partial \mathrm{P}(h \,|\, \mathbf{e})}{\partial \Theta_{c|\mathbf{f}}} = \frac{1}{\Theta_{c|\mathbf{f}}}\big[\mathrm{P}_\Theta(c, \mathbf{f}, h \,|\, \mathbf{e}) - \mathrm{P}_\Theta(h \,|\, \mathbf{e})\,\mathrm{P}_\Theta(c, \mathbf{f} \,|\, \mathbf{e})\big]$$

(where each term $\mathrm{P}_\Theta(\cdot)$ refers to the value computed by the belief net whose parameters are instantiated by Θ), we can obtain the following derivative-free form of Eq. (8):

$$v_{\mathtt{h}|\mathbf{e}}(\mathtt{C}|\mathbf{f}) = \begin{bmatrix} \sum_{\mathtt{c}\in\mathtt{C}} \frac{1}{\mathtt{c}|\mathbf{f}}\big[\mathrm{P}(\mathtt{c},\mathbf{f},\mathtt{h}\,|\,\mathbf{e}) - \mathrm{P}(\mathtt{h}\,|\,\mathbf{e})\mathrm{P}(\mathtt{c},\mathbf{f}\,|\,\mathbf{e})\big]^2 \\ -\big[\mathrm{P}(\mathbf{f},\mathtt{h}\,|\,\mathbf{e}) - \mathrm{P}(\mathtt{h}\,|\,\mathbf{e})\,\mathrm{P}(\mathbf{f}\,|\,\mathbf{e})\big]^2 \end{bmatrix} \tag{A.1}$$

where here (and below) we use $\mathrm{P}(\cdot)$ for $\mathrm{P}_{\hat\Theta}(\cdot)$, where $\hat\Theta$ is the posterior mean. We can trivial read off that $v_{\mathtt{h}|\mathbf{e}}(\mathtt{C}|\mathbf{f})$ (Eq. (A.1)) will be 0 whenever $\mathtt{H}\perp\mathtt{C},\mathbf{F}|\mathbf{E}$.

The following theorem shows that most of these $v_{\mathtt{h}|\mathbf{e}}(\mathtt{C}|\mathbf{f})$ terms are 0, and the remaining ones are trivial to compute.

Theorem 6. *Given the "complete query"* $\mathrm{P}(\mathtt{H}=+\mathtt{h}\,|\,\mathbf{E}=\mathbf{e})$*, where* $\mathbf{E}$ *includes every non-*$\mathtt{H}$ *variable, the only non-*0 $v_{+\mathtt{h}|\mathbf{e}}(\mathtt{C}|\mathbf{F}=\mathbf{f})$ *terms are ...*

- *associated with the query variable* $\mathtt{H}$: *Assuming* $\mathtt{H}$*'s parents are* $\{M_1,\dots,M_k\}$*, and the evidence* $\mathbf{E}=\mathbf{e}$ *includes the* $+\mathbf{m}=[+m_1,\dots,+m_k]$ *assignment to these variables,*

$$v_{+\mathtt{h}|\mathbf{e}}(\mathtt{H}|+\mathbf{m}) = \mathrm{P}(+\mathtt{h}\,|\,\mathbf{e})^2\left[\sum_{\mathtt{h}\in\mathtt{H}}\frac{\mathrm{P}(\mathtt{h}\,|\,\mathbf{e})^2}{\hat\Theta_{\mathtt{h}|+\mathbf{m}}} + \frac{[1-2\mathrm{P}(+\mathtt{h}\,|\,\mathbf{e})]}{\hat\Theta_{+\mathtt{h}|+\mathbf{m}}}\right] \tag{A.2}$$

(*For every other* $-\mathbf{m}\neq+\mathbf{m}$ *assignment to the parents,* $v_{+\mathtt{h}|\mathbf{e}}(\mathtt{H}|-\mathbf{m})=0$.)

- *associated with* $\mathtt{H}$*'s children*: *For each child* $\mathtt{S}$ *of* $\mathtt{H}$*, with parent set* $\{\mathtt{H},\mathtt{U}^1,\dots,\mathtt{U}^r\}$*, where the evidence* $\mathbf{E}=\mathbf{e}$ *includes* $+\mathbf{u}=[+\mathtt{u}_1,\dots,+\mathtt{u}_r]$ *and* $\mathtt{S}=+\mathtt{s}$,

$$v_{+\mathtt{h}|\mathbf{e}}(\mathtt{S}|+\mathtt{h},+\mathbf{u}) = \mathrm{P}(+\mathtt{h}\,|\,\mathbf{e})^2\big[1-\mathrm{P}(+\mathtt{h}\,|\,\mathbf{e})\big]^2\left[\frac{1}{\hat\Theta_{+\mathtt{s}|+\mathtt{h},+\mathbf{u}}} - 1\right] \tag{A.3}$$

and for each $\mathtt{h}'\neq+\mathtt{h}$

$$v_{+\mathtt{h}|\mathbf{e}}(\mathtt{S}|\mathtt{h}',+\mathbf{u}) \;=\; \mathrm{P}(+\mathtt{h}\,|\,\mathbf{e})^2\mathrm{P}(\mathtt{h}'\,|\,\mathbf{e})^2\left[\frac{1}{\hat\Theta_{+\mathtt{s}|\mathtt{h}',+\mathbf{u}}} - 1\right] \tag{A.4}$$

and $v_{+\mathtt{h}|\mathbf{e}}(\mathtt{S}|\mathtt{h},-\mathbf{u})=0$ *for every other parental assignment* $[\mathtt{h},-\mathbf{u}]$ *where* $-\mathbf{u}\neq+\mathbf{u}$.

Connecting this with the general graph in Fig. A.1, we need only deal with the CPtable rows associated with $\mathtt{H}$ and its children, the S_i's. Notice the only values that could influence the variance correspond to these variables and their parents, which is exactly the Markov blanket around $\mathtt{H}$ (which are the double-circled, but not triple-circled, nodes in Fig. A.1). This makes sense, as these are the only values that influence the response itself. Note this means Theorem 6 does not require that the evidence $\mathbf{E}$ include every non-query variable; for this theorem to apply, we need only require that $\mathbf{E}$ include all the variables in the Markov blanket around $\mathtt{H}$.

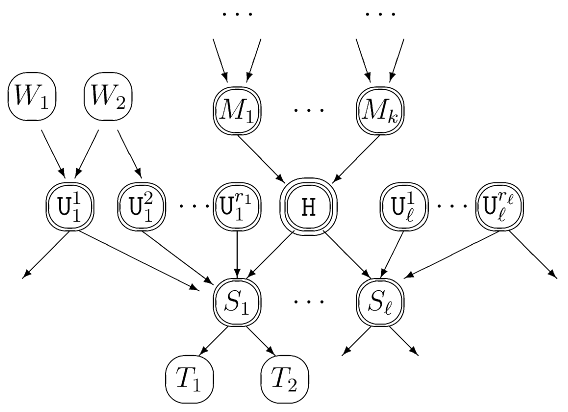


Fig. A.1. Example of a Belief Net structure, for computing $\mathrm{P}(\mathtt{H}=\mathtt{h}\,|\,\mathbf{E}=\mathbf{e})$—used to illustrate Theorem 6.

Here, if H involves $|\mathtt{H}| = r$ classes, and it has ℓ children, then we can compute the variance by adding up only $1 + \ell \times r$ quantities. Moreover, each of these computations is easy, as each $\hat{\Theta}_{\cdot|\cdot}$ is simply a quick look-up, and the $\mathrm{P}(\mathtt{h} \,|\, \mathbf{e})$ terms are trivial given that the evidence $\mathbf{E}$ includes the Markov blanket around H. (Moreover, this quantity has probably already been computed, as it was needed to compute the response—note the basic inference algorithm typically has already computed $\mathrm{P}(\mathtt{h}_i, \mathbf{e})$ for each $\mathtt{h}_i \in \mathtt{H}$.)

We close this section with two further simplifications. First, when dealing with binary classes $\mathtt{H} = \{+\mathtt{h}, -\mathtt{h}\}$, Eq. (A.2) reduces to

$$v_{+\mathtt{h}|\mathbf{e}}(\mathtt{H}| + \mathbf{m}) = \mathrm{P}(+\mathtt{h} \,|\, \mathbf{e})^2\, \mathrm{P}(-\mathtt{h} \,|\, \mathbf{e})^2 \left[\frac{1}{\hat{\Theta}_{+\mathtt{h}|+\mathbf{m}}} + \frac{1}{\hat{\Theta}_{-\mathtt{h}|+\mathbf{m}}} \right]$$

and both Eqs. (A.3) and (A.4) reduce to

$$v_{+\mathtt{h}|\mathbf{e}}(\mathtt{S}|\mathtt{h}, +\mathbf{u}) \;=\; \mathrm{P}(+\mathtt{h} \,|\, \mathbf{e})^2\, \mathrm{P}(-\mathtt{h} \,|\, \mathbf{e})^2 \left[\frac{1}{\hat{\Theta}_{+\mathtt{s}|\mathtt{h},+\mathbf{u}}} - 1 \right]$$

Hence, using the notation defined above, the total variance here is simply

$$\tilde{\sigma}^2_{+\mathtt{h}|\mathbf{e}}(\Theta) \;=\; \mathrm{P}(+\mathtt{h} \,|\, \mathbf{e})^2\, \mathrm{P}(-\mathtt{h} \,|\, \mathbf{e})^2 \sum_{\mathtt{h}} \left[\frac{1}{1 + m_{\mathtt{H}|+\mathbf{f}}} \frac{1}{\hat{\Theta}_{\mathtt{h}|+\mathbf{f}}} + \sum_{\mathtt{S}} \frac{1}{1 + m_{\mathtt{S}|\mathtt{h},+\mathbf{u}}} \left(\frac{1}{\hat{\Theta}_{+\mathtt{s}|\mathtt{h},+\mathbf{u}}} - 1 \right) \right]$$

Second, recall from above that we can ignore "barren nodes"—aka "uninstantiated descendants". For example, imagine that the evidence $\mathbf{E} = \mathbf{e}$ does not include a value for one of the query variable's children nor does not it include values for any of that child's descendants—e.g., in Fig. A.1, imagine the evidence $\mathbf{E}$ does not include values for $\mathtt{S}_1$ nor T_1 nor T_2. We can then ignore this branch. (This follows from the observation that the values of these variables do not contribute to the response [31] and so their associated parameters will also be irrelevant to the variance.)

Note this means we can *always* quickly compute this variance approximation for a Naïve Bayes structure, which will require computing only $1 + \ell' \times r$ quantities, where $\ell' \leqslant \ell$ is the number of H's children that are instantiated.

Appendix B. Proofs

Proof of Theorem 2. The following is a more detailed version of the proof in [40], employing the Delta method. Consider the first-order Taylor expansion

$$q(\Theta) = q(\hat{\Theta}) + L + R$$

where the linear term is

$$L = [\Theta - \hat{\Theta}]^T \cdot q'(\hat{\Theta})$$

and the remainder term is

$$R = \frac{1}{2} \, [\Theta - \hat{\Theta}]^T \cdot q''(\tilde{\Theta}) \cdot [\Theta - \hat{\Theta}]$$

with $\tilde{\Theta} = \hat{\Theta} + a(\Theta - \hat{\Theta})$ for some $a \in (0, 1)$. Here $q'(\hat{\Theta})$ is the vector of first partial derivatives evaluated at $\hat{\Theta}$ (which is composed of the subvectors, $q^{(\mathtt{C}|\mathbf{f})}_{\mathtt{h}|\mathbf{e}}(\hat{\Theta})$), $q''(\hat{\Theta})$ is the matrix of second partial derivatives evaluated at $\hat{\Theta}$, and the superscript ^{T} denotes transposition. Under our asymptotic framework, the variances of the components $\Theta_{c|\mathbf{f}}$ of Θ are of order $\frac{1}{m_{\mathtt{C}|\mathbf{f}}} \to 0$, so Θ converges in probability to $\hat{\Theta}$. Since the components $\hat{\Theta}_{c|\mathbf{f}}$ of $\hat{\Theta}$ are strictly between 0 and 1, it follows that components of the matrix $q''(\Theta)$ remain uniformly bounded for all Θ within an open neighborhood of $\hat{\Theta}$. Consequently, the remainder term R converges to zero at a rate faster than the linear term L, so R is asymptotically negligible compared with L. We approximate $q(\Theta) - q(\hat{\Theta})$ by L, and we define $\tilde{\sigma}^2_{\mathtt{h}|\mathbf{e}}$ to be the variance of the L term.

Now $(q(\Theta) - q(\hat{\Theta}))/\tilde{\sigma}_{\mathtt{h}|\mathbf{e}}$ and $L/\tilde{\sigma}_{\mathtt{h}|\mathbf{e}}$ have the same asymptotic distribution. We claim that this distribution is the standard Normal distribution. First note that $L/\tilde{\sigma}_{\mathtt{h}|\mathbf{e}}$ has mean 0 and variance 1. Asymptotic normality follows from the fact that CPtable rows $\Theta^T_{C|\mathbf{f}}$ are independent and each row (after suitable standardization) is asymptotically multivariate Normal. More precisely, $\sqrt{m_{\mathtt{C}|\mathbf{f}} + 1}\,(\Theta_{C|\mathbf{f}} - \hat{\Theta}_{C|\mathbf{f}})$ converges in distribution to a multivariate Normal distribution with mean vector $(0, \ldots, 0)$ and covariance matrix $\mathrm{Diag}(\hat{\Theta}_{C|\mathbf{f}}) - \hat{\Theta}_{C|\mathbf{f}}\, \hat{\Theta}^T_{C|\mathbf{f}}$, where $\mathrm{Diag}(\hat{\Theta}_{C|\mathbf{f}})$ is a diagonal matrix with the components of $\hat{\Theta}_{C|\mathbf{f}}$ along the diagonal; see [3]. We can thus express $L/\tilde{\sigma}_{\mathrm{h}|\mathbf{e}}$ as a linear combination of independent, asymptotically Normal terms. In this expression, we can standardize each of the independent random variables to have unit variance. We thus write $L/\tilde{\sigma}_{\mathrm{h}|\mathbf{e}} = \sum_r a_r Z_r$, where r indexes all CPtable rows in the network, the Z_r are independent random variables with mean 0 and variance 1, and each Z_r is asymptotically normal. The coefficients a_r do not necessarily remain fixed under our asymptotic framework, but the coefficients must satisfy the constraint $\sum a_r^2 = 1$ since $L/\tilde{\sigma}_{\mathrm{h}|\mathbf{e}}$ has variance one. It then follows that $L/\tilde{\sigma}_{\mathrm{h}|\mathbf{e}}$ is asymptotically standard normal.

The proof is completed by deriving an expression for $\tilde{\sigma}^2_{\mathrm{h}|\mathbf{e}}$, which we defined as the variance of the linear term L. Since CPtable rows are independent, this variance can be decomposed as

$$\tilde{\sigma}^2_{\mathrm{h}|\mathbf{e}} = \sum_{\mathrm{C},\mathbf{f}} q_{\mathrm{h}|\mathbf{e}}^{(\mathrm{C}|\mathbf{f})}(\hat{\Theta})^T \cdot \mathrm{Cov}(\Theta_{\mathrm{C}|\mathbf{f}}) \cdot q_{\mathrm{h}|\mathbf{e}}^{(\mathrm{C}|\mathbf{f})}(\hat{\Theta}) \tag{B.1}$$

where the summation ranges over all CPtable rows and the covariance matrix for $\Theta_{\mathrm{C}|\mathbf{f}}$ is

$$\mathrm{Cov}(\Theta_{\mathrm{C}|\mathbf{f}}) = \frac{1}{m_{\mathrm{C}|\mathbf{f}}+1}\left(\mathrm{Diag}(\hat{\Theta}_{\mathrm{C}|\mathbf{f}}) - \hat{\Theta}_{\mathrm{C}|\mathbf{f}}\hat{\Theta}_{\mathrm{C}|\mathbf{f}}^T\right)$$

It is then straightforward to obtain Eqs. (7) and (8). □

Comment#1: Assumptions underlying Theorem 2. The asymptotic framework in Theorem 2 involves only the posterior distribution, but not its derivation from sample data. The following argument shows that the assumptions are supported by large-sample theory, *provided* the Dirichlet prior distribution is appropriate. Suppose the CPtable parameters are in fact generated by the assumed prior. It then follows, with probability one, that each $\Theta_{\mathrm{c}|\mathbf{f}}$ is strictly between zero and one. Now consider the behavior of the posterior distribution as the number of complete training cases becomes arbitrarily large. In the frequentist perspective, conditioning on the parameters and not the sample data, the effective sample sizes $m_{\mathrm{C}|\mathbf{f}}$ and posterior means $\hat{\Theta}_{\mathrm{c}|\mathbf{f}}$ are random variables, each $m_{\mathrm{C}|\mathbf{f}}$ becomes arbitrarily large (with probability one), and each $\hat{\Theta}_{\mathrm{c}|\mathbf{f}}$ converges in probability to $\Theta_{\mathrm{c}|\mathbf{f}}$. The asymptotic framework in Theorem 2 is similar to this large-sample framework, differing primarily in the assumption (appropriate in a Bayesian context) that the posterior means are fixed.

The text in Section 3 already argued that one can deal with *deterministic links*, within an asymptotic framework. (Here, we basically ignore these CPtable rows.) What happens if instead a Dirichlet prior is incorrectly adopted here? Large-sample theory shows that $\hat{\Theta}$ converges toward the boundary. The variance approximation (Eq. (7)) may still be valid, but this is not guaranteed since higher-order terms in the expansion could be non-negligible. It is also possible that Eq. (7) is valid but asymptotic normality fails. For a simple example of this last scenario, suppose the query is a single belief net parameter that is degenerate: $q(\Theta) = \Theta_{\mathrm{c}|\mathbf{f}} = 0$. Suppose the prior distribution for $\Theta_{\mathrm{c}|\mathbf{f}}$ is $\mathrm{Be}(a,b)$. (Use $a = b = 1$ for a flat prior.) There is zero probability that $(\mathrm{C},\mathbf{F}) = (\mathrm{c},\mathbf{f})$ so the posterior distribution will be $\mathrm{Be}(a, b+m)$ where m is the number of samples with $\mathbf{F} = \mathbf{f}$. The posterior variance is given by Eq. (7), an exact result here and not an approximation. A simple calculation shows that, as $m \to \infty$, the posterior distribution of $q(\Theta)/\tilde{\sigma}_{\mathrm{h}|\mathbf{e}}$ is the Gamma distribution with shape parameter a, mean $\sqrt{a}$, and variance 1. If $a = 1$, then this Gamma distribution is the exponential distribution.

Comment#2: Why $|\tilde{\sigma}^2 - \sigma^2|$ can be large when σ^2 is large. Consider two queries $q_1(\Theta)$ and $q_2(\Theta)$ with corresponding variances $\sigma_1^2 < \sigma_2^2$. The larger variance associated with q_2 may arise from a combination of two factors. First, the components of Θ relevant to q_2 may be more variable than components relevant to q_1—i.e., there may be some Θ_i terms where $\frac{\partial q_2(\Theta)}{\partial \Theta_i}$ is large, which have large (co)variances. The linear approximation describes local behavior of the function (here $q_i(\cdot)$) near its mean, so increased variation may produce a larger remainder term. Second, the first derivatives of q_2 may tend to be larger than those of q_1. In practice, larger linear terms are often associated with larger quadratic terms; e.g., statistical variable selection methods for polynomial models will seldom delete a linear term while retaining a corresponding quadratic term [35]. Larger quadratic terms produce a larger remainder term. Both arguments support the claim that approximations to variance produced by the first-order Delta method can be less accurate when the variance is larger. (We note, again, that in practice, even the largest of these errors is still quite small.)

Proof of Theorem 3. The proof is by induction on the bucket ordering. The basis is the first bucket $b_\emptyset$; it is trivial to compute the derivative function of its output $g_\emptyset$ here, as it is the constant 1. To prove the inductive step, assume we have computed the derivatives of all of functions in the first J buckets (i.e., we know $\frac{\partial y}{\partial f_{i,k}}$ for all $i \in \{\emptyset, 1, \ldots, J-1\}$ and all k) and that $Scheme(\frac{\partial y}{\partial f_{i,k}}) = Scheme(f_{i,k})$, and now we now want to compute the derivatives for functions $f_{J,k}$ in bucket b_J. Recall that BUCKELIM used these values in forming $f_{I,1} = Elim(X_J, Join(f_{J,1}, \ldots, f_{J,k}))$ which appears in the earlier b_I bucket.

$$\begin{array}{lll}
\Longleftarrow & \text{Ordering for } \textbf{Step A} \ (\equiv \text{BUCKELIM}) & \Longleftarrow \\
\Longrightarrow & \text{Ordering for } \textbf{Step B} & \Longrightarrow
\end{array}$$

$$\begin{array}{ccccccc}
b_\emptyset & \ldots & b_I & \ldots & b_J & & \ldots \\
\hline
 & & f_{I,1}(X_I, \ldots) & \ldots & f_{J,1}(X_J, \ldots) & & \\
 & & & & f_{J,2}(X_J, \ldots) & & \\
 & & & & \vdots & & \\
 & & & & f_{J,k}(X_J, \ldots) & &
\end{array} \tag{B.2}$$

To simplify notation, we will simply write $f(\mathbf{x})$ for $f(\mathbf{x}|_{Scheme(f)})$.

Now observe

$$\frac{\partial y}{\partial f_{J,\ell}} = Elim\left(\mathbf{Z}, Join\left(\frac{\partial y}{\partial f_{I,1}}, \frac{\partial f_{I,1}}{\partial f_{J,\ell}}\right)\right) \tag{B.3}$$

where $\mathbf{Z}$ are the variables that are not in $f_{J,\ell}$; see Eq. (B.5) below. (The *Join* here is just a trivial application of the chain rule—also known as "backpropagation" [37].) As $I < J$, by the inductive assumption we can assume that we have already computed the first term $\frac{\partial y}{\partial f_{I,1}}$, and that its arguments are $Scheme(\frac{\partial y}{\partial f_{I,1}}) = Scheme(f_{I,1})$. Moreover, as $f_{I,1}(\mathbf{x}) = Elim(X_J, Join(f_{J,1}, \cdots, f_{J,k})) = \sum_{x_J} \prod_k f_{J,k}(\mathbf{x})$, the second term within the *Join* of Eq. (B.3),

$$\frac{\partial f_{I,1}}{\partial f_{J,\ell}}(\mathbf{x_{1..I}}, x_J) = Join\big(\{f_{J,k} \mid k \neq \ell\}\big) = \prod_{k \neq \ell} f_{J,k}(\mathbf{x_{1..I}}, x_J) \tag{B.4}$$

is just the product of the *other* functions in the Jth bucket [41], where $\mathbf{x_{1..I}}$ corresponds to the variables within $f_{I,1}$.

The result of this computation, $\frac{\partial f_{I,1}}{\partial f_{J,\ell}}$, will involve the variables in $f_{I,1}$ as well as X_J. It is joined with $\frac{\partial y}{\partial f_{I,1}}$ (Eq. (B.3)), to produce a function with arguments in $Scheme(f_{I,1}) \cup \{X_J\}$. As $\frac{\partial y}{\partial f_{J,\ell}}$ should depend on only $Scheme(f_{J,\ell})$, the only remaining step is to marginalize out the extra variables

$$\mathbf{Z} = Scheme(f_{I,1}) \cup \{X_J\} - Scheme(f_{J,\ell}); \tag{B.5}$$

this is done in second part of Eq. (14). This also fulfills the second part of the inductive step, as it insures that $Scheme(\frac{\partial y}{\partial J,\ell}) = Scheme(f_{J,\ell})$. □

Proof of Theorem 4. BUCKELIM$^+$ needs to consider at most $2n$ functions, as Step A (which is BUCKELIM) starts with at most one function (CPtable) for each of the n variables, then adds at most one new function each time a variable is eliminated.

We therefore need only show that the cost of computing the derivative of each of these functions is $O(r^w)$, where r is the largest domain of any of the variables, and w is the tree-width. We will consider the general case, using the notation from the Eq. (B.2), where $f_{I,1} = Elim(X_J, Join(b_{X_J}))$.

Combining Eqs. (B.3) and (B.4), we obtain

$$\frac{\partial y}{\partial f_{J,\ell}} = Elim\left(\mathbf{Z}, Join\left(\frac{\partial y}{\partial f_{I,1}}, \ \{f_{J,k} \mid k \neq \ell\}\right)\right) \tag{B.6}$$

Now consider the variables involved in the join

$$Scheme\left(\frac{\partial y}{\partial f_{I,1}}\right) = Scheme(f_{I,1})$$
$$Scheme(f_{J,k}) \subset Scheme(f_{I,1}) \cup \{X_J\} \quad \forall k$$

which means this entire computation will involve no more than $1 + |\mathit{Scheme}(f_{I,1})|$ variables, which is at most $1 + w$, as $|\mathit{Scheme}(f_{I,1})|$ is at most the tree-width w.

Therefore joining the required tables in Eq. (B.6) requires $O(r^{w+1}) = O(r^w)$ time and space (as r is a constant). Of course, after the join computation is done, the algorithm will then marginalize out the variables in $\mathbf{Z}$ (Eq. (B.5)) to produce a function (for the derivative) that has, at most, w variables. The cost of marginalization is at most the cost of the join, and so the overall complexity of computing a single derivative is $O(r^w)$. □

Proof of Observation 5. As noted above, we need only consider the CPtables of nodes that are not d-separated from the query node H, which means we need consider only this single H node. The variance $\tilde{\sigma}^2_{\mathtt{h}|\mathbf{e}}$ is therefore the variance associated with this single Dirichlet-distributed variable, which is $\sigma^2_{\mathtt{H}} = P(\mathtt{h})\,(1 - P(\mathtt{h}))/(1 + m_{\mathtt{H}})$. (Recall $P(\mathtt{h}) = q(\hat{\Theta})$.) As H has no parents, each of the n training instances contributes to one of its Dirichlet parameters. As H is k-ary and we start with uniform priors, this means $m_{\mathtt{H}} = k + n$. (Note we could also derive this value using Eqs. (A.2) and (A.1).) □

Proof of Theorem 6. As we sweep over the $v_{\mathtt{h}|\mathbf{e}}(\mathtt{C}|\mathbf{f})$ terms, we need to consider three cases, depending on the query variable H: (i) H is C, (ii) H appears in $\mathbf{F}$, or (iii) neither.

Case (iii): If $\mathtt{H} \neq \mathtt{C}$ and $\mathtt{H} \notin \mathbf{F}$, then $v_{\mathtt{h}|\mathbf{e}}(\mathtt{C}|\mathbf{F} = \mathbf{f}) = 0$.[14]

Here, as the evidence variables $\mathbf{E}$ includes every variable except H, it includes both C and all of $\mathbf{F}$. To simplify notation, write $\mathbf{E} = \{\ldots, +\mathtt{c}, +\mathbf{f}, \ldots\}$ to indicate the specific assignments used, where $+\mathbf{f} = [+f_1, \ldots, +f_j]$ refers to $\mathbf{E}$'s assignment to all of C's parents.

Notice $v_{\mathtt{h}|\mathbf{e}}(\mathtt{C}|-\mathbf{f}) = 0$ whenever $-\mathbf{f} \neq +\mathbf{f}$:

$$P(\mathtt{c}, -\mathbf{f}, \mathtt{h}\,|\,\mathbf{e}) = P\big(\mathtt{c}, \boxed{\mathbf{F} = -\mathbf{f}}, \mathtt{h}\,|\,\ldots, \boxed{\mathbf{F} = +\mathbf{f}}, \ldots\big) = 0$$

Similarly $P(\mathtt{c}, -\mathbf{f}\,|\,\mathbf{e}) = 0$, $P(-\mathbf{f}, \mathtt{h}\,|\,\mathbf{e}) = 0$ and $P(-\mathbf{f}\,|\,\mathbf{e}) = 0$.

We therefore consider only $\mathbf{F} = +\mathbf{f}$. Here $P(+\mathbf{f}, \mathtt{h}\,|\,\mathbf{e}) = P(+\mathbf{f}, \mathtt{h}\,|\,\ldots, +\mathbf{f}, \ldots) = P(\mathtt{h}\,|\,\mathbf{e})$ and $P(+\mathbf{f}\,|\,\mathbf{e}) = 1$, which we use to reduce the bottom line of Eq. (A.1):

$$\big[P(+\mathbf{f}, \mathtt{h}\,|\,\mathbf{e}) - P(\mathtt{h}\,|\,\mathbf{e})\,P(+\mathbf{f}\,|\,\mathbf{e})\big]^2 = \big[P(\mathtt{h}\,|\,\mathbf{e}) \;-\; P(\mathtt{h}\,|\,\mathbf{e}) \times 1\big]^2 = 0$$

Now consider the top part of Eq. (A.1), and observe we need only consider the value of $+\mathtt{c}$ value of C within the summation. (For each $-\mathtt{c} \neq +\mathtt{c}$, $P(-\mathtt{c}, +\mathbf{f}, \mathtt{h}\,|\,\mathbf{e}) = P(-\mathtt{c}, +\mathbf{f}, \mathtt{h}\,|\,\ldots, +\mathtt{c}, \ldots) = 0$.) As above, we have that $P(\mathtt{c}, +\mathbf{f}, \mathtt{h}\,|\,\mathbf{e}) = P(\mathtt{c}, \mathtt{h}\,|\,\mathbf{e})$ and $P(\mathtt{c}, +\mathbf{f}\,|\,\mathbf{e}) = P(\mathtt{c}\,|\,\mathbf{e})$. Hence,

$$\begin{aligned}
&\sum_{\mathtt{c}\in\mathtt{C}} \frac{1}{\hat{\Theta}_{\mathtt{c}|+\mathbf{f}}}\big[P(\mathtt{c}, +\mathbf{f}, \mathtt{h}\,|\,\mathbf{e}) - P(\mathtt{h}\,|\,\mathbf{e})\,P(\mathtt{c}, +\mathbf{f}\,|\,\mathbf{e})\big]^2 \\
&\quad = \frac{1}{\hat{\Theta}_{+\mathtt{c}|+\mathbf{f}}}\big[P(+\mathtt{c}, +\mathbf{f}, \mathtt{h}\,|\,\mathbf{e}) - P(\mathtt{h}\,|\,\mathbf{e})\,P(+\mathtt{c}, +\mathbf{f}\,|\,\mathbf{e})\big]^2 \\
&\quad = \frac{1}{\hat{\Theta}_{+\mathtt{c}|+\mathbf{f}}}\big[P(\mathtt{h}\,|\,\mathbf{e}) - P(\mathtt{h}\,|\,\mathbf{e}) \times 1\big]^2 = 0
\end{aligned}$$

Case (i): If $\mathtt{H} = \mathtt{C}$, where H's parents are $\mathbf{M} = \{M_i\}$, then

$$v_{+\mathtt{h}|\mathbf{e}}(\mathtt{H}|+\mathbf{m}) = P(+\mathtt{h}\,|\,\mathbf{e})^2\left[\sum_{\mathtt{h}\in\mathtt{H}} \frac{P(\mathtt{h}\,|\,\mathbf{e})^2}{\hat{\Theta}_{\mathtt{h}|+\mathbf{m}}} + \frac{[1 - 2P(+\mathtt{h}\,|\,\mathbf{e})]}{\hat{\Theta}_{+\mathtt{h}|+\mathbf{m}}}\right] \quad \text{and} \quad v_{+\mathtt{h}|\mathbf{e}}(\mathtt{H}|-\mathbf{m}) = 0$$

for $-\mathbf{m} \neq +\mathbf{m}$.

As argued above, this $v_{+\mathtt{h}|\mathbf{e}}(\mathtt{H}|\mathbf{m})$ value is 0 when $\mathbf{M} = -\mathbf{m} \neq +\mathbf{m}$ (the value used within $\mathbf{e}$). When $\mathbf{M} = +\mathbf{m}$, we can again omit the $\mathbf{M} = \mathbf{m}$ from the various equations; again, this means the second line of Eq. (A.1) is 0. Hence, as the value of the query variable $\mathtt{h} = +\mathtt{c}$,

[14] Note this follows from the observation that this condition means $\mathtt{H}\perp\mathtt{C}, \mathbf{F}|\mathbf{E}$. We include this case as it provides useful analysis and notation.

$$
\begin{aligned}
v_{+\mathrm{h}|\mathbf{e}}(\mathrm{H}|+\mathbf{m}) &= \sum_{\mathrm{h}\in\mathrm{H}} \frac{1}{\hat{\Theta}_{\mathrm{h}|+\mathbf{m}}} \big[\mathrm{P}(\mathrm{h},+\mathbf{m},+\mathrm{h}\,|\,\mathbf{e}) - \mathrm{P}(+\mathrm{h}\,|\,\mathbf{e})\,\mathrm{P}(\mathrm{h},+\mathbf{m}\,|\,\mathbf{e})\big]^2 \\
&= \frac{1}{\hat{\Theta}_{+\mathrm{h}|+\mathbf{m}}} \big[\mathrm{P}(+\mathrm{h}\,|\,\mathbf{e}) - \mathrm{P}(+\mathrm{h}\,|\,\mathbf{e})\,\mathrm{P}(+\mathrm{h}\,|\,\mathbf{e})\big]^2 \\
&\quad + \sum_{\mathrm{h}\neq+\mathrm{h}} \frac{1}{\hat{\Theta}_{\mathrm{h}|+\mathbf{m}}} \big[\mathrm{P}(\mathrm{h},+\mathrm{h}\,|\,\mathbf{e}) - \mathrm{P}(+\mathrm{h}\,|\,\mathbf{e})\,\mathrm{P}(\mathrm{h}\,|\,\mathbf{e})\big]^2 \\
&= \frac{1}{\hat{\Theta}_{+\mathrm{h}|+\mathbf{m}}} \big[\mathrm{P}(+\mathrm{h}\,|\,\mathbf{e})[1-\mathrm{P}(+\mathrm{h}\,|\,\mathbf{e})]\big]^2 + \sum_{\mathrm{h}\neq+\mathrm{h}} \frac{1}{\hat{\Theta}_{\mathrm{h}|+\mathbf{m}}} \big[\mathrm{P}(+\mathrm{h}\,|\,\mathbf{e})\,\mathrm{P}(\mathrm{h}\,|\,\mathbf{e})\big]^2 \\
&= \mathrm{P}(+\mathrm{h}\,|\,\mathbf{e})^2 \left[\frac{[1-\mathrm{P}(+\mathrm{h}\,|\,\mathbf{e})]^2}{\hat{\Theta}_{+\mathrm{h}|+\mathbf{m}}} + \sum_{\mathrm{h}\neq+\mathrm{h}} \frac{\mathrm{P}(\mathrm{h}\,|\,\mathbf{e})^2}{\hat{\Theta}_{\mathrm{h}|+\mathbf{m}}}\right] \\
&= \mathrm{P}(+\mathrm{h}\,|\,\mathbf{e})^2 \left[\sum_{\mathrm{h}\in\mathrm{H}} \frac{\mathrm{P}(\mathrm{h}\,|\,\mathbf{e})^2}{\hat{\Theta}_{\mathrm{h}|+\mathbf{m}}} + \frac{[1-2\mathrm{P}(+\mathrm{h}\,|\,\mathbf{e})]}{\hat{\Theta}_{+\mathrm{h}|+\mathbf{m}}}\right]
\end{aligned}
$$

Case (ii): If $\mathrm{H} \in \mathbf{F}$, then write C's parents as $\mathbf{F} = \{\mathrm{H}\} \cup \mathbf{U}$, and assume the evidence $\mathbf{E} = \mathbf{e}$ contains both $\mathbf{U} = +\mathbf{u}$ and $\mathrm{C} = +\mathrm{c}$. Using the same arguments as above, we know that $v_{\mathrm{h}|\mathbf{e}}(\mathrm{C}|\mathrm{h}, -\mathbf{u}) = 0$ for any $-\mathbf{u} \neq +\mathbf{u}$. We therefore consider only $\mathbf{U} = +\mathbf{u}$, and can then reduce Eq. (A.1) as ...

$$
\begin{aligned}
v_{+\mathrm{h}|\mathbf{e}}(\mathrm{C}|\mathrm{h},+\mathbf{u}) &= \left[\begin{array}{l} \displaystyle\sum_{\mathrm{c}\in\mathrm{C}} \frac{1}{\hat{\Theta}_{\mathrm{c}|\mathrm{h},+\mathbf{u}}} \big[\mathrm{P}(\mathrm{c},\mathrm{h},+\mathbf{u},+\mathrm{h}\,|\,\mathbf{e}) - \mathrm{P}(+\mathrm{h}\,|\,\mathbf{e})\mathrm{P}(\mathrm{c},\mathrm{h},+\mathbf{u}\,|\,\mathbf{e})\big]^2 \\ \quad - \big[\mathrm{P}(\mathrm{h},+\mathbf{u},+\mathrm{h}\,|\,\mathbf{e}) - \mathrm{P}(+\mathrm{h}\,|\,\mathbf{e})\,\mathrm{P}(\mathrm{h},+\mathbf{u}\,|\,\mathbf{e})\big]^2 \end{array}\right] \\
&= \left[\begin{array}{l} \displaystyle\frac{1}{\hat{\Theta}_{+\mathrm{c}|\mathrm{h},+\mathbf{u}}} \big[\mathrm{P}(\mathrm{h},+\mathrm{h}\,|\,\mathbf{e}) - \mathrm{P}(+\mathrm{h}\,|\,\mathbf{e})\,\mathrm{P}(\mathrm{h}\,|\,\mathbf{e})\big]^2 \\ \quad - \big[\mathrm{P}(\mathrm{h},+\mathrm{h}\,|\,\mathbf{e}) - \mathrm{P}(+\mathrm{h}\,|\,\mathbf{e})\,\mathrm{P}(\mathrm{h}\,|\,\mathbf{e})\big]^2 \end{array}\right]
\end{aligned}
$$

(This uses the fact that each term in the summation with $\mathrm{c} \neq +\mathrm{c}$ is 0.)

If $\mathrm{H} = +\mathrm{h}$, this is

$$
\begin{aligned}
v_{+\mathrm{h}|\mathbf{e}}(\mathrm{C}|+\mathrm{h},+\mathbf{u}) &= \left[\begin{array}{l} \displaystyle\frac{1}{\hat{\Theta}_{+\mathrm{c}|+\mathrm{h},+\mathbf{u}}} \big[\mathrm{P}(+\mathrm{h}\,|\,\mathbf{e}) - \mathrm{P}(+\mathrm{h}\,|\,\mathbf{e})\mathrm{P}(+\mathrm{h}\,|\,\mathbf{e})\big]^2 \\ \quad - \big[\mathrm{P}(+\mathrm{h}\,|\,\mathbf{e}) - \mathrm{P}(+\mathrm{h}\,|\,\mathbf{e})\,\mathrm{P}(+\mathrm{h}\,|\,\mathbf{e})\big]^2 \end{array}\right] \\
&= [\mathrm{P}(+\mathrm{h}\,|\,\mathbf{e})(1-\mathrm{P}(+\mathrm{h}\,|\,\mathbf{e}))]^2 \left[\frac{1}{\hat{\Theta}_{+\mathrm{c}|+\mathrm{h},+\mathbf{u}}} - 1\right]
\end{aligned}
$$

but if $\mathrm{h}' \neq +\mathrm{h}$,

$$
v_{+\mathrm{h}|\mathbf{e}}(\mathrm{C}|\mathrm{h}',+\mathbf{u}) = \big[\mathrm{P}(+\mathrm{h}\,|\,\mathbf{e})\mathrm{P}(\mathrm{h}'\,|\,\mathbf{e})\big]^2 \left[\frac{1}{\hat{\Theta}_{+\mathrm{c}|\mathrm{h}',+\mathbf{u}}} - 1\right] \qquad \square
$$

References

[1] B. Abramson, J. Brown, W. Edwards, A. Murphy, R.L. Winkler, Hailfinder: A Bayesian system for forecasting severe weather, Intl. J. Forecasting 12 (1) (March 1996) 57–71.

[2] T.W. Anderson, D.A. Darling, A test of goodness of fit, J. Amer. Statist. Assoc. 49 (268) (December 1954) 765–769.

[3] Y. Akimoto, A note on uniform asymptotic normality of Dirichlet distribution, Math. Japon. 44 (1) (December 1996) 25–30.

[4] A.C. Atkinson, Plots, Transformations, and Regression, Oxford, 1985.

[5] C. Boutilier, N. Friedman, M. Goldszmidt, D. Koller, Context-specific independence in Bayesian networks, in: Twelfth Annual Conference on Uncertainty in Artificial Intelligence (UAI-96), 1996.

[6] J. Binder, D. Koller, S.J. Russell, K. Kanazawa, Adaptive probabilistic networks with hidden variables, Machine Learning 29 (2–3) (1997) 213–244.

[7] J. Cheng, M.J. Druzdzel, Confidence inference in Bayesian networks, in: Seventeenth Conference on Uncertainty in Artificial Intelligence (UAI-2001), Morgan Kaufmann Publishers, 2001, pp. 75–82.
[8] E.F. Castillo, J.M. Gutiérrez, A.S. Hadi, Sensitivity analysis in discrete Bayesian networks, IEEE Trans. Man Cybernet. Syst. 27 (1997) 412–424.
[9] G. Cooper, E. Herskovits, A Bayesian method for the induction of probabilistic networks from data, Machine Learning 9 (1992) 309–347.
[10] P. Che, R.E. Neapolitan, J. Kenevan, M. Evens, An implementation of a method for computing the uncertainty in inferred probabilities in belief networks, in: 9th Annual Conference on Uncertainty in Artificial Intelligence (UAI-93), Morgan Kaufmann Publishers, 1993, pp. 292–300.
[11] D.A. Darling, The Kolmogorov–Smirnov, Cramer–von Mises tests, Ann. Math. Stat. 28 (1957) 823–838.
[12] A. Darwiche, A differential approach to inference in Bayesian networks, in: 16th Annual Conference on Uncertainty in Artificial Intelligence (UAI-93), 2000.
[13] March 1995, Special issue of "Communications of the ACM", on Bayesian Networks.
[14] R. Dechter, Bucket elimination: A unifying framework for probabilistic inference, in: Learning and Inference in Graphical Models, 1998.
[15] A. Darwiche, G.M. Provan, Query DAGs: A practical paradigm for implementing belief network inference, in: Twelfth Conference on Uncertainty in Artificial Intelligence (UAI'96), 1996.
[16] R.B. D'Agostino, M.A. Stephens, Goodness-of-Fit Techniques, Statistics, Textbooks and Monographs, vol. 68, Marcel Dekker Inc., 1986.
[17] R. Greiner, C. Darken, I. Santoso, Efficient reasoning, Computing Surveys, 2001.
[18] Y. Guo, R. Greiner, Discriminative model selection for belief net structures, in: Twentieth National Conference on Artificial Intelligence (AAAI-05), Pittsburgh, July 2005, pp. 770–776.
[19] R. Greiner, A. Grove, D. Schuurmans, Learning Bayesian nets that perform well, in: 13th Annual Conference on Uncertainty in Artificial Intelligence (UAI-97), 1997.
[20] http://www.cs.ualberta.ca/~greiner/RESEARCH/BNvar.
[21] E.H. Herskovits, C.F. Cooper, Algorithms for Bayesian belief-network precomputation, in: Methods of Information in Medicine, 1991, pp. 362–370.
[22] D.E. Heckerman, A tutorial on learning with Bayesian networks, in: M.I. Jordan (Ed.), Learning in Graphical Models, 1998.
[23] D. Heckerman, D. Geiger, D.M. Chickering, Learning Bayesian networks: The combination of knowledge and statistical data, Machine Learning 20 (1995).
[24] P. Hooper, Exact distribution theory for belief net responses, Technical report, University of Alberta, 2007, http://www.stat.ualberta.ca/~hooper/research/papers+talks/exactbeta.pdf.
[25] F. Jensen, F. Jensen, Optimal junction trees, in: Tenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-94), Morgan Kaufmann, San Francisco, CA, 1994.
[26] G. Kleiter, Propagating imprecise probabilities in Bayesian networks, Artificial Intelligence 88 (1996).
[27] R.L. Keeney, H. Raiffa, Decision with Multiple Objectives: Preferences and Value Tradeoffs, John Wiley & Sons, New York, 1976.
[28] H. Korth, A. Silberschatz, S. Sudarshan, Database System Concepts, McGraw Hill, 1998.
[29] U. Kjaerulff, L.C. Van der Gaag, Making sensitivity analysis computationally efficient, in: 16th Conference on Uncertainty in Artificial Intelligence (UAI-00), 2000.
[30] K.B. Laskey, Sensitivity analysis for probability assessments in Bayesian networks, IEEE Trans. Man Cybernet. Syst. 25 (6) (1995) 901–909.
[31] Y. Lin, M.J. Druzdzel, Computational advantages of relevance reasoning in Bayesian beliefs networks, in: Uncertainty in Artificial Intelligence (UAI-97), 1997, pp. 342–350.
[32] C. Lee, R. Greiner, S. Wang, Using variance estimates to combine Bayesian classifiers, in: International Conference on Machine Learning (ICML'06), Pittsburgh, June 2006.
[33] A.W. Moore, Private communication, May 2003.
[34] J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann, 1988.
[35] J.L. Peixoto, Hierarchical variable selection in polynomial regression models, American Statistician 41 (1987).
[36] B. Ripley, Pattern Recognition and Neural Networks, Cambridge University Press, Cambridge UK, 1996.
[37] D.E. Rumelhart, J.L. McClelland, the PDP Research Group (Eds.), Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1: Foundations, The MIT Press, Cambridge, 1986.
[38] A.P. Singh, What to do when you don't have much data: Issues in small sample parameter learning in Bayesian networks, Master's thesis, University of Alberta, 2004.
[39] D.J. Spiegelhalter, S.L. Lauritzen, Sequential updating of conditional probabilities on directed graphical structures, Networks (1990) 579–605.
[40] T. Van Allen, R. Greiner, P. Hooper, Bayesian error-bars for belief net inference, in: 17th Conference on Uncertainty in Artificial Intelligence (UAI-01), Aug 2001.
[41] T. Van Allen, Handling uncertainty when you're handling uncertainty: Model selection and error bars for belief networks, Master's thesis, Department of Computing Science, University of Alberta, 2000.
[42] G.I. Webb, P. Conilione, Estimating bias and variance from data, Technical report, 2005.
[43] S. Wilks, Mathematical Statistics, John Wiley & Sons, 1962.