Probabilistic inference as an exercise in symbolic reasoning (Submitted to *Calculemus 2006*)

Robert Dodier*
Infotility, Inc.

May 1, 2006

Abstract

According to the interpretation of probability sometimes called the logical Bayesian interpretation, probability is the appropriate method for reasoning under uncertainty, in the sense that it is essentially the only generalization of logic to degrees of belief other than 0 and 1, given certain desiderata. However, the broad application of probability to many practical inference problems is hampered by the difficulty of carry out the necessary computations. In particular, exact results are known for a relatively small class of probability distributions, and numerical approximations are applied otherwise. Towards the general goal of applying probabilistic inference in a wider variety of real-world problems, this paper presents a scheme for symbolic inference in general probabilistic models. The general heuristic is to apply symbolic computation when possible, and to fall back on numerical approximations only if the attempt to compute a symbolic result fails. A prototype of the inference system is implemented as a collection of functions in a symbolic computation system. This approach allows for straightforward, comprehensible statement of assumptions and data, from which conclusions are obtained via application of the laws of probability. The conclusions are, to a good approximation, set in stone once the assumptions and data are established; thus we may say, per the dictum of Leibniz, "Calculemus."

1 Introduction

We take as our starting point the notion, captured by R. T. Cox in his delightful paper, "Probability, frequency, and reasonable expectation" [4], that generalizing logic to degrees of belief other than 0 and 1 yields the laws of probability. This is the so-called "logical Bayesian" school of thought; E.T. Jaynes [7] was another notable proponent. From this point of view, reasoning about uncertain propositions, whatever the source of the uncertainty, should be carried out by the laws of probability. We accept this without further discussion and focus on the manner in which such reasoning is carried out.

Certainly, we may endlessly multiple problems in which uncertain propositions naturally appear: global climate, effects of economic policies, reliability of computer hardware, is red wine good for one's health, etc., etc., *ad infinitum.* Analyses of such problems are typically stated without exposing the assumptions which lead to the conclusions, for

*Please address correspondence to: robert.dodier@gmail.com

example, "We estimate that if no measures are taken to hold back the sea, a one meter rise in sea level would inundate 14,000 square miles [in the United States], ..." [14] The reader wants to know from where such a result is derived; the answer is typically given in a lengthy and necessarily incomplete manner in the article text.[1] Ideally, assumptions would be spelled out clearly for the benefit of readers who wish not only to review the reported results, but also to modify the assumptions and then see how the results change. It may also be the case that there may be some assumptions (values for free parameters, for example) which might usefully appear explicitly in the results.

The goal of the present work, therefore, is two-fold: (1) to allow for more succinct and therefore more communicable and more comprehensible statements of problems involving uncertain propositions, and (2) to allow assumptions to intrude on the final result when required by the laws of probability. We attempt to meet these goals with an implementation of probabilistic inference in a symbolic computation system.

2 Benefits of symbolic inference

The present work is motivated by the perceived advantages of symbolic processing in probabilistic inference. The benefits seem to be two-fold. On the one hand, symbolic processing makes it easier to revise assumptions. This might be called a "late to the party" or "looking over the shoulder" mode, because it could be very usefully employed to revise or revisit an analysis made by someone else originally (perhaps the original author at a later date).

On the other hand, a symbolic approach makes it possible to distill the problem into a succinct statement which is more easily communicable and comprehensible. It is often the case in numerical systems that key assumptions and formulas appear as miniscule bits of code deep within a complex program. Ideally, these bits will be extracted and maintained apart from the mass of code which is necessary to get to some final results.

Ideally, the entire game ought to be focused on the problem statement. Once the statement is fixed, conclusions follow by turning the crank, that is, applying the laws of probability. The computational mechanism may be very complex, but we only need to build it once, and it is straightforward to determine whether it is working correctly. So discussion can be focused on the problem statement; once it is decided, we may say, with Leibniz, "Calculemus."

3 Symbolic computation systems and symbolic inference

Symbolic computation systems (also called computer algebra systems) are compilers or interpreters for computer languages which allow indefinite results. A statement such as $Z = X + Y$ is considered an instruction in typical general-purpose languages, and has a definite result, assuming X and Y have definite values. (For simplicity, we'll ignore exceptions and other "out of band" computations). In contrast, symbolic systems accept

[1] The author wishes to make clear that Ref. [14] is a solid and valuable work, and the aim of the current work is not to criticize, but only to suggest ways that such reports might be made even more effective.

definite results but also allow indefinite results, typically evaluating an expression to the extent possible, and carrying forward the partially-evaluated result.

There have been other implementations of probabilistic inference in symbolic computation systems. Typically these implementations handle probability distributions within a restricted class. The Symbolic Probabilistic Inference system (SPI) [12, 8] restricts the probability distributions involved to Gaussian probability distributions. Another implementation [3] restricts distributions to discrete distributions.

In the present work, we shall attempt to generalize previous efforts, which allow symbolic parameters within a specified class of distributions, to allow arbitrary distributions. This generality comes at a price: systems built with a specific class of distributions in mind can be much more efficient, and, more importantly, such systems can handle a broader class of joint probability distributions, in particular those associated with multiply-connected graphs. A fair assessment of the system proposed in this paper is that is suitable for *symbolic analysis of small problems with arbitrary distributions.*

4 Implementation of the polytree algorithm

The polytree algorithm, [10] also known as Pearl's algorithm, is a method of computing posterior distributions in a polytree, that is, an acyclic directed graph in which there is at most one path from one node to another. (Multiply-connected graphs are more general, but also more trouble for computation.) The algorithm is conveniently stated in terms of quantities named π and λ, which are predictive (top-down) distributions and likelihood functions (bottom-up), respectively. Here are the relevant formulas, as given in Ref. [5]; a longer description of the algorithm is found in Ref. [6].

1. Posterior $p_{X|\mathbf{e}}$ for variable X given observed evidence $\mathbf{e}$:

$$p_{X|\mathbf{e}}(x) \;\propto\; \pi_X(x)\,\lambda_X(x) \tag{1}$$

2. Predictive support π_X for X:

$$\begin{aligned}\pi_X(x) &= p_{X|\mathbf{e}_X^+}(x) \\ &= \int du_1 \;\cdots \int du_m \; q_X(x, u_1, \ldots, u_m)\; \pi_{U_1,X}(u_1)\;\cdots\;\pi_{U_m,X}(u_m)\end{aligned} \tag{2}$$

3. Likelihood support λ_X for X:

$$\lambda_X(x) \;\propto\; p_{\mathbf{e}_X^-|X}(x) \;=\; \prod_{j=1}^{n} \lambda_{Y_j,X}(x) \tag{3}$$

4. Predictive message π_{X,Y_k} sent from X to child Y_k:

$$\pi_{X,Y_k}(x) \;=\; p_{X|\mathbf{e}\backslash\mathbf{e}_{X,Y}^-}(x) \;\propto\; \pi_X(x) \prod_{\substack{j=1\\ j\neq k}}^{n} \lambda_{Y_j,X}(x) \tag{4}$$

5. Likelihood message λ_{X,U_k} sent from X to parent U_k:

$$\begin{aligned}\lambda_{X,U_k}(u_k) &= p_{\mathbf{e}\backslash\mathbf{e}^+_{X,U_k}|U_k}(u_k) \\ &\propto \int dx \int du_1 \cdots \int du_{k-1} \int du_{k+1} \cdots \int du_m \\ &\qquad \lambda_X(x)\, q_X(x, u_1, \ldots, u_m) \prod_{\substack{j=1 \\ j\neq k}}^{m} \pi_{U_j,X}(u_j) \end{aligned} \tag{5}$$

These formulas are relatively simple, in a sense. The only operations required are integration and pointwise multiplication of functions. The catch, of course, is that exact results for integrals of probability functions may not be available. One approach, then, is to restrict the class of probability functions to those for which exact results are known. Typically these are discrete and Gaussian probability functions. Other functions are approximated as discrete functions or linear combinations of Gaussian functions. Another approach, which we shall adopt here, is to attempt to compute as much of the result as possible, leaving an explicit representation of the unknown integral in the result if need be.

The above formulas for the polytree algorithm have been implemented in the symbolic computation system Maxima [9]. The code, which is called "Blossom" to suggest (with a little whimsy) the botanical nature of mathematical calculations, is available from the author on request. Systems other than Maxima could have been used equally well. The most important consideration is the symbolic representation of integrals, which is common to many symbolic computation systems.

An effort was made to make the implementation as similar to the formulas which define the polytree algorithm. These formulas plus supporting functions run to about 300 lines of Maxima code.

```
posterior [x] :=
    if noninformativep (lambda% [x])
    then pi% [x]
    else
        normalize (ptwise_lambda_product ([x], pi%[x] * lambda%[x]), x);

pi% [x] :=
   (apply ("*", pi_messages [x]),
    ptwise_lambda_product (cons (x, parents [x]), cpd [x] * %%),
    buildq ([aa : [x],
        expr : multiple_integral (%%, parents [x])],
        lambda (aa, expr)));



lambda% [x] :=
    ptwise_lambda_product ([x], apply ("*", lambda_messages[x]));

pi_message [x, y] :=
```

```
    (ptwise_lambda_product ([y], pi%[y] * lambda_excluding [y, x]),
     normalize (%%, y));


lambda_message [x, y] :=
     if boundp (y) then delta (y - ev(y))
     elseif lambda% [y] = 1 then 1
     else block
       ([p : parents_excluding [y, x],
         m : pi_messages_excluding [y, x]],
         pdf[y] * lambda%[y] * apply ("*", m),
         ptwise_lambda_product (cons (y, p), %%),
         multiple_integral (%%, p));
```

When Maxima can solve the integrals required, the result contains no integrals. But for many distributions, the integrals are intractable, for Maxima and perhaps for any symbolic computation system. Unevaluated integrals are carried through the computation like any other term, and their appearance in a result shows where approximations (in the form of numerical integration) can be applied. This policy, to compute symbolically as much as possible and resort to numerical approximations only when necessary, can be contrasted with another program for inference in general joint distributions, namely BUGS [1], which is strictly numerical: even if an exact result is possible, a numerical method (Gibbs sampling) is applied anyway.

Probabilistic models are represented as unevaluated function expressions; this is the usual manner of constructing objects in Maxima. New operators `-->` and `with` have been defined to make it convenient to represent the relation of a variable to those on which it immediately depends and to specify the domain and conditional probability distribution of the variable. A useful feature of Maxima, in this context, is the ability to define new operators which can be parsed by the standard input parser into an ordinary Maxima expression; this is equivalent to constructing and exposing the data structure that is called the parse tree or abstract syntax tree in some compilers.

A typical specification is similar to

```
[A, B, C] --> X with (dom = interval (minf, inf),
        cpd = gaussian_pdf (X, A + B + C, sigma_X))
```

Here X is conditional on A, B, and C, which are called the parents of X. Without the arrow, a variable is assumed to have no parents. The domain of X and the distribution of X conditional on its parents are specified as `dom` and `cpd`, respectively. A specification for the joint probability distribution for several variables might then be

```
'jpd ([A, B, C] --> X with (dom = interval (minf, inf),
            cpd = gaussian_pdf (X, A + B + C, sigma_X)),
        A with (dom = interval (0, inf), cpd = laplace_pdf (lambda_A)),
        B with (dom = interval (0, inf), cpd = laplace_pdf (lambda_B)),
        C with (dom = interval (minf, inf),
            cpd = gaussian_pdf (0, sigma_C)));
```

Let us now consider some examples of joint distributions specified in this manner, and the results (posterior distributions and other quantities) which can be computed from them.

5 Examples

We consider some toy problems which demonstrate some of the advantages of symbolic probabilistic inference. More elaborate examples are under development.

5.1 Gaussian prior plus Gaussian noise

In this example, there are two variables, X and Y. Y is equal to X plus Gaussian noise, and X has a Gaussian prior. The posterior for Y is again a Gaussian density. The problem statement is

```
'jpd
   (X with cpd = ''(gaussian_pdf (X, mu_X, sigma_X)),
    X --> Y with cpd = ''(gaussian_pdf (Y, X, sigma_Y)));
```

with

```
gaussian_pdf (x, m, s) :=
    exp (-(1/2)*((x - m)^2/s^2)) / (s * sqrt (2 * %pi));
```

This is a classical model, and Maxima can carry out the necessary integration to yield the classical result.

$$\frac{\sqrt{2}\, e^{-\frac{Y^2 - 2\,\mu_X\, Y + \mu_X^2}{2\,\sigma_Y^2 + 2\,\sigma_X^2}}}{2\,\sqrt{\pi}\,\sqrt{\sigma_Y^2 + \sigma_X^2}} \tag{6}$$

5.2 Laplacian prior plus Gaussian noise

Again we assume that Y is X plus Gaussian noise, but this time the prior for X is a Laplacian density function (i.e., $e^{-x/\alpha}/\alpha$ for some α). The problem statement is

```
'jpd
   (X with '(dom = interval (0, inf), cpd = ''(laplacian_pdf (X, alpha_X))),
    X --> Y with cpd = ''(gaussian_pdf (Y, X, sigma_Y)));
```

with

```
laplacian_pdf (x, a) := (1/a) * exp (- x/a);
```

Maxima can carry out the integration required to compute the posterior for Y, and reports the result as

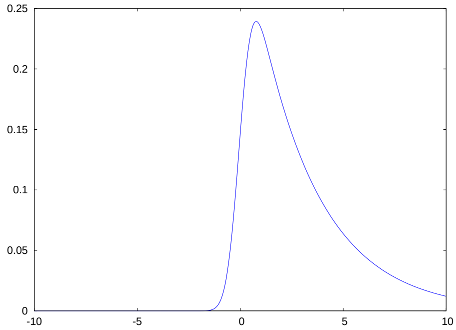

Figure 1: Posterior for Y given Laplacian prior for X, calculated symbolically.

$$\frac{\frac{\sqrt{2}\,\sqrt{\pi}\,\sigma_Y\,e^{\frac{\sigma_Y^2}{2\,\alpha_X^2}-\frac{Y}{\alpha_X}}\,\mathrm{erf}\left(\frac{\sqrt{2}\,\alpha_X\,Y-\sqrt{2}\,\sigma_Y^2}{2\,\alpha_X\,\sigma_Y}\right)}{2}+\frac{\sqrt{2}\,\sqrt{\pi}\,\sigma_Y\,e^{\frac{\sigma_Y^2}{2\,\alpha_X^2}-\frac{Y}{\alpha_X}}}{2}}{\sqrt{2}\,\sqrt{\pi}\,\alpha_X\,\sigma_Y} \tag{7}$$

This function, with $\alpha_X = 3$ and $\sigma_Y = 1/2$, is shown in Figure 1. Qualitatively, it looks like a softened exponential function, as we would expect.

5.3 sech^2 prior plus Gaussian noise

This example differs from the other two in that the integral required to construct the posterior is beyond Maxima's capabilities to compute symbolically. Therefore we will approximate it numerically. The problem statement is

```
'jpd
   (X with '(dom = interval (0, inf), cpd = ''(sech2_pdf (X, alpha_X))),
    X --> Y with cpd = ''(gaussian_pdf (Y, X, sigma_Y)));
```

where

```
sech2_pdf (x, a) := (1/a) * (sech (x/a))^2;
```

In this case, Maxima reports that the posterior p_Y is

$$\frac{\int_0^\infty \mathrm{sech}^{\,2}\left(\frac{X}{\alpha_X}\right)\,e^{-\frac{Y^2}{2\,\sigma_Y^2}+\frac{X\,Y}{\sigma_Y^2}-\frac{X^2}{2\,\sigma_Y^2}}\,dX}{\sqrt{2}\,\sqrt{\pi}\,\alpha_X\,\sigma_Y} \tag{8}$$

The unevaluated integral is carried through into the expression for the posterior. Replacing the symbolic integral with a numerical method (namely algorithm QAGI from QUADPACK [11]) via Maxima's rule-processing mechanism, we can now compute numerical values of the posterior for given assignments to the parameters α_X and σ_Y. The numerically-calculated posterior, for $\alpha_X = 1$ and $\sigma_Y = 2$, is shown in Figure 2.

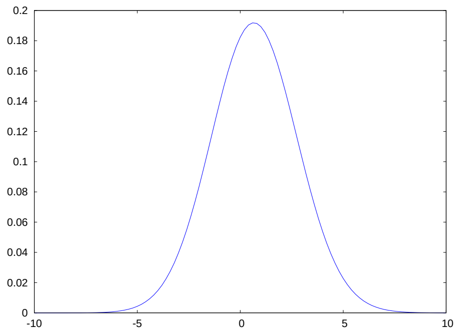

Figure 2: Posterior for Y given sech^2 prior for X, calculated numerically by the QAGI algorithm.

6 Directions for further research

There are some obvious directions for further development of the Blossom package. Whether or not Blossom yields symbolic results depends mostly on the capability of Maxima's integration function. However, there are results known for special cases involving probability functions; a short catalog of such cases is given in Appendix C of Ref. [6]. We could augment Maxima's symbolic integration function with these special cases, perhaps via Maxima's rule-processing mechanism.

A well-known limitation of the polytree algorithm is that it does not apply to joint probability distributions associated with multiply-connected graphs, which comprise a large and practically useful class. We might extend Blossom to multiply-connected graphs by applying a conditioning algorithm, which makes the problem amenable to solution by the polytree algorithm, at the cost of introducing an integration over each variable in the conditioning set. More ambitiously, symbolic methods for directly handling multiply-connected graphs might be developed.

More generally, it would be very useful to extend the scope of Blossom to include decision problems. Assuming a standard decision-theoretic foundation (e.g., [2]), the major piece that is missing at present is the representation of utility functions. Functions to handle the minimization of expected utility, perhaps constrained or unconstrained, would also have a place. The author considers the treatment of decision problems to be the larger goal of the Blossom project.

References

[1] The BUGS Project: Bayesian Inference Using Gibbs Sampling. URL `http://www.mrc-bsu.cam.ac.uk/bugs`

[2] R.T. Clemen. *Making hard decisions.* Belmont, CA: Duxbury Press, 1991.

[3] E. Castillo, J.M. Gutierrez, and A.S. Hadi. *Expert Systems and Probabilistic Network Models.* New York: Springer-Verlag, 1997.

[4] R.T. Cox. Probability, frequency, and reasonable expectation. *Am. J. Physics*, 14(1):1–13, 1946. Reprinted in Ref. [13].

[5] E. Driver and D. Morrell. Implementation of continuous Bayesian networks using sums of weighted Gaussians. In P. Besnard and S. Hanks, editors, *Proc. 11th Conf. Uncertainty in Artificial Intelligence*. San Francisco: Morgan Kaufmann, 1995.

[6] R. Dodier. *Unified Prediction and Diagnosis in Engineering Systems by means of Distributed Belief Networks.* PhD thesis, University of Colorado at Boulder, Dept. of Civil Engineering, 1999.

[7] E.T. Jaynes and L. Bretthorst. *Probability Theory: the Logic of Science*. Cambridge University Press, 2003.

[8] K.-C. Chang and R. Fung. Symbolic Probabilistic Inference with Continuous Variables. *Proc. 7th Conf. Uncertainty in Artificial Intelligence*. San Mateo: Morgan Kaufmann Publishers, pp 77–81, 1991.

[9] Maxima, a computer algebra system. URL `http://maxima.sourceforge.net`.

[10] J. Pearl. *Probabilistic reasoning in intelligent systems.* San Francisco: Morgan Kaufmann, 1988.

[11] R. Piessens, E. de Doncker-Kapenga, C.W. Uberhuber, and D.K. Kahaner. *QUADPACK: A Subroutine Package for Automatic Integration.* Berlin: Springer-Verlag, 1983. URL `http://netlib.org/slatec/quadpack`

[12] R.D. Shachter, B. D'Ambrosio, and B. del Favero. Symbolic Probabilistic Inference in Belief Networks. *Proc. Eighth National Conf. on Artificial Intelligence*, pp 126–131, 1990.

[13] G. Shafer and J. Pearl, editors. *Readings in uncertain reasoning.* San Mateo: Morgan Kaufmann, 1990.

[14] J.G. Titus, R.A. Park, S.P. Leatherman, J.R. Weggel, M.S. Greene, P.W. Mausel, S. Brown, G. Gaunt, M. Trehan, G. Yohe. Greenhouse Effect and Sea Level Rise: The Cost of Holding Back the Sea. *Coastal Management*, 19:171-204, 1991.