UNIFIED PREDICTION AND DIAGNOSIS IN ENGINEERING SYSTEMS BY MEANS OF DISTRIBUTED BELIEF NETWORKS

by

ROBERT H. DODIER

M.S., University of Colorado, Boulder, Colorado, 1995

B.A., Portland State University, Portland, Oregon, 1986

A dissertation submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Civil, Environmental, and Architectural Engineering

1999

This dissertation for the Doctor of Philosophy degree by

Robert H. Dodier

has been approved for the

Department of

Civil, Environmental, and Architectural Engineering

by

_____

Jan F. Kreider

_____

Michael J. Brandemuehl

Date _____

Dodier, Robert H. (Ph.D., Civil Engineering)

Unified Prediction and Diagnosis in Engineering Systems by means of Distributed Belief Networks

Dissertation directed by Professor Jan F. Kreider

This dissertation describes the theory, implementation, and application of a class of graphical probability models, called distributed belief networks, for the purposes of prediction, diagnosis, and calculation of the value of information in engineering systems. Probability models have the very desirable property that several useful operations can be stated as the computation of probability distributions; prediction and diagnosis correspond to the calculation of certain posterior distributions, and the value of information can be interpreted as the calculation of an average decrease of entropy of a posterior distribution. These operations, and others, are different ways of looking at a single model — there is no need for separate models for different operations.

A belief network, for the purposes of this dissertation, is a directed graph associated with a set of conditional probability distributions. Each node in the graph corresponds to a variable in a probability model. A distributed belief network is a belief network implemented on multiple processors. Posterior distributions for variables in the belief network are computed by a message-passing algorithm called the polytree algorithm. For the purpose of modeling engineering systems, it is important that information about the problem domain be represented as the probability distributions natural for the problem, and the polytree algorithm is easily adapted to handling many kinds of distributions. However, the algorithm has only limited applicability to graphs containing undirected cycles.

Some illustrative applications are described, including several sensor models, a model for a mixing box damper, a heating coil, and a belief network to analyze energy use for the purpose of selecting a rate structure.

ACKNOWLEDGEMENTS

I dedicate this dissertation to my mother and father,

Jean Hernandez Dodier and Victor Dodier.

This research was supported by grants from the Edwin Link Foundation, the family of Gustave Larson, and the American Society of Heating, Refrigeration, and Air-conditioning Engineers.

Xing Hai-Yun, Taner Bilgic, Markus Laun, and Michelle Franz aided this project by allowing me to install RISO on their sites. My heartfelt thanks to them.

Finally, special thanks to Margaret Bailey, Kriengkrai Assawamartbunlue, and the members of my committee, especially Peter Curtiss, for insightful discussions.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

Chapter 1

AN INTERCONTINENTAL BELIEF NETWORK

By way of introduction, let us consider a reasoning problem solved by a distributed belief network. Like a proper epic, this dissertation begins *in media res.*

1.1 Has Elvis Presley come back for a visit?

American cultural icon Elvis Presley (1935–77) was a middling talent — "I don't know much about music. In my business, you don't have to" — but that proved no obstacle to his posthumous veneration as the King of Rock 'n' Roll. Every now and then, it seems, he comes back to visit our world from the atemporal realm where he dwells with aliens, Sasquatch, and Princess Grace. Or so the tabloids proclaim; amidst an undifferentiated gangue of conflicting reports, how are we to extract the nuggets of truth? I will argue in Chapter 2 that the proper way to reason in about uncertain propositions is to use the laws of probability. To navigate the shoals of unsteady information, I've constructed a reasoning system called RISO which we can use to compute degrees about propositions such as "Has Elvis come back for a visit?"

We humans can reason quickly and accurately when there are just a few facts to take into account and there is a straightforward relation between a proposition and the background information before which it is asserted. But our capacity to reason carefully in the presence of ambiguity and conflicting sources of information degrades quickly with the number of sources and the decreasing precision of relations, so we need a little help. RISO is a system to aid reasoning in spatially and temporally extended problems, which implements a class of graphical probability models called *distributed belief networks.* A belief network is just what the name suggests — a graph to compute beliefs, and a distributed belief network is a set of belief networks in different places

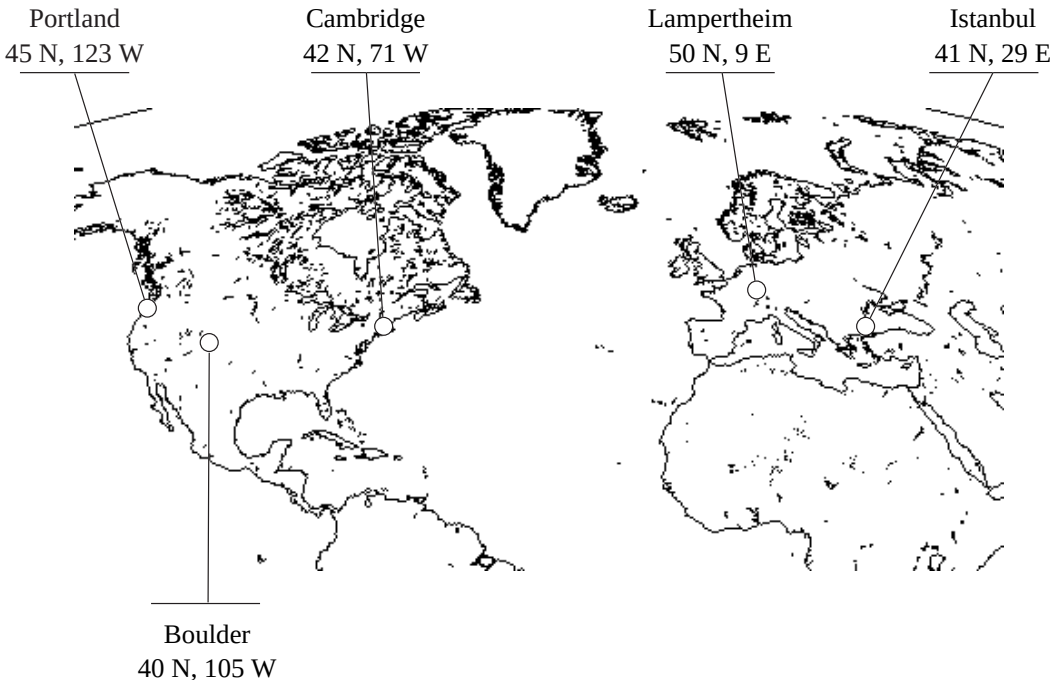

Figure 1.1: Our home planet viewed from about 10,000 km away, showing RISO sites around the world. As the number of sites is increasing 400% per year, we can expect RISO installations to soon outnumber elementary particles, not to mention available IP addresses.

linked together by Internet communications. At present, RISO belief network software has been installed across vast reaches of the Northern Hemisphere (see Figure 1.1) and one hopes that RISO will soon find a home in the southern latitudes as well.

I've constructed a distributed belief network (Figure 1.2) to help us reconcile claims about the presence or absence of Elvis Presley. There are news reports of Elvis from Portland, Oregon, from Istanbul, from Cambridge, Massachussets, and from Lampertheim, Germany. By some good fortune, RISO is running in each of these locations. As shown in Figure 1.2, in each location there is a belief network to assess any evidence observed locally concerning the presence of Elvis there, and a belief network in Boulder to integrate the reports from different locations. The combination of belief networks in

various locations creates a single, distributed belief network.

Each oval, called a *node* in the graphical jargon, in Figure 1.2 represents a proposition, such as $A =$"A cab driver says she saw Elvis." Each arrow shows a dependence of one proposition on another. For example, whether A occurs depends on whether Elvis is in the same city as the cab driver. If Elvis is there, it's much more likely that the cab driver will report that she saw him. But even if he's not there, the cab driver might be mistaken or deliberately try to mislead people by saying that she saw Elvis all the same. In each city, I've constructed a model of observations which could be made of the King, and these observations are assumed to be independent if one knows whether or not Elvis is in town. For example, if I know that Elvis is in Lampertheim, then knowing that he's been sighted at the tavern doesn't tell me anything about whether someone might see him at the hotel. Thus the model for each observation can be constructed separately from the model of every other observation, even the ones which can occur in the same locale. Once the models for all observations in all locations are constructed, the laws of probability tell us how to combine the evidence to come up with a global summary. But we're getting ahead of ourselves; in §1.3 we'll return to the problem of computing inferences in the distributed belief network.

The models for observations in Portland, Lampertheim, and Istanbul all have the same simple form: given that Elvis is in town or is not, each observation is more or less likely. The model for Cambridge is a little more complicated. The cab driver has been talking to a friend, and the friend spoke with another friend. Their reports depend on the cab driver's, but their testimony is weakened by their distance from the scene of the crime, so to speak. Of course, just how much weaker is their testimony is determined by the laws of probability.

Despite his aethereal state, Elvis can still be in only place at a time, so of the propositions "Elvis is in Portland," "Elvis is in Istanbul," etc., at most one can be true. Let us abbreviate the propositions "Elvis is in..." by the first letter of each city, so we

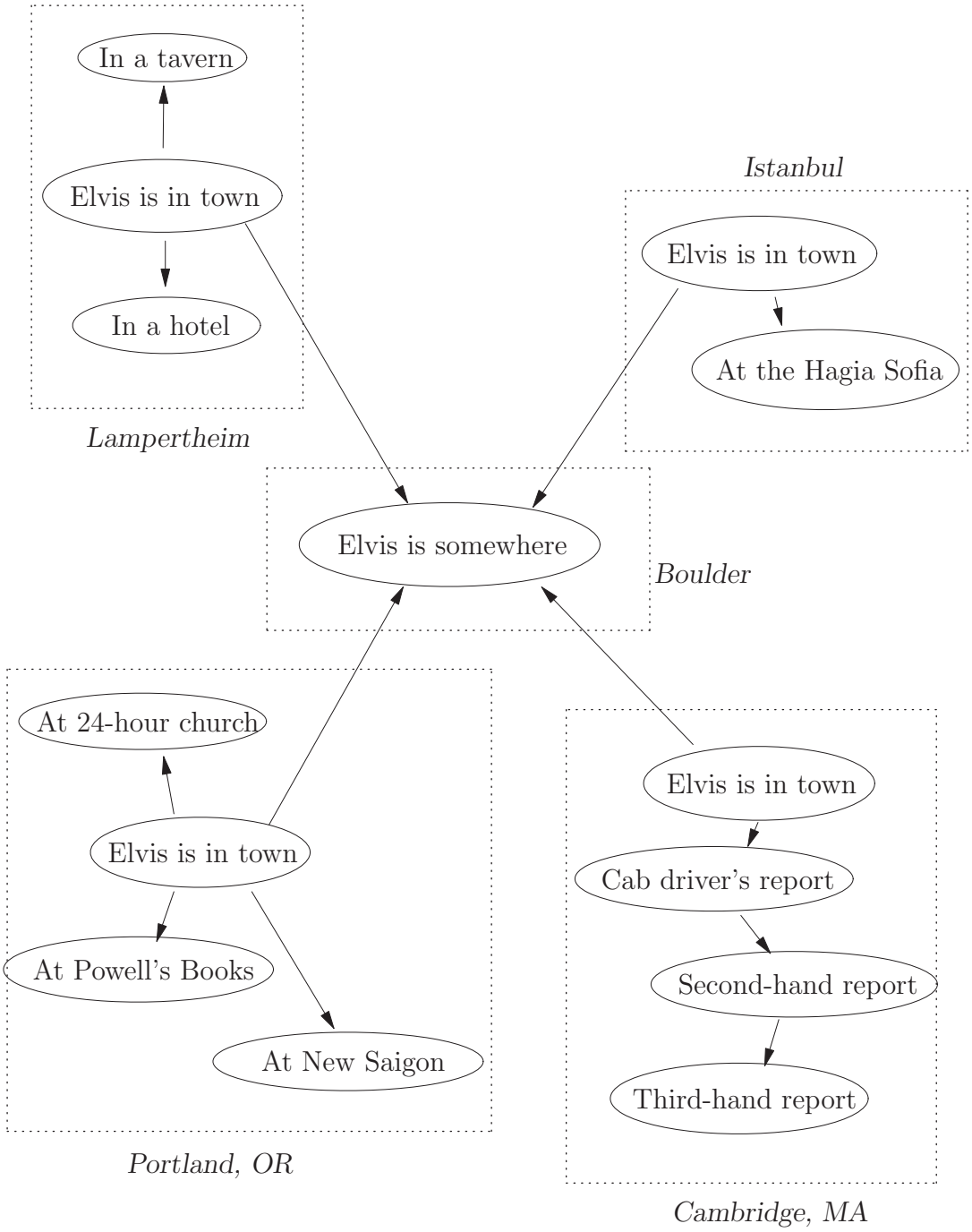

Figure 1.2: A distributed belief network for a simple reasoning problem. Has Elvis Presley come back for visit to our temporal domain? In the text, the abbreviations P, C, I, L, and E correspond to "Elvis is in Portland," "Elvis is in Cambridge," "Elvis is in Istanbul," "Elvis is in Lampertheim," and "Elvis is somewhere in the world," respectively.

have P, C, L, and I. Then

$$\begin{aligned} \text{exactly one of } P, C, L, I \quad = \quad & (P \wedge \neg C \wedge \neg L \wedge \neg I) \\ & \vee (\neg P \wedge C \wedge \neg L \wedge \neg I) \\ & \vee (\neg P \wedge \neg C \wedge L \wedge \neg I) \\ & \vee (\neg P \wedge \neg C \wedge \neg L \wedge I) \end{aligned} \tag{1.1}$$

denoting conjunction, disjunction, and negation as $\wedge$, $\vee$ and $\neg$, respectively. Given that can quantify our beliefs about the elementary propositions P, C, L, and I, how much should we believe the compound proposition? It turns out (§2.1) that any compound proposition can be expressed in terms of conjunction and negation alone, and that these two operations correspond to simple functions of the probabilities of the elementary propositions. In §1.3 we'll use the conjunction and negation functions to compute numerical values of the compound proposition, "Is Elvis in exactly one of the cities?"

1.2 Assignment of numerical probabilities in observation models

The belief network shown in Figure 1.2 requires the specification of a conditional probability for every node which has arrows leading into it, and an unconditional probability for every node which has no arrows leading into it. These specifications are the basis of the calculations which are presented in §1.3.

Let us begin with the so-called *root nodes*, that is, the ones which have no arrows leading into them. In the absence of any evidence, we believe it is unlikely that Elvis is in Lampertheim, Cambridge, or Istanbul, but it's a little more likely that he is in Portland — after all, it is the home of the 24 Hour Church of Elvis. Let us make the following assignments.

$$\Pr(P) = 0.01, \qquad \Pr(C) = 0.001, \qquad \Pr(L) = 0.001, \qquad \Pr(I) = 0.001 \tag{1.2}$$

The corresponding assignment to the negation of L is just $1 - \Pr(L)$, likewise for the other cities.

The conditional probability of the compound proposition E is computed from the probabilities of P, C, L, and I. In §§2.1–2.1.1 we'll see that there are simple formulas for the conjunction and disjunction of independent propositions, and for negation. Calculating the probability of the compound proposition $E =$"Exactly one of P, C, L, I," we find

$$\begin{aligned}\Pr(\text{exactly one of } P,C,L,I) \;=\; & \Pr(P)(1-\Pr(C))(1-\Pr(L))(1-\Pr(I)) \\ & +(1-\Pr(P))\Pr(C)(1-\Pr(L))(1-\Pr(I)) \\ & +(1-\Pr(P))(1-\Pr(C))\Pr(L)(1-\Pr(I)) \\ & +(1-\Pr(P))(1-\Pr(C))(1-\Pr(L))\Pr(I) \quad (1.3)\end{aligned}$$

As evidence is entered into the belief network, the probabilities $\Pr(P)$, $\Pr(C)$, etc., are replaced by $\Pr(P|\text{evidence in Portland})$, $\Pr(C|\text{evidence in Cambridge})$, and so on, but the computation in Eq. 1.3 is carried through just the same.

The observation models in each city have a common form. Each one specifies the probability p that the observed evidence occurs when Elvis is in town, and the probability q that the observed evidence occurs when Elvis is not in town. Such a model can be represented in tabular form like this:

	Observed evidence	Looked for, didn't see evidence
Elvis in town	p	$1-p$
Elvis not in town	q	$1-q$

The parameters p and q are assigned as shown in Table 1.1. In general, some characteristic of Elvis has been identified, and then we check to see if we find that characteristic somewhere in each city. Each characteristic is something we're more likely to find associated with Elvis than otherwise, so every p is greater than the corresponding q.

To complete the specification of the Elvis belief network, it remains only to assess the reliability of the cab driver's friends. A second-hand report can be considered as evidence of the first-hand report, and so a tabular representation analogous to the one shown above for direct observations can be constructed. A third-hand report is evidence

Table 1.1: Parameters for observation models.

Observation	p	q
Sunglasses at hotel (L)	0.7	0.5
Sequins at tavern (L)	0.2	0.05
Sequins at Hagia Sofia (I)	0.2	0.01
Sequins at Powell's Books (P)	0.1	0.01
Sideburns at New Saigon (P)	0.33	0.25
Velvet painting speaks (P)	0.5	0.0001
Cab driver's report (C)	0.1	0.0001

of the second-hand report, and so on, as far as we wish to carry the process. Let's assume that friend number 1 is a little too happy to get his name in the paper:

	Friend 1, positive report	Friend 1, negative report
Cab driver's positive report	1	0
Cab driver's negative report	0.1	0.9

In this little table and the next one, "positive report" is a shorthand for "cab driver says she saw Elvis," likewise "negative report" stands for "cab driver says she did not see Elvis." Even if the cab driver gives a negative report, friend 1 might relay it as a positive report. Friend number 2 is more reliable:

	Friend 2, positive report	Friend 2, negative report
Friend 1, positive report	0.999	0.001
Friend 1, negative report	0.001	0.999

Later on (§1.3) we'll take some testimony from friend 2, but we'll see that the reliability of friend 2 is mostly useless, due to the intervening unreliability of friend 1.

We're now equipped to carry out some inferences in the belief network shown in Figure 1.2.

1.3 Calculation of degrees of belief

Given the structure and the numerical assignments described in preceding sections, let us see how different observations affect our beliefs about the presence of Elvis. We sit down in front of a computer in Boulder (or maybe we telnet from a nearby mountain peak) and tell RISO to connect our local belief network, labeled "Boulder" in Figure 1.2, with the belief networks in other cities. Initially no observations have been made anywhere — how does that affect the degree of belief calculated for the proposition "Elvis is in town" in each locale? In the absence of an observation one way or the other, the degree of belief for each "Elvis is in town" node is just the prior probability that was assigned in §1.2. So the node "Elvis is somewhere" in the Boulder belief network is told that its parents have been assigned the probabilities given in Eq. 1.2. We then use Eq. 1.3 to calculate $\Pr(E) = 0.01293$. In the absence of any evidence, the probability that Elvis has come back is very low — no surprise there.

We'll ask RISO to notify us if there are any changes upstream from Boulder which give us the opportunity to calculate a new probability of Elvis' return. By studying the connections in Figure 1.2, RISO can tell which calculated probabilities need to be revised when evidence is entered or taken away somewhere in the distributed belief network. It turns out that simply being connected by some series of arrows is not the proper way to determine what propositions are affected; the appropriate criterion, described in §2.6.1, is called *d-separation*. Whenever evidence is entered or taken away in a node which is not *d*-separated from E, we need to recalculate the probability of E.

Some observations are made in Lampertheim — the concierge at the hotel in Lampertheim notices a mysterious man wearing sunglasses in the lobby, but the bartender at the tavern hasn't seen anyone wearing a gold sequin suit. If Elvis were in Lampertheim, he would probably be wearing sunglasses, but lots of people wear sunglasses, so that's rather weak evidence for Elvis. Likewise, failing to see anyone in gold sequins isn't proof of his absence — maybe Elvis is wearing his toreador costume. According to model of

observations in Lampertheim, we have

$$\begin{aligned} \Pr(\text{“someone wearing sunglasses”}|L) &= 0.7 \\ \Pr(\text{“someone wearing sunglasses”}|\neg L) &= 0.5 \end{aligned}$$

for the probability of the hotel observation given different values of the parent L, and

$$\begin{aligned} \Pr(\text{“no-one in gold sequins”}|L) &= 0.8 \\ \Pr(\text{“no-one in gold sequins”}|\neg L) &= 0.95 \end{aligned}$$

for the probability of the tavern observation given the parent L. Functions of the form shown here, with the proposition of interest appearing as a variable on the right-hand side of the bar in a conditional probability, are called *likelihood* functions. We should combine these two functions of L by simply multiplying them together — this is a special case of the general rule for propagating information from evidence upward from children to their common parent; the general rule is described in §5.3. For the summarized likelihood, then, we simply compute

$$\begin{aligned} \text{likelihood}(L) &= 0.7 \cdot 0.8 = 0.56 \\ \text{likelihood}(\neg L) &= 0.5 \cdot 0.95 = 0.475 \end{aligned}$$

To compute the posterior probability of L, the likelihood is combined with the prior for L in a simple way — multiply together the prior and the likelihood, and normalize so that the result sums to 1. Again, this is a particular example of a general rule described in §5.3. For the posterior, we have

$$\Pr(L|\text{sunglasses} = \text{yes}, \text{sequins} = \text{no}) = \frac{0.001 \cdot 0.56}{0.999 \cdot 0.475 + 0.001 \cdot 0.56} = 0.001179$$

When the observations are entered into the Lampertheim belief network, Boulder is notified that something has changed, and the updated probability for L is sent along. Upon receiving the message about L, we recompute E (using Eq. 1.3) and find that $\Pr(E|\text{Lampertheim evidence}) = 0.01311$. Hardly any change at all — note that if the likelihood function for L were constant, there would be no change at all. So the

likelihood function supplies information to the extent that different values are assigned for L and $\neg L$; the greater the difference, the greater the changes in the posteriors for L and E.

Over in Cambridge, the cab driver reports that she has seen Elvis. From this, the likelihood function for C is

$$\text{likelihood}(C) = 0.1, \qquad \text{likelihood}(\neg C) = 0.0001$$

The ratio 0.1/0.0001 (called the *likelihood ratio*) equals 1000, which shows that the observation is quite informative. The likelihood for C is combined with the prior to yield the posterior for C by multiplication and normalization, just as for L. This yields

$$\Pr(C|\text{cabbie reports seeing Elvis}) = \frac{0.001 \cdot 0.1}{0.999 \cdot 0.0001 + 0.001 \cdot 0.1} = .5003$$

When the cab driver's report is relayed to Boulder, the recalculated (via Eq. 1.3) probability for E is

$$\Pr(E|\text{Cambridge and Lampertheim evidence}) = 0.5002$$

The cab driver's report is strong evidence for Elvis' presence in Cambridge, but although the likelihood function very strongly favors "Elvis is here," the prior is just as heavily weighted in favor of "Elvis is not here," so the posterior gives just about even odds.

Let's see what's happening in Portland. There's a fellow with sideburns at the New Saigon restaurant on West Burnside, although that's not saying much, since there's plenty of facial hair in Portland. Somebody checked Powell's Bookstore, just up Burnside at 10th (or is it 11th?), but there's nobody in a gold sequin suit wandering the aisles there; perhaps they didn't search carefully enough, it's a big place. The big news from Portland is a *talking velvet painting* of the King at the 24 Hour Church of Elvis. Apparently this work of art spontaneously spouts such words of wisdom as "Lemme be your teddy bear," "I dated your big sister," and "You ain't nothing but a hound dog." Elvis himself hasn't been sighted there, but a mysteriously talking velvet painting is certainly strong evidence that he is in the vicinity. The likelihood function for the

talking painting alone is

$$\begin{aligned} \Pr(\text{talking painting}|P) &= 0.5 \\ \Pr(\text{talking painting}|\neg P) &= 0.0001 \end{aligned}$$

with the very hefty likelihood ratio $0.5/0.0001 = 5000$. The likelihood functions for the other observations are

$$\begin{aligned} \Pr(\text{sideburns at New Saigon}|P) &= 0.33 \\ \Pr(\text{sideburns at New Saigon}|\neg P) &= 0.25 \end{aligned}$$

and

$$\begin{aligned} \Pr(\text{no sequins at Powell's}|P) &= 0.90 \\ \Pr(\text{no sequins at Powell's}|\neg P) &= 0.99 \end{aligned}$$

which have likelihood ratios close to 1; these observations are not very informative. The combined likelihood is

$$\begin{aligned} \Pr(\text{Portland evidence}|P) &= 0.5 \cdot 0.33 \cdot 0.90 = 0.1485 \\ \Pr(\text{Portland evidence}|\neg P) &= 0.0001 \cdot 0.25 \cdot 0.99 = 2.475 \cdot 10^{-5} \end{aligned}$$

which, combined with the prior for P and normalized, yields

$$\begin{aligned} \Pr(P|\text{Portland evidence}) &= \frac{0.1485 \cdot 0.01}{2.475 \cdot 10^{-5} \cdot 0.99 + 0.1485 \cdot 0.01} \\ &= 0.9838 \end{aligned}$$

So, ignoring evidence from other cities, we have a strong degree of belief that Elvis is in Portland. This is due in part to the prior, which gives more weight to "Elvis is in town" than do the priors for other cities, but it is mainly due to the talking velvet painting.

The Portland belief network sends an informational message to Boulder summarizing the Portland evidence. Combining the evidence from Portland with that of Lampertheim and Cambridge, we find

$$\Pr(E|\text{Cambridge, Lampertheim, and Portland reports}) = 0.4987 \tag{1.4}$$

which is actually a little bit less than before, but that shouldn't surprise us. Since Elvis can only be in one place, the strong evidence from Portland conflicts with the slightly weaker evidence from Cambridge.

While all of this is going on, the RISO host in Istanbul becomes unreachable — the machine has crashed or there is a network problem. If some evidence is now observed in Istanbul, we won't find out about it until the connection is reestablished, so a conservative policy is to assume there is no evidence there. So we substitute the prior of I, which was requested by E when the connection to Istanbul was first established, for any informational message from Istanbul. This is a particular case of the general communications policy outlined in Appendix B.

There is a new development in Cambridge — it turns out the evidence was entered incorrectly. The cab driver didn't say she saw Elvis, it was the friend of the friend of the cab driver who said it. In the Cambridge belief network, the "cab driver says she saw Elvis" evidence is erased, and "friend of a friend says cab driver says she saw Elvis" is entered. This triggers a reassessment of E back in Boulder — Cambridge tells Boulder that the previous message is now invalid (because evidence in Cambridge changed), and Boulder requests a new message. The evidence in Cambridge is at the bottom of the heap — information bubbles up to C from the friend's friend's report, using the rules described in §5.3. The probability for "Elvis is in Cambridge" is revised,

$$\Pr(C|\text{friend's friend's report}) = 0.001878 \tag{1.5}$$

which is greater than the prior probability (0.001) for C, but not by much. Back in Boulder, E receives a message from C which incorporates the revised evidence. The revised posterior for E is

$$\Pr(E|\text{Cambridge, Lampertheim, and Portland reports}) = 0.9798 \tag{1.6}$$

Thus we now have a strong belief that Elvis is afoot somewhere in the world, and examining the probabilities for each location, we see that Cambridge and Lampertheim have very low probabilities, while Portland has a high probability; it appears that Elvis is in Portland.

1.4 Comments on the "Where is Elvis?" problem

The trivial problem which we have studied in this chapter illustrates some important points about belief networks. Among the useful properties of belief networks, described at greater length in §§2.7–2.8, is the capability to fuse or merge together different sources of information. Each source can be constructed independently, and the laws of probability prescribe the algorithm for combining the sources. In the Elvis belief network, each city can be considered a source, and even within the models for each city, each observation can be considered separately, and again the laws of probability tell how evidence from different observations should be combined. At the level of a single proposition, top-down (prior) information is combined with bottom-up (likelihood) information; the posterior is computed by combining the prior and likelihood by pointwise multiplication. Although the posterior probabilities which are computed will change when some observation models are added to the belief network or removed from it, the models for the existing or remaining observations need not be modified. Thus the belief network accomodates the combination or fusion of information on at least three different levels — between networks, between nodes in a single network, and within a single node.

The reader will note that the Elvis belief network required a substantial investment of time and thought to construct. A belief network is a quantitative representation of relations between propositions, and so it is a statement of knowledge about the problem domain. The ability of a belief network to make interesting computations (§2.7) is limited by the quality of the knowledge by which the belief network was constructed; a belief network computation essentially reexpresses or rerepresents the information which is built into the network. A belief network cannot turn straw into gold, but, happily, it can turn ingots into coins, foil, bracelets, and so on. Lest the reader become discouraged by the effort required to build a belief network, it is possible to identify structures and probability models for several common classes of problems, which should greatly speed the development of new belief networks. Some of these common models are described in Chapters 8 and 9, and especially Chapter 6.

To clarify an expression used repeatedly in §1.3, a message from one node A to another B contains the information from nodes connected to A, but the evidence is not represented explicitly. The information is present in a digested form, and expressed as a probability distribution or likelihood function. Thus it is generally not possible to reconstruct the evidence from messages, because there are usually many arrangements of the evidence which yield the same messages. From the recipient's point of view, the distilled information in the message is more useful than the raw evidence, because the recipient can ignore the structure of the belief network attached to the sender once the message is received.

On a philosophical note, one might wonder how to interpret the probabilities computed in §1.3. If we compute $\Pr(E) = 1$, does that mean that Elvis is truly here among us? No, a probability of 1 means that we are completely certain of the proposition, not that the proposition is true, and probability 0 means we are completely certain of the negation of the proposition, not that the proposition is false. In Chapter 2 probability is constructed as an extension of classical logic, and for better or worse, probability inherits the interpretational baggage of logic. Logic has a notorious "garbage in, garbage out" feature: the truth of deductions is no different than the premises upon which they are based. "If today is Tuesday, this must be Belgium" and "This is not Belgium" together imply "Today is not Tuesday," which is correct but not necessarily true. In the exactly the same way, probability calculations can be correct and not necessarily true. The premises upon which a belief network inference rests are the assignments of conditional and prior probabilities in the belief network; given inappropriate assignments, we can compute as much garbage as we want. The essential problem is the lack of a necessary connection between beliefs and the world "out there," which is discussed in a little more detail in §3.3.

1.5 What's to come in the dissertation

We have begun our journey through the world of belief networks with an enjoyable diversion through a simple belief network example. Several questions are raised by the

example — What does a belief network represent? How is its structure determined? Where do the numbers come from? How do we use a belief network to answer our queries? — which will be resolved in the chapters to follow. In broad outline, this dissertation will put a particular theory of probability on a solid foundation and then show how belief networks based on that theory can be used to solve practical problems. It is time to resume the main development — let us see where the remainder of the dissertation is going.

In Chapter 2, a derivation of probability from logic is sketched, and this provides the foundation for everything else in the dissertation. Useful operations such as prediction and diagnosis are interpreted in probabilistic terms, and some of the ways in which belief networks merge different sources of information are described.

In Chapter 3, a brief account of the history of the idea of probability is sketched. There is a brief discussion of the problem of induction.

In Chapter 4, the RISO belief network system is described. RISO was used in this chapter to implement the Elvis belief network and carry out the computations required. Less trivial examples of distributed belief networks are given in §§ 4.4, 8.1, and 9.1.

In Chapter 5, the inference algorithm which is used by RISO is described in detail. This algorithm, called the *polytree algorithm*, is well-suited to handling the various kinds of distributions which arise in engineering problems, although there are difficulties with belief networks which contain more than one path from some node to another. The polytree algorithm has been around for a number of years, but it is usually assumed that the distributions involved all belong to the same class. I have extended the polytree algorithm to handling different kinds of distributions, to better accomodate the demands of engineering models. This heterogeneous polytree algorithm, for all its limitations, is the major technical result of this dissertation.

In Chapter 6, some commonly-occurring belief network structures are described. These structures are models for sensors, and it is foreseen that such "off-the-shelf" models will be useful in engineering applications.

In Chapter 7, the problem of selecting a electricity rate schedule is solved by modeling

energy use with a belief network and computing expected demand and energy costs. The demand calculation requires that the distribution of the maximum of a number of variables be computed. The electricity rates problem is the most substantial application of a belief network in this dissertation.

In Chapter 8, two additional belief network applications are described. One is a model of a heating coil and the other is a model of a mixing box. I believe these models are typical of the kinds of engineering problems for which belief networks are well-suited.

In Chapter 9, a class of models for inferring model parameters is presented. I believe that such models will be useful for tuning parameters *in situ*, and so these models are an excellent topic for future investigation. This chapter, which is the final chapter of the dissertation, also contains some summary remarks.

There are several appendices which contain details of calculations or programming which would only burden the main development. In particular, Appendix C contains many special cases of the general inference rules described in Chapter 5.

Let us start over with the fundamentals: we need a means of reasoning correctly in an uncertain world. Chapter 2 opens with a search for the principles of uncertain reasoning.

Chapter 2

INTRODUCTION TO GRAPHICAL PROBABILITY MODELS

In the previous chapter, probability was accepted without discussion as the foundation on which the intercontinental belief network was constructed. Let us step back for a moment and consider the reasons for choosing probability instead of some other formalism for reasoning in an uncertain world. Uncertainty arises in many ways; in engineering problems we may distinguish uncertainty in experimental data (including noise and missing data), in model parameters, and in hypotheses, and still other forms of uncertainty may arise in other fields. It might be supposed that there are several reasoning formalisms, each of which is best-suited to handle a different kind of uncertainty, but it turns out there is no need to enumerate or distinguish different forms of uncertainty, because all forms can be handled by one formalism, namely probability, in a uniform way. It is not difficult to show that, given some reasonable axioms, the generalization of ordinary logic to degrees of belief other than 0 and 1 yields a system which is isomorphic to the usual formulation of probability. The development is not described in detail, but the essentials sketched in §2.1 below and further references are given.

As explained further in this chapter, probability is the basis for the implementation of the automated reasoning system described in this dissertation. The circles and arrows, so prominent in diagrams of belief networks such as Figures 2.1 and 2.2, are merely convenient representations of probabilistic concepts; the object of study is always the underlying probability model. However, as an aid to thought, the role of a graphical representation is very substantial, since the graph of a belief network is a quantitative representation of the independence properties of the probability model. The independence of two variables in a probability model can be determined by studying the graph

of the corresponding belief network, without referring to numerical values at all; the graphical criterion for independence, called *d*-separation, is described in §2.6.1.

In the remainder of this chapter, the customary metaphysical foundation of classical physics is assumed without question — this is mostly deterministic materialism, the dualism of *res cogitans* and *res extensa*, and the correspondence theory of truth. Like all such assumptions, these may be called into question in various ways; in particular, modern physics, in the shape of the quantum and relativity theories and the study of nonlinear dynamical systems, seems to undermine the classical foundations, which in the West have become enshrined as simple common sense. Troublesome questions are relegated to a brief discussion in Chapter 3, and for the remainder of the dissertation conventional views will be assumed.

2.1 Generalizing logic into probability

According to the interpretation of probability proposed by J.M. Keynes, R.T. Cox, and E.T. Jaynes, the domain of probability is the same as ordinary propositional logic.[1] Probability and propositional logic both operate on variables called "statements" or "propositions," which are generally interpreted as statements of fact. Each proposition A is associated with a function which maps A into the set $\{0, 1\}$, with 0 conventionally identified with certainty of the negation of A and 1 identified with certainty of A. Simple propositions are the primitives of probability and propositional logic, since complex propositions are formed by applying logic operations to two or more simple propositions. It can be shown [51, App. A] that all operations on propositions may be reduced to just a few: it turns out that all logic operations may be expressed in terms of conjunction and negation alone. Thus the task is to obtain probabilistic generalizations of conjunction and negation; given this, the probability of any complex proposition can be obtained (in principle, at least) by reducing it to simple propositions and applying the rules for

[1] One is tempted to call it the "logical" interpretation of probability, except that name has already been taken by a rather different theory devised by R. Carnap.

combining simple propositions by conjunction and negation.

This dissertation faithfully follows the exposition of Jaynes [41], who did much to popularize the 'probability as extended logic' theory. Jaynes' approach is based on the work of R.T. Cox [18, 19, 20], who in turn was inspired by J.M. Keynes [45]. The work of Jaynes cited here is highly recommended to the reader, as it contains a thorough discussion of fundamentals, interesting digressions on several topics, and a lengthy annotated bibliography. The numerous screeds directed against R.A. Fisher, the very prophet of frequentism, are the only drawback of Jaynes' work; at least his rants make for good entertainment.

In what follows, propositions will be written in the form $A|B$, which indicates a proposition A given that proposition B holds. Degrees of belief generally depend not only on a particular proposition A in question, but also on the sum-total of background information before which A is asserted. This implied background, which consists of all information pertaining to A, is sometimes denoted "&c." (that is, "et cetera"). A probability $\Pr(A|B)$ is called the conditional probability of A given B, and it is more carefully written as $\Pr(A|B, \&c.)$. An "unconditional" probability $\Pr(A)$ is one which is conditioned only on the implied background, and it is more carefully written as $\Pr(A|\&c.)$. This illustrates a fundamental tenet of our approach, that all degrees of belief are conditional on *something*, which is restated more precisely as Desideratum 3(b) below.

The probability $\Pr(A|B, \&c.)$ is to be interpreted as the rational degree of belief (d.o.b.) of A given B and the background information. The d.o.b. of A is derived by fixed rules from d.o.b.'s of B and the background; in this sense the d.o.b. is "objective" and not personal — anyone, indeed a machine, starting from the same information will compute the same d.o.b. of A. Degree of *belief* suggests, perhaps too strongly, a personal, subjective element in the calculations; Jaynes wisely phrases the search for the principles of reasoning in terms of the operating principles of a robot, which presumably introduces no personal element into the calculations. However, the d.o.b.'s of elementary propositions, such as $\Pr(B|\&c.)$ in the present example, are not computed

by the laws of probability and enter the calculations through some extra-probabilistic considerations; see §2.3. The conclusion of a probability calculation may be interpreted as, "If you hold the stated d.o.b.'s for the elementary propositions in this problem, and you accept the desiderata listed below, then your rational d.o.b. of the proposition in question has the computed value." Thus the laws of probability can be seen as a mechanism for expressing the hidden implications of one's assumptions, embodied in d.o.b.'s and desiderata; one can hardly hope for more than this.

Jaynes begins the search for a generalization of propositional logic with the following set of desired characteristics of the resulting system, which he calls "desiderata." These desiderata are sufficient to ensure that there is essentially only one extension of propositional logic satisfying these criteria.

1. Degrees of belief are represented by real numbers.

2. The reasoning system should have qualitative correspondence with common sense and propositional logic:

   2(a). The degree of belief of a conjunction $(A \wedge B)|C$ is a function of the degrees of belief of $A|(B \wedge C)$ and of $B|C$, or a function of the degrees of belief of $B|(A \wedge C)$ and of $A|C$.

   2(b). The degree of belief of a negation $\neg A|B$ is a function of the degree of belief of $A|B$ alone.

   2(c). Greater degree of belief is represented by a greater number.

   2(d). The rules governing degree of belief must hold under all numerical assignments of the d.o.b.'s of the elementary propositions of the problem.

3. Consistency:

   3(a). If a conclusion can be reasoned out in more than one way, all ways must lead to the same result.

3(b). The reasoning system takes into account all information relevant to a question.

3(c). Equivalent states of knowledge are represented by equivalent degrees of belief.

This list of desiderata differs slightly from that given by Jaynes in that a few items which he did not state directly are included here. The importance of Desideratum 2(d) has been lately emphasized [77, 38], since counterexamples to the results cited below can be constructed if 2(d), which is sometimes termed a "density" assumption, is denied.

A consideration of Desideratum 2(a) leads us to conclude

$$F(F(x,y),z) = F(x,F(y,z)) \tag{2.1}$$

where F is the degree of belief function mentioned in 2(a), and x, y, and z are the degrees of belief of some propositions $A|(B \wedge C)$, $B|C$, and C, respectively; $F(x,y)$ is the degree of belief of the compound proposition $(A \wedge B)|C$, likewise $F(y,z)$ is the d.o.b. of $B \wedge C$, and both sides of Eq. 2.1 are equal to the d.o.b. of $A \wedge B \wedge C$. A solution of Eq. 2.1, which is called the "associativity equation" by Jaynes, may be found easily [18, 41] by assuming that F is twice differentiable, and in a more roundabout way [1] by not assuming differentiability. It transpires that the associativity equation is satisfied by

$$F(x,y) = w^{-1}(w(x)w(y)) \tag{2.2}$$

where w is any positive monotonic function of a real variable. A particular choice of $w(x) = x$ yields the law of conjunction which we find most familiar; a different choice would yield a law with a different form, but the same content.

Likewise, a consideration of Desideratum 2(b) leads to

$$S(S(x)) = x \tag{2.3}$$

where S is the degree of belief function mentioned in 2(b). Imposing the conventions that $S(0) = 1$ and $0 \leq x \leq 1$, we find

$$S(x) = (1 - x^m)^{1/m} \tag{2.4}$$

where m is a positive real number. Again, the particular choice $m = 1$ yields the law of negation which seems familiar to us, but the choice is only a matter of taste and convenience.

To summarize, Desiderata 2(a) and 2(b) (with other desiderata taken into account as necessary) yield the conjunction and negation formulas

$$\begin{aligned} \Pr(A \wedge B|C, \&c.) &= \Pr(A|B \wedge C, \&c.)\ \Pr(B|C, \&c.) \\ &= \Pr(B|A \wedge C, \&c.)\ \Pr(A|C, \&c.) \qquad (2.5) \\ \Pr(\neg A|\&c.) &= 1 - \Pr(A|\&c.) \qquad (2.6) \end{aligned}$$

with Pr denoting the degree of belief of a proposition. Using these two rules and the conjunctive normal form [51] of a complex proposition, we can express the degree of belief of any complex proposition in terms of products and differences of degrees of belief of simple propositions.[2]

2.1.1 *A useful elementary result: the disjunction rule*

A very useful result is to compute the probability of a disjunction. Since $A \vee B = \neg(\neg A \wedge \neg B)$, we have

$$\begin{aligned} \Pr(A \vee B) &= 1 - \Pr(\neg A \wedge \neg B) = 1 - \Pr(\neg A|\neg B)\ \Pr(\neg B) \\ &= 1 - (1 - \Pr(A|\neg B))\ \Pr(\neg B) \end{aligned}$$

At this point, note $\Pr(A|B)\ \Pr(B) = \Pr(A \wedge B) = \Pr(B|A)\ \Pr(A)$ for any two propositions A and B, so we have

$$\begin{aligned} \Pr(A \vee B) &= 1 - \Pr(\neg B) + \Pr(\neg B|A)\ \Pr(A) \\ &= 1 - (1 - \Pr(B)) + (1 - \Pr(B|A))\ \Pr(A) \\ &= \Pr(A) + \Pr(B) - \Pr(B|A)\ \Pr(A) \qquad (2.7) \end{aligned}$$

[2] "It is often said, and rightly, that reasons should not be counted, but weighed; however, no one has yet provided us with the scales on which to weigh the cogency of reasons," as Leibniz remarked in 1697. Would he accept Cox's rules as the appropriate scales?

In the special case that A and B are mutually exclusive, so $\Pr(B|A) = \Pr(A|B) = 0$, we have

$$\Pr(A \vee B) = \Pr(A) + \Pr(B) \tag{2.8}$$

It is hoped that the derivation of $\Pr(A \vee B)$ illustrates a useful result based on the conjunction and negation rules; in the interest of brevity, further demonstrations will be much condensed or omitted entirely.

2.2 Properties of joint, conditional, and marginal densities

In applications, the propositions of interest are often of the form "$X \in A$" where X is a variable which represents a numerical quantity in the problem domain, and A is a set of the values which X can take on. It is often convenient to work with *probability density* functions instead of the probability function itself, $\Pr(\text{“}X \in A\text{”})$. A probability density function p_X is the Radon-Nikodym derivative (w.r.t. the underlying measure, usually the Lebesgue measure on the reals or the uniform discrete measure on a subset of the integers) of the corresponding probability function $\Pr(\text{“}X \in A\text{”})$, so by definition it has the property

$$\Pr(\text{“}X \in A\text{”}) = \int_A p_X(a)\, da \tag{2.9}$$

interpreting the integration as a summation, if appropriate. Probability densities inherit the conjunction rule from probability functions: writing the joint density of X and Y as p_{XY}, we have

$$p_{XY}(a,b) = p_{X|Y}(a,b)\, p_Y(b) = p_{Y|X}(b,a)\, p_X(a) \tag{2.10}$$

where the conditional density $p_{X|Y}$ is defined as the Radon-Nikodym derivative of the conditional measure

$$\lim_{\Delta b \to 0} \Pr(\text{“}X \in A\text{”} | \text{“}Y \in B(b, \Delta b)\text{”}) \tag{2.11}$$

(denoting the open ball centered on b with radius Δb as $B(b, \Delta b)$), and likewise for $p_{Y|X}$. Densities also have the extremely useful *marginalization* property,

$$p_X(a) = \int p_{XY}(a,b)\, db, \qquad p_Y(b) = \int p_{XY}(a,b)\, da \tag{2.12}$$

again interpreting the integration as a summation, if appropriate. In the equations in this section, X and Y may be one-dimensional or many-dimensional. The conjunction and marginalization rules are not difficult to prove, but require some care in the handling of conditional densities; see Ref. [62] for an elementary treatment. These properties of probability densities will be used very extensively in Chapter 5.

2.3 Assigning numerical values to probability distributions

We now have rules (Eqs. 2.5 and 2.6) for combining degrees of belief. This brings us to the question of how numerical values are assigned to degrees of belief of elementary propositions. Strictly speaking, this is an extra-probabilistic problem; when we invoke the machinery of probability in a problem, it is assumed that numerical values are already known for the elementary propositions in question. As to problem of determining some numbers so that the calculations can get started, formal methods are known only for the expression of complete ignorance, but so far only heuristics are available for the much more interesting and complex problem of expressing partial knowledge. Let us consider the simpler problem of ignorance first.

2.3.1 *Numerical probabilities via exchangeable labels*

Let us suppose that we are interested in a variable which can take on a finite number of values, and these values are identified by labels which can be exchanged without changing our degrees of belief; this is the simplest case for which numerical probabilities can be derived. As an example, we might consider the probability of head or tail on the toss of a coin; it is customary to assume it doesn't matter which side is called head and which is called tail, so these labels are exchangeable. Likewise, with the throw of a die, it is usually assumed that the numbering of the sides is irrelevant, so that labels 1 through 6 are exchangeable. Exchangeability implies equal probability, through Desideratum 3(c) above; Jaynes [41] works through the details. Thus the probability of each proposition associated with an exchangeable label is just $1/N$, where N is the total number of labels, since all the probabilities are equal and sum to unity.

This argument is easily generalized to a common case involving non-exchangeable labels. Suppose the labels are non-exchangeable, but can be decomposed into more elementary labels which are exchangeable. For example, "die shows 1 or a prime number" and "die shows a composite number" are not exchangeable, but each may be decomposed into statements about the labels 1 through 6, and those labels are exchangeable. Now

$$\begin{array}{rcl} \text{die shows 1 or a prime} & \text{iff} & \text{die shows 1 or 2 or 3 or 5} \\ \text{die shows a composite} & \text{iff} & \text{die shows 4 or 6} \end{array}$$

Since only one label can appear at a time, the simple propositions "die shows 1," "die shows 2," etc. are mutually exclusive, and the special disjunction rule, Eq. 2.8, yields

$$\Pr(\text{die shows 1 or a prime}) = \frac{1}{6}+\frac{1}{6}+\frac{1}{6}+\frac{1}{6}=\frac{2}{3} \tag{2.13}$$

$$\Pr(\text{die shows a composite}) = \frac{1}{6}+\frac{1}{6}=\frac{1}{3} \tag{2.14}$$

A moment's thought shows that

> The probability of an event is the ratio of the number of cases favorable to it, to the number of all cases possible when nothing leads us to expect that any one of these cases should occur more than any other, which renders them, for us, equally possible.

as Laplace put it [23]; this is now called the classical definition of probability. Unfortunately, this definition has rather limited applicability — it works very well for discrete variables with exchangeable labels, and is not very useful for anything else. In particular, one runs into trouble with the definition of "exchangeable" in the case of a continuous variable, because the parametrization of the problem makes a difference in the results; it makes a difference whether one works with distance per time or time per distance, for example. See Ref. [12] for a discussion of the difficulties involved, and some indications of a solution.

It is important to note that probabilities calculated above are not based on the physical properties of the coin or die, but on the state of our knowledge about the coin

or die. Two labels are exchangeable *if nothing tells us they are not exchangeable;* thus exchangeability is an expression of ignorance. Ignorance is supplanted at any moment by additional information, which will lead us to compute some other probabilities. A detailed consideration of the mass removed by drilling dots into the die may lead us to revise slightly the probability of "die shows 1 or a prime number", but until that information is available, our degree of belief stands at 2/3.

2.3.2 *Heuristics for expression of partial knowledge*

The extremely important task of expressing or encoding partial knowledge in probability distributions is, unfortunately, only poorly understood. See Refs. [42, Ch. 3] and [13, Ch. 5] for an introduction to the art of constructing belief network models. Several ways in which partial knowledge is encoded in the structure of a belief network are described in this section.

Precise relation of child to parents. Suppose that we know a variable Y is a precise (i.e., noiseless) function F of some parents $X = (X_1, X_2, X_3, \ldots)$. To encode this in a belief network, we make Y the child of parents X, with a conditional distribution which is a delta function:

$$p_{Y|X}(y, x_1, x_2, x_3, \ldots) = \delta(y - F(x_1, x_2, x_3, \ldots)) \tag{2.15}$$

Formally, there is no problem with a delta function in the computation of π- and λ-messages described in Chapter 5, since the presence of the delta only simplifies some of the integrations.

Noisy relation of child to parents. With Y and $X = (X_1, X_2, X_3, \ldots)$ as before, suppose the relation F is distorted by noise. We can model a relation with additive noise ϵ as

$$p_{Y|X}(y, x_1, x_2, x_3, \ldots) = p_\epsilon(y - F(x_1, x_2, x_3, \ldots)) \tag{2.16}$$

with p_ϵ usually taken as a Gaussian distribution with mean zero, and we can model a relation with multiplicative noise η as

$$p_{Y|X}(y, x_1, x_2, x_3, \ldots) = p_\eta(y/F(x_1, x_2, x_3, \ldots)) \tag{2.17}$$

with p_η usually taken as a lognormal distribution, such that $\log \eta$ has mean zero. Other noise distributions are certainly possible.

Child is a noisy observation of parent. A child $\widehat{X}$ which represents a observation or measurement of a parent X is a special case of the preceding example. As before, one must decide what kind of noise affects the measurement. For additive noise, if the noise distribution is p_ϵ, the distribution of the child given the parent is $p_\epsilon(\widehat{X} - X)$. In the case of additive Gaussian noise, it is convenient to represent the child's distribution as a conditional Gaussian with multiplier 1, offset 0, and conditional variance equal to the variance of p_ϵ.

Child is a classification of parents. Sometimes diagnostic models are constructed with observable variables as the inputs and a discrete hypothesis variable as the output. Models of this type include generalized linear models [54] and squashing neural networks [71]. The output of classification models can be interpreted as a probability distribution over the possible values of the category hypothesis. A classification model interpreted as a conditional distribution appears in the belief network *monitor2* described in §4.4.

Conditional derived from a joint distribution. It is sometimes convenient to construct a model of a relation between two variables first as a joint distribution, and then derive a conditional distribution from that, especially in problems in which the conditional variance of the child variable may change depending on the parent's value, a condition called "heteroskedasticity." Deriving a conditional from a joint distribution is especially easy in problems based on measured empirical data, as in the model of temperature and relative humidity described in §6.7, in which the joint distribution was constructed as a mixture of Gaussians, and the conditional of relative humidity given temperature was derived using the results described in Appendix E.

The methods of model construction listed in this section are important ways of introducing knowledge about the problem structure into the belief network. There is a another broad aspect of knowledge representation, which concerns setting parameters

to represent prior information within a predetermined structure. For example, to say that the relation of a child to its parents is given by a neural network, is to specify a certain structure for the relation, and that specification rules out many alternatives. However, one can go farther, and say that the output of the neural network should be similar to another neural network, which implies the parameters of the new neural net should be similar to those of the old one. Quantifying the similarity between two sets of parameters is a little subtle, and it is fair to say that definitive solutions have not yet been found. However, the broad framework of Bayesian modeling, which takes prior information into account when finding model parameters, seems appropriate for this problem. Some aspects of this important and interesting topic are discussed in Refs. [53] and [56].

2.4 Shortcomings of probability

There are at least two important classes of problems for which probability alone is not enough. One such kind of problem concerns making decisions when there is more than one agent making decisions. The results of classical utility theory (see Refs. [59] and [16] for an introduction to this vast subject) apply only to a single decision maker — the presence of others complicates the situation greatly. Some progress on the topic of game theory (as multiple agent decision theory is called; see Ref. [82] for an introduction to this equally vast subject) has been made, but the easy result of single-agent decision theory, namely the best action is that which maximizes expected gain, no longer applies. For the purposes of this dissertation, we shall assume that decisions are made by a single agent, whose state of knowledge is represented by a probability distribution; this is enough to carry out predictions and diagnoses.

The other kind of problem not covered by probability alone concerns the representation and, especially, the inference of causation. It is well known that probability can represent association and correlation but not causation; and it is conventional to therefore leave causation aside without further discussion. However, causation is an extremely useful concept (ignoring physicists who tell us there is no such thing). A

promising new formal theory of causation has been proposed by J. Pearl [64].

Pearl's theory of causality is focused on the need for a formal algebra of causality. Although causation was banished from statistics by Karl Pearson early in the 20th century, there is still a need to reason carefully about interventions such as holding a variable fixed or randomizing an experiment with respect to a variable, but under the influence of Pearson such considerations are almost always carried out informally. Pearl uses a graphical theory to reason about causation, similar to the probability theory of belief networks but enriched with additional concepts. It is certainly true that probability alone is not sufficient to discuss causation properly, but instead of ignoring the topic altogether, we need to devise the formal tools which will allow us to reason correctly — this is Pearl's motivation.

In engineering applications, automatically inferring cause and effect from raw data would be very useful, but, sadly, there is not room in the present work to explore this extremely interesting topic.

2.5 Conventions of notation

Before moving on to graphical probability models, let us take a moment to review the notations which were introduced in this chapter, and list others that are used in the remainder of this dissertation.

1. A variable is a place-holder in an equation or other relation. That is, a variable is defined only by its relations with other variables — this is the way variables work in logic and mathematics.

2. There is no distinction between a variable and an instance of that variable, as no distinction is drawn in ordinary scientific and mathematical problems. For example, if I say "T is temperature," then the proposition "$T > 100$" is well-formed.

3. "$\Pr(A|B, \&c.)$" is read as "probability of A given B and implied background,"

and often abbreviated as "$\Pr(A|B)$."

4. We write p_X for the probability density of the probability function $\Pr(\text{“}X \in S\text{”})$.

5. The notation $X \,|\, Y$, pronounced "X given Y," indicates that any distribution function or probability density associated with the variable X will be a function of the variable Y.

6. "$X \sim N(\mu, \sigma^2)$" is a shorthand for

$$p_X(u) = 1/(\sigma\sqrt{2\pi}) \exp(-(1/2)(u-\mu)^2/\sigma^2)$$

"$X \,|\, Y \sim N(F(Y), \sigma^2)$" is a shorthand for

$$p_{X|Y}(u, v) = 1/(\sigma\sqrt{2\pi}) \exp(-(1/2)(u-F(v))^2/\sigma^2)$$

where F is some function.

7. "$g(X; \mu, \sigma)$" is a shorthand for

$$g : X, \mu, \sigma \mapsto 1/(\sigma\sqrt{2\pi}) \exp(-(1/2)(X-\mu)^2/\sigma^2)$$

That is, g is the density function of a variable X such that $X \sim N(\mu, \sigma^2)$.

8. p_X, p_Y, $p_{X|Y}$, p_{XY} are all functions; $p_X(0)$, $p_Y(50)$, $p_{X|Y}(10, 28)$, $p_{X,Y}(17, 45)$ are numbers.

I have tried to use a notation which is as general and unambiguous as possible; the reader may judge whether I have succeeded.

2.6 Elements of graphical probability models

Let us establish some background material on the graphical properties of belief networks. Refs. [63] and [13] contain comprehensive expositions on these properties. A belief network, for our present purposes, is a collection of conditional probability functions

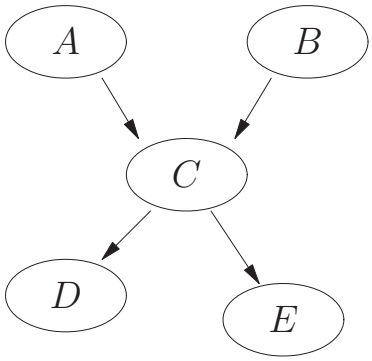

Figure 2.1: An illustration of graphical model terminology. The probability model represented by this figure is $p_{ABCDE} = p_{D|C}\, p_{E|C}\, p_{C|AB}\, p_A\, p_B$. Nodes A and B have no parents; the parents of C are A and B; the parent of D is C; and the parent of E is C. C is the child of A and B; D and E are the children of C; and D and E have no children. Note that the assignment of arrows leading into each node in the graph corresponds exactly to the list of variables on the right-hand side of each conditional probability in the model. Parents are conventionally placed above children on the page, but the direction of the arrows is what matters.

associated with a directed graph. (Sometimes the definition of belief network is extended to include undirected graphs, or graphs with both directed and undirected edges, but these extensions aren't needed for this dissertation.) Each variable X in the probability model is associated with a node of the same name in the graph. The parents of a variable X are the variables that appear on the right-hand side of the vertical bar in a conditional probability. The graph has an arrow (i.e., a directed edge) leading from each parent node into X. If U is a parent of X, then we say that X is a child of U. Figure 2.1 illustrates the terms introduced in this paragraph.

A node is said to be an evidence node if the corresponding variable in the probability model has a known value. The set of all evidence nodes is conventionally denoted $\mathbf{e}$. Any node can be an evidence node, not just those corresponding to sensor observations or other kinds of measurements.

2.6.1 *A graphical criterion for independence*

Two variables X and Y are said to be independent if their joint distribution is equal to the product of their marginal distributions,

$$p_{XY}(a,b) = p_X(a)\, p_Y(b) \tag{2.18}$$

Whether two variables in a belief network are independent (a mathematical notion) can be determined by studying only the graph of the belief network. If two nodes in the graph have a property called "d-separation," then the corresponding variables are independent. However, if the two nodes are not d-separated, they might or might not be dependent. It turns out that it is more important to establish independence, since if independence is assessed incorrectly the calculations of posterior probabilities will be incorrect. On the other hand, incorrectly assuming dependence leads only to unnecessary computations.

Before stating a graphical criterion for independence, we need a few more technical terms. A directed path from some node A to another, B, is a sequence of arrows such that A is at the tail of the first arrow, the head of each arrow meets the tail of the one following, and B is at the head of the last arrow. An undirected path between A and B is a sequence of edges (i.e., ignoring the direction of arrows) such that A is at one end of the first edge, each succeeding edge is joined to the one previous, and B is at one end of the last edge. B is a descendent of A if there is a directed path from A to B, in which case A is an ancestor of B.

Consider an undirected path between A and B. A node X (other than A or B) in the path is called a converging node if its predecessor and successor are both parents of X. It is called a diverging node if its predecessor and successor are both children of X. Finally, it is called a linear node if the predecessor is a parent and the successor is a child, or vice versa. Figure 2.2 shows a belief network and some paths within it, and classifies the nodes in those paths.

We can now state, in graphical terms, a criterion for the independence of two variables. See especially Ref. [14] for a comprehensible treatment of this topic.

> *(The d-separation criterion.)* Two nodes A and B are d-connected w.r.t. evidence $\mathbf{e}$ if there is an undirected path between them such that every converging node is in $\mathbf{e}$ or has a descendent in $\mathbf{e}$, and all diverging and linear nodes are not in $\mathbf{e}$. If A and B are not d-connected, they are d-separated. If they are d-separated, A and B are independent w.r.t. $\mathbf{e}$.

The d-separation criterion is illustrated in Figure 2.2. It is not trivial to prove that the graphical d-separation criterion implies the mathematical independence property; see Ref. [63] for a proof.

If $X \in \mathbf{e}$ is d-connected to A through one of the parents of A, X is said to be "upstream" evidence w.r.t. A; if X is d-connected through one of the children of A, X is said to be "downstream" evidence w.r.t. A.

Note that adding a converging node X to the evidence $\mathbf{e}$ or adding a descendent of X to $\mathbf{e}$ makes the ancestors of X d-connected. On the other hand, adding a diverging node Y to $\mathbf{e}$ makes the descendents of Y d-separated (if there are no paths connecting those descendents except through Y), and adding a linear node Z to $\mathbf{e}$ makes the ancestors of Z d-separated from the descendents of Z (again assuming there are no paths except through Z). Thus whether two variables A and B are independent is not a static property of the probability model, but it changes with the evidence $\mathbf{e}$. RISO uses d-separation to determine which quantities (in particular, posterior distributions) need to be recomputed when some variable X is added to $\mathbf{e}$ or removed from $\mathbf{e}$. Every calculation associated with a variable which is d-connected to X becomes stale, and must be recomputed, taking the change in the evidence into account. According to the general policy of laziness in RISO, stale calculations are erased, but not recomputed until there is a request for those calculations; the pertinent details are described in Appendix B.

2.7 Probabilistic interpretation of prediction and diagnosis

Part of the power of a probabilistic approach to modeling is that several important and interesting operations can be defined as the calculation of probability distributions. Thus one can define a single probabilistic model and examine it in different ways to

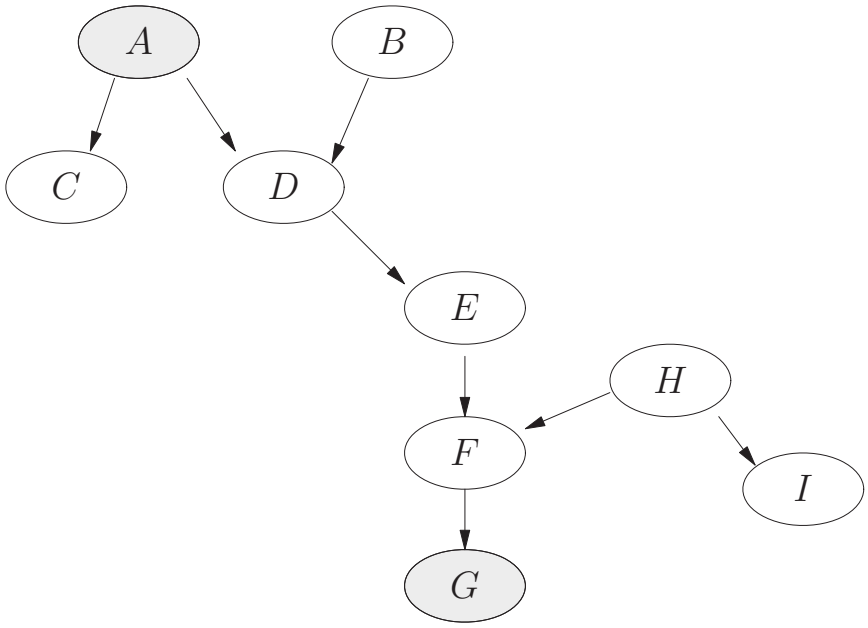

Figure 2.2: Two illustrations of the *d*-separation criterion. Evidence nodes are shaded: $\mathbf{e} = \{A, G\}$. (i) The undirected path $B \rightarrow D \rightarrow E \rightarrow F \leftarrow H \rightarrow I$ joins B and I. F is converging; H is diverging; D and E are linear. B and I are *d*-connected by this path, therefore they are not *d*-separated w.r.t. the evidence $\mathbf{e}$. (ii) The undirected path $C \leftarrow A \rightarrow D \rightarrow E \rightarrow F \leftarrow H$ joins C and H. F is converging; A is diverging; D and E are linear. A is a diverging node which is in $\mathbf{e}$, therefore this path does not *d*-connect C and H; there are no other undirected paths between C and H, therefore C and H are *d*-separated, therefore C and H are independent w.r.t. the evidence $\mathbf{e}$.

perform prediction and diagnosis, calculate value of information, and carry out function inversion, revision of hypotheses, "what if?" reasoning, and rule out hypotheses; see Ref. [42] for a lengthy discussion of belief network operations. In this section, these operations will be defined and illustrated with some trivial examples, and references to substantial examples elsewhere in this dissertation are given.

2.7.1 *Revision of hypotheses*

Suppose we have only one evidence variable X in a belief network. We dutifully compute the posterior of an hypothesis variable H given X, $p_{H|X}$. On the basis of the evidence $\mathbf{e} = \{X\}$, H has the distribution $p_{H|X}$. When additional evidence Y is observed, our beliefs about H become out-of-date. We should now base our belief about H on the combined evidence $\mathbf{e} = \{X, Y\}$. This is no problem: we simply compute the posterior $p_{H|X,Y}$. We can repeat this process as often as we want, sometimes adding variables to $\mathbf{e}$, sometimes taking them away. At each step, the newly-computed posterior $p_{H|\mathbf{e}}$ states our belief about H on the basis of the evidence $\mathbf{e}$. The algorithm for computing posterior distributions is described in detail in Chapter 5.

The revision of hypotheses with the increase or decrease of available evidence is a fundamental operation on a belief network. In some cases, the arrival of new evidence requires that all previous results be invalidated, while in others, partial results based on previous evidence are not affected by new evidence. To be more precise, a result is invalid if it is associated with a variable which is not d-separated (§2.6.1) from the variable which is new evidence or which is evidence no longer. Whether or not a partial result is invalid can be determined automatically by d-separation, so that every posterior computed in the belief network correctly takes into account the current set of evidence variables.

2.7.2 *Prediction and "What if?" scenarios*

Probability models are usually constructed with variables which are considered "causal" or "exogenous" as the parents for other variables which are considered "effects" or

"endogeneous." Let us name the parent variables generically as U, and the children as X. The model is specified generically as $p_{UX} = p_{X|U}\, p_U$; there may be any number of other variables in the model, but the essential point is that the causal variables are upstream from the effects variables.

We understand *prediction* as the computation of a posterior for a downstream variable given upstream evidence. If the belief network expresses a temporal structure, with nodes for variables in the past and future, the evidence may include stated values for past variables, and the posterior may be computed for future variables. The posterior can also be computed for a past variable given observed values in the present, in which case the posterior might be called a 'retrodiction.'

The computation of an estimated yearly energy use could be computed by a belief network with nodes for, say, hourly temperature, time of day, architectural parameters, and other influential variables. A calculation using actual values yields an estimate of actual energy use, but there's nothing to stop us from modifying some of the parameters and recomputing the posterior over annual energy use. Tinkering with the parameters while constructing or calibrating a model is a time-honored practice, and it is supported in belief network computations as the calculation of posterior distributions conditional on the tinkered-with variables.

2.7.3 *Diagnosis and ruling out hypotheses*

For our purposes, we will define *diagnosis* as the computation of the posterior distribution of a discrete variable given downstream evidence. In many applications, discrete upstream variables will represent hypotheses about equipment status or mode of operation, and downstream variables will represent observables such as temperatures, pressures, electrical demand, etc. Models for such systems are typically specified with the observables (say X) as children of the hypothesis variables (say H), and the goal is to compute $p_{H|X}$.

It is typically the case that the hypothesis variable H has several possible values, which correspond to a set of mutually exclusive hypotheses, for example, normal oper-

ation of a chiller and several mutually exclusive failure modes. The posterior $p_{H|X}$ is a list of numbers which tells the probability for each of the possible hypotheses; generally if the probability of one or more failure modes is large, we will need to take some kind of action. Note that there is no separate "detection" calculation which precedes the computation of the the probability of each kind of failure; the probability of one or more failures of any kind is just $\sum_k \Pr(\text{"failure type } k\text{"})$, so in order to calculate an aggregated failure probability, it is necessary to carry out the whole calculation of $p_{H|X}$, which obviates the need for detection.

It is worth noting in this context that the machinery developed in Chapter 5 is devoted largely to automatically computing extensions to Bayes' rule. For example, if a model is specified as $p_{HX} = p_{X|H}\, p_H$, we can obtain the posterior for the hypothesis variable via Bayes' rule,

$$p_{H|X}(h,x) = \frac{p_{X|H}(x,h)\, p_H(h)}{\sum_{h'} p_{X|H}(x,h')\, p_H(h')} \tag{2.19}$$

Now suppose that our model contains three variables X, Y, and H, with $p_{XYH} = p_{Y|X}\, p_{X|H}\, p_H$. Suppose we've observed Y but not X. The appropriate inference rule is now

$$p_{H|Y}(h,y) = \frac{p_H(h) \int p_{Y|X}(y,x)\, p_{X|H}(x,h)\, dx}{\sum_{h'} p_H(h') \int p_{Y|X}(y,x)\, p_{X|H}(x,h')\, dx} \tag{2.20}$$

which is not too difficult to derive; but if the model is, say, $p_{X_1X_2Y_1Y_2Y_3H} = p_{Y_1Y_2|X_1}\, p_{X_1|H}\, p_{H|X_2}\, p_{X_2|Y_2}$, and we are asked to find the posterior of H given Y_1, Y_2, and Y_3, it's not immediately obvious the inference rule should be

$$p_{H|Y_1Y_2Y_3}(h,y_1,y_2,y_3) = \frac{\int p_{Y_1Y_2|X_1}(y_1,y_2,x_1)\, p_{X_1|H}(x_1,h)\, dx_1 \cdot \int p_{H|X_2}(h,x_2)\, p_{X_2|Y_3}(x_2,y_3)\, dx_2}{\sum_{h'} \int p_{Y_1,Y_2|X_1}(y_1,y_2,x_1)\, p_{X_1|H}(x_1,h')\, dx_1 \cdot \int p_{H|X_2}(h',x_2)\, p_{X_2|Y_3}(x_2,y_3)\, dx_2} \tag{2.21}$$

and, except to promote mental dexterity, it is not a very enlightening calculation; as with breaking rocks, it is better to have a machine do it. Figure 2.3 shows belief networks for the models to which this paragraph refers.

We may know, through a site inspection or by other means, that some hypotheses are ruled out, that is, impossible. We can express this information with a prior p_H which

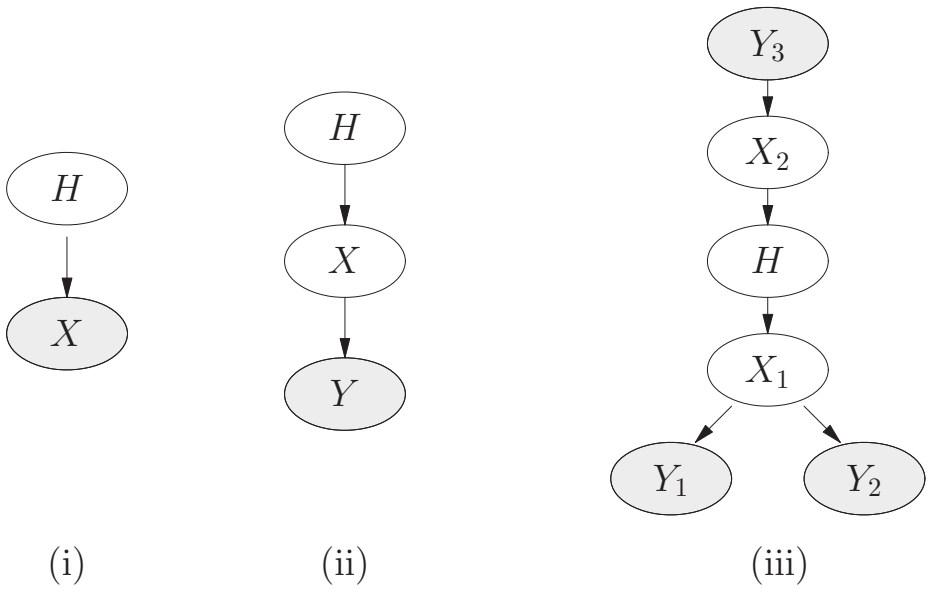

Figure 2.3: Belief networks to illustrate generalizations of Bayes' rule. Evidence nodes are shaded. (i) $p_{H,X} = p_{X|H}\, p_H$. (ii) $p_{H,X,Y} = p_{Y|X}\, p_{X|H}\, p_H$. (iii) $p_{X_1,X_2,Y_1,Y_2,Y_3,H} = p_{Y_1,Y_2|X_1}\, p_{X_1|H}\, p_{H|X_2}\, p_{X_2|Y_2}$.

puts zero mass on the hypotheses which are impossible, and compute the posterior as usual. The remaining hypotheses which are not ruled out will be assessed greater probabilities, since $\sum_{h'} p_{H|\mathbf{e}}(h')$ must still be 1. Furthermore, H may interact with other hypothesis variables elsewhere in the system through observable variables, so a prior p_H which rules out some hypotheses will affect degrees of belief for other hypothesis variables, as one expects. In a sense, ruling out hypotheses is a kind of "negative evidence," since it does not tell what the hypothesis is, only what it is not. The laws of probability can handle this case with ease.

2.7.4 *Function inversion*

A special case of diagnosis which is arises in many engineering problems is determining inverse functions — more precisely, determining preimages of functions. Generally, one has a function $Y = F(X_1, X_2, X_3, \ldots)$, and given Y and some of the X_k's, one wishes to compute distributions over the remaining parent variables. The relation between Y and the X_k's may be deterministic (i.e., if all the parents are known exactly, then Y can be calculated exactly) or stochastic (i.e., the function F is distorted by random noise). Examples of deterministic relations include thermodynamic laws, and examples of stochastic relations include many empirical laws such as regressions of heat transfer coefficients on Reynolds and Nusselt numbers. But even if a relation is deterministic, a distribution over some parent might have considerable dispersion, due to dispersion in the distributions of the other variables to which it is related. For example, the ideal gas law states $pV = nRT$. If any two of p, V, and T are known, the third can be determined exactly, but in the presence of measurement uncertainty in the two, we will find a distribution over the third instead of obtaining an exact value.

An example of a belief network model which can be used for function inversion is described in §8.1. In that model, a function relating water and air temperatures in a heating coil can be inverted to yield an inferred distribution over the heat transfer coefficient. There are several parent variables, but they need not all be measured. In fact, if some measurements are missing, we can still compute a distribution over the

heat transfer coefficient, although this will have a larger variance than if an additional measurement were available; this graceful degradation of the result is a feature of the use of probability.

2.7.5 *Value of information*

It is obvious that information has value, since with more information we can make better decisions (usually). However, the value of information is difficult to quantify. Perhaps the most principled solution is to define a quantity which represents the value of information with respect to decisions and outcomes in a specific problem; this approach is proposed by Arrow [5]. Such a problem-specific index is difficult to construct, in large part because the economic values of alternative decisions are difficult to asses. Typically one retreats from a problem-specific formulation to a general index of the value of information, which is precisely the approach taken in this dissertation.

Problem-independent indices of the value of information are often formulated in terms of entropy or other information theory quantities. The general idea is that information is equivalent to a change in a probability distribution, that is, a change in degrees of belief. Two indices of this type are employed in this dissertation. The first is the *Kullback-Leibler divergence* [50, §1.3], $KL(p, q)$, which measures the difference between two probability distributions p and q over the same variable,

$$KL(p, q) = \int p(x) \log \frac{p(x)}{q(x)} \, dx \tag{2.22}$$

The Kullback-Leibler divergence can be interpreted as an index of the informativeness of an observation: to assess the value of a specific observation of, say, $Y = y$, compute $KL(p_{X|Y=y}, p_X)$, where $p_{X|Y=y}$ is the posterior of X conditioned on the observation of Y, and p_X is the posterior of X omitting the observation. Heuristically, an observation of Y which results in a large change in the posterior $p_{X|Y=y}$ yields a large $KL(p_{X|Y=y}, p_X)$, and is interpreted as a 'very informative' datum. The technical interpretation is given in terms of the degree by which X can be compressed if Y has the observed value.[3]

[3] If one is designing communications networks, then compression is indeed an economic criterion, but

The units of KL, and likewise the units of MI (discussed in the next paragraph), are called 'bits' if the logarithm in Eq. 2.22 is the base-2 logarithm, and 'nats' or 'nits' if the logarithm is the natural logarithm.

The quantity $KL(p_{X|Y=y}, p_X)$, which is the informativeness of a particular observation, can be averaged over possible values of Y to obtain a second index, an index of the general informativeness of Y w.r.t. X. It is interesting to note that averaging $KL(p_{X|Y=y}, p_X)$ w.r.t. the distribution p_Y yields the mutual information of X and Y:

$$\begin{aligned} \int KL(p_{X|Y=y}, p_X)\, p_Y(y)\, dy &= \int \int p_{X|Y}(x,y)\, p_Y(y)\, \log \frac{p_{X|Y}(x,y)}{p_X(x)}\, dx\, dy \\ &= \int \int p_{XY}(x,y)\, \log \frac{p_{X|Y}(x,y)}{p_X(x)}\, dx\, dy \\ &= MI(X,Y) \end{aligned} \tag{2.23}$$

Note further that while the Kullback-Leibler divergence is asymmetric, $KL(p,q) \neq KL(q,p)$, the mutual information is symmetric, $MI(X,Y) = MI(Y,X)$. In the absence of an observation — say, before making an observation — mutual information gives an average informativeness of one variable on another. This can be used to determine which variable to observe next, often meaning which experiment to carry out; further discussion of mutual information will be found in §3.4, and some minor results which are used elsewhere in this dissertation are established in Appendix F.

2.8 Probability is the glue that holds the world together

It is common in engineering problems to have several sources of information which should be accounted for, merged, and reconciled. Probability models naturally integrate data in several ways — this is one of the strongest arguments for using such models. We considered in §2.7 several operations, which are essentially different ways of looking at the same thing; let us now review some ways of composing or integrating different things into a unified whole. We will examine how probability accounts for different

its applicability in other contexts is questionable; the reader will forgive me for blandly refraining from this question. But see the quotations from Ernst Mach on the philosophy of science in Chapter 3.

sources of information, and each of these connective functions will find use in some model elsewhere in this dissertation.

2.8.1 *Express relation of one variable to others with conditional probability*

The relationships between a large number of variables can be decomposed into conditional probability models which tell the relation of each variable to a subset of the other variables. Usually such local models are easier to describe than a global model containing all the variables. However, a decomposition into conditional probability models preserves all the properties of the joint distribution of all variables — the joint distribution is immediately recovered by taking the product of all the conditional distributions.

A wide variety of relations can be expressed as conditional distributions, such as thermodynamic relations and other constraints, relations determined from regression, classification, and expert knowledge in the form of rules of thumb. Some of these relations were discussed in §2.3.2.

2.8.2 *Treat uncertainty in measurements, parameters, and hypotheses symmetrically*

Under the laws of probability, all uncertain propositions are handled in the same way. The origin of the uncertainty is immaterial, so uncertainty arising from sensor noise, uncertainty about governing parameters, and uncertainty in hypotheses can all be treated in the same way. Different rules for different kinds of uncertainty need not be derived.

The uniform manner in which rational degree of belief handles all forms of uncertainty is in stark contrast to some other interpretations of probability. For example, in the frequentist theory, a probability is a long-term frequency, which is physical property of a system — thus there cannot be a probability distribution over a parameter or hypothesis. We are still obliged to take non-frequency uncertainties into account, so various kludges have been invented; for example, "confidence" in an hypothesis is an attempt at handling a non-frequency uncertainty within the frequentist paradigm. There is no need for adhockeries, however, since rational d.o.b. can be assessed for any uncertain proposition.

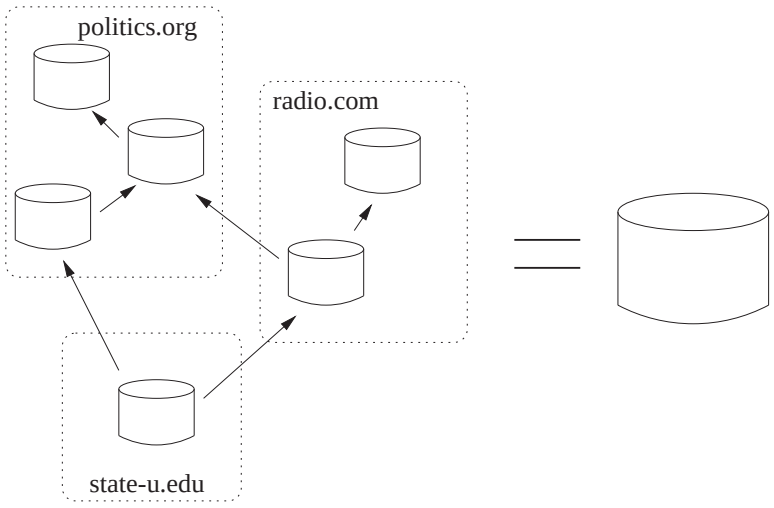

Figure 2.4: A distributed belief network, comprising several local belief networks, is a distributed database.

2.8.3 *"Think locally, compute globally"*

A decomposition into local probability models makes it natural to model functionally or geographically distributed systems with belief networks. Models for the functional or geographic units can be developed separately and perhaps by different people, and then the units joined by conditional distributions which have a variable in one unit conditioned on variables in another unit or units. Since each unit is a belief network, the result is again a belief network, distributed over two or more processors (Figure 2.4). The analysis which works at the level of the unit belief network immediately scales up to handle the distributed belief network — new rules of inference need not be derived. If we consider a belief network as a specialized database, a distributed belief network amounts to a distributed database, with rules for organization, query, and update given by the laws of probability.

2.8.4 *Laws of probability permit incremental development*

Distributed belief networks support "afterthought" analysis — it is easy to construct a new belief network which refers to nodes in one or many existing belief networks and makes use of the results already computed within those other networks, for some purpose not originally foreseen. For example, there could be a number of belief networks at various sites representing the HVAC systems of each site, and a research center might publish total yearly energy use for each site, or categorize the faults observed at all the sites. The summarizing belief network can be created without requiring that the other networks to which it is connected be restarted, and the new network can also be destroyed without affecting the others.

A local probability model in a belief network remains valid even when removed from that belief network and plugged into some other belief network, because the interactions between different parts of a belief network are not determined by its specification; rather, interactions are computed at run-time. Thus one may simply copy a local model from one belief network to another; this greatly speeds the development of new belief networks.

2.9 A glance forward

Several topics were covered in this chapter. In §§ 2.1–2.5 the formal properties of probability were described, and in §§ 2.6–2.8 the use of probabilistic models — that is, the connection between the formal world of probability and the "real world" in which we need to solve problems — was introduced. The remainder of this dissertation is largely devoted to describing general aspects of probabilistic models (Chapters 4–5) and some particular applications of such models (Chapters 6–8), but before we move on to practicalities, let us make a slight detour to explore the origins of the concept of probability, which plays so large a role in this work.

Chapter 3

ON THE CONCEPT OF PROBABILITY

3.1 Historical remarks

The concept of probability has a long and fascinating history. Let us review very briefly some aspects of that history, relating especially to the evolution of the 'rational degree of belief' interpretation of probability which is the basis of this dissertation, and concerning also the nature of scientific induction. The interested reader is referred to accounts of the history of probability [37, 78] and the philosophy of science [8, 73] for more details on these topics; see also the excellent bibliographies and historical notes in Refs. [41] and [25].

Probability is best understood as a branch of the theory of epistemology, that is, that branch of philosophy which studies knowledge. The first advances that resulted in recognizably modern doctrines were made by Skeptic philosophers of the school of Carneades (ca. 214–129 B.C.), who in that era was the head of the Academy in Athens. Skepticism was first formulated by Pyrrho (ca. 360–272 B.C.), who held that all appearances are deception, and all conclusions baseless; a wise man will suspend judgement about everything. The origins of Pyrrho's opinions are obscure, but some see the influence of Indian philosophers, whom Pyrrho seems to have met when he traveled to India with the court of Alexander the Great. Skeptical views had been expressed in Greek philosophy before Pyrrho's time — Xenophanes (born ca. 580 B.C.) remarked, "For even if one chances for the most part to say what is true, still he would not know; but every one thinks he knows" [31, Fragment 14] — but Pyrrho was the first to propose systematic skepticism.

The philosophy of the Skeptics was mostly negative, but Carneades and his followers did begin a positive development, which unfortunately was not completed for the better

part of two millenia. Though knowledge of truth escapes us, Carneades taught, we still act according to the strength of our opinions, of which he recognized four degrees: "(i) implausible; (ii) plausible (i.e. appear true 'to an intense degree'); (iii) irreversible (i.e. plausible and confirmed by other impressions); and (iv) tested (i.e. irreversible and tested by the scrutiny of surrounding circumstances)" [36]. These degrees of plausibility were illustrated by Carneades with the example of a rope on the floor of a darkened house, which appears to be a snake. The idea that action should be contingent on strength of belief is clearly allied with modern probability and decision theory, but Carneades' successors seem to have not further developed his doctrine of plausibilty. It is also noteworthy that Carneades did not identify plausibility with being nearer to truth, but only with strength of subjective opinion [36]. This, again, is compatible with modern interpretations of probability.

Not until the 17th century was the study of probability revived. In the intervening centuries doubtless many dice were thrown, and many ropes mistaken for snakes, but none thought to analyze these phenomena. But the Chevalier de Méré, fond of losing other people's money at the gambling table like so many aristocrats, had the good fortune to pose some questions to his acquaintance Blaise Pascal. There followed a correspondence (1654) between the latter and Pierre Fermat, which was the basis for the first systematic treatise on probability, *De ratiociniis in ludo aleae* by Christiaan Huygens.[1] Around the same time the study of social phenomena began, of which perhaps the earliest example is John Graunt's study of mortality (1662); the study of insurance and annuities originated about the same time. These developments were followed some decades later by the treatises of Jakob Bernoulli (*Ars conjectandi*, 1714) and Abraham de Moivre (*Doctrine of Chances*, 1718), which established many of the now-familiar algebraic results of probability, especially relating to the binomial distribution.

The mathematical theory of probability continued to develop throughout the 18th

[1] The Latin edition appeared in 1657, but it was followed just three years later by an edition in the vernacular, *Van Rekeningh in Spelen van Geluck*, which suggests the book had practical value: the title translates "On reasoning in games of chance."

century, as did the much messier study of social phenomena. It was noted early on that the proportions of male and female children born are not equal, with male children exceeding female children by a few percent. On this evidence, John Arbuthnot published "An Argument for Divine Providence, taken from the constant Regularities observ'd in the Births of both Sexes" in 1710. (Jaynes cites this as the first rejection of a null hypothesis, namely that the birth ratio is 1:1, on the basis of observations which are unlikely under the null hypothesis.) The question of the birth ratio was still of interest in the early 19th century, when the Marquis de Laplace asked the French government to carry out a careful census of male and female births.[2] Laplace also asked Alexander von Humboldt to investigate the birth ratio in the New World; Humboldt confirmed that male births exceed female births in Mexico as in France. The investigations of Humboldt and Laplace are described in the *Essai philosophique sur les probabilités* by the latter [25]. Also in relation to social questions, in 1738 Daniel Bernoulli introduced the concept of *moral expectation*, which is what is now called expected utility. The adjective 'moral' refers to society or custom, and Bernoulli's work was the beginning of modern decision theory.

In the middle of the 18th century, an obscure cleric published a brief paper which has since come to be seen as the foundation of the Bayesian interpretation of probability, which is essentially the interpretation adopted in this dissertation. The cleric, of course, was Thomas Bayes, and his paper was "An Essay towards solving a Problem in the Doctrine of Chances" (1764), which presented a solution of the so-called inverse probability problem. Given that one has observed a certain number of heads or tails of a coin toss, what is the probability of observing a head on the next toss? The paper itself is quite limited in scope, and it is questionable whether Bayes himself subscribed to the convoluted philosophy which characterizes the modern Bayesian theory; the reader can doubtless name at least two other movements, known by their founders' names, which

[2] The census was taken in 30 departments in the years 1800, 1801, and 1802. It is curious that the government had the time and money to spend on Laplace's research project while carrying on with Napoleon's wars.

would likewise be unrecognizable to the one to whom they are nominally most indebted.

Of all the philosophers and mathematicians before the 19th century, Gottfried von Leibniz came closest to expressing the concept of probability as extended logic. Perhaps this need not surprise us; Leibniz wrote voluminously on many topics, and was not much concerned to make the whole of his work consistent, so whatever your point of view, there may well be support for it in Leibniz.[3] Be that as it may, let us review the opinions of Leibniz, which very succinctly capture the spirit of the degree of belief theory.

Leibniz recognized, like Carneades, that we must often work with incomplete information. Except for phenomena which we witness directly, our knowledge is based entirely on opinion at lesser or greater remove from the facts. We may be very sure of our opinions, and they may be shared by many people, but they remain opinions nonetheless. The following is found in Leibniz' commentary [79] on Locke's *Essay on human understanding*.

> Perhaps *opinion*, based on likelihood [*vraisemblance*], also deserves the name of knowledge; otherwise nearly all historical knowledge will collapse, and a good deal more. But, without arguing about names, I maintain that *the study of the degrees of probability* would be very valuable and is still lacking, and that this is a serious shortcoming in our treatises on logic. For when one cannot absolutely settle a question one could still establish the degree of likelihood on the evidence, and so one can judge rationally which side is the most plausible.

Despite our knowledge being mostly limited to opinion, we may still hope to find a means of rational assessment of degrees of belief, and this is precisely the motivation behind the desiderata presented in §2.1.

In a famous passage, Leibniz speculates that reasoning may be as mechanizable as adding figures in a ledger. He dreamed of a *characteristica universalis* or 'universal symbolism,' a kind of algebraic writing system, which would be to ordinary human discourse as the notation of algebra is to mathematics. Apart from expressing ideas in a universal form, the *characteristica* would make it possible to easily detect errors

[3] For my part, I will conveniently ignore his monadology, and his theory that ours is the best of all possible worlds.

of reasoning, just as mathematical notation is a clearer and more precise expression of concepts that can also be expressed verbally. As Leibniz [72, p 592] envisions the scene,

> If controversies were to arise, there would be no more need of disputation between two philosophers than between two accountants. For it would suffice to take their pencils in their hands, to sit down at their slates, and to say to each other (with a friend as a witness, if they liked): Let us calculate.

This seems very close to the motivation for belief networks: once a problem is expressed in terms of probability (the *characteristica universalis* of the world of belief networks), any question can be answered as a simple calculation. The catch, of course, is in the construction of a belief network suitable for a given problem. Leibniz seems to have underestimated the effort required to simply restate a verbal problem in algebraic form. There is the further complication that calculations which are conceptually 'simple' may be intractable in practice, but new algorithms and faster computers should help us solve that problem.

Leibniz, though he contributed technical writings in several fields of mathematics and philosophy, was only a commentator in the field of probability. The development of probability through the 18th century followed the lead of de Moivre and Jakob Bernoulli, which had a much more restrictive scope than the expansive musings of Leibniz; dice, coins, balls in urns, etc., remained the standard examples of probabilitistic phenomena. The classical development culminated in the monumental *Théorie analytique des probabilités* of Laplace, of which his *Essai philosophique sur les probabilités* is a popularized summary. These works, which appeared early in the 19th century, were based on papers Laplace wrote during the later decades of the 18th century. To Laplace, the future is the necessary outcome of the past, the development being governed by deterministic laws; apparent randomness is due simply to ignorance of initial conditions. This view was stated [24, p 5] by Laplace in majestic and yet sublime rhetoric:

> Thus we should visualize the present state of the universe as the effect of its past state, and as the cause of that which is to follow. An intelligence which, at a given instant, comprehends all the forces by which Nature is animated, and the respective positions of the things which compose it — if it were so vast as to submit these

> data to analysis — would encompass in the same formula the movements of the largest bodies in the universe and the those of the lightest atom: nothing would be uncertain to it, and the future, like the past, would be present before its eyes.

Yet even if initial conditions are known with arbitrary precision, outcomes may become unpredictable even a short time into the future. This sensitive dependence on initial conditions is a hallmark of that property called 'chaos.' Henri Poincaré, who understood better than anyone in his time the implications of nonlinear dynamics, asked the pointed rhetorical question [66, p 69],

> Why is it that showers of rain, the storms themselves, seem to us to come at random, in such a way that so many people find it perfectly natural to pray for rain, but they think it laughable to pray for an eclipse?

In a chaotic system, even a supreme Intelligence might have trouble predicting the future. But the statistical regularity of a chaotic system might be enough to work with, as in the kinetic theory of gases.

The frequency interpretation of probability, which is at present the conventionally accepted definition, has had a relatively short history. The origin of the frequency definition dates to the mid-19th century, and is associated with John Venn, R.L. Ellis, and Antoine Cournot. In this definition, a probability is a long-term frequency, which is well-defined for repetitive physical events or ensembles of similar instances. Probabilities are not defined for any other kind of phenomenon. In the first half of the 20th century, the frequency interpretation was accepted and promoted whole-heartedly by R.A. Fisher, arguably the most influential figure in the history of statistics. Fisher used tests of a null hypothesis in a wide variety of problems with great success, and conventional textbook expositions of statistics contain many of his methods. As he was tremendously intelligent, he could always find a way to solve any particular problem he faced, and had little need of fundamentals. His emphasis on particular rather than general methods colors conventional statistics down to the present day.[4]

[4] There are those who have a none-too-exalted view of their own profession: "Statistics is defined as *the science of building and evaluating tools for data analysis.* The word 'tools' is chosen on purpose

Fisher's lack of interest in fundamentals is not shared by all frequentists. A statement of the frequentist conceptual landscape was given by von Mises [80], which is recommended for its clarity and concision. Richard von Mises accepted the application of probability only to a phenomenon which had the characteristics of a *collective*, which is essentially an ensemble of similar, random instances, and rejected all other uses. In practice, the kinds of phenomena which fulfill the definition of a collective are social phenomena, games of chance, and thermodynamic ensembles. He stated emphatically (p 9, op. cit.),

> Our probability theory has nothing to do with questions such as: 'Is there a probability of Germany being at some time in the future involved in a war with Liberia?'

But against the conceptual foundation of frequentism, Ernst Mach wrote (quoted in Ref. [47, p 179]),

> Suppose we were to attribute to nature the property of producing like effects in like circumstances; just these like circumstances we should not know how to find. Nature exists once only. Our schematic mental imitation alone produces like events.

This passage occurs in a critique of induction, but it serves just as well as a criticism of frequentism. The classification of 'similar' events, so far from being an objective observation of physical phenomena, is itself a subjective construction. Frequentism is in part an attempt to make probability objective, and remove the subjective element, but the attempt is unsuccessful.

The idea of probability as extended logic began its technical development with Augustus de Morgan in the mid-19th century. The extended logic interpretation was accepted by J.M. Keynes and Harold Jeffreys in their works in the 1920's. However, Fisher's influence outweighed theirs, and the frequency definition prevailed. In 1946, R.T. Cox published a brief, elegant article [18] which is the original statement of the development presented in Chapter 2. Only recently has the extended logic interpreta-

here. It indicates that statistics is close to engineering, and in some instances perhaps even close to carpentry." [26]

tion found widespread acceptance, and then largely in computer science; conventional statistics is still dominated by Fisher.

3.2 On induction

Having arrived, in Chapter 2, at an acceptable theory of probability, it is natural to ask what that theory might say about the natural world. In the classical scientific view which, by now, has become part of common sense, an induction based on empirical data is a meaningful statement about unobserved events, whether in the past, present, or future, because of the uniformity of Nature. There are real objects out there in the world, and these interact according to natural laws; we may not understand these laws, but natural law is as much a permanent, real feature of the world as are rocks and trees. Thus it is possible, at least, to find the connections between events. Yet this common-sense view has been attacked with philosophical arguments which thus far have been equally difficult to accept and to refute.

3.3 Hume's critique of induction

If deduction is the art of drawing necessary conclusions, then induction is the art of drawing unnecessary conclusions. In a sense, a deduction does not produce new information, because the conclusion is implicit in the premises; for this reason deduction is called *non-ampliative*. Induction, on the other hand, is *ampliative*: an induction contains a statement about phenomena which have not been observed, such as a future phenomenon, a present phenomenon which is hidden from view, or a past phenomenon of which we have incomplete information. Thus an induction contains more than the information in the observations on which the induction is based. The conclusion of a deduction is entailed by the premises, so no contradictory conclusion can validly be obtained from the premises. On the other hand, the conclusion of an induction is not so entailed; from a set of observations, one can make as many generalizations as one likes, some or all of them mutually inconsistent.

How, then, are we to judge which generalizations are 'reasonable' and which are bogus? David Hume (1711–76) set out to seek an answer this question; he never arrived at one, but along the way destroyed the reliance on induction which had become popular in the West with the rise of science. (It is fair to say that a satisfactory response to Hume's criticism of induction has not yet been found, and if the problem seems to cause little trouble these days, it is only by ignoring it.) In his *Treatise of Human Nature* (1740), Hume emphasized the gulf between our beliefs and events in the world on which they are based. I have seen the Sun rise many times, and I expect that it will do so tomorrow, but there is no *necessary* connection between the Sun rising in the past and rising tomorrow. I could establish, deductively, that the Sun will rise tomorrow if there were a Uniformity Principle which stated that the future will be similar to the past; but such a principle could only be established inductively, by observing that past futures were similar to past pasts. Hume stated that our apparent reliance on induction is only a psychological curiosity, and we carry on, despite the shaky foundations of induction, only because of ingrained habit or custom. It would seem that the question of the reasonableness of generalizations has an obvious answer: all generalizations to unobserved phenomena are unjustified, and unjustifiable.

Hume's conclusions (which are very hastily sketched here — see Ref. [8] for one of many treatments) are psychologically difficult to accept, and yet difficult to disprove philosophically. One response is to limit the applicability of induction: a statement such as "The Sun will rise tomorrow" is not an assertation of a physical phenomenon, but only a description of a state of mind. This sidesteps the difficulties Hume raised, because induction no longer concerns the world, but only the mind. In a sense, deduction is in the same boat — the clarity and precision of deduction only obtain by identifying certain discrete chunks[5] of the world with abstract symbols, and pretending that there are no relevant data aside from the premises of the deduction. But a perfect deduction,

[5] "In mentally separating a body from the changeable environment in which it moves, what we really do is to extricate a group of sensations on which our thoughts are fastened and which is of relatively greater stability than the others, from the stream of all our sensations." Ernst Mach [47, p 180].

like a perfect circle, exists only in the mind and not in the world. This comforting view is a little too easy, for it cannot explain the 'unreasonable' success we have in dodging cars and finding groceries, not to mention scientific investigation.

Since a necessary connection between observed and unobserved events seems impossible to establish, we might soften inductive statements to be probabilistic rather than categorical, for example, "The Sun will rise tomorrow with probability 0.999." But this really gets us nowhere, philosophically speaking. If a probability is a physical property, there is no necessary connection between the physical process and the stated number, and if a probability is a degree of belief, the statement remains trapped in the mind, without a connection to the world. In either case, we have not achieved a necessary connection between the induction and an unobserved phenomenon. Several other kinds of rescue plans have been put forth to lift induction from the abyss of subjectivity, but none has yet succeeded; Ref. [73] describes several such plans and their shortcomings.

3.4 A probabilistic interpretation of 'falsifiability'

Some aspects of Karl Popper's theory of science, which emphasizes deduction as opposed to induction and falsification as opposed to verification, can be interpreted in probabilistic terms. As a way of defeating Hume's attack on induction, Popper held [68, 69] that science is exclusively deductive, essentially interpreting the apparent inductive process of science as a deductive process. The only inference in science, he held, is the deduction of consequences from general laws. We then use those predictions to devise experiments, and prune away the hypotheses which fail experimental tests — that is, we keep only those hypotheses which have not been falsified. An hypothesis which resists falsification is said to be "corroborated." Popper held that corroboration is not the same as probability — in the terms used in this dissertation, we are not justified in having a greater degree of belief in a corroborated hypothesis, for it has not been *verified*, only *not falsified*.

Popper further argues that the information content of a scientific theory is, paradoxically, inversely proportional to its probability. If a theory is compatible with all

possible observations, the falsity of the hypothesis cannot be decided on the basis of observation. On the other hand, an hypothesis which is compatible with only a narrow range of observations is, in a sense, less probable *a priori*, since fewer outcomes favor it. This is directly related to the falsifiability requirement — to seek an informative theory is to seek a theory which is incompatible with many observations, which is to say, one which is falsifiable.

It is obvious that despite Popper's claims to the contrary, corroboration works like degree of belief: we base our actions (experiments, industries, buildings, etc.) on those hypotheses which are corroborated; according to Bruno de Finetti, the quantities which represent uncertainty in an economic analysis must behave according to the laws of probability. (This is the import of the so-called 'Dutch book' theorem, described, e.g., in Ref. [16].) Popper's theory of science can be interpreted in probabilistic terms as follows.[6] Let us agree to measure the informativeness of hypotheses according to their mutual information with observables — that is, if we have hypotheses H_1 and H_2, then we say H_1 is more informative than H_2 when $MI(H_1, X) > MI(H_2, X)$, where MI is the mutual information

$$MI(H, X) = \int \sum_h p_{HX}(h, x) \log \frac{p_{HX}(h, x)}{p_H(h)\, p_X(x)}\, dx \tag{3.1}$$

As mentioned in §2.7.5, the mutual information is a measure of the average number of bits by which we can compress the observations X; in assessing hypotheses according to MI, we are essentially committing ourselves to a particular interpretation of science — namely, that the goal of scientific inquiry is data compression.[7] The important term in the MI definition is the logarithm of the ratio of joint and marginal probabilities,

$$\log \frac{p_{HX}(H, X)}{p_H(H)\, p_X(X)} \quad = \quad \log \frac{p_{H|X}(H|X)}{p_H(H)} \quad = \quad \log \frac{p_{X|H}(X|H)}{p_X(X)} \tag{3.2}$$

[6] To the knowledge of the author, this interpretation has not been published elsewhere; but as it is a very natural development, doubtless it has occurred to someone before.

[7] Ernst Mach was perhaps the most well-known proponent of this view, as stated, for example, in the selection on "The economical nature of scientific inquiry" in Ref. [47]. "The goal which it [physical science] has set itself is the *simplest* and *most economical* abstract expression of facts."

If the distribution over X changes greatly when H is known, i.e., there is a large difference between p_X and $p_{X|H}$, then the logarithm is substantially different from zero, and the larger the magnitude of the logarithm, on the average, the greater the mutual information. If $p_{X|H} = p_X$, that is, the hypothesis H is flatly irrelevant to X, the logarithm is identically zero and the mutual information is likewise zero. If we can choose between two or more observations $X_1, X_2, X_3, \ldots$, none of which have been realized, we should choose the one with greatest mutual information on the hypothesis of interest. The more informative observation will be the one which has a very different distribution $p_{X_k|H}$ compared to p_{X_k}.

We see, then, that three important aspects of Popper's theory can be formally captured by probability. (i) The consequences of each hypothesis are deduced according to the laws of probability from the hypothesis; (ii) an experiment which can falsify an hypothesis is found by the mutual information criterion; and (iii) corroboration is interpreted as the posterior probability of the hypothesis. This tidy account omits two substantial problems. The first is the origin of hypotheses: is there a formal process for generating new hypotheses for previously unknown phenomena? The second is the relation of degrees of belief to real events: we may compute a high degree of belief, but as Hume informed us, a belief cannot have a necessary connection to future events. If we haven't yet solved these problems, at least we can console ourselves that no-one else has either.

Chapter 4

OVERVIEW OF THE RISO BELIEF NETWORK SYSTEM

Let us put the degree of belief concept to work in a system for the construction of distributed belief networks, which are useful models for diagnosis and prediction in geographically or functionally distributed systems. Belief networks for such systems have lately been the object of study by Xiang [86] and his students [15, 39] who refer to "multiply-sectioned Bayesian networks" which have a certain strict definition. In this dissertation the term "distributed belief network" will be used rather loosely to mean a probability model represented as an acyclic directed graph and implemented on multiple processors which communicate by means of a standard networking protocol; that part of a distributed belief network implemented on a particular processor will be referred to as a "component network" or "subnetwork." These definitions are deliberately vague, so as to include a wide range of interesting architectures and implementations. Broadly speaking, we will be as much interested in the "distributedness" as in the "belief-networkness" of distributed belief networks, and the interaction between these two aspects will lead to interesting problems.

Figure 4.1 shows a typical distributed belief network. Each one of the components A, B, C, and D is itself a belief network. The variables within the belief networks are identified by *network.variable*. For example, variable t in subnetwork A is referred to as "$A.t$." This scheme avoids name conflicts when variables in two networks have the same name. Note that in the example network, there is one loop (that is, an undirected cycle) wholly contained within subnetwork A, but there is another loop induced by the dependencies of B and C on variables in A. Such induced dependencies make life difficult for inference algorithms, because the known algorithms for handling loops require non-local information — that is, it is not possible (yet) to compute inferences

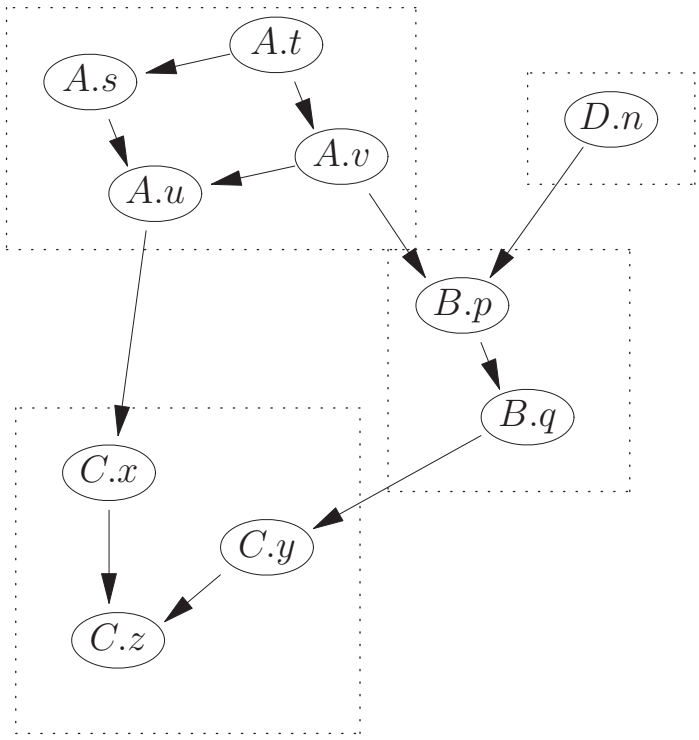

Figure 4.1: An example of a distributed belief network, composed of four subnetworks, A, B, C, and D.

in terms of a single variable and its immediate neighbors. Rather, groups of variables must be handled all at once, and the groups may be very large even for simple problems. A local algorithm for inference in a distributed system[1] is desirable, but may not be feasible in the presence of loops. A general inference algorithm for belief networks with loops has not yet been attempted for the RISO project; some comments on this topic will be found in §5.9.

The representation of geographically distributed systems is perhaps the prototypical application of distributed belief networks, although by no means the only application.

[1] "Distributed systems — What are they made of? Where do they come from? They're made of *string!* Yes, good old string. You know where you are with string!" — Attributed to a computer science professor.

It is natural to represent each geographical unit with a belief network, and to represent the flow of information from one locale to another by edges connecting variables in separate belief networks; different kinds of messages travel with the arrows and against the arrows. Such a scheme is all the more reasonable if the cost, time lapse, or difficulty of transferring information from one locale to another is large compared to the computations required for inference within one belief network. In that case, we will want to carry out as much computation as possible locally, and only transmit messages when necessary.

In addition to modeling geographically separate systems as separate belief networks, one can use distributed belief networks as a means of decomposing a modeling problem so that modeling of the pieces can proceed independently. For the same reasons that other kinds of software are customarily constructed piece by piece, it could be convenient to construct sub-networks separately, then connect them together through common variables. A decomposition into sub-networks will make it easier to comprehend the model and present it to other people. The components of a distributed belief network can be treated as independent software modules and reused for purposes other than those for which they were originally constructed. A great deal of work may go into the design and implementation of a belief network for a particular engine, reactor, pump, compressor, etc., and it will greatly speed development for new applications if the belief networks previously devised for similar systems can be reused. Generic belief networks for some special purposes, such as sensor models, time-series filtering, and hidden Markov models, can be easily constructed; it is foreseen that one could make available a library or archive of belief network components, which could be used as templates or building blocks from which particular applications could be constructed.

4.1 Features of the RISO system

A prototype implementation of distributed belief networks, named RISO,[2] has been constructed. Progress has been made in several areas, in both theoretical and practical matters. Let us survey the accomplishments of the RISO project.

4.1.1 *Representation of belief networks with heterogeneous distributions*

A necessary first step towards a flexible and extensible modeling system for general applications is the ability to represent belief networks composed of arbitrary conditional probability distributions. In order to make the belief network more comprehensible to domain experts, and to simplify the specification and modification of the belief network, probability distributions from the problem domain should be directly represented in the belief network. Software has been implemented in RISO which makes it possible to represent several kinds of basic distributions, and RISO is designed so that additional types of distributions can be implemented without changing the existing code at all. In contrast, most currently available belief network software can only handle a few well-known types of distributions, and it is assumed that if a distribution arising in a problem does not fall into a known category, then that distribution will be represented by an approximation belonging to a known category.

4.1.2 *Inference in polytrees with arbitrary distributions*

RISO implements a "just in time" approximation scheme for inference with arbitrary conditional distributions described in Chapter 5. The RISO inference algorithm is based on the polytree algorithm for belief network inference, in which "messages" (predictive distributions called π-messages and likelihood functions called λ-messages) are computed. The posterior distribution for a given variable depends on the messages sent to it by its parents and children, if any. In this scheme, an exact result is computed if such

[2] So called because the nodes in belief network diagrams look like nice plump grains of rice; see Figure 4.1.

a result is known for the incoming messages, otherwise an approximation is computed, which is a mixture of Gaussians or a monotone spline. The approximation may then be propagated to other variables. Approximations for likelihood functions (λ-messages) are not computed; the approximation step is put off until the likelihood function is combined with a probability distribution — this avoids certain numerical difficulties. In contrast with standard polytree algorithms, which can only accomodate distributions of at most a few types, this heterogeneous polytree algorithm can, in principle, handle any kind of continuous or discrete conditional distribution.[3] With standard algorithms, it is necessary to construct an approximate belief network, in which one then computes exact results; the heterogeneous polytree algorithm, on the other hand, computes approximate results in the original belief network. The advantages are that the approximations computed by the new algorithm are all one-dimensional and thus easier to compute, and, more importantly, that the belief network can be directly represented using the conditional distributions most appropriate for the problem domain.

4.1.3 *Implementation of distributed belief networks*

In extending the usual single-processor computational model to multiple processors, several interesting problems arise, which must be solved for the successful implementation of a distributed belief network. In keeping with the distributed computational model, no single processor has information about the structure of the entire distributed belief network, and inferences are to be computed using only local quantities. However, this leads to difficulties (which have not yet been resolved) when there are loops in the distributed belief network which contain nodes in two or more component networks. Also, temporal dependencies in one network lead to temporal dependencies in any other network connected to some of its variables; this may lead to intractable computations, especially if some of the dependencies involve different time scales. While these problems of induced dependencies have only been identified, and not resolved, progress has been been made in other areas. A general policy is proposed for publishing information

[3] Some other approaches for inference in heterogeneous belief networks are described in §5.9.

as belief networks. A modeling language for the representation of distributed belief networks has been devised, and software has been implemented to compile the modeling language and carry out inferences.

A prototype implementation of some of these ideas about distributed belief networks has been constructed using the Java programming language, in particular the Remote Method Invocation (RMI) mechanism for passing information between sub-networks. As a Java application, RISO will run on any hardware that supports the Java virtual machine. Furthermore, RISO has been designed without a graphical user interface, to make it possible to run the software on machines without a display; a rudimentary user interface has been constructed for testing purposes.

The feature of greatest interest in RMI is the representation of remote variables (Java program variables on another processor) as local references. This makes it possible to write code without worrying too much if the variables involved are local or remote — the RMI software on the local processor will communicate with the corresponding software on the remote processor as needed. Thus RMI is a very convenient form of interprocess communication, because from the programmer's point of view the communication looks like calling functions, passing arguments, and return values, while bothersome details of socket connections remain well-hidden.

A parser for the RISO belief network grammar has been implemented; the RISO grammar (Appendix A) is loosely based on the original proposal of the Belief Network Interchange Format (BNIF) grammar [3].

In contrast to some other implementations of distributed systems for inference (e.g., the Integrated Diagnostic System [85]) RISO is not designed to link diagnostic systems based on different software architectures and different reasoning paradigms. Probability was chosen as the sole medium for the expression and exchange of information, for theoretical reasons cited in Chapter 2, which suggest that other paradigms for reasoning under uncertainty are either essentially the same as probability or else fundamentally weaker. Choosing only one representation of uncertainty also greatly decreases the complexity of the software implementation. Choosing a single implementation language,

and a single collection of classes within that language, also greatly simplifies the implementation. The choice of a strictly probabilistic system implemented in Java has the advantages of unity, simplicity, and comprehensibility, both conceptual and practical.

4.2 Communication in distributed belief networks

In this section, let us briefly review some of the issues surrounding communication in distributed belief networks.

4.2.1 *Local versus global control of communication.*

Throughout this dissertation it will be assumed that communications between the components of a distributed belief network will be restricted to immediate neighbors. That is, belief network A can communicate with another, B, only if some variable in A has a parent or child in B. This local communication model, inspired by the idea of belief networks representing independent agents, is to be contrasted with a global communication model in which a central mechanism organizes communications between all components of a distributed belief network. The local communication model is better suited to a system in which component belief networks are created and executed independently, for potentially different purposes.

4.2.2 *Publishing information as distributed belief networks.*

There is considerable promise in the idea of publishing information on the Internet in the form of belief networks. Just as people can now connect to data sources in the form of files or Common Gateway Interface (CGI) and then reprocess the information for their own purposes, it should be possible to connect to belief network interfaces to obtain information in the form of probability and likelihood messages, which can then be used for purposes not specifically foreseen by the originators of the belief network. The original belief network together with any variables which are created as descendants of its interface variables will comprise a larger belief network. The new child variables may

in turn be linked to still others through their own interface variables; a belief network of any size could be constructed in this piecemeal fashion.

Allowing a belief network to reference variables in another raises interesting questions concerning the permission to pass messages within the overall belief network. Strictly speaking, entering evidence in a subnetwork requires that information be passed to other subnetworks, so that attaching a subnetwork and entering evidence will change the beliefs computed in existing subnetworks. It is easy to construct examples in which the agent represented by a component of a distributed belief network would not want to accept information supplied from variables outside its component. For example, an energy use forecasting service would doubtless have parent variables such as current and forecasted temperature which represent the influence of weather on energy use. However, the weather service which maintains temperature variables may wish to avoid having their distributions modified by downstream evidence, over which the weather service has no control.

We could control the influence of downstream evidence by establishing message passing permissions within distributed belief networks. Let us assume that a belief network can be linked to another only in such a way that does not disturb existing conditional probability assessments in the linked-to belief network. This rules out linking one belief network to another by specifying that some variable in the first is a parent of some variable in the second; but we can specify that some variable in the first is a child of some variable in the second without changing any conditional probabilities in the second network. Then there is essentially only one kind of message-passing permission to consider, namely

> Permission to affect belief computations in a subnetwork containing a parent node by entering evidence into a remote child node.

There are several subtleties to consider in this apparently simple problem.

1. Loops: in a polytree, "A does not accept messages from B" is equivalent to "A ignores evidence in B," but these two propositions are not equivalent in the

presence of loops.

2. Transitivity: A accepts information from B and B accepts information from C implies A accepts information from C. What if A says, "I don't want information from C," but B has accepted a π- or λ-message from C?

3. Access: for a given component belief network A, which other belief nets may create children of variables in A?

The first two topics mentioned, loops and transitivity, refer to technical problems of a graphical flavor. The third, access, only concerns software design choices we can make and so it is easier to handle. Let us dispose of the easier topic first.

Access. Access to variables requested by a remote belief network could be handled at several levels: by the name of the remote belief network, by the host on which the remote belief net is running, or by the domain in which the host lives. It is reasonable to assume, for security purposes, that any access permission not explicitly granted is withheld. One might expose all the variables in a belief network, or only certain interface variables might be exposed; in either case the name of the belief network and any remotely visible variables would be published in a digest or directory, as hypertext, say.

Transitivity. If we wish to preserve locality of communication, it seems we must allow transitivity of message acceptance. However, transitivity could lead to undesired consequences. If A does not accept messages from C but B does, and A accepts from B, then A must somehow sort out the information from C and ignore it when A receives messages from B. Of course, such a separation is not possible given the usual message-passing algorithms, which compute a message representing the relevant information in a form which does not reveal or identify the source of the information. It would appear that the only alternative to requiring transitivity is to tag each message according to the belief networks which contributed information to it, and to transmit

multiple tagged messages when a standard algorithm would only transmit one, each tagged message based on different evidence. Although a little clumsy, such a scheme is simple conceptually and easy to implement, and perhaps necessary if we are to avoid transitive message passing.

Loops. The computation of probabilities in the presence of loops in the belief network requires the use of non-local information, and for this reason may be difficult to implement in a distributed belief network. Recall that we have assumed there is no central mechanism to coordinate the individual belief networks which are the components of a distributed belief network, and that all coordination must occur through local communication. To make matters worse, local changes to a component may introduce loops into the distributed belief network in a way that is not immediately apparent from the components point of view. Local editing of a component may even introduce directed cycles — this must be detected and prevented; but who will give the approval for proposed changes to a belief network structure, if not a central authority? It may be possible for the processors to cooperatively detect and handle loops; see §5.9 for some comments on this topic.

4.2.3 *Computing inferences in distributed belief nets*

To accomodate the geographically and functionally distributed belief networks which are the focus of this dissertation, it is of great importance that the inference algorithms used allow for locality of computations and heterogeneous conditional distributions. Toward this end, the central inference algorithm of RISO is the polytree algorithm [63]. Unfortunately, this algorithm has limited applicability to belief networks which contain loops (that is, undirected cycles). In RISO loops will eventually be handled by a conditioning algorithm [63, 13], using a loop-cutset algorithm [7], but the details of this scheme have not yet been worked out. On the brighter side, the polytree algorithm does lend itself well to accomodating different types of conditional distributions.

4.3 Solutions to communications problems

Several interesting problems arise when a belief network is implemented in a distributed computing system. The usual network communication problems take on forms peculiar to belief network computations. Among these problems, the following have been handled in the design of RISO. Appendix B describes further details of the RISO communications architecture.

4.3.1 *Locating and connecting belief networks on different hosts*

When a belief network is loaded, RISO advertises its name in a globally–visible list, called the "registry." Belief networks in other processes on the same host or on different hosts can use the registry to obtain a pointer to any registered belief network. If the parent *spruce/weather.humidity* referred to by a belief network cannot be located in the registry on the host *spruce*, an attempt will be made to have the *weather* belief network loaded onto *spruce* so that *weather* becomes available; this is similar in spirit to the resolution of function references in libraries. If the host name is omitted, the host of the belief network in which the reference occurs is assumed. There will usually be a process running on each host which can load any belief network from a description on the file system of that host, or from a description string transmitted over the Internet. However, only a short program need be installed on each host, and additional software can be loaded as needed from another host. In particular, classes named in a belief network description will be copied (by the Java runtime software) to the host on demand; the complete RISO software need be installed on just one machine.

4.3.2 *Communicating π- and λ-messages between belief networks*

The messages transferred between belief networks are probability distributions and likelihood functions. These are expressed in parametric forms, which may be very short (e.g., a Gaussian can be described by just a few numbers) or very long (e.g., a mixture of Gaussians may contain an arbitrary number of parameters). Probabilities and

likelihoods are represented as objects within the RISO software, and these objects are automatically converted into a block of data which is transferred across a socket connection. Thus a request for a message from a remote belief network is implemented as a function call, and the message which is returned appears to be the return value of the function. As part of the general "lazy" computational policy of RISO, messages are requested only when they are needed; this cuts down on the relatively large overhead of passing messages between remote belief networks.

Aside from π- and λ-messages, other kinds of messages are passed in RISO distributed belief networks. When evidence is entered or removed from a node X in the network, messages are sent to all parents and children of X, telling them that the π- and λ-messages originating from X are no longer valid; these "invalid π- or λ-message" messages are propagated as appropriate to other nodes not d-separated from X. Any node receiving such a message knows that its posterior must be recomputed, but the computation is postponed until a request for the posterior is received.

4.3.3 *Coping with communication failures*

Communication failures can occur, for instance, when a host crashes or the process running a belief network is killed. When a π- or λ-message is required and an attempt to communicate with the corresponding parent or child node fails, RISO attempts to re-establish the link using the same registry look-up algorithm by which the link was originally established. If the attempt to reconnect fails as well, in order to make some progress RISO assumes that no evidence is available through the lost parent or child. Due to the distinction between parents and children in the computation of a posterior distribution, lost parents are treated differently from lost children. A lost child is simply dropped from the list of children of the variable which requested the λ-message, since in the absence of evidence from the child, the child has no effect on the computation of the posterior. A lost parent must be kept on the list of parents, but until the parent becomes available again (through restarting the process running the belief network), the prior for the parent will be substituted for any request for a π-message from that

parent, since in the absence of evidence the π-message from the parent will simply be the parent's prior. The parent's prior is sent from the parent to any remote child when the child first makes contact with the parent.

4.3.4 *Security issues*

In distributed belief networks, there are the usual problems of who can access which data, and these problems can be handled by well-known authentication, authorization, and encryption algorithms. However, there is at least one security problem which is peculiar to belief networks, namely that the naming X as the parent of Y implies (according to the laws of probability) the transfer of information from the child Y to the parent X as well as from X to Y. This suggests that we could affect degrees of belief in a network maintained by the Weather Service (let us say) by connecting some child variables and then introducing evidence into our sub-network. It is foreseen that the each belief network should be able to specify which others can propagate information up from child nodes, but this policy has not been implemented yet in RISO.

4.3.5 *Parallel computation*

The mechanism described for creating distributed belief networks — scope rules plus RMI — could be automated. One could analyze a belief network to determine which parts are the "most independent," then ship the parts to other hosts; inferences could be managed as if the entire network were on a single host, perhaps using a message-passing policy outlined by Pearl [63, pp 219–222]. This approach is reminiscent of data-flow analysis in parallelizing compilers, as described, for example, in Ref. [2].

4.4 Example: Monitoring a Distributed System

To illustrate the particulars of RISO which have been described in this chapter, let us consider a more realistic belief network than the one presented in Chapter 1. Our example is a simple distributed belief network for monitoring a distributed system. We

will examine the mechanical details of the representation of the domain model as a belief network and some inferences on the model. More substantial applications, with less RISO-specific detail, will be presented in Chapters 6, 7, and 8.

Consider the problem of monitoring a geographically distributed system. Let us model each component of the system as a separate belief network, each of which has a discrete status variable and one or two variables on which we can make measurements; in what follows we'll call these "measurable" variables. (We will distinguish between the variable we are trying to measure and the measurement itself. The measurement may be more or less accurate, and can be affected by a failure of the sensor or measuring device. The *monitor3* belief network shows a simple measurement model, comprising a measurable variable, the sensor status, and the measurement.) The monitor is represented as a belief network which contains only one variable, a "or" gate which computes the probability that at least one of the status variables is non-zero. We adopt the convention that status equal to 0 represents the normal status, and any other value represents an abnormal status.

The RISO code for the monitor subnetwork names the parents of the "or" gate as the status variables of the three individual monitor subnetworks.

```
BeliefNetwork combine
{
  Variable or
  {
    type discrete { "all OK"
       "at least one failure" }
    parents { phoenix.ef.boun.edu.tr/monitor1.s
       heaventemple.mit.edu/monitor2.s
       www.cdsoft.de/monitor3.s }
    distribution OrGate
  }
}
```

In this belief network and others, the "`BeliefNetwork`" tag not only begins the belief

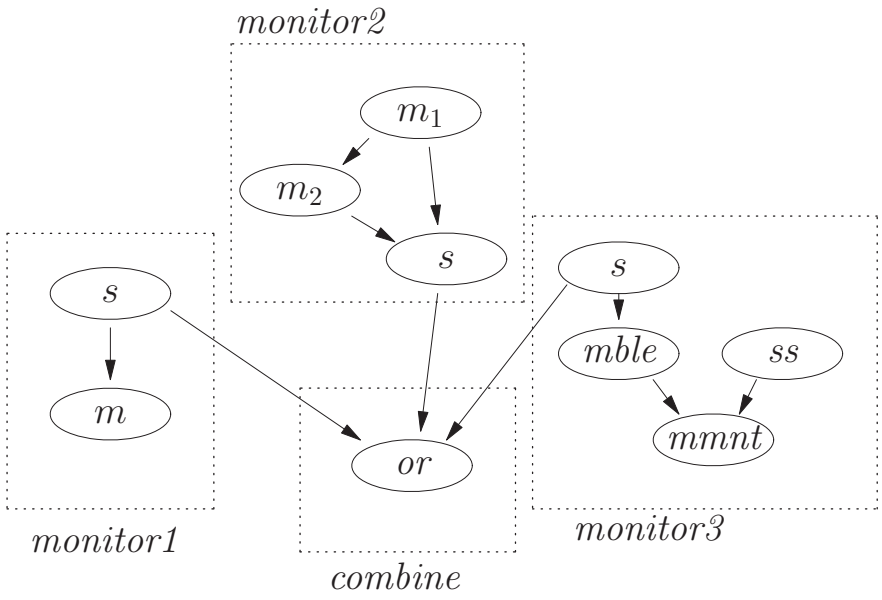

Figure 4.2: A distributed belief network for monitoring geographically distributed equipment. Key: s = "status," m or *mble* = "measurable," *mmnt* = "measurement," and *ss* = "sensor status."

network description, but names the Java class which knows how to parse the description and which implements the various message-passing functions necessary for computing inferences. Likewise, "`Variable`" begins the description of a variable within the belief network and also names the class which parses the description and implements the functions needed for a variable. In this scheme, one could implement software which accepts an alternative description by simply extending the `BeliefNetwork` or `Variable` class. For example, no provision has been made in RISO for representing display information such as the color and position of variables' nodes, but such information could be stored by a `FancyVariable` which extends `Variable`. More importantly, a conditional distribution represented in a RISO belief network description is tagged with the name of the class which implements it, and that class contains the code to parse the description and implement the probability functions needed for the inference algorithm. Thus special-purpose distributions can be created as the need arises in an application, and existing code for belief network objects and for variables need not be changed, including, above all, the inference algorithm.

Let us briefly consider the distributions encoded in the belief networks in this example.[4] In the interest of brevity, only the description of *monitor2* is shown. The *monitor1* belief network contains a simple "naïve Bayes" model. In *monitor2*, variable *mmnt* has a conditional Gaussian dependence on *mble*, and *s* is a logistic discriminant model with two classes. Thus *monitor2* shows a conventional classification model, which computes class probabilities conditional on its inputs *m1* and *m2*, integrated into the monitoring system.

[4] The belief networks discussed in this example are intended for illustration only, in particular the transmission of messages between variables. The complete RISO description files for these belief networks can be found at `http://civil.colorado.edu/~dodier`. For clarity, some identifiers have been abbreviated in this chapter.

```
BeliefNetwork monitor2
{
  Variable s {
    type discrete { "OK" "goofed" }
    parents { m1 m2 }
    distribution SquashingNetworkClassifier
      { ...  } }

  Variable m1 { distribution Gaussian
      { mean 30 std-deviation 4 } }

  Variable m2 {
    parents { m1 }
    distribution ConditionalGaussian {
      conditional-mean-multiplier { 0.3 }
      conditional-mean-offset { 8 }
      conditional-variance { 28 }
    }
  }
}
```

In *monitor3*, a naïve Bayes model is extended with a simple measurement model; more sophisticated measurement models are described in Chapter 6. The measurable variable *mble* is not directly known; only the measurement *mmnt* is observed. The measurement depends on the status of the sensor *ss*, as well as the measurable variable. The measurement is a noisy function of the measurable variable when the sensor is working correctly, and the measurement is just zero when the sensor is broken; this is common in sensors which output a voltage. However, in the model specified in *monitor3*, a measurement of zero can also occur when the sensor is working correctly. Finally, *combine* contains a single variable, *or*, which represents the probability that any of its parents (the status variables *monitor1.s*, *monitor2.s*, and *monitor3.s*) is non-zero.

Each belief network is running on a different host: belief network *monitor1* on host

phoenix.ef.boun.edu.tr, *monitor2* on *heaventemple.mit.edu*, *monitor3* on *www.cdsoft.de*, and *combine* on *sonero.colorado.edu*. The first and last are Linux machines, and the other two are running Windows. The description of *combine* specifies, in the list of parents for the *or* variable, which belief network is running on which host.

First, let us set *monitor2.m1* to -150. Querying *monitor2.s* (that is, computing the its posterior distribution), we find $\Pr(s = \text{“OK”} | m1 = -150) = 0.2974$. This result averages over values of the missing variable *monitor2.m2*; querying *m2* we see that its posterior is a Gaussian distribution with mean -37 and standard deviation 5.2.

So far, no messages have been passed from one belief network to another. Now let's set *monitor3.mmnt* to 23 and query *combine.or*. A π-message is sent from each monitoring belief network to the “or” gate. For *monitor1.s*, this is just its prior since no evidence has been introduced, and for *monitor2.s*, the π-message is just the posterior which was computed a moment ago. But for *monitor3.s*, computing the π-message to *combine.or* requires that evidence be propagated up from *monitor3.mmnt*; *ss* sends a π-message to *mmnt*, which sends a λ-message to *mble*, which sends a λ-message to *s*, which then sends the π-message to *combine.or*. Note that none of the messages in *monitor3* were computed until *combine.or* asked for a π-message. We find the posterior of *combine.or* has the probability that *or* = “all OK” is 0.2759, given the upstream evidence in *monitor2* and *monitor3*. The message-passing scheme is described in generality in Chapter 5.

Introducing evidence *monitor1.m* = 57, the posterior probability of *or* = “all OK” drops to 0.01705. Now there is evidence in all three monitor belief networks. Figure 4.3 shows the messages which have been communicated between the variables in the four belief networks comprising our distributed monitoring system. Each message is requested from the variable which needs the it — a message is not constructed unless it is required; this is the “lazy” computational policy. Also, messages within a belief network (that is, within the boxes shown in Figure 4.3) are transferred as return values from functions, while messages between belief networks are transmitted as blocks of data on a socket connection, and reconstituted into program objects by their recipient.

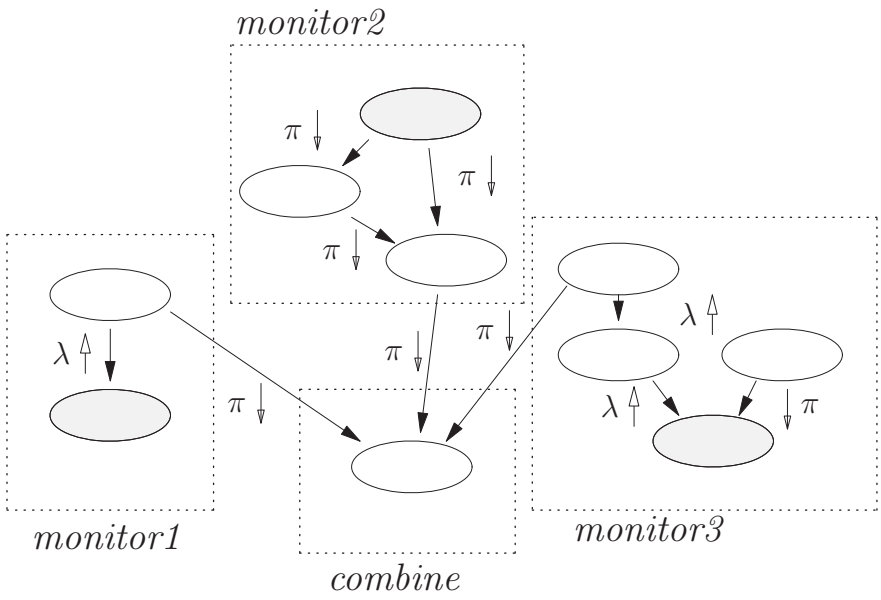

Figure 4.3: π- and λ-messages transmitted within a distributed belief network to satisfy a query on *combine.or*. Evidence nodes are shaded.

Let's see what happens in two scenarios involving a failure. The first scenario is a communication failure: *heaventemple* has crashed, leaving *monitor2* inaccessible. If we query *combine.or*, an attempt is made to contact *monitor2* to supply a π-message. The attempt fails, so *combine.or* uses the marginal prior for *monitor2.s* which was computed when *combine.or* first connected to *monitor2.s*. The prior for *monitor2.s* gives probability of "OK" equal to 0.03227, and this value is used to update *combine.or*, yielding the probability of "all OK" equal to 0.001833. Note that the prior over *monitor2.s* must be computed by integrating over its parents *m1* and *m2*; priors for status variables in *monitor1* and *monitor3*, which are root variables, are specified directly.

The second scenario is a sensor failure in *monitor3*. The sensor fails, and its output is zero. Querying *monitor3.ss*, we find that Pr("sensor OK"$|mmnt = 0) = 0.2309$. Although the likelihood for "sensor OK" indicates that the measurement $mmnt = 0$ is

much more typical of a failed sensor than one operating correctly, with

$$\frac{\Pr(mmnt = 0 | \text{“sensor OK”})}{\Pr(mmnt = 0 | \text{“sensor not OK”})} = 0.003032$$

the posterior for “sensor OK” is appreciably greater than zero due to the 99 : 1 prior odds in favor of “sensor OK.” Since *mmnt* is the common descendent of *s* and *ss*, information can travel from *ss* to *s* via *mmnt*. Querying *s*, we see that the posterior probability of “OK” is 0.9213, not much different from its prior value of 0.9000. However, if we set the sensor status *ss* equal to “OK,” we find a greater change in the posterior of *s*. Now the probability of *s* = “OK” is 0.9755. The effect of the measurement on the status variable *s* was weakened because the evidence favored a failed sensor.

4.5 Where is the magic hidden?

In this chapter, the broad design of a system for probabilistic reasoning was described, and a simple monitoring application was given. However, the calculations which go on in such a system were glossed over. Once the conditional and prior probability distributions are established for all the nodes in a belief network model, various intergrations and other transmogrifications are necessary to obtain a final result. In the next chapter, we will examine the formulas and algorithms by which the probability distributions which comprise a belief network description are transformed into useful results.

Chapter 5

AN INFERENCE ALGORITHM FOR POLYTREES WITH HETEROGENEOUS DISTRIBUTIONS

This chapter describes a general scheme for accomodating different types of conditional distributions in a belief network. The algorithm is based on the polytree algorithm for belief network inference, in which "messages" (probability distributions and likelihood functions) are computed. The posterior for a given variable depends on the messages sent to it by its parents and children, if any. In this scheme, an exact result is computed if such a result is known for the incoming messages, otherwise an approximation is computed, which is a mixture of Gaussians. The approximation may then be propagated to other variables. Approximations for likelihood functions (λ-messages) are not computed; the approximation step is put off until the likelihood function is combined with a probability distribution — this avoids certain numerical difficulties. In contrast with standard polytree algorithms, which can only accomodate distributions of a few types at most, this heterogeneous polytree algorithm can, in principle, handle any kind of continuous or discrete conditional distribution. With standard algorithms, it is necessary to construct an approximate belief network, in which one then computes exact results; the heterogeneous polytree algorithm, on the other hand, computes approximate results in the original belief network. The most important advantage of the new algorithm is that the belief network can be directly represented using the conditional distributions most appropriate for the problem domain.

5.1 Overview of the inference problem

The computation of posterior distributions is much simplified if the conditional distribution for each variable comes from a certain small set of distributions. The best-known

case is that of discrete conditional distributions; several algorithms have been developed (see, for example, Ref. [63]) for computing posterior distributions, and there are algorithms which can handle discrete networks which contain loops. Likewise, conditional Gaussian distributions are also well known [63]. An algorithm is also known [29] for a belief network in which each conditional distribution is a mixture of conditional Gaussians (with a linear dependence of the mean and no dependence of the variance), but this algorithm applies only to a polytree network. This algorithm has been extended [30] to a polytree containing both continuous variables (with mixtures of conditional Gaussian distributions) and discrete variables.

In many problems, though, the most natural probability distributions are neither discrete nor Gaussian, and constructing approximations within those well-known classes may yield a representation which is verbose[1] and which obscures the conceptual basis of the natural model. It seems desirable, then, to represent the natural probability distributions directly, and to only compute approximations (from the class of mixtures of Gaussians, for example) when necessary. The representation problem is easily solved by equipping each variable with type information and suitable parameters, and the difficult part is the computation of posterior distributions. Only for certain kinds of conditional distributions will it be possible to compute the partial results necessary for the posterior, so in many cases some kind of approximation is necessary. The following approach is taken in RISO:

> Directly represent the conditional distributions from the problem domain; compute an exact result for the posterior when possible, and otherwise compute an approximation to the exact result.

Note that this is different from the approach implicit in present belief network implementations, which is this: if a distribution does not fall into a well-behaved category (discrete or Gaussian), make a well-behaved approximation, then compute an exact

[1] For example, a discretized model for a small problem involving waste water treatment requires more than 300 kilobytes to store the model, and the engineering knowledge in the model is completely obscured.

posterior using the approximation.

We want to keep representations in a belief network close to the original problem domain. This will make it easier to describe and comprehend a belief network, especially for domain experts who are not particularly well-acquainted with belief networks; if one views a belief network as a kind of probabilistic database [63] then it is important that such people find it intuitive and productive to work with belief networks. As an important part of this approach, the conditional distributions in the network must match the ones which domain experts are accustomed to working with.

The reader will note that the scheme described in this chapter is similar in spirit to the adaptive discretization algorithm described in Ref. [48] and the combination of exact inference and Gibbs sampling in HUGS [46]. In each case, exact and numerical methods are combined to extend belief network inference to a larger class of problems.

5.2 Additional nomenclature of polytrees

The inference algorithm described in this chapter is based on a few simple concepts which are expressed in a terminology which may be unfamiliar to the reader, so let us take a moment to review the nomenclature associated with polytree graphs. Some of these terms were introduced in §2.6, but for clarity of exposition we will review them, and introduce several more which are peculiar to the description of the polytree inference algorithm.

Figure 5.1 shows a typical polytree. There is at most one undirected path (i.e., ignoring direction of the arrows) between any two nodes. Suppose an arrow is added between the two nodes at upper left, as shown in Figure 5.2. The result is not a polytree, because there are at least two undirected paths between some two nodes; for example, there are two paths from the node labeled A to the node B. Non-polytree belief networks are important in applications, but require a more sophisticated inference algorithm than the one used by RISO.

Let us focus on a particular node within a polytree belief network, as illustrated in Figure 5.3. This node represents one variable, say X. A node at the tail of an arrow

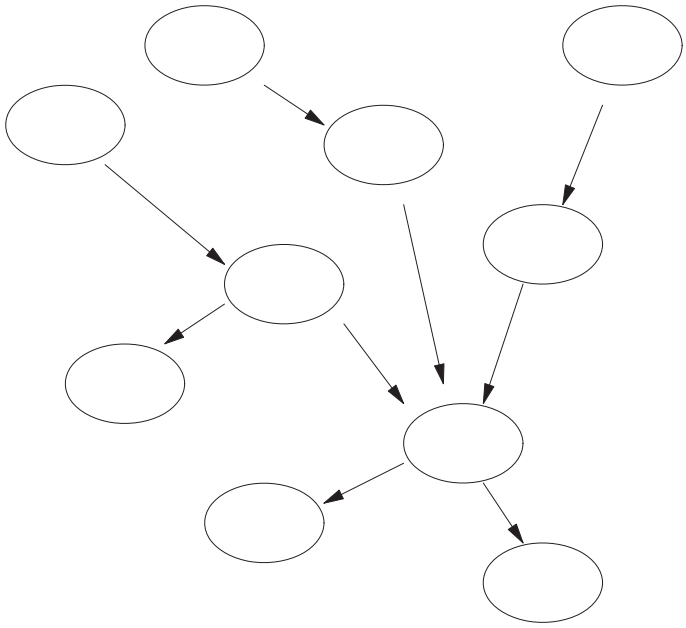

Figure 5.1: A typical polytree belief network.

leading into X is called a *parent* of X, while a node at the head of an arrow leading out of X is called a *child* of X. A node with no parents is called a *root node*, and a node with no children is called a *leaf node*. As described in §5.3, the polytree inference algorithm is defined in terms of probability distributions and likelihood functions called π-messages and λ-messages, respectively. The messages associated with a particular node X are shown in Figure 5.4. π-messages (predictive distributions) are sent along the arrows from a parent to a child, and λ-messages (likelihood functions) travel against the arrows from a child to a parent.

A node is called an *evidence node* if the variable associated with the node has a known value, for example, a sensor reading or known status. The set of all evidence nodes in a belief network is called the *evidence* of the belief network, and conventionally denoted by a boldface $\mathbf{e}$. It is convenient to divide the evidence $\mathbf{e}$ into subsets according to its placement in the belief network relative to a typical node X. Let us imagine for a moment that we have removed X from the belief network, so that several disconnected

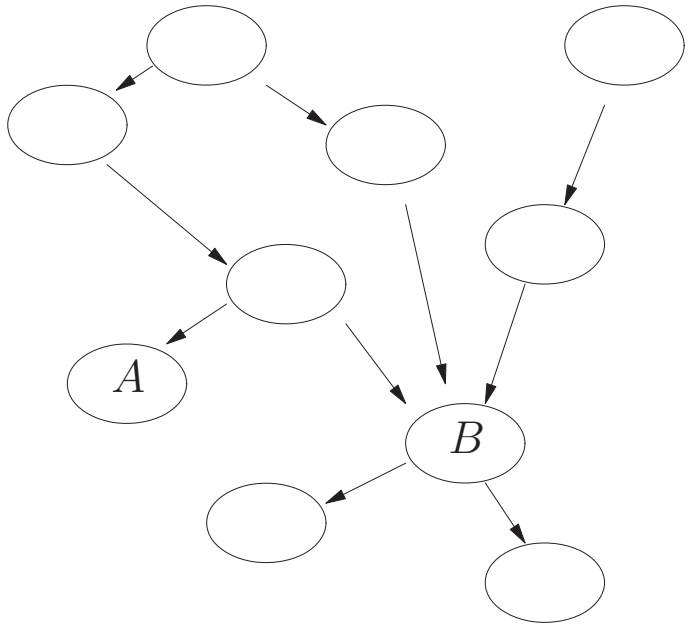

Figure 5.2: An example of a belief network which is not a polytree.

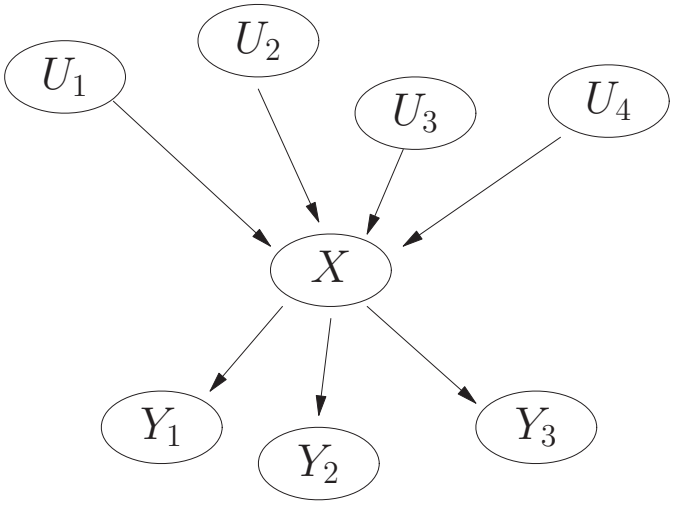

Figure 5.3: A typical node X in a polytree belief network. U_1, U_2, U_3, and U_4 are parents of X, and Y_1, Y_2, and Y_3 are children of X.

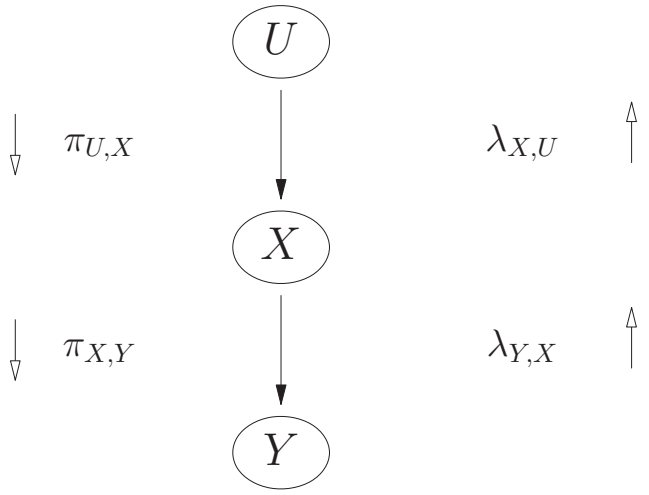

Figure 5.4: Messages associated with a typical node X. π-messages travel from parents to children, and λ-messages travel from children to parents.

subnetworks (one for each parent and each child) remain. The union of the subnetworks connected to parents is called the *upstream* part of the belief network (relative to X), while the union of the subnetworks connected to the children is called the *downstream* part of belief network. Now let us examine the evidence from the point of view of X, as shown in Figure 5.5. The set of all evidence in Figure 5.5 is $\mathbf{e} = \{A, B, E, Y_1, G, H\}$. The evidence upstream from X is denoted $\mathbf{e}_X^+ = \{A, B, E\}$, and the evidence downstream is $\mathbf{e}_X^- = \{Y_1, G, H\}$. The upstream evidence may further be divided according to which parent subtree contains the evidence: in the example, we have $\mathbf{e}_{X,U_1}^+ = \{A, B\}$ and $\mathbf{e}_{X,U_2}^+ = \{E\}$. Likewise, the downstream evidence may be divided into $\mathbf{e}_{X,Y_1}^- = \{Y_1\}$ and $\mathbf{e}_{X,Y_2}^- = \{G, H\}$.

We are now prepared to discuss the inference algorithm, called the polytree algorithm, which RISO uses.

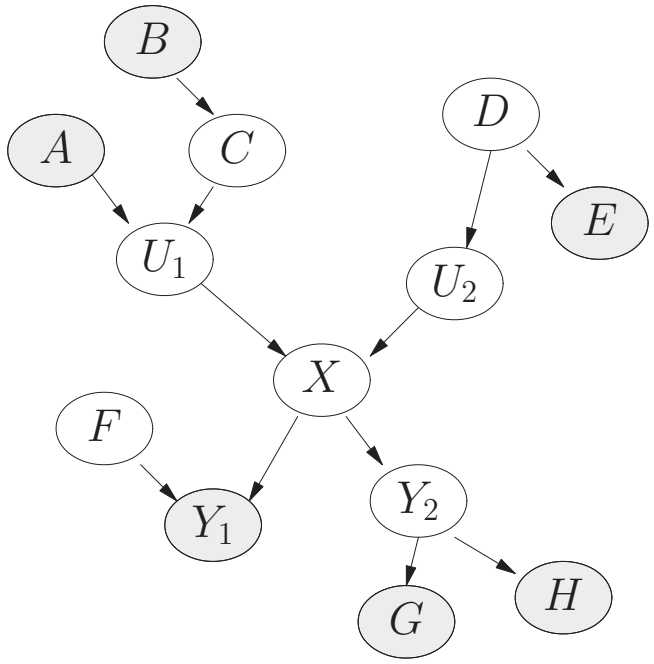

Figure 5.5: Evidence in a belief network, from the point of view of a typical node X.

5.3 The polytree inference algorithm

The polytree algorithm for computing the posterior of any variable in a belief network without loops exploits the fact that each variable X *d*-separates the network into two disjoint parts, one upstream from X and one downstream. The computation of the distribution of X given all the evidence $\mathbf{e}$ in the network, $p_{X|\mathbf{e}}$, is expressed in terms of predictive messages $\pi_{U,X}$ which are passed down from each parent U of X, and likelihood messages $\lambda_{Y,X}$ which are passed up from each child Y of X. These messages are combined to yield the predictive support π_X, and the likelihood support λ_X, and the posterior for X is just the normalized product of π_X and λ_X. The polytree algorithm computes the posterior of one variable at a time, by computing the predictive and likelihood support for that variable, and computing any messages needed. So long as the evidence variables in the network do not change, the same messages may be re-used to compute the posterior distributions for other variables as well.

The polytree algorithm is defined in terms of π's and λ's as follows.[2] Denote the distribution of X given its parents $U_1, \ldots, U_m$ as q_X. That is, q_X is a shorthand for $p_{X|U_1,\ldots,U_m}$. Let $\mathbf{e}$ be the set of all evidence variables within the network, with $\mathbf{e}_X^+$ denoting all the evidence above X in the polytree and $\mathbf{e}_X^-$ denoting all the evidence below X in the polytree. Also, let $\mathbf{e}_{X,U}^+$ denote the evidence above X which is also above its parent U, and let $\mathbf{e}_{X,Y}^-$ denote the evidence below X which is also below its child Y. The backslash denotes set difference, e.g. $\mathbf{e}_1 \backslash \mathbf{e}_2$ represents the set obtained by removing elements of $\mathbf{e}_2$ from $\mathbf{e}_1$. Then the π and λ functions are defined in terms of probabilities involving X, the parents and children of X, and the evidence in the network. There are five important equations in the polytree algorithm, as derived, for example, in Ref. [29].

1. Posterior for variable X:

$$p_{X|\mathbf{e}}(x) \;\propto\; \pi_X(x)\, \lambda_X(x) \tag{5.1}$$

2. Predictive support for X:

$$\begin{aligned} \pi_X(x) &= p_{X|\mathbf{e}_X^+}(x) \\ &= \int du_1 \cdots \int du_m \; q_X(x, u_1, \ldots, u_m)\; \pi_{U_1,X}(u_1) \cdots \pi_{U_m,X}(u_m) \end{aligned} \tag{5.2}$$

3. Likelihood support for X:

$$\lambda_X(x) \;\propto\; p_{\mathbf{e}_X^-|X}(x) \;=\; \prod_{j=1}^{n} \lambda_{Y_j,X}(x) \tag{5.3}$$

4. Predictive message sent to child Y_k:

$$\pi_{X,Y_k}(x) \;=\; p_{X|\mathbf{e}\backslash\mathbf{e}_{X,Y}^-}(x) \;\propto\; \pi_X(x) \prod_{\substack{j=1 \\ j\neq k}}^{n} \lambda_{Y_j,X}(x) \tag{5.4}$$

[2] A particularly clear exposition of the polytree algorithm was given in Ref. [29].

5. Likelihood message sent to parent U_k:

$$\begin{aligned}\lambda_{X,U_k}(u_k) &= p_{\mathbf{e}\setminus\mathbf{e}^+_{X,U_k}|U_k}(u_k) \\ &\propto \int dx \int du_1 \cdots \int du_{k-1} \int du_{k+1} \cdots \int du_m \\ &\qquad \lambda_X(x)\, q_X(x, u_1, \ldots, u_m) \prod_{\substack{j=1 \\ j\neq k}}^{m} \pi_{U_j,X}(u_j) \end{aligned} \tag{5.5}$$

Note that since a likelihood function is not a probability density, it need not be normalized to 1; a likelihood function may integrate to any positive number, or, indeed, it need not be integrable at all. Any positive multiple of a likelihood function is again a likelihood function; likelihood functions are unique only up to a positive constant factor. For this reason, expressions involving likelihood functions are given as porportionalities instead of equalities in Eqs. 5.1 through 5.5; the constants of proportionality for Eqs. 5.1, 5.2, and 5.4 are simply those numbers which make the left-hand sides integrate to 1. The constants of proportionality for Eqs. 5.3 and 5.5 are arbitrary; it is sometimes convenient to make the likelihood function integrate to 1. However, for some purposes, the normalization of likelihood functions does matter, and incorrect results obtain if the constants of proportionality are ignored. This problem arises in the computation of π's and λ's when the distributions involved are mixtures; this point is discussed at greater length in §5.3.1.

For some combinations of types of functions, the result can be computed symbolically. Otherwise, an approximate result must be computed. A general scheme for computing such approximations is presented in §5.5. However, symbolic results can be obtained for a wider range of distributions than the ones mentioned already (discrete, conditional Gaussian, mixtures of conditional Gaussian). A catalog of the π and λ computations which can be carried out exactly, or with a close approximation, is listed in Appendix C. The catalog is useful even if not all the functions of interest (π_X, λ_X, etc.) can be computed exactly; if some result can indeed be computed exactly, let us do so, and postpone approximations until they are finally necessary.

5.3.1 *π's and λ's for mixture distributions*

Mixture distributions, especially Gaussian mixture distributions, are very useful for constructing belief networks, since a mixture with enough components can approximate any smooth density. Within RISO, mixtures are computed as approximations to π and λ messages, as described in the next section. Mixtures may also arise when integrating over a discrete variable which has a continuous child — generally speaking, the mixture contains one component for each possible value of the discrete parent. For these reasons, mixture distributions will appear frequently in inference computations, and it is worthwhile to consider how the presence of mixtures affects the computation of π and λ functions.

It is easy to show that computing a π or λ function from mixture distributions will yield mixture of π or λ functions from each combination of single components. From Eqs. 5.1 through 5.5 we see that the quantities of interest are either products of distributions, or integrals of products of distributions. So let us focus on a product of mixture distributions. (If there is a distribution which is not a mixture, it can always be considered a mixture with a single component.) Let $p_1, \ldots, p_n$ be mixture distributions, each of which has some mixing parameters α_{kl} and component distributions p_{kl}. The density of each distribution is given by

$$p_k(x) = \sum_{l=1}^{N_k} \alpha_{kl}\, p_{kl}(x) \tag{5.6}$$

Then the product of these distributions is

$$\begin{aligned} \prod_{k=1}^{n} p_k(x) &= \prod_{k=1}^{n} \sum_{l=1}^{N_k} \alpha_{kl}\, p_{kl}(x) \\ &= \sum_{l_1=1}^{N_1} \cdots \sum_{l_n=1}^{N_n} \prod_{k=1}^{n} \alpha_{k,l_k}\, p_{k,l_k}(x) \end{aligned} \tag{5.7}$$

This shows that the result is again a mixture, with the number of components equal to the product $N_1 \cdots N_n$ of the numbers of components of each multiplicand, and mixing parameters equal to products $\alpha_{1,l_1} \cdots \alpha_{n,l_n}$ of the mixing parameters of the multiplicands.

5.4 Implementation of the inference algorithm

The polytree algorithm lends itself well to the following "lazy" or "just in time" computational scheme:

> To compute the posterior for X (or to compute π_X, or λ_X, or a predictive or likelihood message), compute only those functions which are required, then use those functions to compute the quantity of interest.

Probability distributions are represented within RISO as classes in the Java programming language. To compute the posterior (or π_X, etc.), the inference code first computes any necessary partial results, such as incoming messages. Then the inference code uses the types of the partial results to search the local filesystem for a helper class which contains a function to compute the quantity of interest. The helper classes are grouped together into "packages" according to their purpose: all classes to compute a posterior distribution will be found in the package `computes_posterior`, classes to compute π_X will be found in `computes_pi`, and so on for classes to compute λ_X, etc.

For example, if the posterior is to be computed and π_X is represented by an object of class A and λ_X is represented by an object of class B, then the inference code attempts to find a helper class in the `computes_posterior` package which accepts arguments of types A and B. If no such helper exists, the inference code attempts to locate a class which accepts arguments of types S and T, where S is A or a superclass of A, and T is B or a superclass of B. This scheme makes it possible to construct code which handles both special cases (for the subclasses) or handles general cases (for the superclasses).

All classes which represent a conditional probability distribution are subclasses of an abstract class which represents a conditional distribution, and all that represent unconditional distributions are subclasses of an abstract class which represents an unconditional distribution. If an exact symbolic result is known for a π- or λ-message or some other required function, that result will be computed by a helper class named by the subclasses. Otherwise, we fall back on a helper class named according to the superclasses. The approximation scheme described in this document is implemented by

helpers for the abstract base classes; exact results are implemented by subclass helpers. A list of the special cases which RISO can detect is tabulated in Appendix C. An approach for computing approximate results is described in the next section.

Since each type of distribution and each type of helper is represented by a different class, new types can be added without requiring modification of existing type definitions, and, above all, without requiring modification of the inference algorithm. This allows one to create the types suitable for particular applications, as described, for example, in Chapters 6, 7, and 8.

5.5 Approximating π's on the fly

To construct an approximation for π_X or a π-message, RISO minimizes the cross-entropy between the target (the distribution to be approximated) and a Gaussian mixture. The cross-entropy calculation is just

$$H_{p,q}(\theta) = -\int p(x)\ \log q(x|\theta)\ dx \tag{5.8}$$

denoting a target density as p and its Gaussian mixture approximation as q; the parameters of the approximation are denoted θ. The cross-entropy can be considered the continuous analog of the negative log-likelihood which appears in approximation problems based on measured data. Values of $p(x)$ are computing by directly evaluating the appropriate equation — in the case of π_X and λ_{X,U_k}, this requires numerical evaluation of integrals. Evaluating the cross-entropy itself also requires a numerical integration.

Likelihoods are not targets because they need not be normalized nor normalizeable, and so there may be no well-defined approximation. Posterior distributions are approximated by monotone cubic splines, as described in §5.7 and Appendix D. It is much faster to fit a spline than to compute a mixture approximation, but splines are unwieldy in calculations. Since a posterior distribution is a final result in the polytree algorithm, it need not have a form which is convenient for further computations. Spline approximations are also computed for certain π_X, namely distributions of sums and products, as described in §C.3.17.

The cross-entropy for the mixture approximation problem is minimized by an expectation-maximization (EM) algorithm, also employed by Poland [67]; the following discussion of convergence of the algorithm is based on Wu [84]. EM algorithms are usually described in terms of discrete data, but the development can be extended readily to the approximation of a continuous function. Let x denote the variable or variables on which the target p and its approximation q are defined, and let y denote the "unobserved" variable or variables; in the mixture estimation problem, the mixture selector is the unobserved variable. Consider the expectation of the logarithm of the joint distribution of x and y under the approximation model with respect to possible instantiations of the unobserved variable,

$$\begin{aligned} Q(\theta, \theta') &= \int p(x) \int q(y|x, \theta') \log q(x, y|\theta) \, dy \, dx \\ &= -H_{q,q}(\theta, \theta') - H_{p,q}(\theta) \end{aligned} \tag{5.9}$$

writing the cross-entropy of $q(y|x, \theta')$ and $q(y|x, \theta)$ as

$$H_{q,q}(\theta, \theta') = -\int p(x) \int q(y|x, \theta') \log q(y|x, \theta) \, dy \, dx \tag{5.10}$$

Thus Q is related to the cross-entropy as

$$H_{p,q}(\theta) = -H_{q,q}(\theta, \theta') - Q(\theta, \theta') \tag{5.11}$$

Applying Gibbs' inequality to

$$\int q(y|x, \theta') \log \frac{q(y|x, \theta')}{q(y|x, \theta)} \, dy \tag{5.12}$$

we find

$$H_{q,q}(\theta', \theta') \leq H_{q,q}(\theta, \theta') \tag{5.13}$$

Let us suppose we have some initial value for the parameters, denoted θ'. The EM algorithm can be stated as these two steps:

- *E step:* Compute $Q(\theta, \theta')$.
- *M step:* Maximize Q over θ and assign the result to θ'.

These two steps are repeated until θ' seems not to change much, or Q seems not to change much, or we run out of patience. Each two-step iteration decreases the cross-entropy $H_{p,q}$: suppose we have found θ such that $Q(\theta, \theta') > Q(\theta', \theta')$. Then from Eqs. 5.11 and 5.13 we find

$$\begin{aligned} H_{p,q}(\theta) &< -H_{q,q}(\theta', \theta') - Q(\theta', \theta') \\ &= H_{p,q}(\theta') \end{aligned} \tag{5.14}$$

If, in a sequence of parameter updates we consider θ' to be a previous value and θ to be new value, the EM algorithm decreases the cross-entropy $H_{p,q}$. If the entropy of the target exists, the cross-entropy must have a point of accumulation, since the cross-entropy decreases with EM iterations and is bounded below by the entropy of the target distribution p.[3] It does not necessarily follow that the parameters θ likewise converge, although if the cross-entropy is unchanged, whether the parameters converge is inconsequential in the function approximation problem.

The EM algorithm is applied to the mixture approximation problem as follows. Let i denote the component selector; the number of components is m. The parameters are $\theta = (\alpha_1, \ldots, \alpha_m, \mu_1, \ldots, \mu_m, \sigma_1, \ldots, \sigma_m)$. The joint distribution of x and i is

$$q(x, i|\theta) = \alpha_i \; g(x; \mu_i, \sigma_i) \tag{5.15}$$

Denote the mixture approximation as

$$q(x|\theta) = \sum_{i=1}^{m} \alpha_i \; g(x; \mu_i, \sigma_i) \tag{5.16}$$

with g being the Gaussian density function, $g(x; \mu, \sigma) = \exp(-(x-\mu)^2/(2\sigma^2))/(\sigma\sqrt{2\pi})$. With these definitions, we have

$$Q(\theta, \theta') = \int p(x) \sum_{i=1}^{m} \frac{q(x, i|\theta')}{q(x|\theta')} \; \log q(x, i|\theta) \; dx \tag{5.17}$$

[3] However, there exist proper distributions for which the entropy does not exist, such as $p(x) = x^{-1}(\log x)^{-2}$ for $x > e$. It is not clear how large a class of distributions this is.

$$= \sum_{i=1}^{m} \int p(x) \frac{\alpha_i' \, g(x; \mu_i', \sigma_i')}{q(x|\theta')} \log \alpha_i \, dx$$
$$+ \sum_{i=1}^{m} \int p(x) \frac{\alpha_i' \, g(x; \mu_i', \sigma_i')}{q(x|\theta')} \log g(x; \mu_i, \sigma_i) \, dx \tag{5.18}$$

To maximize Q over the α_i, we need consider only the first term in Eq. 5.18. Let us define the "integrated responsibility" as

$$IR_i(\theta) = \int p(x) \frac{\alpha_i \, g(x; \mu_i, \sigma_i)}{q(x|\theta)} \, dx \tag{5.19}$$

Applying Gibbs' inequality again, we see that

$$\sum_{i=1}^{m} \int p(x) \frac{\alpha_i' \, g(x; \mu_i', \sigma_i')}{q(x|\theta')} \log \alpha_i \, dx \leq \sum_{i=1}^{m} IR_i(\theta') \log IR_i(\theta') \tag{5.20}$$

thus to maximize Q we choose

$$\alpha_i \leftarrow IR_i(\theta') \tag{5.21}$$

As for the other parameters μ_i and σ_i, we seek a stationary point of the second term in Eq. 5.18. Computing the gradient with respect to the μ_i and σ_i and setting the gradient to zero, we obtain

$$\mu_i \leftarrow \frac{1}{IR_i(\theta')} \int x \, p(x) \, \frac{\alpha_i' \, g(x; \mu_i', \sigma_i')}{q(x|\theta')} \, dx \tag{5.22}$$

$$\sigma_i^2 \leftarrow \frac{1}{IR_i(\theta')} \int (x - \mu_i')^2 \, p(x) \, \frac{\alpha_i' \, g(x; \mu_i', \sigma_i')}{q(x|\theta')} \, dx \tag{5.23}$$

In general, one should consider a maximum of Q at the boundaries of the parameter space as well as any stationary points within the parameter space. Since the μ_i may take on any real value, the only boundary to worry about is $\sigma_i = 0$. If the target density p is smooth, none of the components of the mixture approximation will have zero variance and we can safely ignore the boundary. On the other hand, if the target density has nonzero mass at some points (for example, if it is a mixture of discrete and smooth densities) then Q will reach a maximum for some $\sigma_i = 0$; the required derivatives do not all exist, and the assignment in Eq. 5.23 is not applicable. In this case, it seems the best course of action will be to place zero-width components at the points supporting discrete masses, and carry out the EM algorithm on the smooth density which remains.

Since over the course of several iterations of the cross-entropy minimization algorithm the target function will be evaluated repeatedly at the same or nearly the same argument, we can speed up the calculations by cacheing values of the target function. The cacheing algorithm employed by RISO is based on a self-balancing binary tree called a "top-down splay tree" [75]. Each node in the splay tree stores a key x and its associated function value $f(x)$; the nodes are ordered by increasing values of x. When a value of $f(x)$ is needed, the splay tree is searched for x. If x is contained in some node, the associated $f(x)$ is returned. Otherwise, if x is between two nearby values, the values associated with the neighbors are interpolated and the result is returned. Otherwise x is less than the least key in the tree or greater than the greatest key, or the neighbors of x are too far away; the value of $f(x)$ is computed, stored in the tree with key x, and returned.

On the average, searching a top-down splay tree requires a number of operations proportional to the logarithm of the number of keys in the tree. These operations are relatively fast, such as dereferencing memory addresses and comparing numbers. Since the target function may be defined in terms of numerical integrations which are relatively time-consuming, using a splay tree as a cache can yield a significant speed-up.

In this scheme which computes approximations to explicitly computed target functions, there is substantial bookkeeping. Target functions are constructed by keeping references to the necessary components functions (π- or λ-messages and the conditional distribution for a given variable), and then evaluating the components (integrating them if necessary) when an output of the target function is needed.

It has proven useful to compute a summation over a discrete variable as a special case of the numerical integration code, so in order to compute a multiple integral it is necessary to know whether each variable is discrete or continuous. Also, it is very useful to "skip over" a variable in a multiple integration, that is, to assign the variable a certain value, and to not integrate over that variable. If a variable is an evidence variable, or if it is an argument to the integral (as is the parent variable to which we send a λ-message), then we skip over the integration of that variable.

5.6 Numerical subtleties of cross-entropy calculations

Finding the "effective support." To make numerical integrations easier, RISO tries to make the region over which we integrate as small as possible. Let us refer to a region which contains at least a mass $1-\epsilon$, for a small number ϵ, as an "effective support" of the integrand. Note that a region which contains a mass $1-\epsilon$ is not unique; RISO makes an effort to find the smallest effective support. It is important to obtain a small effective support because numerical integrations may fail if the integrand varies on scales that are much larger or much smaller than the region over which the integration is carried out. Also, to initialize a Gaussian mixture approximation (as described below), RISO attempts to find peaks in the target distribution over the effective support, and this search is more efficient when the effective support is as small as possible.

Integration algorithm. Numerical integrations in more than one dimensions are difficult. In the current implementation of RISO, multidimensional integrations are computed by a quasi Monte Carlo (QMC) algorithm. Such an algorithm is much like an ordinary Monte Carlo integration in that the estimate of an integral is taken as

$$\int f(x)\,dx \approx \frac{1}{N}\sum_{k=1}^{N} f(x_k)$$

where the (x_k) are a sequence of points. However, in a QMC algorithm the sequence is constructed to be more uniform and more regular than an ordinary pseudo-random number sequence, and for this reason QMC algorithms are often called *methods of low discrepency*. If the number of dimensions of the integration is not too high, QMC often yields a more accurate estimate for the same number N of function evaluations, compared to ordinary Monte Carlo. See Ref. [58] for a comprehensive treatment of QMC algorithms. TOMS algorithm 659 [9] was used to generate low-discrepency sequences used in QMC integrations; source code for algorithm 659 is available at `www.netlib.org`.

One-dimensional integrations are carried out by an adaptive region-splitting algorithm based on a Gauss-Kronrod 21-point rule. (The code is a translation of the QAGS

algorithm from QUADPACK, a collection of quadrature algorithms available from `www.netlib.org`.) The adaptive quadrature algorithm can be "fooled" if the integrand varies on a scale much smaller than $I/42$, where I is the length of the interval of integration. For this reason, RISO tries to find the smallest effective support of the integrand, as described under the preceding heading.

Initial approximations. Since the EM algorithm yields only a local minimum of the cross-entropy, the initial approximation should be not too far from correct — otherwise we might come to a high local minimum of cross-entropy. In particular, it pays to search for peaks in the target density, and initialize the approximation with corresponding peaks. RISO constructs an initial mixture approximation with a certain number of components with equal mass and variance, with their centers placed at regularly spaced intervals; these components are called "pavers," since they cover the support of the target like flagstones. Also, a mixture component is allocated for each peak which is found in the target density, as described in the following paragraphs.

RISO implements the following scheme for locating peaks in the target density. A uniform grid x_0, $x_1 = x_0+h$, $x_2 = x_0+2h, \ldots, x_n$ is placed over the effective support of the target density. If a point seems to be a local maximum (i.e., if the density is greater at x_i than at x_{i-1} and x_{i+1}), then a component is added to the initial approximation mixture with mean equal to x_i and standard deviation calculated by fitting a Gaussian bump to the curvature of the target density. For convenience, translate from x_i to 0, with $u = x - x_i$. Let q_0 denote a function proportional to the Gaussian density with mean 0 and variance σ^2. Denote the mass $\int q_0(u)\,du$ as α. Then q_0 and its first two derivatives are

$$q_0(u) = \frac{\alpha}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\frac{u^2}{\sigma^2}\right) \tag{5.24}$$

$$q_0'(u) = -\frac{u}{\sigma^2}\, q_0(u) \tag{5.25}$$

$$q_0''(u) = \frac{1}{\sigma^2}\left(\frac{u^2}{\sigma^2} - 1\right) q_0(u) \tag{5.26}$$

From this it follows that

$$q_0''(0) = -(1/\sigma^2)\, q_0(0) \tag{5.27}$$

Now given the curvature of the target density, estimated as

$$q_0''(0) \approx \frac{p(x_{i+1}) - 2p(x_i) + p(x_{i-1})}{h^2} \tag{5.28}$$

we can solve for σ to get an approximate standard deviation:

$$\sigma \approx \sqrt{-q_0(0)/q_0''(0)} \tag{5.29}$$

The mass of the peak is estimated as

$$\alpha \approx p(x_i)\, \sigma\, \sqrt{2\pi} \tag{5.30}$$

The corresponding mixing parameter is set to somewhat more than the estimated mass of the peak: a mixture component is allocated with its weight proportional to

$$\frac{1}{n} + p(x_i)\, \sigma\, \sqrt{2\pi} \tag{5.31}$$

where n is the number of peaks plus the number of pavers in the initial mixture. A paver is assigned a weight proportional to $1/n$.

Delaying integration of likelihood functions. The likelihood function of a discrete variable has a known domain: the domain is just $0, 1, 2, \ldots, \#X - 1$, where $\#X$ is the number of states of the variable X. So for a discrete variable, RISO constructs a table of probability values and evaluates the likelihood function over its domain, and stores the results. The table is propagated through further computations.

However, for a continuous variable the likelihood need not be normalizeable, so it may not have an effective support smaller than all the reals. RISO does not integrate over a likelihood until there is a predictive distribution (which is guaranteed to be normalized) in the integrand. For this reason approximations are not generated for a likelihood function of a continuous variable (i.e., λ_X or a λ-message). Gaussian mixture approximations are only computed for π_X or a π-message of a continuous variable. This is perhaps too pessimistic; there are examples of likelihood functions of a

continuous variable which do have bounded support, and which are therefore amenable to approximation.

Removing "unneeded" components. If an accurate approximation can be constructed using some number of components, then over iterations of the EM algorithm any additional components often become nearly the same (i.e., having the same mean and variance) as some other component. RISO makes an effort to find and remove such redundant components, by comparing the mean and variance of each component against the mean and variance of every other component. If the means are close enough and the variances are close enough, the components are redundant: one of them is removed and its weight is given to the other component. "Close enough" is assessed as follows:

> Let $r = \sigma_1/\sigma_2$, $\sigma^2 = 1/(1/\sigma_1^2 + 1/\sigma_2^2)$, and $\Delta\mu = \mu_1 - \mu_2$. If $\Delta\mu/\sigma < \beta$ and $|r - 1| < \gamma$, then components 1 and 2 are redundant.

Of course, the numerical factors β and γ can be adjusted to suit one's tastes; there does not appear to be a principled means of adjusting these parameters. At present, these are assigned the values $\beta = 0.25$ and $\gamma = 0.2$.

If, at the end of an EM iteration, a component has a mass less than some threshold, it is removed. In the present implementation, the mass threshold is 0.005.

Other heuristics for reducing the number of components in a mixture have been described [17, 6]. The application in these papers was to mixtures of discrete distributions, but the basic ideas should also apply to Gaussian mixtures as well.

Stopping criterion for cross-entropy minimization. It is not yet clear how to determine when an approximation computed by minimizing cross-entropy is "close enough." At present, the approximation algorithm runs for a fixed number of cycles; this yields acceptable results.

5.7 Constructing monotone spline approximations

So-called monotone cubic splines [33] are useful as approximations to probability density functions. Such a spline is monotone between two adjacent support points, thus if the

function to be approximated is nonnegative at all support points, then the spline is nonnegative as well. As with other cubic splines, it is easy to express derivatives and antiderivatives of the spline in terms of its coefficients.

A spline approximation to a density function is constructed by evaluating the density a number of points and then calculating the spline coefficients according to the formulas given in Appendix D. The total area under the spline is calculated and the coefficients are scaled so that the spline integrates to unity. After this adjustment, the spline is a proper density function; this is especially convenient if the density to be approximated is not normalized, for example, if the density is a posterior distribution expressed as $\pi_X \lambda_X$, because it is easier to normalize the spline than to normalize the density to be approximated.

Spline approximations are much simpler and faster to construct than Gaussian mixture approximations, but they have some drawbacks. A spline approximation may be very verbose, because a large number of support points may be required. In RISO, verbose splines are often the output of a numerical convolution (§C.3.17); all the operands in the convolution must be discretized with the same interval Δx, which is determined according to the operand with least variance. In calculations described elsewhere in this dissertation (e.g., §8.2.4), splines with as many as several thousand support points were constructed.

A more serious problem than simple verbosity is the present lack of exact results for a posterior distribution, π_X, λ_X, etc. with spline approximations as the π- or λ-messages. Some exact results are possible — for example, the convolution of cubic splines is a higher-order polynomial spline — but it is not clear how useful such results will be. Integrals of cubic splines will generally contain higher-order polynomial terms, and these terms, when integrated again in another π_X or π-message calculation, will yield terms of still-higher order; this is bad news. Perhaps any terms of degree greater than 3 could be evaluated at support points to yield a cubic spline which is an approximation of an exact higher-order result. The good news is that if all terms in a calculation (posterior, π_X, λ_X, etc.) are piecewise polynomials, then the result is also a piecewise

polynomial. Polynomial spline representations of conditional distributions have not been used in belief network computations, to my knowledge, but the construction of such splines should be straightforward.

5.8 Exploiting parallelism for faster inferences

It is well known that belief network computations can be parallelized in some part. Making best use of parallel computation is especially important when the "just in time" approximation algorithm is employed, since computing each π or λ function by an approximation may be time-consuming. The original description of the polytree algorithm [63] refers to each node in the network as a separate processor. While computations proceed sequentially on each processor, each processor may be computing a different partial result necessary for the computation of a posterior distribution.[4] The parallelism exploited by such a system of processors is sometimes referred to as "topological parallelism." Other forms of parallelism can be exploited by more sophisticated inference algorithms [49, 65] to gain greater speed-ups.

RISO makes use of topological parallelism in an obvious way. A variable wanting messages from other variables issues a request for each message, then waits until all the messages have arrived — thus other processors may compute the messages in parallel. In at least one case of practical interest, this yields substantial parallelism: in distributed diagnostic monitoring problems, there may be many networks which represent different equipment units and which run on separate processors.

5.9 Extending the polytree algorithm

At present, the most important unsolved problem in RISO is handling networks which contain loops. A conditioning algorithm is planned, perhaps using algorithms [87, 27] to cooperatively detect loops in a distributed environment. However, it is not clear that

[4] It is conceivable that a large network might be divided into interconnected sub-networks, and the resulting pieces farmed out to different processors. Such a scheme is not implemented by RISO.

it will be feasible to carry out conditioning in the presence of hetergeneous distributions — in particular, conditioning on continuous variables may well be troublesome. A conditioning algorithm devised for a particular problem is described in §7.2, but it can handle only one kind of loopy network.

There remain some numerical problems. While convergence of the EM algorithm is comforting, other algorithms such as the quasi-Newton algorithm might be faster. Sometimes the EM algorithm finds a local optimum which is clearly inferior; perhaps several random starting points should be tried, and the least cross-entropy approprox-imation kept. There is anecdotal evidence to suggest that ordinary Monte Carlo is superior to quasi Monte Carlo in more than 8 dimensions or so; perhaps RISO ought to switch over to ordinary Monte Carlo in high-dimensional problems. It might also be profitable to combine exact inferences with Gibbs sampling, in the style of HUGS [46].

Chapter 6

BELIEF NETWORK IDIOMS FOR SENSORS AND SO ON

A belief network for an engineering problem will very often contain variables which represent measurements of physical quantities. Usually our belief about a measured variable X will be influenced by the status (correctly functioning or otherwise) of the sensor which measures X, past values of X, measurements of variables with which X is correlated, and, of course, measurements of X itself. The dependencies among all these variables may be modeled by a fairly complex belief network, but several commonly-encountered structures have been identified, and these structures are described in this chapter as belief network "idioms" — the well-worn phrases of belief network expression.[1] These stuctures can be used as cookie-cutters or patterns for the construction of belief networks containing measured variables for particular applications. A library of such patterns will reduce the time spent on extraneous variables — which might well be called "nuisance variables" — and speed the development of the application proper. Although the emphasis in this chapter is on models which are used as parts of larger networks, sometimes, as in the case of weather variables, a model of a measured variable may be an end in itself.

In addition to the idioms for measured variables, two models which aren't particularly concerned with sensors or measurements are also described. One is a model which can answer the query "is X larger or smaller than expected?" and the other is an "alternative grouping" model. So as not to lose them amid the profusion of sensor models, these non-sensor models are considered first, in §§ 6.1 and 6.2.

Except for the simple sensor model, which has been described before [57], descriptions of the models discussed in this chapter have not been published before. However,

[1] The very appropriate term "idiom" is due to Martin Neil and Norman Fenton.

this is probably due mostly to a lack of interest in the models themselves; the models have doubtless appeared in applications without any specific attention drawn to them. This chapter should be considered a convenient catalog, rather than a list of fundamental results.

The idioms described in this chapter are conveniently implemented as subnetworks in a distributed belief network. The subnetworks can be constructed and developed separately from the applications which may use them, and then linked into application models, via the measured variables, as needed. It might be convenient to allow the user to specify the name of a pattern and supply a few parameters, and have RISO construct the appropriate subnetwork; however, such a template expansion capability has not yet been implemented.

6.1 A model for the "strange magnitude" problem

In many engineering problems, there is some variable of interest which has a nominal value, and we want to know if the actual value of the variable is substantially different in magnitude from the nominal value. This model is very simple, but may appear repeatedly in a belief network application.

In this model, X_{nom} is the nominal or expected value of some variable X, while X_{act} is the actual value, and α is a multiplier: if X_{nom} and α are known, then $X_{act} = \alpha X_{nom}$. The usual problem is to infer α from a more-or-less known value of X_{act} and the distribution over X_{nom} inferred from the upstream evidence $\mathbf{e}^+$. (A measurement model for X_{act} is omitted in the interest of clarity.) If both the nominal and actual are known, then α is just the ratio X_{act}/X_{nom}, thus the posterior over α is essentially the computation of the distribution of the ratio. Except when the distributions involved are lognormal, the ratio will not have any simple form, and an approximation must be computed.

As a simple example, suppose the upstream evidence is such as to yield $p_{X_{nom}|\mathbf{e}^+}$ as a truncated Gaussian distribution with $\mu = 120$ and $\sigma = 30$, truncated to the interval $[0, +\infty)$, and α is given a uniform prior over $[0, 4]$. Suppose we have $X_{act} = 100$. Then

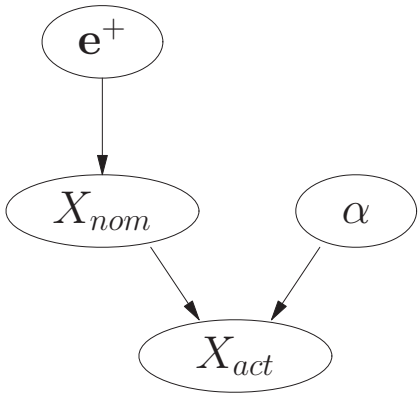

Figure 6.1: A model for "strange magnitude." To be specified: p_α (prior belief about the multiplier) and $p_{X_{nom}}$ (prior over nominal value — may want to make this conditional on other variables). The conditional distribution $p_{X_{act}|\alpha,X_{nom}}$ is always $\delta(X_{act} - \alpha X_{nom})$.

the posterior for α is skewed to the right, with a mode near 0.79, mean 0.9897, and standard deviation 0.3759; such a skewed distribution is typical for the distribution of a ratio. The posterior, shown in Figure 6.2, is computed by RISO as a spline approximation (Appendix D) with about 200 support points.

6.2 Alternative groupings

Some operations, such as summation or logical conjunction or disjunction, yield the same results no matter what the order of the arguments, and results are easily defined as the same operation applied to partial results. In some engineering problems, it is of interest to consider a collection of variables combined into different groups by the same operation. An example will clarify the idea: we may need to compute the total energy use of a group of buildings, dividing the buildings according to geographic region and computing a subtotal for each region, and also dividing the buildings into classes according to predominate use (residence, retail, school, office, etc.) and computing a subtotal for each usage class. Or we may be monitoring the buildings for problems in the HVAC systems, in which case the operation is not summation but disjunction — each

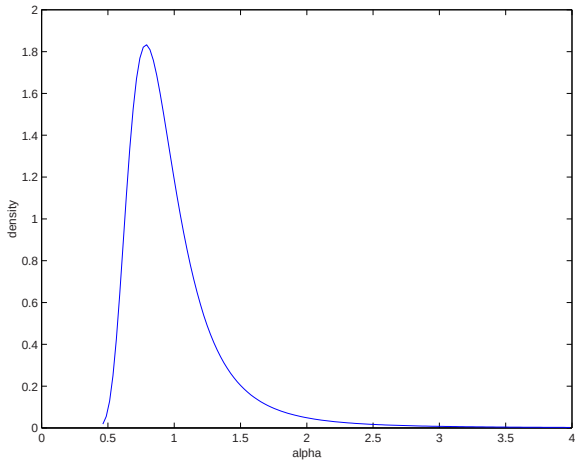

Figure 6.2: Posterior for α, with $p_{X_{nom}|\mathbf{e}^+}$ a truncated Gaussian distribution ($\mu = 120, \sigma = 30$), and $X_{act} = 100$, as a spline approximation comprising about 200 support points.

partial result has the form "There is a problem in such-an-such a group of buildings," obtained from "Building A has a problem or building B has a problem or ...."

So long as evidence is not introduced into the variables downstream from $\sum_i X_i$ (or whatever the operation), the various groupings have no effect on each other, so new groupings can be introduced as needed without affecting previously computed results. Operations can be arranged in hierarchies if need be, again without affecting already-computed results.

As a simple example, suppose that the variables in Figure 6.3 are assigned the

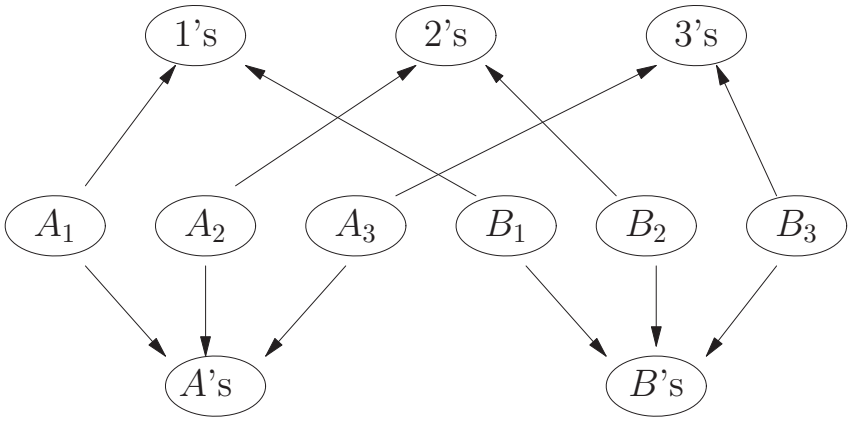

Figure 6.3: Alternative groupings for a collection of variables. One grouping distinguishes variables by letter: all the A's are in one group and all the B's in another. Another group distinguishes variables by index: all the 1's are together, all the 2's are together, and all the 3's are together.

following distributions:

$$A_1 \sim N(50, 7^2) \tag{6.1}$$

$$A_2 \sim N(110, 20^2) \tag{6.2}$$

$$A_3 \sim \text{Gamma}(8, 2) \tag{6.3}$$

$$B_1 \sim \text{Lognormal}(2, 1.2) \tag{6.4}$$

$$B_2 \sim N(32, 5^2) \tag{6.5}$$

$$B_3 \sim N(14, 17^2), \quad \text{left-truncated at 11} \tag{6.6}$$

and suppose further that the nodes "1's," "2's," and "3's" are the sums A_1+B_1, A_2+B_2, and $A_3 + B_3$, respectively; likewise "A's" and "B's" are the sums $A_1 + A_2 + A_3$ and $B_1 + B_2 + B_2$, respectively. RISO can tell that the sum $A_2 + B_2$ can be computed exactly as a Gaussian distribution, but the other distributions are computed as spline approximations, with the support points given by numerical convolutions (as described in §C.3.17) of the density functions involved. The splines have from 900 to 6600 points

— the wider the support of the distributions involved, the more support points are allocated; the effective support of the lognormal distribution of B_1 is about $[0, 1000]$, and this forces the use of a large number of points for $A_1 + B_1$ and $B_1 + B_2 + B_3$. In the interest of brevity, the distributions of the summation variables are not plotted.

As a sanity check, the distributions of "1's + 2's + 3's" and "A's + B's" are computed by RISO as identical up to numerical errors, as indeed they should be.

6.3 A simple sensor model

We begin our development of models for measured variables by distinguishing between the variable in which we are interested, and its measurement. We directly observe only the measurement, and the variable of interest must be inferred, as when we observe a thermometer showing "52° F" and we infer that the temperature throughout the room is more or less 52° F. The reader will immediately note that several unhappy accidents can occur: the air in the room is not well mixed, and near the ceiling it is substantially warmer than below; the observation was made several hours ago; the thermometer is miscalibrated; the lights are out and it's too dark to read the thermometer. There is always a greater or lesser distance, so to speak, between the observation that we can carry out and the variable we need to know about — the point of modeling the measurement process is to detect measurement problems, on the one hand, and to work as best we can with whatever information is available, on the other.

A simple, but useful, model which embodies some of the above considerations is shown in Figure 6.4. A model of this kind was proposed in Ref. [57], and is commonly used in many applications. We are interested in the variable X; its measurement is denoted $\widehat{X}$, and the status of the sensor which measures X is denoted S. Three distributions are specified for this model:

- p_X — In the absence of any further information concerning X, p_X describes our beliefs about X. This distribution is commonly a density model of historical data, or is constructed from prior information about the typical values of X. A

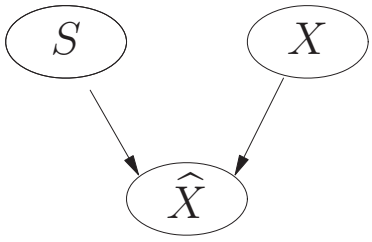

Figure 6.4: A simple sensor model. To be specified: p_S (reliability), p_X (long term distribution), and $p_{\widehat{X}|S,X}$ (measurement model).

convenient specification is $X \sim N(\mu_X, \sigma_X^2)$, where μ_X and σ_X take on any suitable values, but other forms of models such as lognormal, gamma, uniform, truncated, etc., are possible, and easy to work with in RISO.

- p_S — This discrete distribution is interpreted as a specification of the reliability of the sensor. The possible states of S are "correct operation" and one or more failure modes; let us agree to the convention that $S = 0$ corresponds to the "correct operation" state, and $S \geq 1$ corresponds to some failure mode. Ordinarily, most of the mass is on $p_S(0)$ and the remainder is split up among the failure modes.

- $p_{\widehat{X}|S,X}$ — This distribution describes the dependence of measurements on the measured variable and the sensor status. In the "correct operation" state, $p_{\widehat{X}|S,X}$ is typically a conditional Gaussian of the form $\widehat{X} \,|\, X \sim N(X, \sigma_{\widehat{X}}^2)$, where $\sigma_{\widehat{X}}$ is identified with the measurement error. If the sensor is not working correctly, any one of several problems might occur, and each will have a characteristic effect on $\widehat{X}$. One very common failure mode is an open circuit (e.g., a broken wire) on a sensor which transduces a physical quantity to a voltage. In this case, the output of the sensor is reported as a constant value, denoted here $\widehat{X}_0$, which is the result of mapping the zero voltage to a number via the calibration function. It is worth remarking that in the "open circuit" state, X has no effect on $\widehat{X}$, so the λ-message from $\widehat{X}$ to X will generally be a mixture which contains a noninformative

component.

Let us illustrate the working of the model with a few queries on an instance of this class of models. The distributions are specified as follows: let $\mu_X = 50$, $\sigma_X = 20$, $\widehat{X}_0 = -46$, $\sigma_{\widehat{X}} = 1$, and $p_S(0) = 0.99$. Let $p_{\widehat{X}|S=0,X}$ be a conditional Gaussian with mean X and standard deviation $\sigma_{\widehat{X}}$, and let $p_{\widehat{X}|S,X}$ be an unconditional Gaussian with mean $\widehat{X}_0$ and standard deviation $\sigma_{\widehat{X}}$.

The typical use of the model is to compute the posterior marginals $p_{X|\widehat{X}}$ and $p_{S|\widehat{X}}$. When $\widehat{X}$ is well away from $\widehat{X}_0$, $p_{X|\widehat{X}}$ is very nearly $N(\widehat{X}, \sigma^2_{\widehat{X}})$, but close to $\widehat{X}_0$, $p_{X|\widehat{X}}$ is essentially the same as $N(\mu_X, \sigma^2_X)$, the prior for X. Properly speaking, the posterior $p_{X|\widehat{X}}$ is always a mixture, with one component corresponding to $S = 0$ and the other to $S = 1$, but for most values of $\widehat{X}$ one component or the other has almost all the mass. However, there is a transition region near $\widehat{X}_0$ in which both components have appreciable mass. For example, at $\widehat{X} = -42$, we find

$$p_{X|\widehat{X}}(X, -42) = 0.278\ g(X; -41.8, 0.999) + 0.722\ g(X; \mu_X, \sigma_X) \tag{6.7}$$

for the posterior of the measured variable; for the posterior of the sensor status, we have

$$p_S(0) = 0.278, \qquad p_S(1) = 0.722 \tag{6.8}$$

Despite its simplicity, the sensor model shown in Figure 6.4 will find a place in a wide variety of applications, and the more complex models shown in the following sections are largely elaborations of the simple sensor model.

An example of the use of the simple sensor model in an application is given in Figure 8.2, which shows a belief network model for a heating coil. The variables UA, C_{pa}, and C_{pw} are all non-measured variables with fixed values, while $T_{db,ent}$, m_w, m_{air}, T_w, and $T_{db,lvg}$ are all measured. Each measured variable is represented by a belief network of the same name, containing nodes for the actual value, the measured value, and the sensor status. In the RISO belief network description, the list of parents for $T_{db,lvg}$ refers to the measured values by the name of the belief network and the name

of the variable representing the actual value within it: `Tdbent.actual, mw.actual, mair.actual` etc. References to two different nodes which are both named "actual" are disambiguated by the names of the subnetworks containing them, using the naming scheme described in §5.4. The RISO belief network description of Figure 8.2 may be found at `http://civil.colorado.edu/~dodier` under the heading, "Example RISO belief networks."

6.4 A sensor model with temporal dependence

Consider the problem of measuring atmospheric pressure, temperature, or relative humidity. From physical principles, we know that the measured variable at the current time is correlated with the value at previous times. We also know that the status of the measuring device also persists through time: if it is malfunctioning, it is unlikely to spontaneously fix itself. So our belief about the measured variable at the current time is based not only on the current measurement, but also on measurements made at past times. Likewise, the current sensor status is also inferred from current and past measurements.

The influence of the past on the present can be captured in a simple extension of the sensor model of §6.3. Let us suppose that the measured variable X_t is positively correlated with its value at the immediately preceding time step, X_{t-1}, and that the sensor status S_t, in the absence of measurements $\widehat{X}$, forms a Markov chain. The transition matrix for $S_t \,|\, S_{t-1}$ gives high probability to transitions from one state into the same state, and low probability for other transitions; in particular, the transition from a malfunctioning state in the "OK" state is very low. This sensor model with temporal dependencies is represented graphically in Figure 6.5. This kind of model has been called a "factorial hidden Markov model" [34], an "HMM(1,2) model" [76], and a "dynamic network model" [22].

Let us assign numerical values to the temporal dependency sensor model as follows.

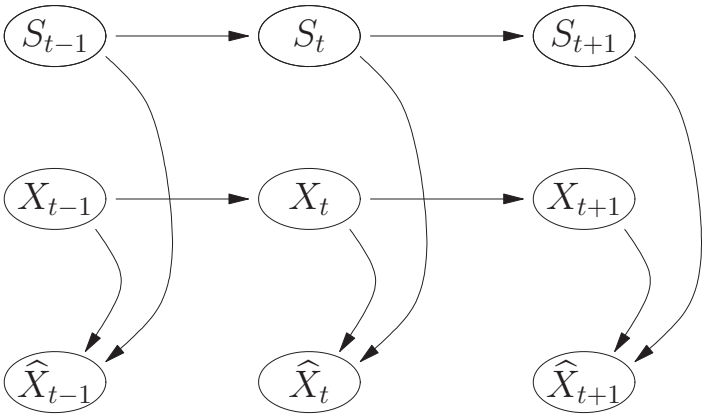

Figure 6.5: A sensor model which represents temporal dependence of sensor status from one time-step to the next, likewise the temporal dependence of the measured variable. To be specified: $p_{S_t|S_{t-1}}$ (sensor status transition), $p_{X_t|X_{t-1}}$ (evolution of measured variable; typically a decay model), and $p_{\widehat{X}_t|S_t,X_t}$ (measurement model).

The transition model for the measured variable is a simple linear correlation model,

$$X_t = \rho\, X_{t-1} + \epsilon \tag{6.9}$$

where ρ is a constant and $\epsilon \sim N(0, \sigma_\epsilon^2)$. The transition model for the sensor status is a column-stochastic matrix, shown in Table 6.1. Numerical values for the parameters of the sensor status transition model are given in Table 6.2.

Table 6.1: A model for sensor status transitions.

		From:		
		"OK"	"open circuit"	"strange"
	"OK"	$1-(\alpha_{10}+\alpha_{20})$	α_{01}	α_{02}
To:	"open circuit"	α_{10}	$1-(\alpha_{01}+\alpha_{21})$	α_{12}
	"strange"	α_{20}	α_{21}	$1-(\alpha_{02}+\alpha_{12})$

Table 6.2: Numerical values for the sensor transition model.

ρ	0.99
σ_ϵ	1
α_{10}	0.001
α_{20}	0.009
α_{01}	0.0001
α_{21}	0.009
α_{02}	0.009
α_{12}	0.001

The linear correlation model predicts that the uncertainty in X_t grows without bound as $t \to \infty$. This does not correspond with physical reality: typically the uncertainty grows rapidly for some time, then saturates near the size of the typical range of the measured variable. For chaotic systems, the long-term uncertainty is described by the invariant measure, while the short-term rate of growth of uncertainty is described by local Lyapunov exponents; error models for such systems are described in Ref. [44]. A more realistic transition model for $X_t \mid X_{t-1}$ should take both short-term growth and long-term saturation into account.

The computation of inferences for the temporal sensor model cannot be handled by the polytree algorithm alone, since the graph shown in Figure 6.5 contains loops. The loops are broken by instantiating the sensor status variables S_t and carrying out the polytree algorithm for each instantiation of the S_t; this is an example of the conditioning algorithm mentioned in §7.2. Although conceptually simple, the conditioning algorithm for the temporal sensor model requires computation time which grows like 3^N where N is the number of time steps. As mentioned in §5.9, more sophisticated algorithms for inference in heterogeneous belief networks are under investigation.

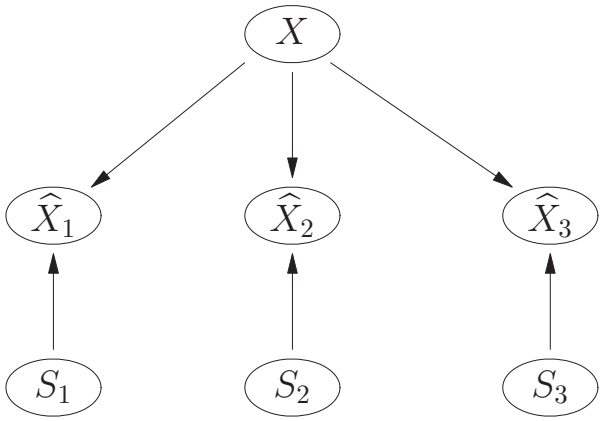

Figure 6.6: A model of redundant sensors measuring the same variable. X is the measured variable, and $\widehat{X}_1, \widehat{X}_2$, and $\widehat{X}_3$ are three measurements of X. To be specified: p_{S_k} (reliability of each sensor), p_X (long term distribution of measured variable), and $p_{\widehat{X}_k|S_k,X}$ (measurement model).

6.5 A model of redundant sensors

It is sometimes possible to install multiple sensors which all measure the same variable. In this case, each estimate serves to reduce the uncertainty of the measured variable, and we can check each sensor for consistency with the others to help detect sensor problems. A model of redundant sensors is shown in Figure 6.6. This model is similar to the simple sensor model shown in Figure 6.4, with more than one measurement. The conditional distributions of the measurements, $p_{\widehat{X}_k|S_k,X}$, $k = 1, 2, 3, \ldots$, need not be the same, and in an application it may be desirable to make measurements based on different physical principals so as to better cancel out the idiosyncrasies of each sensor. With redundant sensors, it becomes possible to distinguish between a sensor which has failed in an obvious way (e.g., failed with an open circuit) and a sensor which is only giving a measurement which is "strange" compared to the other sensors. For example, if the measurement error is about 1° F, then any measurement which is more than 3 or 4 ° F away from the others is apparently unreliable. Sensors can effectively "vote"

against outliers, with the posterior over the measured variable being dominated by the largest group of more-or-less consonant measurements. Let us consider a specific model which displays these qualitative characteristics.

For simplicity, let us assume all sensors are identical: the reliability is the same for all, and the measurement model is the same for all. As ever, the numbers specified in the following can be modified to better suit a particular application — the important thing is the structure, and we can adjust the numbers to suit our tastes.

- p_{S_k} — Let us assign $S = 0$ to the "sensor OK" state, $S = 1$ to the "open circuit" state, and $S = 2$ to the "strange measurement" state. Let us take $p_{S_k} = [0.990, 0.001, 0.009]$ for all sensors. That is, the sensor is usually working correctly, and if it's not, "strange" values are nine times as common as open circuits.

- $p_{\widehat{X}_k|S_k,X}$ — Let us take the distribution of the measurement $\widehat{X}_k$ given its parents S_k and X to be the same as the distribution of $\widehat{X}$ specified in §6.3. Let us also take $p_{\widehat{X}_k|S_k=2,X}$ to be a Gaussian distribution with a very wide variance, say $2\sigma_X$. In summary, we have

$$\widehat{X}_k \,|\, S_k = 0, X \;\sim\; N(X, \sigma^2_{\widehat{X}}) \tag{6.10}$$

$$\widehat{X}_k \,|\, S_k = 1, X \;\sim\; N(-46, \sigma^2_{\widehat{X}}) \tag{6.11}$$

$$\widehat{X}_k \,|\, S_k = 2, X \;\sim\; N(\mu_X, 2^2\sigma^2_X) \tag{6.12}$$

- p_X — Let us take the distribution of the measured variable the same as in §6.3, namely $X \;\sim\; N(\mu_X, \sigma_X)$.

Here are a few typical inferences computed from a model of three redundant sensors.

All sensors more or less agree. Suppose we have the measurements $\mathbf{e} = \{\widehat{X}_1 = 23, \widehat{X}_2 = 24, \widehat{X}_3 = 25\}$. Then RISO computes the posterior $p_{X|\mathbf{e}}$ as a Gaussian with mean 24.02 and standard deviation 0.5771. Note that the posterior is very nearly a Gaussian with mean $(23+24+25)/3$ and standard deviation $1/\sqrt{3}$; the difference is due to the influence of the prior p_X.

One sensor apparently has an open circuit. Suppose we have the measurements $\mathbf{e} = \{\widehat{X}_1 = 23, \widehat{X}_2 = 24, \widehat{X}_3 = -45.8\}$. Now RISO computes the posterior $p_{X|\mathbf{e}}$ as a Gaussian with mean 23.53 and standard deviation 0.7067. This posterior is very nearly a Gaussian with mean $(23+24)/2$ and standard deviation $1/\sqrt{2}$. That is, the apparently erroneous reading $\widehat{X}_3$ has been effectively excluded from the posterior computation. Examining the posterior of S_3, we see that most of the mass (0.9871) is placed on the "open circuit" state, while a little bit (0.01287) is on the "strange measurement" state. (Although the measurement is very close to the open circuit value, it is also well within the range of "strange" readings.)

Two sensors vote against the third. Suppose we have the measurements $\mathbf{e} = \{\widehat{X}_1 = 23, \widehat{X}_2 = 24, \widehat{X}_3 = 12\}$. In isolation, each of the three seems reasonable, since they are all well within the prior range of X and none is close to the open circuit value -46. However, $\widehat{X}_3$ doesn't agree well with the other two. In this case, RISO again excludes $\widehat{X}_3$ from the posterior, which again is very nearly Gaussian with mean $(23+24)/2$ and standard deviation $1/\sqrt{2}$. However, if we examine the posterior for S_3, this time we find all of the mass is on the "strange measurement" state.

The sensors all disagree. Suppose we have the measurements $\mathbf{e} = \{\widehat{X}_1 = 10, \widehat{X}_2 = 24, \widehat{X}_3 = 35\}$. Now the posterior is a Gaussian mixture of three components, each one corresponding to a measured value. Each status variable has almost all its mass on the "strange measurement" state.

Thus a belief network model of redundant sensors automatically gives us outlier detection and excludes apparently abnormal measurements from our beliefs about the measured variable. We needn't try to formulate rules for the combination or exclusion of measurements; these are determined automatically, according to the laws of probability, from the specification of each sensor.

6.6 Modeling a predictable measured variable

In some problems, we may be able to model a measured variable X which is to some degree predictable from the time of day, day of the week, or day of the year. For example, outdoor temperature is a variable of this kind. There are two sources of information about X: the time prediction, which supplies a π-message, and the sensor measurement, which supplies a λ-message. As usual, the posterior over X will just be the product of the π- and λ-messages; if the sensor is working correctly, the variance of the λ-message will be smaller, and it will dominate the posterior. Otherwise, the π-message will dominate the posterior. This model provides an automatic, gradual trade-off between the relative accuracies of each source of information about X.

The measurement model can be constructed without reference to the prediction model, and one might use the simple model described in §6.3.

The prediction model is typically a regression model of the form $p_\epsilon(X - F(t))$. The regression function F will often be a harmonic function such as $A \sin \omega t + B \cos \omega t$, with $\omega = 2\pi/(24\text{ h})$ or $\omega = 2\pi/(365\text{ d})$. One can employ as many frequencies as one likes, of course. The function F may also be a look-up table containing one entry for each hour in the day or each day in the year. Both harmonic functions and look-up tables are easily constructed from empirical data.

A prior p_t is formally required, but it will not enter the calculations unless there is no clock reading. If the clock is unreliable, it may be necessary to introduce a more complicated model for t, but in most applications there will always be an accurate time measurement. Typically p_t can be taken as a uniform distribution over $[0, 24)$ hours.

A model for outdoor dry-bulb temperature T was constructed from some data for a

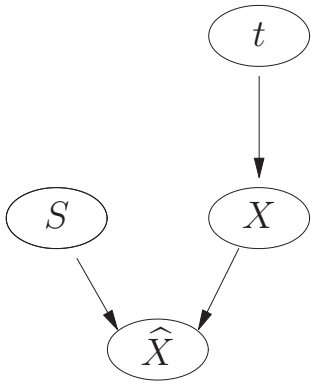

Figure 6.7: A measured variable which can be predicted by time (of day, of week, of year, etc.). To be specified: p_t (typically uniform over $[0, 24)$ hours), p_S (sensor reliability), $p_{X|t}$ (prediction model), and $p_{\widehat{X}|S,X}$ (measurement model).

site in Wisconsin. A regression of T on daily and yearly frequencies yields

$$p_{T|t}(T,t) = p_\epsilon(T - F(t)), \tag{6.13}$$

$$p_\epsilon(u) = g(u; 0, \sigma_\epsilon), \quad \text{with } \sigma_\epsilon = 9.521, \tag{6.14}$$

$$\begin{aligned} F(t) = {} & 49.82 - 3.629 \cos\frac{2\pi t}{24\text{ h}} - 2.978 \sin\frac{2\pi t}{24\text{ h}} \\ & -20.00 \cos\frac{2\pi t}{8760\text{ h}} - 8.452 \sin\frac{2\pi t}{8760\text{ h}} \end{aligned} \tag{6.15}$$

Let us assume the now-familiar the sensor reliability and measurement models from §6.3. Suppose we have a measurement taken just after midnight on July 2: $\mathbf{e} = \{t = 4369\text{ h}, \widehat{T} = 55°\text{ F}\}$. RISO reports the posterior for T as

$$T \,|\, \mathbf{e} \;\sim\; N(55.11, 0.9945^2) \tag{6.16}$$

We can see what the prediction alone says about temperature by omitting the observation, so $\mathbf{e} = \{t = 4369\}$. Then

$$T \,|\, \mathbf{e} \;\sim\; N(65.47, 9.520^2) \tag{6.17}$$

so the predicted temperature is somewhat greater than the observed; for this reason the posterior over T, taking the observation into account, is a little greater than the

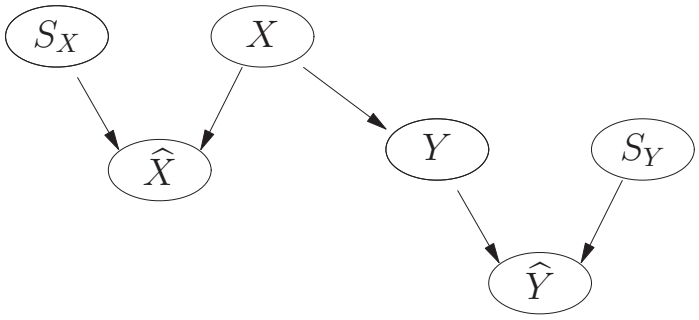

Figure 6.8: Measurement of related variables. A measurement of X also provides information about Y, and vice versa. To be specified: p_{S_X}, p_{S_Y} (reliability of each sensor), p_X (long term distribution of X), $p_{Y|X}$ (relation between X and Y), and $p_{\widehat{X}|S_X,X}$, $p_{\widehat{Y}|S_Y,Y}$ (measurement models).

observed value. However, the effect isn't much since the variance of the prediction is about 100 times greater than that of the observation. If there is an observation, but it is erroneous, e.g. $\mathbf{e} = \{t = 4369 \text{ h}, \widehat{T} = -46.3° \text{ F}\}$, the posterior for T is the same as if the observation were missing:

$$T \,|\, \mathbf{e} \;\sim\; N(65.47, 9.520^2) \tag{6.18}$$

Thus this model of a predictable variable is much like the simple sensor model of §6.3, but whereas the simple model has a fixed prior, the prediction yields a sharper posterior for the variable of interest in the absence of a reliable sensor reading.

6.7 A model of correlated measured variables

It is often the case that measured variables may be correlated to a greater or lesser extent; in engineering contexts this is often due to thermodynamic relations or unmodeled common influences. We can exploit the relation between two or more measured variables to reinforce beliefs about each one, if the sensors are working, and to substitute, in effect, an estimated value when a sensor is not working.

Let us consider a simple form of a "correlated variables" model with two measured variables, X and Y, as shown in Figure 6.8. There are measurements of each variable, which reinforce each other through the relation $p_{Y|X}$ of X and Y. There are six distributions to be specified in this model, most of which are now familiar from §§6.3–6.6.

- p_{S_X}, p_{S_Y} — As before, these are sensor reliability descriptions.
- $p_{\widehat{X}|S_X,X}$, $p_{\widehat{Y}|S_Y,Y}$ — These are measurement models.
- p_X — Description of prior beliefs about X.
- $p_{Y|X}$ — This distribution describes the relation between X and Y. This can take any convenient form; conditional Gaussian and regression models will be suitable for many problems. The relation may be established from empirical data or from first principles, or both.

As an example, consider a model which combines measurements of temperature T and relative humidity RH with a model of the relation between T and RH. A conditional Gaussian model for $RH \,|\, T$ was constructed from TMY2 data (shown in Figure 6.9) for Denver, Colorado. Some of the parameters for $p_{RH|T}$ are shown in Table 6.3; the complete description, which is a little verbose, may be found on the RISO web site, `http://civil.colorado.edu/~dodier`. The $RH \,|\, T$ model is a conditional distribution derived from a joint distribution of T and RH, which was fit by the expectation–maximization algorithm; see E for details on the calculation of conditional Gaussian distributions. The T–RH belief network also contains a sensor measurement and sensor status variable for T, likewise for RH. Here are the results of a few example calculations.

$\widehat{T} = 60$, *no RH measurement.* The T–RH model provides, in effect, a substituted value for RH when a direct measurement is missing. RISO computes the posterior as a bimodal mixture of Gaussians distribution, as shown in Figure 6.10. The peaks near 30° F and 61° F appear in the empirical data from which the conditional mixture was constructed; they may be due to different physical processes governing the relation between T and

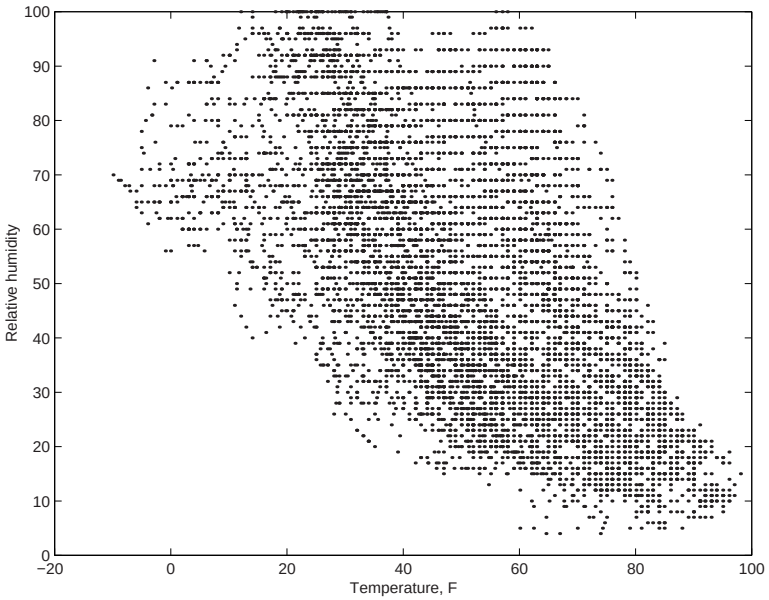

Figure 6.9: Temperature (horizontal axis) and relative humidity (vertical) data for Denver, Colorado, from the TMY2 database; 8760 points.

k	a_k	b_k	σ_k^2
1	-0.5657	64.23	81.50
2	-1.704	129.2	133.1
3	-1.455	146.4	72.01
4	0.02897	71.61	171.3

Table 6.3: Parameters for the conditional distribution $RH \,|\, T$. The form of each component is $p_{RH|T}(RH, T, k) = g(RH; a_k\, T + b_k, \sigma_k^2)$; additional parameters for computing the mixing coefficients are omitted, but may be found in the RISO description file.

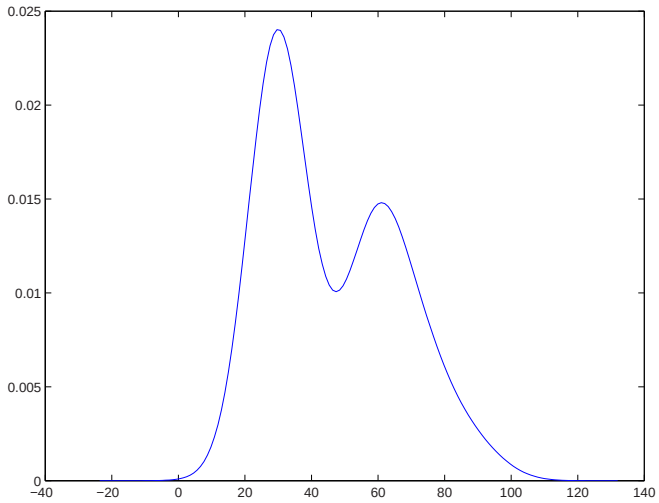

Figure 6.10: Posterior for RH given $\widehat{T} = 60$. The posterior is computed as a Gaussian mixture with eight components.

RH (say high and low pressure weather systems), or they may simply be artifacts of the particular data set.

$\widehat{T} = 10$, *no RH measurement.* A glance at Figure 6.9 shows that the cross-section of RH is narrowest at relatively low and high temperatures; as expected, the dispersion of the posterior for $RH \,|\, \widehat{T} = 10$ is substantially less than in the previous case, $RH \,|\, \widehat{T} = 60$. RISO reports the posterior as a mixture of Gaussians; the posterior is unimodal, with mean 71.29 and standard deviation 14.84, and only slightly non-Gaussian. The mixture of conditional Gaussians has the very attractive capability of modeling variables for which the shape of the conditional distribution varies over the range of the parent variable, a so-called "heteroskedastic" model.

$\widehat{T} = -45.3, \widehat{RH} = 80$. In this case, there is a temperature measurement, but it is apparently spurious. RISO computes a spline approximation to the posterior $T \,|\, \widehat{RH}$, shown in Figure 6.11. The posterior of T is inferred largely from the measurement of RH (the prior of T has a slight effect); although the model of the relation $RH \,|\, T$ is

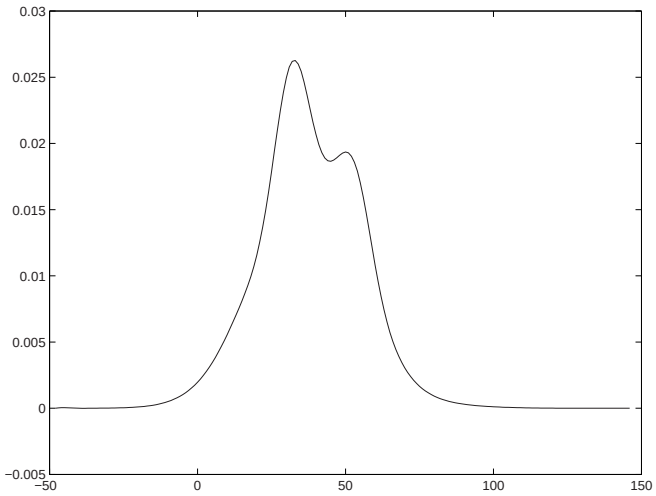

Figure 6.11: Spline approximation to the posterior $T \,|\, \widehat{T} = -45.3, \widehat{RH} = 80$. The spline comprises 300 support points.

expressed with RH on the downstream side, information propagates upward from $\widehat{RH}$ in the form of a λ-message.

6.8 Learning a predictive sensor model

From the extended logic point of view, there is nothing essentially different about parameters as opposed to measured variables or hypothesis variables in a belief network, since all kinds of uncertainty are handled by the laws of probability. We can profitably represent as a belief network a problem in which the parameters of a prediction model appear as variables; then we can incorporate prior information about the parameters, update beliefs about the parameters with incoming data, and propagate uncertainty about the parameters into the predictions we make. Such a model is an example of a "belief network for learning;" see Ref. [11] for an excellent introduction.

An example of a model which shows these qualitative features in shown in Figure 6.12. This belief network represents a prediction model with autoregressive noise; the "prediction" part is the model which propagates upstream evidence $\mathbf{e}^+$ into X, and

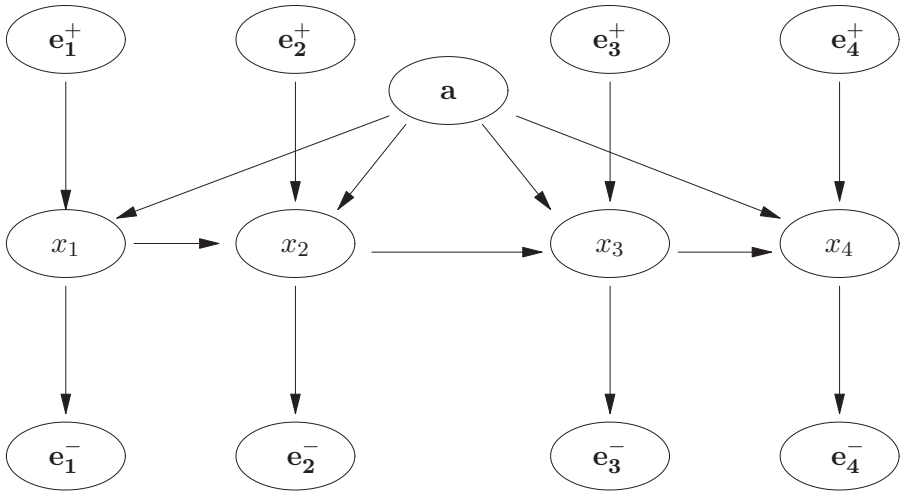

Figure 6.12: Learning a predictive sensor model with autoregressive noise.

the "autoregressive" part describes how information X from one time step can sharpen beliefs about X at the next time step. The transition from X_t to X_{t+1} is modeled as

$$X_{t+1} \,|\, X_t \;\sim\; N(\rho X, \sigma_\epsilon) \tag{6.19}$$

which is equivalent to an AR(1) model with correlation coefficient ρ and additive noise magnitude σ_ϵ. The parameters ρ and σ_ϵ, and any other parameters for the transition model, may be considered components of α in the diagram, so α is a multi-dimensional variable. The prediction model $p_{X_t|\mathbf{e}_t^+}$ may take any convenient form, for example, a mixture of conditional Gaussians, a look-up table, or a sum of squashing functions. To complete the belief network, additional downstream evidence $\mathbf{e}^-$ enters the model; such evidence often represents sensor measurements.

A belief network of the form shown in Figure 6.12 has been constructed, but detailed results are not yet available. See also §9.1 for a discussion of a related class of models.

Chapter 7

APPLICATION: SELECTING RATES TO MINIMIZE ENERGY AND DEMAND COSTS

At present, most electricity users have one electricity supplier, which is determined by geography, and their energy and demand schedules are selected perhaps once a year and not changed otherwise. However, under a scheme known as "retail wheeling," there are several suppliers which all share the same distribution lines, and users may select from any supplier. Thus the customer may select from several rate schedules, which give different trade-offs between energy and demand costs; each supplier generally tries to encourage users, through its rate schedule, to use electricity in such a way as to minimize the supplier's total capital and operating costs. (For example, a supplier may set a low energy rate but a high demand rate; this encourages users to make their electricity demand as constant as possible over the course of a day.) The schedules may be arbitrarily complex; energy and demand costs usually vary according to the time of day, day of the week, and amount (in kWh or kW) requested by the customer, and there may be many other factors taken into account.

Retail wheeling is not widely available in the United States, but it is foreseen that in the next several years, state utility commissions will require suppliers to offer their services in a retail wheeling scheme. This will offer customers the opportunity of reducing their electricity costs by selecting the supplier which will give them the best rate schedule for the customer's purposes. So let us now consider the problem, from the customer's point of view, of selecting the rate schedule which yields the lowest expected total electricity cost. The problem, from the supplier's point of view, of setting the energy and demand rates according to its economic circumstances, is somewhat more difficult and will not be considered in this dissertation.

The customer's problem, as stated, is not strictly a problem in belief network modeling. As probability models, belief networks handle only beliefs, but not values (e.g., monetary costs) or decisions. The larger framework of decision theory is needed to take values into account and make a decision. For this problem, I will describe a belief network that could be used to compute a probability distribution over different energy use scenarios, then show how to incorporate the cost information to compute expected energy costs. For simplicity, I will assume that the value of money is proportional to its quantity, and that its value does not change over time. However, in a more careful treatment, other value functions such as a logarithmic function might be considered (which makes the first dollar more valuable than the last), and the difference between the nominal value of money and its present value should be taken into account.

Let us assume the customer selects a rate schedule, just before the beginning of each billing period (typically a calendar month), to minimize expected total electricity cost over the billing period. Given this, the customer's decision proceeds as follows.

1. *Construct a belief network model of energy use.* The customer constructs a model of its electricity use over the billing period, which is typically one month. The belief network needs to encompass all the variables within the billing period, since the demand cost depends on every recorded energy use during the billing period.

2. *Compute distributions for the belief network variables.* The customer computes a distribution over the energy use in each time slice, and then uses the distributions over energy use in each time slice (typically one hour, but sometimes shorter) to compute a distribution over the maximum energy use.

3. *Computing expected costs using the rate schedules.* The customer uses the distributions over energy use in each time slice to compute expected energy cost, and uses the distribution over maximum energy use to compute the expected demand cost.

4. *Select a rate schedule.* The decision to use one rate schedule or another will simply be to choose the rate schedule which yields lowest expected cost.

We shall now consider each of these steps in turn.

7.1 Specification of a belief network for rate selection

To illustrate the use of a belief network to compute the distributions over total and maximum energy use which are needed for a calculation of expected cost, a simple building model was constructed as a belief network, shown in Figure 7.1. Since the problem requires predicting energy use at future times, a relatively detailed model, based on first principles, should be constructed. (Regression-based models, I believe, are more suitable to applications in which the data available at the time predictions are needed is more certain to be similar to the data that were available for training. However, in the absence of engineering knowledge, regression models could be used.) This model is a temporal belief network, with one slice per hour in the electricity billing period, usually one calendar month. The electricity use in the building is divided into four components:

- Cooling load due to heat transfer into the building through the envelope.
- Cooling load due to occupants. Only the sensible load is considered.
- Cooling load due to sensible heat gain in the outside air brought in for ventilation.
- Electricity use by lights and equipment in the building.

The model building is located in Denver. For simplicity, the building is assumed to be low and flat, so that heat gains through the roof dominate envelope heat transfer. The roof consists of 4 inches of concrete. Heat transfer through the roof is modeled by a discrete convolution of the effective outside temperature $T_{sol-air}$, called the "sol-air" temperature, with a transfer function describing the response of the roof to changes in

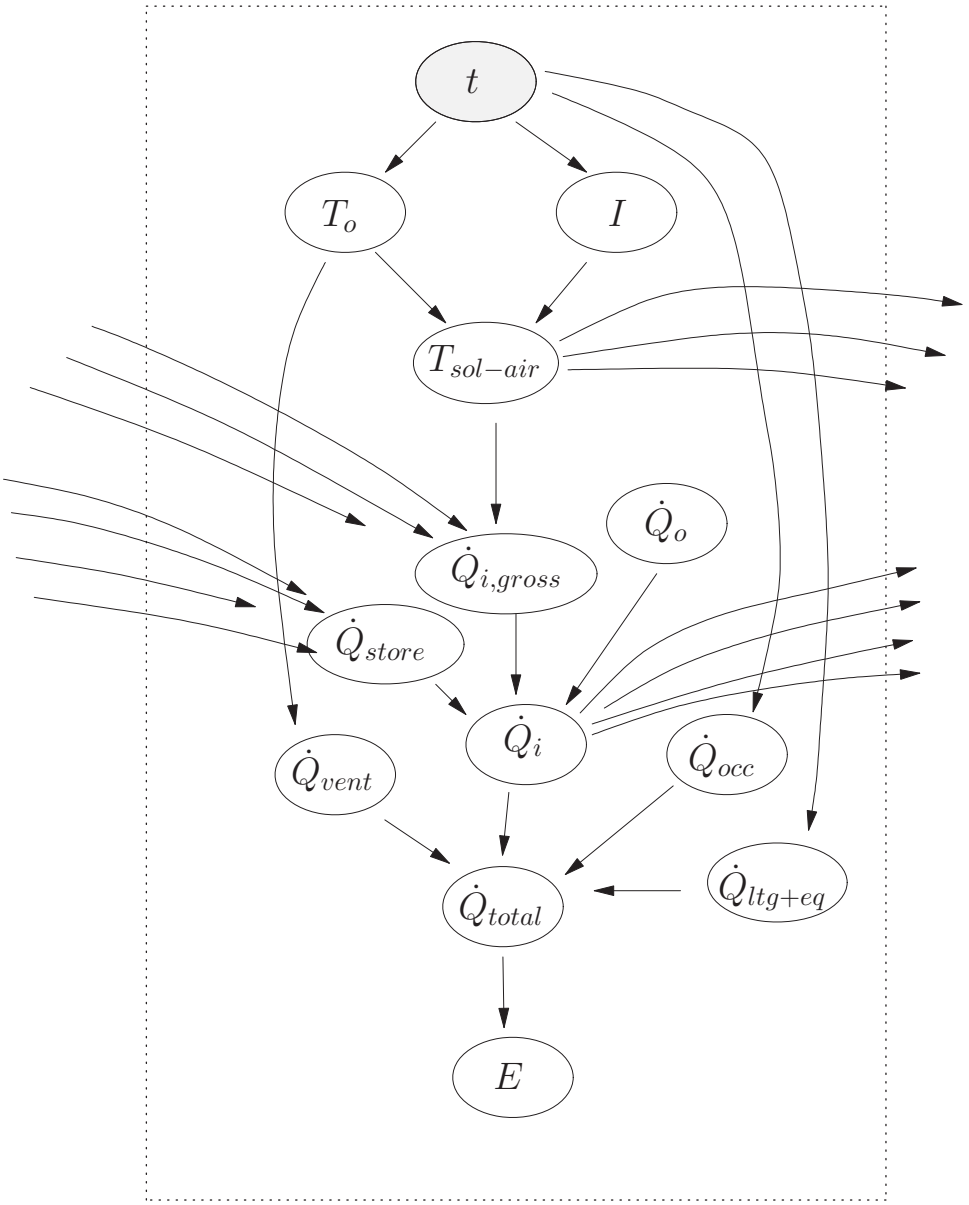

Figure 7.1: One slice of a belief network to predict energy use in a building, showing the decomposition of the modeling problem into energy use attributed to lighting and equipment, occupants, heat gain due to ventilation, and envelope loads. Time is t, the net heat gain through the envelope is $\dot{Q}_i$, and the electrical demand is E. Arrows leading into the slice show the dependence of variables on past times; likewise, future variables are dependent on this slice, as shown by the arrows leading out. Arrows showing dependence on a constant variable (namely, the envelope area and the indoor air temperature) have been omitted for clarity. One slice is created for each hour of the month.

temperature. The heat transfer, per hour, through the roof into the conditioned space is given by [55]

$$\dot{Q}_i(t) = A \sum_{k=0}^{m} b_k \, T_{sol-air}[t-k] - \sum_{k=1}^{m} d_k \, \dot{Q}_i[t-k] - A \, T_i \sum_k c_k \tag{7.1}$$

where A is the surface area, T_i is the indoor temperature (assumed in this study to be a constant 72° F), and the b_k, d_k, and c_k are constants which depend on the material of the envelope. The following values [55, Table 7-24] were used.

	$k=0$	$k=1$	$k=2$	$k=3$	$\sum_k c_k$
b_k	0.0078	0.0705	0.0355	0.0011	0.1149
d_k	-0.8789	0.0753	-0.0001	0	

Note that time t affects energy use only through its effects on intermediate variables such as temperature and occupancy. One could therefore substitute more accurate predictions for the intermediate variables, should they become available. For example, the National Weather Service may predict that the winter will be unusually warm — this prediction could be used to construct a distribution over temperature which is substituted in place of the prediction based solely on time.

Figures 7.2 and 7.3 shows how sub-networks for energy use during one time slice can be linked together to form a belief network for prediction of the maximum energy use over the course of the demand cost billing period. Belief networks for different billing periods need not be linked. Note that this grouped or linked belief network only computes a distribution over the maximum energy use — it does not compute the demand cost. The computation of energy and demand costs is considered in §7.3.1.

7.1.1 *On the stability of a transfer function model*

As noted above, the transfer function model of net inward heat transfer $\dot{Q}_i$ expresses $\dot{Q}_i[t]$ as a linear combination of $\dot{Q}_i[t-1]$, $\dot{Q}_i[t-2]$, $\dot{Q}_i[t-3]$, ..., and other terms. Thus the transfer function model is an infinite impulse response (IIR) linear filter of the sequence $\dot{Q}_i$. The output at a time t can be expressed exactly in terms of t and

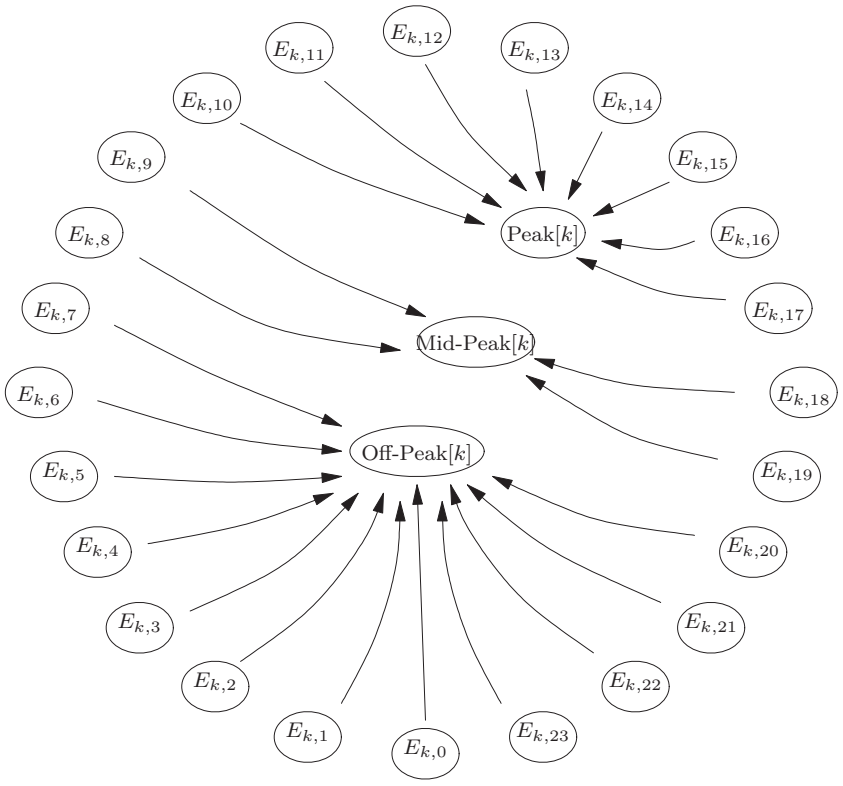

Figure 7.2: Energy use predictors from each time slice, $E[k]$, grouped together into a belief network to compute a distribution over the daily maximum demand for day k. Only the variable E from each time slice is shown here; the parents of E, which are shown in Figure 7.1, are omitted for clarity.

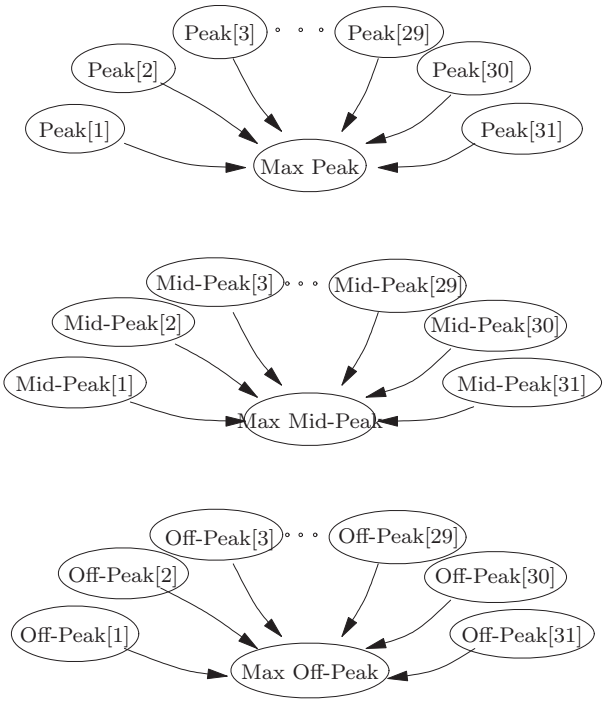

Figure 7.3: Combining daily maxima over the billing period to obtain the maximum for each category (peak, mid-peak, and off-peak) in the demand rate schedule. The parents of the variables Peak[k], which are shown in Figure 7.2, are omitted here for clarity.

the roots of the characteristic polynomial associated with the filter; if some of the roots have magnitude greater than 1, the filter will have modes which grow exponentially with t, and the filter is said to be *unstable* [70, § 12.9]. The characteristic polynomial is defined in terms of the coefficients of the $\dot{Q}_i[t-k]$: if the filter is

$$\dot{Q}_i[t] = -d_1\,\dot{Q}_i[t-1] - \cdots - d_m\,\dot{Q}_i[t-m] + \text{ terms not involving any } Q_i[t-k] \quad (7.2)$$

then the characteristic polynomial is

$$z^m + \sum_{k=1}^{m} d_k\, z^{m-k} \quad (7.3)$$

Note that only the terms involving past values of $\dot{Q}_i$ affect the stability of the filter.

It turns out that the mean and variance of $\dot{Q}_i[t]$ can be described in a simple way in terms of the means and variances, respectively, of past values $\dot{Q}_i[t-k]$. Recall that the mean of a linear combination is a linear combination of the mean of each term, and the variance of a linear combination is a linear combination of the variance of each term (replacing each coefficient in the combination by its square); these relations hold for any kind of distribution. So we have

$$E[\dot{Q}_i[t]] = A \sum_{k=0}^{m} b_k\, E[T_{sol-air}[t-k]] - \sum_{k=1}^{m} d_k\, E[\dot{Q}_i[t-k]] - A\,T_i \sum_{k=0}^{m} c_k \quad (7.4)$$

for the expected value, and

$$\mathrm{Var}[\dot{Q}_i[t]] = A^2 \sum_{k=0}^{m} b_k^2\, \mathrm{Var}[T_{sol-air}[t-k]] + \sum_{k=1}^{m} d_k^2\, \mathrm{Var}[\dot{Q}_i[t-k]] \quad (7.5)$$

So we have here two linear IIR filters, one for the mean of $\dot{Q}_i[t]$ and one for the variance. The characteristic polynominal for the mean is just Eq. 7.3, but the filter for the variance has the characteristic polynomial

$$z^m - \sum_{k=1}^{m} d_k^2\, z^{m-k} \quad (7.6)$$

Even if the filter for the mean is stable, the filter for the variance may not be. For this reason, the transfer function coefficients given in reference works are not necessarily

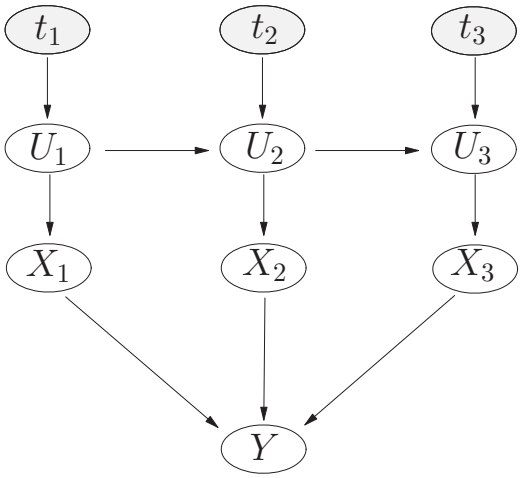

Figure 7.4: A belief network, essentially similar to the building model, for which the polytree algorithm fails. The parents of Y, which may be the maximum or sum of the X's, are not independent. But if the U's are given, the X's become independent — this is the basic idea of the conditioning algorithm.

usable in a calculation (such as a belief network inference) which propagates variance as well as expected value. Generally, structures with greater thermal mass have larger values for the d_k; for example, the parameters [55, Table 7-24] for a 4 inch heavy concrete wall yield a stable variance, but the parameters for an 8 inch heavy concrete wall do not. Stability could probably be recovered by working with a time interval shorter than one hour, but this has not yet been investigated.

7.2 A conditioning algorithm for inferences in the building model

Because of the dependence of the heat storage term on past values of $\dot{Q}_i$ and the dependence of the inward heat transfer term on past values of $T_{sol-air}$, the polytree algorithm cannot correctly compute a distribution over daily or monthly maxima. Figure 7.4 illustrates the problem. The problem is that the parents of the maximum (called Y in the figure) are not independent in the absence of downstream evidence. Thus their

joint distribution does not factor as indicated in Eq. 5.2 into a product of π-messages. One way to cope with this is to instantiate the U variables which link the time slices together. According to the rules of *d*-separation (§2.6.1), this breaks the dependencies between the slices, and then the polytree algorithm can be applied to compute the posterior for Y. The general approach exemplified by this tactic is called *conditioning*: if U is a variable or set of variables which make a loopy graph into a polytree when instantiated (a so-called "cut-set"), then we can express a calculation of $p(Y|\mathbf{e})$ as a weighted average over instantiations of U, like so:

$$p(Y|\mathbf{e}) = \int p(Y|U,\mathbf{e})\; p(U|\mathbf{e})\; dU \tag{7.7}$$

Since U is a cut-set, the polytree algorithm applies to $p(Y|U,\mathbf{e})$. Typically, the integration over U is carried out as an average over some number of discrete instantiations of U; thus the result $p(Y|\mathbf{e})$ is a mixture of all the $p(Y|U,\mathbf{e})$, usually weighted equally.

In the case of the building model belief network, the variables $\{T_o[k],\, T_{sol-air}[k],\, \dot{Q}_i[k]\}$ are a convenient cut-set. Instantiations of the cut-set are generated by setting the time variable, $t[k]$, in each time slice, computing the posterior of each cut-set variable in turn, and instantiating each cut-set variable to a random value sampled from its posterior distribution. This amounts to generating a "typical" trajectory or simulation of the cut-set variables. Once all the cut-set variables (three samples in each time slice) have been sampled, the computation of the daily and monthly maxima and summations can be computed by the polytree algorithm. Six sampling runs were made for the results presented in this chapter; a larger number would yield more accurate results, but it was found the maxima and summations did not vary too much from run to run, so six is enough, we hope.

The conditioning algorithm was built on top of RISO for this particular inference problem; RISO really needs its own built-in algorithm (whether conditioning or something else) for coping with loopy belief networks, as discussed at slightly greater length in §5.9.

7.2.1 *Computation of a distribution over maximum energy demand*

Once the cut-set variables are instantiated, the inference problem becomes much easier. To obtain a distribution over the maximum energy demand, the electrical demand is first computed for each time slice, and these variables are independent given the cut-set variables. The distribution of the maximum of a set of independent variables is most easily stated in terms of the distribution function, which is the antiderivative of the probability density. Suppose the variables $X_1, \ldots, X_n$ have distribution functions $F_1, \ldots, F_n$. Then the maximum of $X_1, \ldots, X_n$ has the distribution function

$$F_{max}(x) = F_1(x) \cdots F_n(x)$$

That is, the distribution function of the maximum is just the product of the individual distribution functions. The probability density of the maximum is found by differentiating this expression. The distribution of the maximum may be harder to work with than the individual distributions, but it is not difficult to compute expected values numerically, since all that is needed is a 1-dimensional integration. Note that even if the energy demand variables in every time slice have Gaussian distributions, the maximum energy demand will not have a Gaussian distribution.

7.3 Computing expected costs using rate schedules

7.3.1 *General approach for cost calculations*

Rate schedules, for both energy and demand costs, are generally similar to this table.

Block	On-peak	Mid-peak	Off-peak
0–10 MWh	0.12 \$/kWh	0.10 \$/kWh	0.04 \$/kWh
10–20 MWh	0.10 \$/kWh	0.08 \$/kWh	0.02 \$/kWh
$>$ 20 MWh	0.08 \$/kWh	0.06 \$/kWh	0.02 \$/kWh

Rate schedules may be arbitrarily complicated, and sometimes require a small book to completely describe them. For purposes of illustration, we will consider energy and demand schedules of the following tabular form. Each column in a rate schedule, such as

the one shown above, is considered a separate table and applied to a different variable.

Lower limit	Cost per unit
α_1	β_1
α_2	β_2
α_3	β_3
$\vdots$	$\vdots$
α_n	β_n

Thus there are a number of simple tables equal to the number of columns in the energy cost schedule plus the number of columns in the demand cost schedule. Each table is applied to calculating the cost of a different variable: the on-peak maximum demand, the on-peak total energy, the mid-peak maximum demand, the mid-peak total energy, etc. These variables will be generically denoted x in the equations which follow. In this scheme, the cost of x units (kWh or kW) is

$$C(x) = \Delta\alpha_1\beta_1 + \Delta\alpha_2\beta_2 + \cdots + \Delta\alpha_{k-1}\beta_{k-1} + (x - \alpha_k)\beta_k \tag{7.8}$$

with $\Delta\alpha_i = \alpha_{i+1} - \alpha_i$ and $k = \arg\max_i \alpha_i < x$. Usually $\alpha_1 = 0$. The expected value of C with respect to the distribution p of x is

$$\begin{aligned}
E[C] &= \int_0^{+\infty} C(x)\, p(x)\, dx \\
&= \int_{\alpha_1}^{\alpha_2} C(x)\, p(x)\, dx + \int_{\alpha_2}^{\alpha_3} C(x)\, p(x)\, dx + \cdots \\
&\quad + \int_{\alpha_{n-1}}^{\alpha_n} C(x)\, p(x)\, dx + \int_{\alpha_n}^{+\infty} C(x)\, p(x)\, dx \\
&= \int_{\alpha_1}^{\alpha_2} (x-\alpha_1)\beta_1\, p(x)\, dx + \int_{\alpha_2}^{\alpha_3} (\Delta\alpha_1\beta_1 + (x-\alpha_2)\beta_2)\, p(x)\, dx + \cdots \\
&\quad + \int_{\alpha_n}^{+\infty} (\Delta\alpha_1\beta_1 + \Delta\alpha_2\beta_2 + \cdots + (x-\alpha_n)\beta_n)\, p(x)\, dx \\
&= \beta_1 \int_{\alpha_1}^{\alpha_2} (x-\alpha_1)\, p(x)\, dx + \beta_2 \int_{\alpha_2}^{\alpha_3} (x-\alpha_2)\, p(x)\, dx + \cdots \\
&\quad + \beta_n \int_{\alpha_n}^{+\infty} (x-\alpha_n)\, p(x)\, dx \\
&\quad + \Delta\alpha_1\beta_1(1 - F(\alpha_2)) + \Delta\alpha_2\beta_2(1 - F(\alpha_3)) + \cdots \\
&\quad + \Delta\alpha_{n-1}\beta_{n-1}(1 - F(\alpha_n)) \qquad (7.9)
\end{aligned}$$

noting that $F(+\infty) = 1$.

In general, F is a product of cumulative distribution functions, and for numerical purposes it is easier to work with F than its derivative p, the density function. The integrals in Eq. 7.9 can be expressed in terms of the c.d.f.,

$$\int_{\alpha_k}^{\alpha_{k+1}} (x-\alpha_k)\, p(x)\, dx = (\alpha_{k+1} - \alpha_k)\, F(\alpha_{k+1}) \; - \int_{\alpha_k}^{\alpha_{k+1}} F(x)\, dx \qquad (7.10)$$

by integrating by parts. Substituting Eq. 7.10 into 7.9 yields a simpler formula,

$$E[C] = \beta_1 \int_{\alpha_1}^{\alpha_2} 1 - F(x)\, dx + \cdots + \beta_{n-1} \int_{\alpha_{n-1}}^{\alpha_n} 1 - F(x)\, dx + \beta_n \int_{\alpha_n}^{+\infty} 1 - F(x)\, dx \qquad (7.11)$$

Once distributions have been computed over the energy use for each time slice within a billing period, and a distribution over the maximum energy use has also been computed, it is easy to compute the expected energy and demand costs for the billing period. The total expected cost is just the sum of the expected energy cost and the expected demand cost.

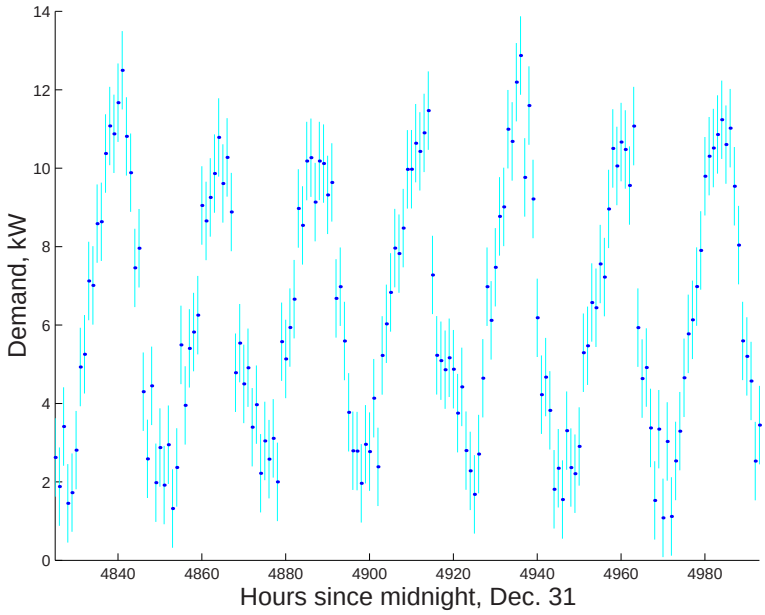

Figure 7.5: A simulation of typical electrical demand in the month of July; this figure shows seven days in the middle of the month. This sequence is a by-product of the conditioning algorithm described in §7.2. The bars represent the standard deviation of the posterior distribution of electrical demand, which is nearly constant from hour to hour and approximately equal to 1.

7.3.2 *Computation of costs for the model building*

Calculations of demand and energy costs were carried out for the month of July, using the building model belief network and the rate structures shown in Tables 7.1 through 7.4. Posterior distributions over the monthly peak, mid-peak, and off-peak demand were computed by the conditioning algorithm described in §7.2, as were posterior distributions for the monthly peak, mid-peak, and off-peak energy use. As part of the calculation, distributions over the energy demand at each hour of the month were computed; one sequence of these distributions, which corresponds to one instantiation of the cut-set variables $\{T_o[k], T_{sol-air}[k], \dot{Q}_i[k]\}$, is shown in Figure 7.5. The distribution over outdoor temperature is assumed to be a Gaussian distribution having a standard

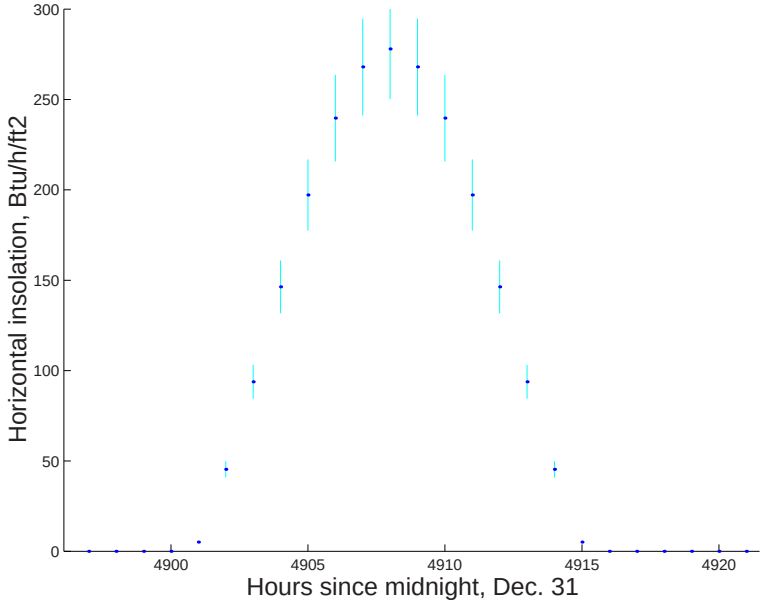

Figure 7.6: Posterior distributions over hourly horizontal insolation for one day in July; these is almost identical to the distributions calculated for other days in July. This figure shows the distribution, represented as a point for the mean value and a bar for the standard deviation, of insolation at each hour. In the building model belief network, it is assumed the error in the calculated insolation is proportional to the magnitude of the insolation, so the error bars are wider at midday than in the morning or evening.

deviation of 10° F and a mean value given by a sinusoid with two components, one for daily variation and one for yearly variation. The distribution over horizontal insolation is illustrated in Figure 7.6. Every day in July has a daily insolation sequence which is almost the same as the one shown in Figure 7.6. It is assumed the error in the calculated insolation is multiplicative, with a standard deviation equal to 10% of the calculated value.

The posterior distributions for the maximum demand during peak, mid-peak, and off-peak hours are all shown in Figure 7.7. Each of these distributions is a mixture of the posterior distributions computed from the six instantiations of the cut-set variables;

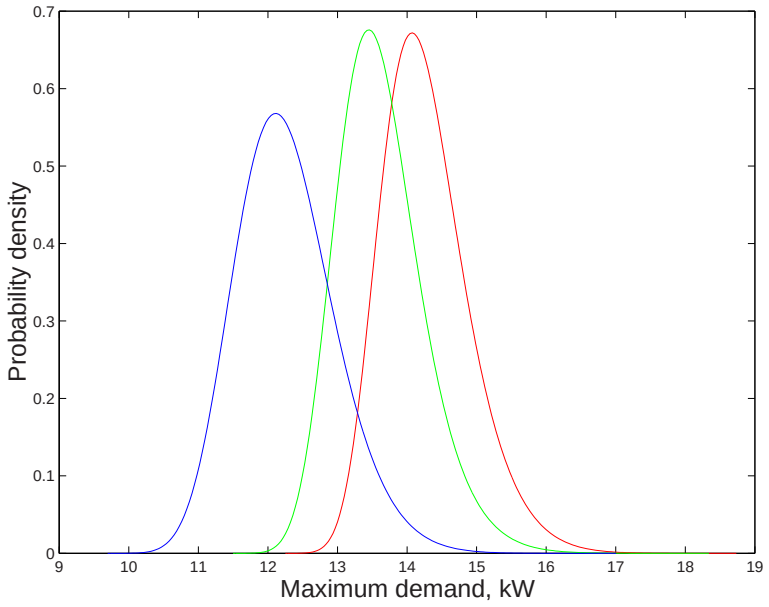

Figure 7.7: Posterior distributions for the maximum demand during peak (at right), mid-peak (center), and off-peak hours (at left). Each one is slightly skewed to the right; this is typical of the distributions of maxima.

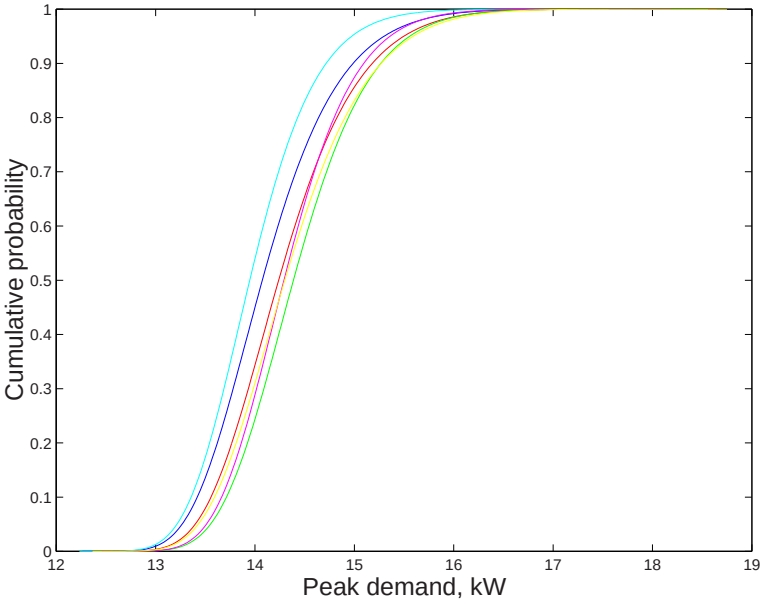

Figure 7.8: Cumulative distribution functions for six instances of the posterior of maximum peak-hours electricity demand.

a spline approximation (Appendix D) was constructed for each distribution computed from one instantiation of the cut-set, so the distributions shown in Figure 7.7 are mixtures of spline approximation densities. There was not much variation among the posteriors computed from each instantation; for example, Figure 7.8 shows that the six posteriors computed for the maximum demand during peak hours are all quite similar, and their median values are all in the range 13.8 to 14.4; the median of the mixture of these six distributions is about 14.21. This suggests that six instantiations is enough to capture the variability electrical demand due to the cut-set variables.

The posterior distributions for the maximum demand at different hours of the day are needed for the calculation of expected energy costs, as described in §7.3.1. Expected costs were computed according to the cost schedules presented in Tables 7.1 through 7.4.

The total costs (energy and demand) for each of the two utilities are compared in Table 7.5. As can be seen in the table, Utility A gives a better energy rate for our

Table 7.1: Energy cost schedule for Utility A. Units are cents per kWh.

Range	Peak	Mid-Peak	Off-Peak
0 – 1000 kWh	6	5	2
1000 — 2000 kWh	5	4	1
2000 kWh & up	4	3	1

Table 7.2: Energy cost schedule for Utility B. Units are cents per kWh.

Range	Peak	Mid-Peak	Off-Peak
0 – 500 kWh	9	6	4
500 — 1500 kWh	7	5	3
1500 kWh & up	5	4	3

example, while Utility B gives a better demand rate; the overall cost is somewhat lower for Utility B ($551) compared to Utility A ($608). The grand total of energy use for the month of July, including all hours of the day, has a Gaussian distribution with mean 4538 kWh and standard deviation 49 kWh; this yields blended energy and demand rates equal to 13.4 cents per kWh and 12.1 cents per kWh for A and B, respectively. To complete our planning process, we would want to compute energy and demand costs for every month in the year, and each month choose the utility whose rates give the lowest total cost.

Table 7.3: Demand cost schedule for Utility A. Units are dollars per kW.

Range	Peak	Mid-Peak	Off-Peak
0 – 12 kW	14	10	8
12 — 15 kW	10	8	6
15 kW & up	8	6	4

Table 7.4: Demand cost schedule for Utility B. Units are dollars per kW.

Range	Peak	Mid-Peak	Off-Peak
0 – 10 kW	9	8	7
10 — 13 kW	7	7	5
13 kW & up	5	6	4

Table 7.5: Comparison of total (energy and demand) costs, computed according to rate schedules from Utilities A and B.

		Utility A	Utility B
Energy	Peak	$ 116	$ 148
	Mid-Peak	47	52
	Off-Peak	24	48
	(subtotal)	188	248
Demand	Peak	191	117
	Mid-Peak	133	105
	Off-Peak	97	81
	(subtotal)	421	303
		$ 608	$ 551
Blended rate, cents/kWh		13.4	12.1

Chapter 8

ADDITIONAL BELIEF NETWORK APPLICATIONS

8.1 A model of a heating coil

A schematic diagram of a heating coil is shown in Figure 8.1. This heating coil could be used to recover heat rejected from a refrigeration system, for the purpose of space or process heat. In this scheme, there is another heat exchanger (not shown) between the refrigeration system, which uses a halocarbon as its working fluid, and the heating coil, which uses water as its working fluid. A belief network model of the heating coil is shown in Figure 8.2. The heating coil model was constructed by Kriengkrai Assawamartbunlue as part of his research (to be reported in his dissertation), and minor modifications were made by the present author. The dry-bulb temperature of the exhaust air is calculated by the following equations.

$$C_{air} = m_{air} \cdot C_{pa} \tag{8.1}$$

$$C_w = m_w \cdot C_{pw} \tag{8.2}$$

$$C_{min} = \min(C_{air}, C_w) \tag{8.3}$$

$$C_{max} = \max(C_{air}, C_w) \tag{8.4}$$

$$C = \frac{C_{min}}{C_{max}} \tag{8.5}$$

$$NTU = \frac{UA}{C_{min}} \tag{8.6}$$

$$n = NTU^{-0.22} \tag{8.7}$$

$$\eta = 1 - \frac{\exp\left(\exp(-NTU \cdot C \cdot n) - 1\right)}{C \cdot n} \tag{8.8}$$

$$Q = \eta \cdot C_{min} \cdot (T_w - T_{db,ent}) \tag{8.9}$$

$$T_{db,lvg} = \frac{Q}{C_{air}} + T_{db,ent} \tag{8.10}$$

This is a standard heat transfer model, found, for example, in Ref. [43].

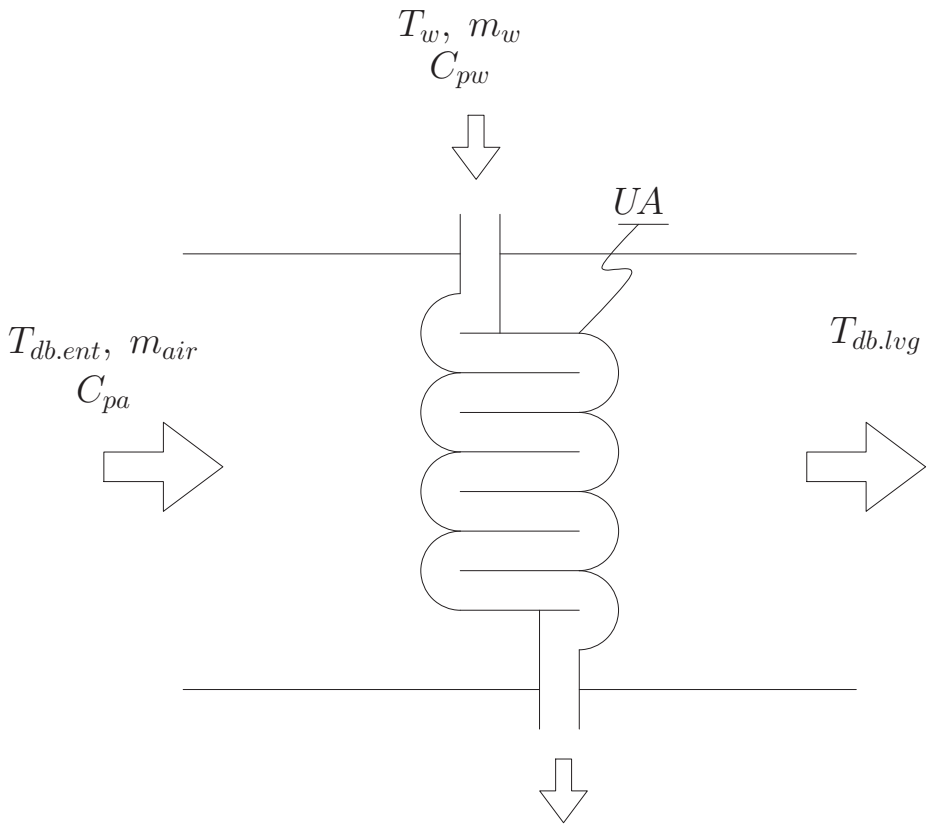

Figure 8.1: Schematic diagram of a heating coil, showing the variables which appear in the belief network shown in Figure 8.2.

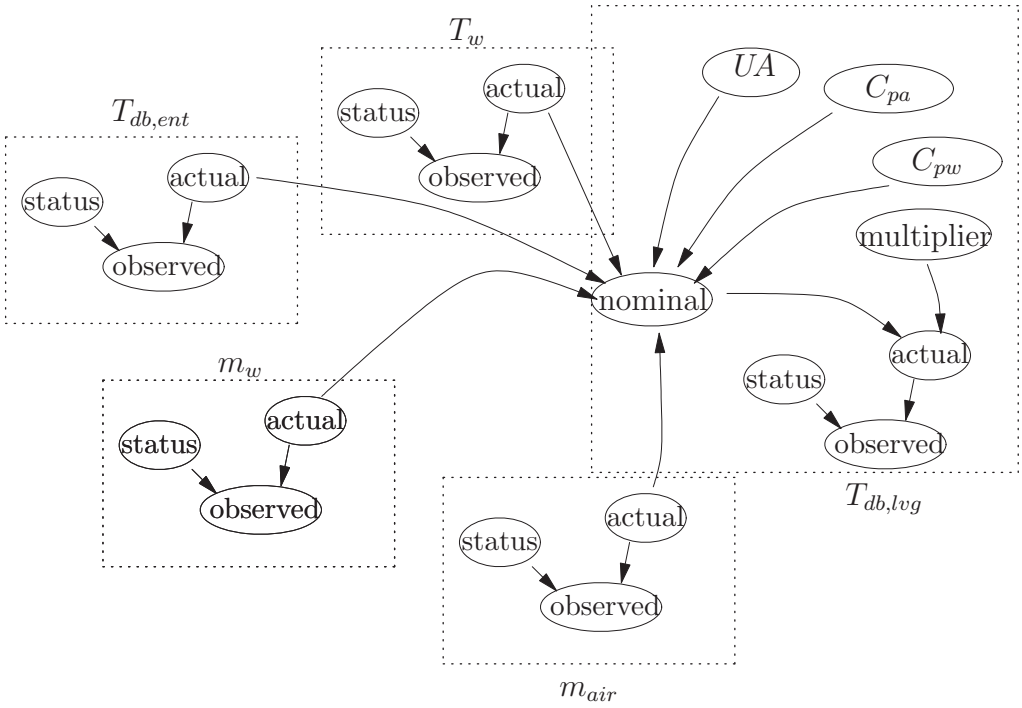

Figure 8.2: A belief network for the heating coil. The relation of the variable $T_{db,lvg}.nominal$ to its parents is described by the heat transfer model, Eqs. 8.1–8.10. Measured variables are described by simple sensor models; the unmeasured variables UA, C_{pa}, and C_{pw} are simple variables, with values which are usually fixed. This belief network is implemented as a distributed belief network: each dotted box encloses the variables in a sub-network, with the name given at the margin of the box; variables with the same name in different sub-networks are distinguished by including the name of their sub-network, e.g., $T_w.actual$ is distinct from $m_{air}.actual$.

To decompose the modeling problem into a slightly more comprehensible form, the belief network was implemented as a distributed belief network, with the variables corresponding to each sensor model in a separate sub-network. The names of the variables are abbreviations, as shown in this list.

$T_{db,ent}$	Dry-bulb temperature, air influx
$T_{db,lvg}$	Dry-bulb temperature, air exhaust
T_w	Temperature of water entering heat exchanger
m_{air}	Mass flow rate of air
m_w	Mass flow rate of water through heat exchanger
UA	Overall heat transfer coefficient
C_{pa}	Specific heat (constant pressure) of air
C_{pw}	Specific heat of water

An observed value of a variable is denoted by a caret, e.g. $\widehat{T}_w$ is a measurement of T_w. In the belief network description, actual and observed values are two nodes in a small belief network which has the same name as the variable of interest, e.g., $T_w.actual$ is the actual value of water temperature and $T_w.observed$ is its measured value. These two notations will be used interchangeably.

Substantial engineering knowledge is expressed in the belief network shown in Figure 8.2. Now let us see how this knowledge can be exploited by two different operations on the heating coil belief network.

8.1.1 *Assessing value of information of measurements by MI*

The first operation is the assessment of the value of information yielded by different measurements. Observations can be made on four of the upstream variables: $\widehat{T}_w$, $\widehat{m}_w$, $\widehat{m}_{air}$, and $\widehat{T}_{db,ent}$. Not surprisingly, each of the four has a different relevance to the variable of greatest interest, $T_{db,lvg}$. The value of information of measured variables was assessed as the mutual information MI of a group of measurements with $T_{db,lvg}$;

average conditional mutual informations were computed and combined according to the identity, Eq. F.2, to yield joint mutual information.

From Table 8.1, we see that $\widehat{T}_{db,ent}$ appears to be the most important measurement — it has a higher mutual information with $T_{db,lvg}$ than any other single variable, and all of the groups with high mutual information contain it. We might rank groups of variables as follows: the most important single variable is $\widehat{T}_{db,ent}$; the most important pair is $\widehat{T}_w$ and $\widehat{T}_{db,ent}$; and the most important set of three is $\widehat{T}_w$, $\widehat{m}_{air}$, and $\widehat{T}_{db,ent}$. On the other hand, $\widehat{m}_w$ has a very low individual mutual information with $T_{db,lvg}$, and none of the most informative groups contain it. This suggests that if $\widehat{m}_w$ is difficult to measure, or if there are funds for only a limited number of sensors, we could omit measurements of $\widehat{m}_w$ with very little effect on the posterior distribution for $T_{db,lvg}$. Note that this does not mean we will rewrite the model to exclude m_w; it will still appear in the heating coil model, but its prior will be used instead of a distribution derived from a measurement. We can omit $\widehat{m}_w$, but not m_w.

The Kullback-Leibler divergence between two Gaussian densities has a simple form, given by Eq. F.4. Two special cases can help us gauge the informativeness of measurements: (i) if $p_1(x) = g(x; 0, \sigma_1)$ and $p_2(x) = g(x; 0, \sigma_2)$, then

$$KL(p_1, p_2) = \frac{1}{2}\left(\frac{\sigma_1^2}{\sigma_2^2} - 1\right) - \log\frac{\sigma_1}{\sigma_2} \qquad \text{[nats]} \tag{8.11}$$

(ii) if $p_1(x) = g(x; \mu_1, 1)$ and $p_2(x) = g(x; \mu_2, 1)$, then

$$KL(p_1, p_2) = \frac{1}{2}(\mu_1 - \mu_2)^2 \qquad \text{[nats]} \tag{8.12}$$

Thus a mutual information equal to M bits is approximately equivalent to reducing the standard deviation by a factor $1/(2^M\sqrt{e})$, on the average, without moving the location, or moving the location by $\sqrt{2M\log 2}$, on the average, without changing the standard deviation. Since the posterior for $T_{db,lvg}$ is approximately Gaussian whatever the evidence (the functional relation is only mildly nonlinear and the π-messages are Gaussian), the column labeled "2^{bits}" can be interpreted in terms of average reduction in standard deviation from prior to posterior, or in terms of moving the posterior away from the prior.

Table 8.1: Groups of variables in the heating coil model, ranked by mutual information with $T_{db,lvg}$. Mutual information was computed for the variables in each row marked with plus signs.

$\widehat{T}_w$	$\widehat{m}_w$	$\widehat{m}_{air}$	$\widehat{T}_{db,ent}$	nats	bits	2^{bits}
+	+	+	+	1.783	2.57	5.95
+		+	+	1.584	2.29	4.87
+	+		+	1.242	1.79	3.46
+			+	1.150	1.66	3.16
	+	+	+	1.144	1.65	3.14
		+	+	1.065	1.54	2.90
	+		+	0.9043	1.31	2.47
			+	0.8551	1.23	2.35
+		+		0.09667	0.14	1.10
+	+	+		0.08813	0.13	1.09
	+	+		0.04949	0.071	1.05
+				0.04579	0.066	1.05
+	+			0.04196	0.061	1.04
		+		0.03855	0.056	1.04
	+			0.009176	0.013	1.01

The low mutual information $MI(\widehat{m}_w, T_{db,lvg})$ may be due, in part, to the relatively low accuracy of the $\widehat{m}_w$ sensor compared to the range of m_w. The effect of a more accurate sensor could be studied by assigning a smaller conditional variance to the distribution of $\widehat{m}_w$ given m_w, and recomputing the mutual information.

8.1.2 *Is $T_{db,lvg}$ higher or lower than expected?*

A belief network fragment of the form shown in Figure 6.1 was grafted into the $T_{db,lvg}$ network shown in Figure 8.2. This enables us to answer the question "Is $T_{db,lvg}$ higher or lower than expected?" by computing a posterior distribution over $T_{db,lvg}.multiplier$. The variable $T_{db,lvg}.actual$ is just the product of its parents $T_{db,lvg}.nominal$ and $T_{db,lvg}.multiplier$, so the distribution over the multiplier is just the distribution of a ratio.

Let us consider a few example calculations. With the measurements

$\widehat{T}_w$	195 °F
$\widehat{m}_w$	8.7 Klbm/h
$\widehat{m}_{air}$	79.5 Klbm/h
$\widehat{T}_{db,ent}$	104 °F
$\widehat{T}_{db,lvg}$	122.5 °F

we find the posterior distribution of the multiplier is sharply peaked near 0.90; the π-message sent to $T_{db,lvg}.actual$ from $T_{db,lvg}.nominal$ is nearly Gaussian with a mean 136.1 and standard deviation 2.159, which is just about 10% greater than the observed value $\widehat{T}_{db,lvg} = 122.5$. However, if the $\widehat{T}_{db,ent}$ measurement is missing, the π-message from $T_{db,lvg}.nominal$ is much broader; in this case it is again nearly Gaussian with mean 127.9 and standard deviation 12.93. Now the posterior of the multiplier is much broader than before, with mean 0.9674 and standard deviation 0.1013; it is noticeably skewed toward the right, which is typical of densities of ratios. The two posterior distributions are shown in Figure 8.3.

We could use the posterior distribution of $T_{db,lvg}.multiplier$ to generate warning

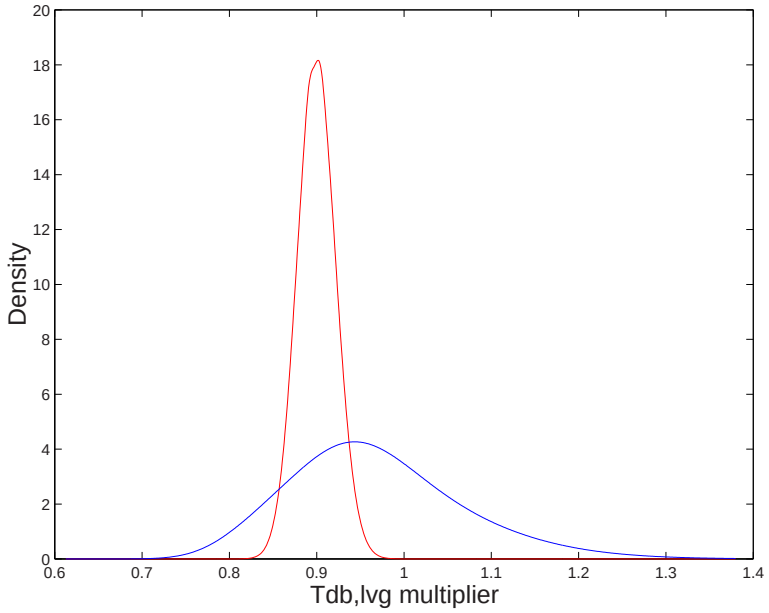

Figure 8.3: Two posterior distributions of the $T_{db,lvg}$ multiplier. The narrower distribution was computed with measurements $\widehat{T}_w$, $\widehat{m}_w$, $\widehat{m}_{air}$, $\widehat{T}_{db,ent}$, and $\widehat{T}_{db,lvg}$. The wider distribution was computed using only $\widehat{T}_w$, $\widehat{m}_w$, $\widehat{m}_{air}$, and $\widehat{T}_{db,lvg}$, and not $\widehat{T}_{db,ent}$.

messages. The general scheme is to set limits on the allowable range of the multiplier, saying, for instance, that values within the range 0.95 to 1.05 are allowable, but values outside that range indicate there is some kind of problem. Then we just compute the probability mass which falls in each interval, $[0, 0.95]$, $[0.95, 1.05]$, and $[1.05, +\infty)$. Actions such as message generation are considered only incidentally in this dissertation, so let it be enough for now to state that we want to see a relatively large mass in the middle interval, and if the lower or higher interval has enough mass, we'll generate a message.

8.2 A model of a mixing box damper

At the terminal end of the duct which supplies conditioned air from the air handling unit to a building zone, there is typically a device called a mixing box which controls the delivery of air to the zone. There may be dozens or even hundreds of these devices in a large building, so even though mixing boxes are usually very reliable, there may well be a few which are malfunctioning. Given the number of units which would have to be inspected, automated diagnostics are attractive. A further consideration is that mixing boxes are usually built as inexpensively as possible, so it is desirable to implement diagnostics with as few sensors as possible, beyond those that are already necessary for the operation of the mixing box.

The mixing box design which is the subject of this section was the subject of a report published elsewhere [28]. The mixing box, which is installed in the Larson HVAC laboratory at the University of Colorado, is shown schematically in Figure 8.4. A belief network for the damper alone is shown in Figure 8.5. The belief network consists of a number of time slices, which represent the state of the damper and associated variables at uniform time intervals. In experimental testing, four variables were measured: zone temperature offset (ZTO), supply duct static pressure (p_0), supply duct damper position (DP), and total electric power (P). P comprises fan power and reheat coil power; electric power is not relevant to diagnosis of damper problems, so P was omitted from the belief network for the damper. The damper is controlled according to ZTO and p_0 so that the

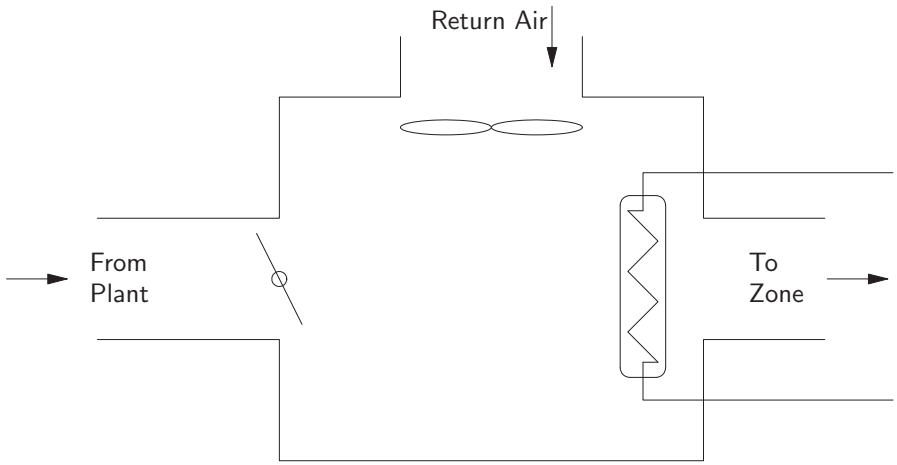

Figure 8.4: Schematic diagram of a typical mixing box, showing the damper at left, reheat coils at right, and the return fan above. The static pressure p_0 is measured in the supply duct (at left). The zone temperature offset ZTO is calculated as the temperature in the conditioned space minus the thermostat setpoint.

volume of supply air at large temperature offsets is approximately constant. Further details of the experimental set-up are given in Ref. [28].

8.2.1 *Local models in the damper belief network*

Local conditional probability models were constructed from both laboratory data and from engineering knowledge[1] — the model of correct operation of the damper was derived from empirical data, while models of other states of damper operation were constructed from engineering principles, likewise the models of correct and incorrect sensor operation were constructed from engineering knowledge of the sensors.

There are three parts of the mixing box damper belief network. Let us specify the parameters for each part, then consider some inferences which illustrate the interaction of the various parts. The complete temporal belief network, in RISO format, may be

[1] The author gratefully acknowledges the assistance of Peter Curtiss.

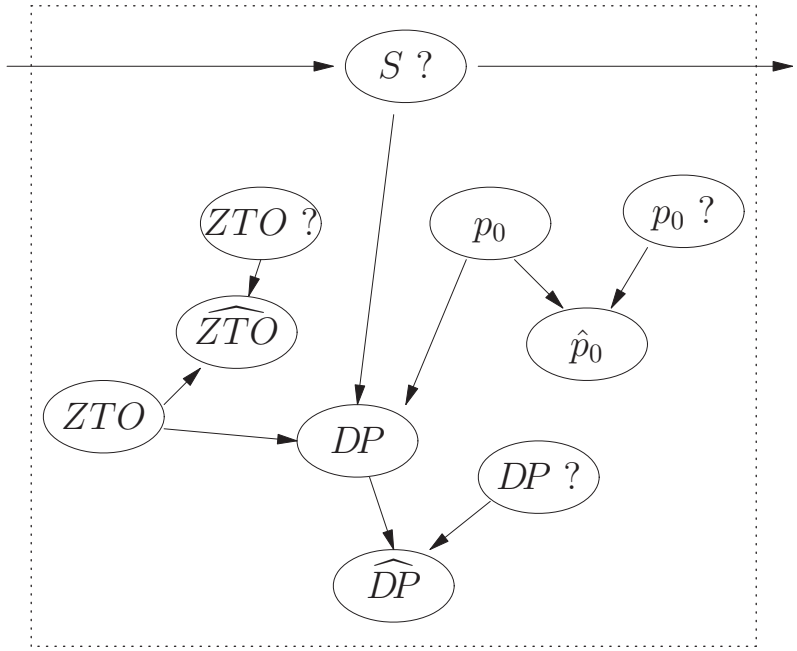

Figure 8.5: One slice of a temporal belief network for the mixing box damper. The variables are damper status, $S?$, zone temperature offset, ZTO, supply duct static pressure p_0, and damper position, DP. Observed values (i.e., sensor readings) are distinguished by a caret, and status variables are distinguished by a question mark. E.g., $\hat{p}_0$ is the observed value of p_0, and $p_0?$ is the pressure sensor status variable. The arrows leading into and out of $S?$ are links to the status variables in the previous and succeeding time slices, respectively.

found in the file `damper-tbn.riso` under the heading "Example RISO belief networks" at the web site, `http://civil.colorado.edu/~dodier`.

Transitions of S? The transition of damper states $S?$ from one time step to the next; in the absence of any sensor observations, the successive variables $\ldots, S?[t-1], S?[t], S?[t+1], \ldots$ are a simple Markov chain. (The damper status variable is denoted $S?$ when there is no danger of ambiguity, and by $S?[t]$ when it is necessary to distinguish the time slice.) Four states are distinguished: (i) correct operation of the damper, (ii) damper stuck open, (iii) damper stuck closed, and (iv) unknown state. The last state is a catch-all for the damper stuck in an intermediate position and any other kind of failure not foreseen. The transition matrix for these states is the following:

$$\text{from} \quad \begin{pmatrix} 1-\alpha & \alpha/3 & \alpha/3 & \alpha/3 \\ \beta_1 & 1-(\beta_1+\beta_2) & 0 & 0 \\ \beta_1 & 0 & 1-(\beta_1+\beta_2) & \beta_2 \\ 1/4 & 1/4 & 1/4 & 1/4 \end{pmatrix} \qquad \text{(8.13)}$$

to

The parameters are assigned as

$$\alpha = 10^{-3}, \quad \beta_1 = 10^{-6}, \quad \beta_2 = 9 \times 10^{-6} \qquad \text{(8.14)}$$

This transition matrix expresses the following: if the damper is operating correctly, usually it will continue to operate correctly, but with low probability it may transition into one of the other states. If the damper is stuck open or closed, it is almost certain to stay in that state, and very unlikely to transition back to the correct state, although with somewhat greater probability it may transition into the "unknown" state. The "unknown" state may transition into any other state with equal probability.

It is clear that constructing a transition matrix brings substantial prior information to bear on the diagnosis problem. As ever, one can spend as much effort as one likes on a single conditional probability model in a belief network, and the laws of probability

will ensure that the information in that local model is combined with the rest of the belief network to yield correct global inferences.

Model of DP given S?, p_0, and ZTO. The damper operates in different ways depending on the zone temperature offset, duct static pressure, and operating status. These different operating modes are expressed as *generative* models for damper operation — that is, models which predict where the damper will be, depending on the particular combination of driving variables p_0, ZTO, and $S?$. The generative model of DP is coded as a set of four distributions indexed by $S?$, with one distribution in the set for each state of $S?$.

(i) The model for $DP \mid S? = 0,\ p_0,\ ZTO$ is a neural network with two inputs (one for p_0 and one for ZTO), six hidden units, and one output (for DP). The number of hidden units was chosen by minimizing the so-called "Bayesian Information Criterion" or BIC [60] on a training data set, comprising several thousand data obtained in the HVAC laboratory. The BIC is defined (ignoring constants) as

$$BIC = \log MSE + p\,\frac{\log n}{n} \tag{8.15}$$

where MSE is the mean-square error of prediction on the training set, p is the number of free parameters (weights and biases in a neural network) in the model, and n is the number of data in the training set. The term $(p/n)\log n$ tends to weigh against networks with many hidden units, unless the training data set is very large. It has been found empirically that selecting a neural network which minimizes BIC on a given training set usually yields the best out-of-sample error. The residual root mean-square error on the training set was 0.4344, which is taken as the magnitude of the Gaussian additive noise associated with predictions of DP.

(ii) The model for $DP \mid S? = 1,\ p_0,\ ZTO$ is a Gaussian distribution of DP with mean 9.6 and standard deviation 0.4344.

(iii) The model for $DP \mid S? = 2,\ p_0,\ ZTO$ is a Gaussian distribution of DP with mean 4.1 and standard deviation 0.4344.

(iv) The model for $DP \mid S? = 3,\ p_0,\ ZTO$ is a uniform distribution of DP over the range $[4.5344, 9.1656]$. Like the distributions for $DP \mid S? = 1$ and $DP \mid S? = 2$, it is independent of p_0 and ZTO.

These generative models, which give the conditional distribution over damper positions depending on the damper status, are essentially "inverted" according to the laws of probability to obtain a distribution over the damper status.

Sensor models. Each observed variable has a sensor model. A distribution over the real or actual value is inferred from the measurement. All three sensors are reported in volts, without translation into engineering units. The scale of the ZTO and DP sensors is zero to 10 volts, while the scale of the p_0 sensor is zero to 0.5 volts. The open-circuit measurement is, therefore, zero volts.

When the sensor operates correctly, the measurement is assumed to be the actual value distorted by Gaussian additive noise. The only failure mode modeled is the open circuit. So we have a common model for all three sensors, with different parameters:

$$\widehat{X} \mid X,\ X? = 0 \quad \sim \quad N(X, \sigma^2_{\widehat{X}}) \tag{8.16}$$

$$\widehat{X} \mid X,\ X? = 1 \quad \sim \quad N(0, \sigma^2_{\widehat{X}}) \tag{8.17}$$

Here X stands for one of ZTO, DP, or p_0, and the sensor states are 0 for "correct operation" and 1 for "open circuit." The magnitude of the additive noise for each sensor is the following:

$$\sigma_{\widehat{ZTO}} = 0.1, \quad \sigma_{\widehat{DP}} = 0.2, \quad \sigma_{\hat{p}_0} = 0.025 \tag{8.18}$$

The prior probability of failure for each of the sensors is 0.01.

Additional information about sensors, such as models of other kinds of sensor failures, could be incorporated without changing the $S?$ transition model or the generative models of damper operation. Probabilistic modeling allows the decomposition of the modeling problem into independent pieces, which are then combined according to the laws of probability.

8.2.2 *Belief revision in a temporal belief network*

In a belief network, every posterior probability is contingent upon the evidence available at the time it was computed, and if new evidence is acquired, old beliefs might have to be revised — specifically, the posterior of every variable d-connected to the new evidence must be recomputed. So the computation of a posterior distribution for $S?[6]$, say, is not the final word about the dampers status at time slice 6. Time slices 7, 8, 9,... will bring new evidence which leads us to revise our beliefs about time slice 6. Whereas we might initially suspect a problem, further evidence might lead us to conclude there was no problem at all, and vice versa. Let us consider a specific example of belief revision in the mixing-box damper belief network.

Table 8.2 shows data collected in the HVAC laboratory at one minute intervals. These data show how the damper opens as the zone temperature offset increases — this is about half of one typical cycle of operation, and the damper closes as zone temperature offset decreases in the other half of the cycle. Ten time slices of the damper belief network were created, and the values shown in the first five rows were copied into time slices 1 through 5 as evidence. The values of $\widehat{ZTO}[6]$ and $\hat{p}_0[6]$ were copied verbatim, but the value 4.1 was copied into $\widehat{DP}[6]$ instead of the correct value, 6.460. We will compute the posterior of $S?[6]$ with increasing evidence to see how the belief network handles the glitched datum.

The posterior for $S?[6]$ was computed with evidence $\mathbf{e}_1$ through $\mathbf{e}_5$, then $\mathbf{e}_1$ through $\mathbf{e}_6$, $\mathbf{e}_1$ through $\mathbf{e}_7$, and so on. The posterior for $S?[6]$ is shown at each time step in Table 8.3. The λ-message sent from $DP[6]$ to $S?[6]$ is

$$\lambda_{DP[6],S?[6]} = [\ 3.071 \times 10^{-5},\quad 3.764 \times 10^{-84},\quad 0.9961,\quad 0.003848\ ] \tag{8.19}$$

so the likelihood ratio for "stuck closed" against "normal operation" is $0.9961/3.071 \times 10^{-5} \approx 32440$. Due to this strong evidence in favor of "stuck closed," the posterior for $S?[6]$ given $\mathbf{e}_1$ through $\mathbf{e}_6$ puts most of its mass (0.9297) on the "stuck closed" state. However, with additional evidence $\mathbf{e}_7$ which overwhelmingly supports "normal operation" against "stuck closed" (with a likelihood ratio 2.6×10^{26}), the posterior for

Table 8.2: Mixing-box data collected in the HVAC laboratory at one minute intervals. The value shown in parentheses for $\widehat{DP}[6]$ was observed, but the value 4.1 was assigned as evidence instead. All measurements are in volts.

k	$\widehat{ZTO}$	$\hat{p}_0$	$\widehat{DP}$
1	3.994	0.3271	4.170
2	4.341	0.4102	4.185
3	4.702	0.3906	4.180
4	5.005	0.4053	4.644
5	5.352	0.3516	5.347
6	5.649	0.3223	4.1 (6.460)
7	5.981	0.1514	7.856
8	6.245	0.2197	9.512
9	6.509	0.1904	9.658
10	6.704	0.1758	9.673

Table 8.3: Posterior for $S?[6]$ with increasing evidence, $\mathbf{e}_1 \cup \cdots \cup \mathbf{e}_k, k = 5, \ldots, 10$.

Evidence	Posterior for $S?[6]$			
$\mathbf{e}_1 \cup \cdots \cup \mathbf{e}_5$	0.9988	3.651×10^{-4}	4.263×10^{-4}	3.751×10^{-4}
$\mathbf{e}_1 \cup \cdots \cup \mathbf{e}_6$	0.06713	3.008×10^{-84}	0.9297	0.003160
$\mathbf{e}_1 \cup \cdots \cup \mathbf{e}_7$	0.9415	5.733×10^{-41}	5.081×10^{-4}	0.05803
$\mathbf{e}_1 \cup \cdots \cup \mathbf{e}_8$	0.9564	9.586×10^{-41}	3.485×10^{-4}	0.04327
$\mathbf{e}_1 \cup \cdots \cup \mathbf{e}_9$	0.9551	1.024×10^{-40}	3.614×10^{-4}	0.04450
$\mathbf{e}_1 \cup \cdots \cup \mathbf{e}_{10}$	0.9546	1.051×10^{-40}	3.672×10^{-4}	0.04506

$S?[6]$ is revised to favor "normal operation." Further evidence $\mathbf{e}_8, \mathbf{e}_9, \mathbf{e}_{10}$, continues to support $S?[6] = 0$.

The transition matrix for $S?$ assigns a very small probability to the sequences such as "normal" → "stuck closed" → "normal." Thus glitches are smoothed out, since it is more probable that the state was normal and the datum is strange than the state changed from a failure to normal operation. We might report an error at time step 6, since the probability of the normal damper state is small at that time. However, we would want to retract the alarm after receiving more evidence which shows the data at time step 6 were just a glitch. Perhaps for unimportant problems, we would always wait some time before issuing an error message, to be more certain that the problem is real. On the other hand, for costly problems, we would issue messages right away, even if there is a higher risk of false alarms.

8.2.3 *Strengthening repeated weak evidence*

An hypothesis which is only weakly supported by the evidence in one time slice may increase in probability if evidence in other time slices supports the same hypothesis. Suppose that we consider 10 time slices of the mixing-box damper belief network. Initially, there is evidence only in slice 1, with

$$\mathbf{e}_1 = \{\widehat{ZTO} = 5.6494134, \quad \widehat{DP} = 9.1, \quad \hat{p}_0 = 0.15\} \tag{8.20}$$

Taking the prior distribution over $S?$ as one which heavily favors "normal operation" (state 0) over the others,

$$p_{S?[1]} = [\ 0.97, \quad 0.01, \quad 0.01, \quad 0.01\ ] \tag{8.21}$$

we find the posterior distribution still favors "normal operation," with $\Pr(S?[1] = 0 \,|\, \mathbf{e}_1) = 0.9732$ and $\Pr(S?[1] = 1 \,|\, \mathbf{e}_1) = 0.02526$. States 0 and 1 ("normal operation" and "stuck open") have increased probability, at the expense of states 2 and 3 ("stuck closed" and "unknown"). But if the same evidence is assigned in time slice 2, so $\mathbf{e}_2 = \mathbf{e}_1$, we find the probability of "normal operation" decreases, with

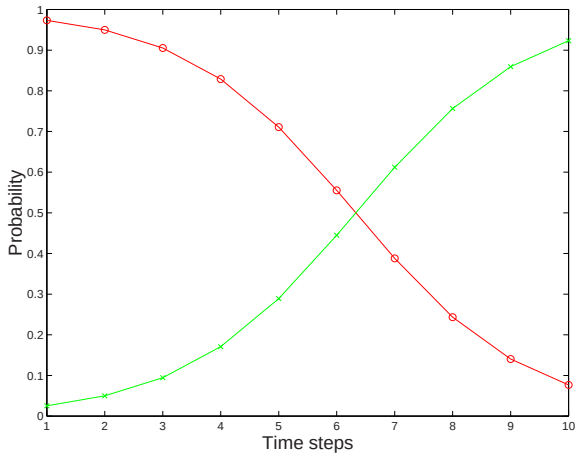

Figure 8.6: Repeated weak evidence in a temporal belief network eventually supports a strong conclusion. This graph shows $\Pr(S?[k] = 0 \,|\, \mathbf{e}_1 \cup \cdots \cup \mathbf{e}_k)$ as circles, and $\Pr(S?[k] = 1 \,|\, \mathbf{e}_1 \cup \cdots \cup \mathbf{e}_k)$ as crosses, for $k = 1, 2, 3, \ldots, 10$. (Probabilities for states 2 and 3 of $S?[k]$ are not shown, as they are very small in every time slice.) The evidence in each slice $j = 1, \ldots, 10$ is denoted $\mathbf{e}_j$, and for each slice it is the same: $\mathbf{e}_j = \{\widehat{ZTO} = 5.6494134, \widehat{DP} = 9.1, \hat{p}_0 = 0.15\}$.

$\Pr(S?[1] = 0 \,|\, \mathbf{e}_1 \cup \mathbf{e}_2) = 0.9497$ and $\Pr(S?[1] = 1 \,|\, \mathbf{e}_1 \cup \mathbf{e}_2) = 0.04989$. Putting the same evidence into each time slice, $\mathbf{e}_k = \mathbf{e}_1$, and computing the posterior of $S?[k]$, we find the probability of "normal operation" gradually declines, and the probability of "stuck open" rises, as shown in Figure 8.6. The λ-message (likelihood function) which is sent from $DP[k]$ to $S?[k]$ is the same in each time slice, with

$$p_{\mathbf{e}_j \,|\, S?[k]} = \lambda_{DP[k], S?[k]} = [\ 0.2841, \quad 0.5588, \quad 8.245 \times 10^{-65}, \quad 0.1571\ ] \tag{8.22}$$

This likelihood function supports the "stuck open" hypothesis at about 2 to 1 odds over the "normal operation" hypothesis, while the "unknown status" hypothesis also has substantial support. In the absence of any evidence from other time slices, the weak support for "stuck open" would be swamped out by the prior odds in favor of "normal operation," which in these computations were 97 to 1. However, the presence of weak

support for "stuck open" in every time slice accumulates, and as Figure 8.6 shows, after a few time slices, the posterior probability for "normal operation" decreases, and after 10 time slices "stuck open" has much higher posterior probability than "normal operation."

Since the evidence in each time slice affects the posterior of $S?[k]$ only through the λ-message $\lambda_{DP[k],S?[k]}$, any evidence which yielded the same λ-message would yield the same posterior probabilities. So it was only for convenience that the evidence was taken as the same in every time slice; the reinforcement of weak evidence is a general property of the belief network, and not especially associated with having the same evidence from one time slice to the next.

8.2.4 *Predictions from the damper belief network*

In the treatment of the glitched datum in §8.2.2, we may wonder what is the "usual" value of the damper position, if not the value 4.1 which was assigned. We can use the belief network in a predictive sense to obtain a distribution over the damper position measurement, to see what observation we could expect for $\widehat{DP}$ given the other observations in the belief network. Recall that we observed $\widehat{ZTO}[6] = 5.649$ and $\hat{p}_0[6] = 0.3223$. Let us assume the damper is operating correctly and the sensors are working correctly. Then the posterior of $\widehat{DP}[6]$ is computed as a nearly Gaussian distribution with mean 6.652 and standard deviation 0.5495. Table 8.2 shows the actual measurement was $\widehat{DP}[6] = 6.460$, which is not far from the predicted value; the non-glitched value would have been strong evidence in favor of the "normal operation" state.

We might just as well ask what combination of the other observables (zone temperature offset and duct static pressure) correspond to the glitched value $\widehat{DP}[6] = 4.1$. That is, when is 4.1 an expected or predictable value? Setting $\widehat{DP}[6]$ to the glitched value and holding the duct pressure at the observed value 0.3223, the posterior for $\widehat{ZTO}[6]$ was computed as a monotone cubic spline approximation (Appendix D) with about 4500 support points. As shown in Figure 8.7, the posterior for $\widehat{ZTO}[6]$ is heavily skewed toward the left — in a sense, the observed value $\widehat{ZTO}[6] = 5.649$ is "too high," and it

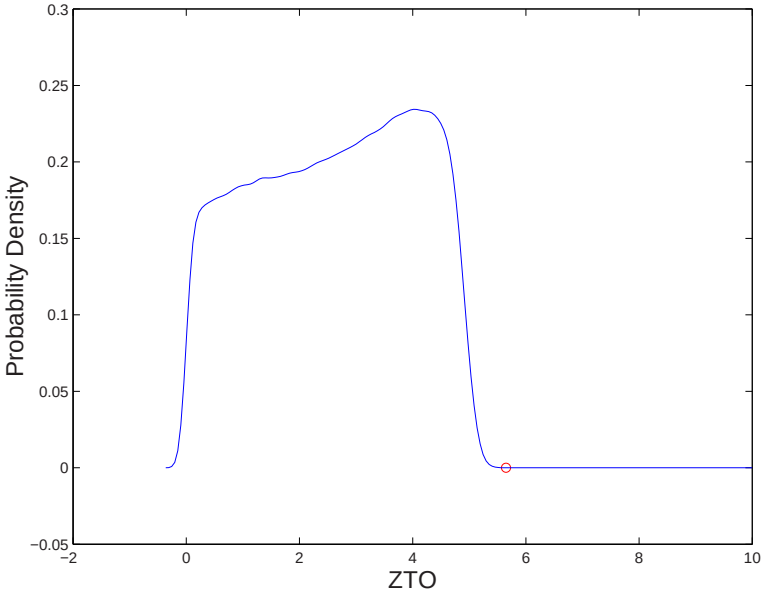

Figure 8.7: Posterior distribution for $\widehat{ZTO}[6]$ given $\widehat{DP}[6] = 4.1$, $\hat{p}_0[6] = 0.3223$, and assuming the damper is operating correctly. This density is approximated as a monotone spline with about 4500 support points. The measurement $\widehat{ZTO}[6] = 5.649$ is marked with an open circle.

should be smaller to fit the glitched damper position measurement.

Chapter 9

CONCLUDING REMARKS

This dissertation has expounded the theoretical basis of distributed belief networks, described a particular implementation of the theory, and presented several applications. Let us close with a few remarks on an important class of belief network applications which remain unexplored. Of all the problems which might be described by belief networks, I believe the models described in §9.1 are among the the most promising. We will briefly consider three belief networks for updating or tuning parameters of an equipment model "on the fly."

9.1 Calibration *in situ* and other learning applications

Exemplars of mass-produced equipment function more or less the same. Often there is enough prior information to set up the general form of a model for the equipment as designed, but some parameters need to be adjusted to better fit a particular instance. For example, the maximum and minimum damper positions or the power use levels (fan alone, or fan and heating coils) in a mixing box are parameters which are important to know for models of operation in failure states. Without knowing the heating coil power use, for example, it is not possible to identify a state in which the heating coil failed to operate. We can construct models in the factory which mention power levels or extreme damper positions, but the most appropriate values for these parameters will depend on accidents of manufacture and installation.

One approach to handling parameters in need of calibration is to make them variables in a belief network which describes observables and status variables. Since configurable parameters usually modify the relations between other variables, they will appear as parents (and not as children) in the belief network. A very generic representation of this

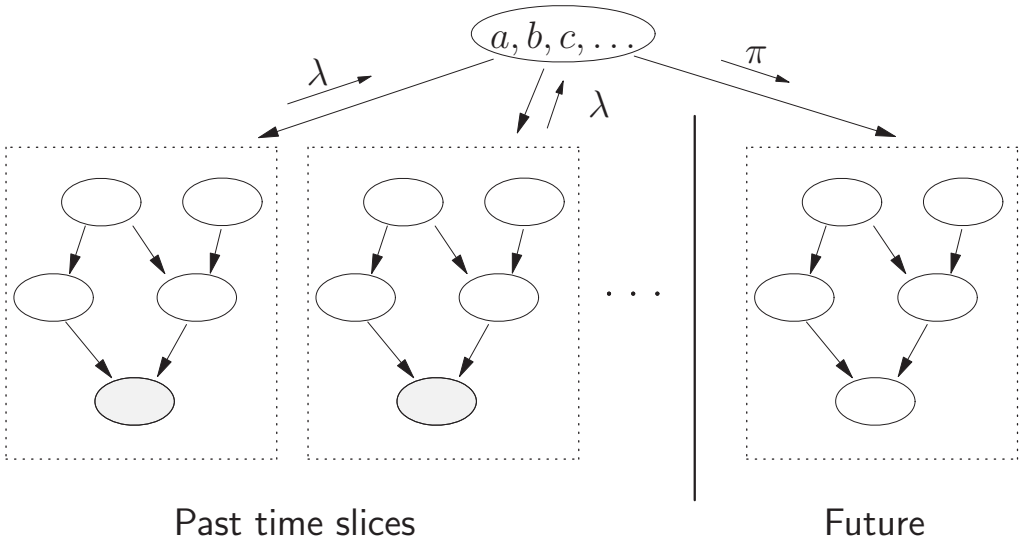

Figure 9.1: A generic representation of a belief network for adjustment of model parameters *in situ*. Observed variables are shaded.

arrangement is shown in Figure 9.1. This belief network has a commonly-encountered form: relations among the physical variables in a system are represented by a subnetwork which is replicated into many time slices, and the parameters which govern these relations are the parents of at least one variable in every time slice. Information flows from past observations into the present and future via the common parents. Likelihood functions (λ-messages) are calculated from observations in past time slices, and transmitted to the nodes for the model parameters. When inferences are computed for variables in future time slices, prior information about the parameters, in the form of their prior distribution, is combined with information from observations to yield a π-message which summarizes all that is known about the parameters. As ever, prior information dominates when there are few observations, while sufficient observed data will wash out the influence of the prior.

Each installed exemplar of a certain kind of equipment tells something about how that kind of equipment works. When a new unit is installed, it would be helpful to

exploit the information gathered at other sites to predict and diagnose the behavior of the new unit. A distributed belief network which generalizes Figure 9.1 is shown in Figure 9.2. In this belief network, information gathered at each installation updates the local parameters (a_1, b_1, c_1 and a_2, b_2, c_2 in the figure) and influences future inferences at each site, but λ-messages are also passed back to a central site — perhaps maintained by the manufacturer — which establishes a generic distribution over the equipment parameters, expressed as a prior distribution for the "meta-parameters" $\tilde{a}, \tilde{b}, \tilde{c}$. In the absence of observations at a new installation, the parameters at the new site will reflect both the factory prior and the accumulated information from other sites, which will gradually be modified or displaced by local observations.

Another very general form of belief network for learning is shown in Figure 9.3. We may know that a system contains equipment of one of several kinds, but we would like to automatically determine which particular kind it is. For example, we may know the equipment is from one of several manufacturers but not know which one, or again, we may know the equipment is one of several models in a product line but not know which one. In this case, the appropriate probabilistic model depends on which particular type of equipment is present. So our belief network contains a node for the equipment type, which is parent of the model parameters, since for each type there will correspond a set of parameters appropriate for that type. The equipment type and its parameters are together the parents of variables in all time slices of the system model. As in Figure 9.1, information flows from past observations into future predictions and diagnoses. One inference of great interest will be the posterior over the equipment type — if the observations are sufficient to distinguish different kinds of equipment, we could hope the belief network will help us determine what kind of equipment is present in the system.

While conceptually simple, the calculations required for belief networks of the form shown in Figure 9.1 can be extremely burdensome. For some special cases, fast, exact calculations are possible, but these happy circumstances are rare. An excellent overview of the conceptual and computational problems involved, with some exact results and suggestions for computing approximations, is given in Ref. [11]. Given the broad appli-

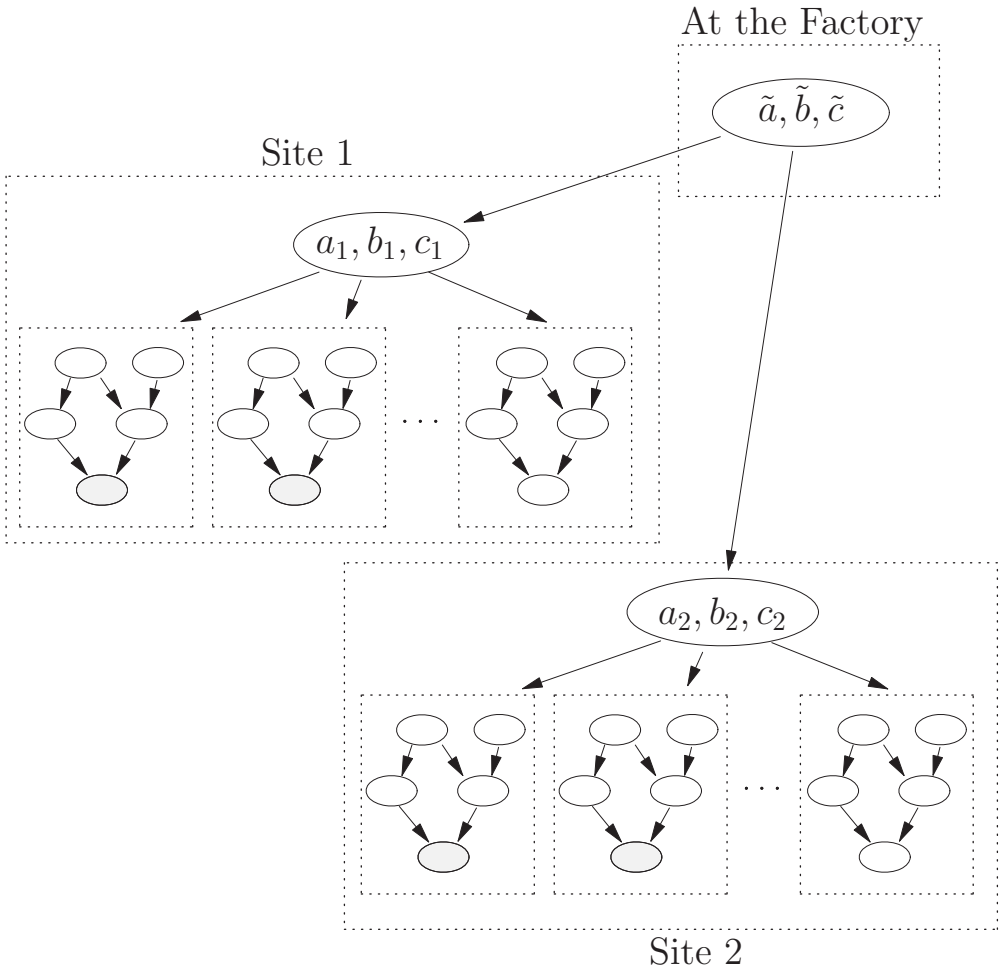

Figure 9.2: A distributed belief network to allow messages about equipment parameters to pass between different installation sites.

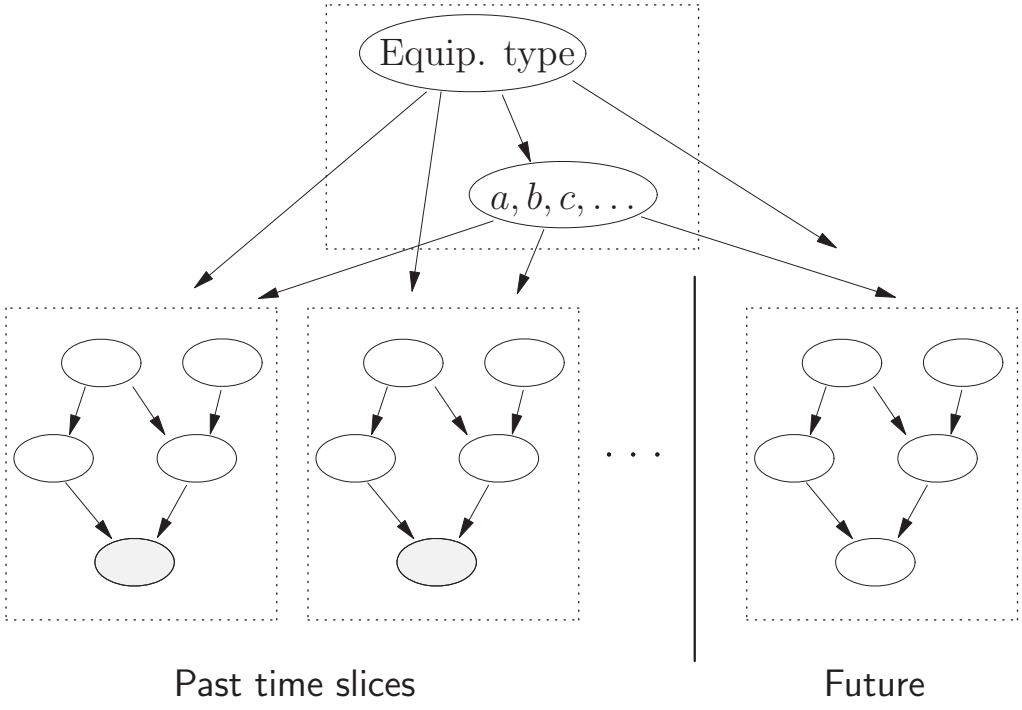

Figure 9.3: A belief network to infer equipment type from observations. Given equipment type, this belief network reduces to the form shown in Figure 9.1.

cability of *in situ* model adjustment, further research in this direction seems well worth the trouble. It may be possible to invent approximate inference algorithms which are well-suited to belief network structures of the kinds shown in Figures 9.1–9.3.

9.2 In closing

In this dissertation, I have tried to describe how we can use the laws of probability to obtain interesting inferences from whatever data we may have. Along the way, some elementary examples were described; if the results ever seemed strange, the problem could not lay in the laws of probability, for they only encode the desiderata which we set forth in Chapter 2. Strange results might be the outcome of models which are a poor fit to an actual system, or mistakes or poor approximations in calculations, or the omission of critically relevant information — thus strange results indicate we need to revise our belief networks and debug our numerical algorithms or invent new ones. Unless we change our desiderata, we need not search further for a reasoning framework:

> When I try my system out I'm going to get nonsense answers. When this happens I want to know that the problem is in how I encoded the knowledge, not in my uncertainty calculus,

a comment attributed to Eugene Charniak. Interesting problems will require substantial effort, but we may safely detour the construction of new reasoning systems and direct our effort entirely toward formalizing the problem and computing any inferences we find useful.

The framework developed in Chapters 2, 4, and 5 exhibits several interesting qualitative properties, as illustrated in the examples from Chapters 6 and 8. These qualitative properties, including the expression of prior knowledge, fusion of different sources of information, representation of prediction and diagnosis as different operations on a single model, quantification of the value of information, and other properties discussed at length in §§2.7–2.8, are the best reasons to use probabilistic models. Particular inference algorithms and model types will come and go with the varying demands of applications, but the foundation of probability, like the Parmenidean One, remains.

BIBLIOGRAPHY

[1] J. Aczel. *Lectures on functional equations and their applications*. Academic Press, New York, 1966.

[2] G. Almasi and A. Gottlieb. *Highly Parallel Computing*. Redwood City, CA: Benjamin/Cummings Publishing Co., 1989.

[3] Belief network file format for XML. URL `http://www.research.microsoft.-com/research/dtg/bnformat`, 1999.

[4] DNET-1 File Format. URL `http://www.norsys.com/dl/DNET_File_Format.txt`, 1995.

[5] K. Arrow. *The economics of information*. Harvard University Press, Cambridge, MA, 1984.

[6] M. Baioletti. *Metodi computazionali per l'inferenza bayesiana con dati incompleti*. PhD thesis, Universitá degli Studi di Perugia, 1996. URL `http://gauss.stat.-unipg.it/~asterix/pubbl/tesi.ps.gz`.

[7] A. Becker and D. Geiger. Approximation algorithms for the loop cutset problem. In R. Lopez de Mantaras and D. Poole, editors, *Proc. 10th Conf. on Uncertainty in Artificial Intelligence*, pages 60–68. San Francisco: Morgan Kaufmann, 1994.

[8] A. Bird. *Philosophy of science*. UCL Press, 1998.

[9] P. Bratley and B. L. Fox. Sobol's quasirandom sequence generator for multivariate quadrature and optimization. *ACM Trans. on Mathematical Software*, 14(1):88–100, 1988. Algorithm 659, `http://www.netlib.org/toms/659`.

[10] E.O. Brigham. *The fast Fourier transform*. Prentice-Hall, Englewood Cliffs, N.J., 1974.

[11] W.L. Buntine. Operations for learning with graphical models. *J. Artificial Intelligence Research*, 2:159–225, 1994. URL `http://www.cs.washington.edu/-research/jair/home.html`.

[12] W.L. Buntine. Prior probabilities. Tutorial given at NATO Workshop on Learning in Graphical Models, Erice, Italy. URL `http://www.ultimode.com/wray/-refs.html`, September 1996.

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

[14] E. Charniak. Bayesian networks without tears. *AI Magazine*, 12(4):50–63, 1991.

[15] T. Chu and Y. Xiang. Exploring parallelism in learning belief networks. In D. Geiger and P. Shenoy, editors, *Proc. 13th Conf. on Uncertainty in Artificial Intelligence*. San Francisco: Morgan Kaufmann, 1997.

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

[17] R.G. Cowell, A.P. Dawid, and P. Sebastiani. A comparison of sequential learning methods for incomplete data. In J.M. Bernardo, editor, *Bayesian Statistics 5: Proceedings of the Fifth Valencia International Meeting*, pages 533–542. Oxford: Clarendon Press; New York: Oxford University Press, 1996.

[18] R.T. Cox. Probability, frequency, and reasonable expectation. *Am. J. Physics*, 14(1):1–13, 1946. Reprinted in Ref. [74].

[19] R.T. Cox. *The algebra of probable inference*. Johns Hopkins Press, Baltimore, 1961.

[20] R.T. Cox. *Of inference and inquiry: an essay in inductive logic*. 1978. pp 119–167 in Ref. [52].

[21] F. Cozman. The interchange format for Bayesian networks. URL `http://www.-cs.cmu.edu/afs/cs/user/fgcozman/www/Research/InterchangeFormat/`.

[22] P. Dagum, A. Galper, E. Horvitz, and A. Seiver. Uncertain reasoning and forecasting. *Int'l J. Forecasting*, 11(1):73–87, 1995.

[23] Pierre-Simon de Laplace. *Théorie analytique des probabilités*. Courcier Imprimeur, Paris, 1812.

[24] Pierre-Simon de Laplace. *Essai philosophique sur les probabilités*. Courcier Imprimeur, Paris, 3rd edition, 1816. (See also the translation by Dale [25], which has extensive notes.).

[25] Pierre-Simon de Laplace. *Philosophical essay on probabilities*. Springer-Verlag, New York, 1995. Translated by A.I. Dale from the fifth French edition (1825).

[26] Jan de Leeuw. Statistics and the sciences. URL `http://www.stat.ucla.edu/-papers/preprints/152.ps.gz`, 1994.

[27] F.J. Díez. Local conditioning in Bayesian networks. *Artificial Intelligence*, 87:1–20, 1996.

[28] R. Dodier, P.S. Curtiss, and J.F. Kreider. Small-scale on-line diagnostics for an HVAC system. *ASHRAE Transactions*, 104(1), 1998.

[29] 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.

[30] E. Driver and D. Morrell. A new method for implementing hybrid Bayesian networks. Unpublished technical report, 1998.

[31] A. Fairbanks. *The first philosophers of Greece*. 1898. Translations of the extant fragments of Xenophanes may be found at URL `http://history.hanover.edu/-texts/presoc/xenophan.htm`.

[32] A. Fraser and A. Dimitriadis. *Forecasting probability densities by using hidden Markov models*. 1994. pp 265–282 in Ref. [83].

[33] F.N. Fritsch and J. Butland. A method for constructing local monotone piecewise cubic interpolants. *SIAM J. Sci. Stat. Comp.*, 5(2):300–304, 1984.

[34] Z. Ghahramani and M. Jordan. Factorial hidden Markov models. In D. Touretzky, M. Mozer, and M. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, Cambridge, MA, 1996. MIT Press.

[35] W.R. Gilks, A. Thomas, and D.J. Spiegelhalter. A language and program for complex Bayesian modelling. *The Statistician*, 43:169–178, 1994.

[36] L. Groarke. *Skepticism, Ancient*. 1998. In Ref. [88].

[37] I. Hacking. *The emergence of probability: a philosophical study of early ideas about probability, induction and statistical inference*. Cambridge University Press, London, 1975.

[38] J. Halpern. A counterexample to the theorems of Cox and Fine. *J. AI Research*, 10:76–85, 1999.

[39] J. Hu and Y. Xiang. Learning belief networks in domains with recursively embedded pseudo independent submodels. In D. Geiger and P. Shenoy, editors, *Proc. 13th*

Conf. on Uncertainty in Artificial Intelligence. San Francisco: Morgan Kaufmann, 1997.

[40] R.A. Jacobs, M.I. Jordan, S. Nowlan, and G.E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

[41] E.T. Jaynes. *Probability theory: the logic of science*. 1996. Unpublished MS; URL `ftp://bayes.wustl.edu/Jaynes.book/`.

[42] F.V. Jensen. *An introduction to Bayesian networks*. Springer, New York, 1996.

[43] W.M. Kays and A.L. London. *Compact Heat Exchangers*. McGraw-Hill, New York, 1964. 2nd Edition.

[44] M. Kennel, H.D.I. Abarbanel, and J.J. Sidorowich. Prediction errors and local Lyapunov exponents. URL `http://xxx.lanl.gov/ps/chao-dyn/9403001`, 1994.

[45] J.M. Keynes. *A treatise on probability*. Macmillan, London, 1921.

[46] U. Kjaerulff. HUGS: Combining exact inference and Gibbs sampling in junction trees. In P. Besnard and S. Hanks, editors, *Proc. 11th Conf. Uncertainty in Artificial Intelligence*. San Francisco: Morgan Kaufmann, 1995.

[47] J. Kockelmans. *Philosophy of science: the historical background*. The Free Press, New York, 1968.

[48] A. Kozlov and D. Koller. Nonuniform dynamic discretization in hybrid networks. In D. Geiger and P. Shenoy, editors, *Proc. 13th Conf. Uncertainty in Artificial Intelligence*. San Francisco: Morgan Kaufmann, 1997.

[49] A. Kozlov and J.P. Singh. Parallel implementations of probabilistic inference. *IEEE Computer*, 29(12):33–40, 1996.

[50] S. Kullback. *Information theory and statistics*. Dover Publications, Mineola, NY, 1968.

[51] E.J. Lemmon. *Beginning logic*. Hackett Publ. Co., Indianapolis, 1978.

[52] R.D. Levine and M. Tribus, editors. *The maximum entropy formalism*. MIT Press, Cambridge, MA, 1978.

[53] D. MacKay. A practical Bayesian framework for backpropagation networks. *Neural Computation*, 4:448–472, 1992.

[54] P. McCullagh and J.A. Nelder. *Generalized linear models*. Chapman and Hall, New York, 1983.

[55] F.C. McQuiston and J.D. Parker. *Heating, ventilating, and air conditioning: analysis and design*. Wiley and Sons, New York, 1982.

[56] R. Neal. *Bayesian Learning for Neural Networks*. PhD thesis, University of Toronto, 1994.

[57] A.E. Nicholson and J.M. Brady. Dynamic belief networks for discrete monitoring. *IEEE Trans. Systems, Man, and Cybernetics*, 24(11):1601ff, 1994.

[58] H. Niederreiter. *Random number generation and quasi-Monte Carlo methods*. Society for Industrial and Applied Mathematics, Philadelphia, 1992.

[59] D.W. North. A tutorial introduction to decision theory. *IEEE Trans. Systems Science and Cybernetics*, 4(3), 1968. Reprinted in Ref. [74].

[60] D. Nychka, S. Ellner, R. Gallant, and D. McCaffrey. Finding chaos in noisy systems. *J. Royal Statistical Society, Series B*, 54(2):399–426, 1992.

[61] E. Ott. *Chaos in dynamical systems*. University Press, 1993.

[62] A. Papoulis. *Probability, random variables, and stochastic processes*. McGraw-Hill, New York, 1984.

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

[64] J. Pearl. The new challenge: From a century of statistics to the age of causation. *Computing Science and Statistics*, 29(2):415–423, 1997.

[65] D.M. Pennock. Polylogarithmic time parallel Bayesian inference. In G.F. Cooper and S. Moral, editors, *Proc. 14th Conf. Uncertainty in Artificial Intelligence*. San Francisco: Morgan Kaufmann, 1998.

[66] H. Poincaré. *Science et Méthode*. Flammarion, Paris, 1909.

[67] W. Poland. *Decision analysis with continuous and discrete variables*. PhD thesis, Stanford University, Dept. of Engineering-Economic Systems, 1994.

[68] K. Popper. *Logic of scientific discovery*. Hutchinson, London, 1959.

[69] K. Popper. *Conjectures and refutations*. Routledge, London, 1963.

[70] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes in C*. Cambridge University Press, Cambridge, 1988.

[71] B. Ripley. *Pattern recognition and neural networks*. Cambridge U. Press, Cambridge, UK, 1996.

[72] B. Russell. *History of western philosophy*. Simon & Schuster, New York, 1945.

[73] W.C. Salmon. *The foundations of scientific inference*. University of Pittsburgh Press, 1967.

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

[75] D. Sleator and R. Tarjan. Self-adjusting binary search trees. *J. Assoc. Computing Machinery*, 32(3):652–686, 1985.

[76] P. Smyth, D. Heckerman, and M. Jordan. Probabilistic independence networks for hidden Markov models. Microsoft Research technical report MSR-TR-9603, 1996.

[77] P. Snow. On the correctness and reasonableness of Cox's theorems for finite domains. *Computational Intelligence*, 14:452–459, 1998.

[78] I. Todhunter. *A history of the mathematical theory of probability from the time of Pascal to that of Laplace*. Cambridge, 1865.

[79] G. von Leibniz. *New essays on human understanding*. Cambridge University Press, 1982. Translated by P. Remnant and J. Bennet from the edition of 1705.

[80] R. von Mises. *Probability, statistics, and truth*. George Allen and Unwin Ltd., London, 1957.

[81] R. von Mises. *Mathematical theory of probability and statistics*. Academic Press, New York, 1964.

[82] J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, Princeton, New Jersey, 2nd edition, 1947.

[83] A. Weigend and N. Gershenfeld, editors. *Time series prediction*, volume XV of *Santa Fe Institute Studies in the Sciences of Complexity*. Addison-Wesley, Reading, MA, 1994.

[84] C.F.J. Wu. On the convergence properties of the EM algorithm. *Annals of Statistics*, 11(1):95–103, 1983.

[85] R. Wylie, R. Orchard, M. Halasz, and F. Dub. IDS: Improving aircraft fleet maintenance. In *Proc. 14th Nat'l Conf. Innovative Applications of Artificial Intelligence (IAAI-97)*, pages 1078–1085, 1997.

[86] Y. Xiang. A probabilistic framework for cooperative multi-agent distributed interpretation and optimization of communication. *Artificial Intelligence*, 87(1):295–342, 1996.

[87] Y. Xiang. Verification of DAG structures in cooperative belief network based multi-agent systems. *Networks*, 31:183–191, 1998.

[88] E. Zalta, editor. *Stanford Encyclopedia of Philosophy*. 1998. ISSN 1095-5054. URL `http://plato.stanford.edu`.

Appendix A

THE RISO BELIEF NETWORK GRAMMAR

This appendix describes the grammar of the belief network description language implemented by RISO. The RISO grammar is divided into a grammar for the belief network proper (Table A.1), a grammar for variables (Table A.2), and a grammar for conditional distributions, an example of which is given in Table A.3. The grammar has been so divided in order to make it easier to replace part of the grammar. Although is is unlikely that someone will need to replace the grammar for the belief network proper, it will commonly be useful to extend the grammar for variables with additional productions, and for each new representation of a conditional distribution a new grammar analogous to Table A.3 will be added. In the rest of this paper, "the grammar" without qualification refers to the sum total of productions for the belief network proper, variables, and distributions.

Some features of the RISO grammar have been invented solely to improve the readability of belief network descriptions. The grammar is organized so that all the descriptive data pertaining to a variable are found together. (In some other belief network formats, data which pertain to the same variable are found in different locations within the description file — this is the case with the existing BNIF [3].) Grouping all of a variable's data together should make the description more comprehensible. Also, each input token is labeled with the name of the parameter which it represents, if applicable; this makes it easier to recall the meaning or purpose of a particular number or string.

The following notation is used in the grammars shown in the tables in this appendix. $[x|y]$ means x or y may occur, or neither, but not both. $[x|y]!$ means either x or y occurs, but not both. $[x]$ means x may occur once or may not occur at all. $[x]*$ means x occurs zero or more times. $[x]+$ means x occurs one or more times. Nonterminals are shown in italics, and terminals are set in a fixed-width font. Single-character terminals, such

Table A.1: Productions of the grammar for the belief network proper; productions for variables are shown in Table A.2, and productions for one type of distribution are shown in Table A.3. Nonterminals are shown in italics. Terminals are shown in fixed-width font. The nonterminals *host*, *domain*, and *java-classname* must follow the rules established for Internet host names and domain names, and Java class names, respectively. For clarity, single-character terminals are enclosed in quote marks.

1	*belief-network*	::=	*bn-type bn-name* [“{” [*variable-description*]+ “}”]
2	*variable-description*	::=	*variable-type variable-name* [“{” *var-descript-data* “}”]
3	*bn-type*	::=	*java-classname*
4	*variable-type*	::=	*java-classname*
5	*bn-name*	::=	*local-identifier*
6	*variable-name*	::=	[[*host* [“.” *domain*] [“:” *port*] “/”] *namespace-name* “.”] *local-identifier*
7	*namespace-name*	::=	*identifier*
8	*local-identifier*	::=	*identifier*
9	*identifier*	::=	*nonnumeric-char* [*nonnumeric-char* | [“0”-“9”]!]*
10	*nonnumeric-char*	::=	[“A”-“Z” “a”-“z” “@” “$” “-” “_” “?”]!
11	*port*	::=	*unsigned-integer*
12	*unsigned-integer*	::=	[“0” | [[“1”-“9”]![“0”-“9”]*]]!

as curly braces and parentheses, are quoted to better distiguish them from the special characters of the grammar. Of course, the quote marks wouldn't appear in a belief network description file.

A.1 Implementation details

The Java programming language was chosen to implement a parser for the RISO grammar. One feature of Java is particularly important for the parser: namely the class loader, which makes it possible to load compiled Java code at run time.

Like the grammar shown in Tables A.1–A.8, the implementation of the parser is divided into three parts. The parser for the grammar of the belief network proper is implemented by a function of the class `BeliefNetwork`; the parser for the grammar for variables is implemented by a function of the class `Variable`; and every class which

Table A.2: Productions of the grammar for variables. The nonterminal *distribution-description* is parsed by a method of a class, named as *distribution-type*, which inherits from the abstract base type `ConditionalDistribution`.

1	*var-descript-data*	::=	[`type` [`continuous` | [`discrete` [*states-list*]]]!] [`parents` [*parents-list*]] [`distribution` *distribution-type* [“{” *distribution-description* “}”]]
2	*parents-list*	::=	“{” [*simple-parent* | *prev-parent*]+ “}”
3	*simple-parent*	::=	*variable-name*
4	*prev-parent*	::=	“`prev[`” [*simple-parent* | *prev-parent*] “`]`”
5	*distribution-type*	::=	*java-classname*
6	*states-list*	::=	“{” [*quoted-string*]+ “}”
7	*quoted-string*	::=	“`"`” [“\01” - “\0377”]* “`"`”

Table A.3: Productions of the grammar for the conditional discrete distribution. The class `ConditionalDiscrete` inherits from `ConditionalDistribution`.

1	*cond-discrete*	::=	`ConditionalDiscrete` “{” `nparents` *unsigned-integer* `parents-dimensions` “{” [`unsigned-integer`]+ “}” `probabilities` “{” [*float*]+ “} }”
2	*float*	::=	*unsigned-integer* [“`.`” *unsigned-integer* [“`e`” [“`-`”|“`+`”] *unsigned-integer*]]

Table A.4: Productions of the grammar for the unconditional discrete distribution.

1	*discrete*	::=	`Discrete` “{” `dimensions` “{” [`unsigned-integer`]+ “}” `probabilities` “{” [*float*]+ “} }”

Table A.5: Productions of the grammar for the Gaussian distribution.

1	*gaussian*	::=	`Gaussian` “{” `mean` *float* `std-deviation` *float* “}”

Table A.6: Productions of the grammar for an unconditional mixture distribution. The nonterminal *unconditional-distribution* is a description of any type of unconditional distribution, including mixture types. The "regularization gammas" are parameters which come into play in maximum a posteriori estimation of mixing proportions from observed data, and do not affect computations of posterior distributions in belief network inferences.

1	*mixture*	::=	`Mixture` "{" `ncomponents` *unsigned-integer* `mixing-proportions` "{" [*float*]+ "}" [`regularization-gammas` "{" [*float*]+ "}"] `components` "{" [*unconditional-distribution*]+ "}" "}"

Table A.7: Productions of the grammar for monotone spline (Appendix D) distribution. The five floating-point numbers in each node description are the abscissa of the node, the ordinate, the slope at the abscissa, and two additional parameters describing the second and third derivatives of the spline.

1	*spline-density*	::=	`SplineDensity` "{" "{" [*node-description*]+ "}"
2	*node-description*	::=	*float float float float float*

Table A.8: Productions of the grammar for a density based on a regression model. The nonterminal *noise-model-description* can be any unconditional distribution. The nonterminal *regression-model-description* is the description of a model such as a squashing network, harmonic function, or polynomial.

1	*regression-density*	::=	`RegressionDensity` "{" `regression-model` *regression-model-description* `noise-model` *noise-model-description* "}"

represents a conditional distribution will have a parser for its own grammar. This will make it easier to extend the belief network grammar with additional attributes for variables and additional types of distributions. A general approach for creating such extensions is described in §A.1.4.

A.1.1 *Temporal dependence*

A primitive means of specifying temporal dependencies has been implemented in RISO; this allows the construction of Markov models and allied forms of belief networks. A temporal dependence is specified in the parents list of a variable (Production 2 in Table A.2). A temporal dependence is shown by a parent name of the form `prev[X]` where X is the name of a variable in some belief network. Such references can be nested, e.g., `prev[prev[prev[Y]]]` refers to the variable Y three time slices back.

A.1.2 *Arbitrary continuous/discrete conditional densities*

The Java type `ConditionalDistribution` is an abstract representation of a conditional probability distribution.[1] In order to be considered a concrete realization of the abstract conditional distribution type, a class must implement at least these functions:

- A function which reads a description of the distribution from a string.
- A function which writes a description of the distribution to a string. The output format should be the same as the input format.
- A function which computes the probability at a point, given values for any parent variables there may be.
- A function which returns an effective support of the distribution, that is, an interval which contains a mass of at least $1 - \epsilon$, where ϵ is a small number.

[1] To use the Java terminology, `ConditionalDistribution` is an interface.

It is certainly possible that a distribution might implement additional functions; the functions mentioned provide useful basic capabilities.

A variable may be described by any class which implements the `ConditionalDistribution` functions; the class need not be part of the RISO software. The name of the class which describes the variable is specified before the name of the variable. Any data needed to specify the distribution follows the variable's name. By convention, the format of this data follows the general belief network format: if non-empty, the data is contained within curly braces; each attribute is specified by a single token, or by multiple tokens contained within curly braces.

As an unconditional distribution is a particular type of conditional distribution, all unconditional distributions are also derived from `ConditionalDistribution`. Thus an unconditional distribution type-name may appear in place of the nonterminal *distribution-type* in Table A.1. Such a type is only appropriate for a variable which has no parents, of course.

A.1.3 *Identifier scope rules*

In the RISO grammar, each belief network is allocated a namespace. Within each namespace, identifiers must be unique. However, the same local identifier may occur in two different namespaces. To resolve an ambiguous reference, an identifier is qualified with the "." operator; an unqualified identifier is assumed to exist in the current namespace. In the current implementation, namespaces cannot be nested, although a future version of RISO may allow nested namespaces.

A belief network file may contain more than one belief network. At least one of the names of the belief networks in a file must coincide with the name of the file which contains it. The extension of the filename is "`riso`". For example, a file named "`sensor-diagnosis.riso`" must contain a belief network named "*sensor-diagnosis*," and it may contain other belief networks as well.

A qualified variable name of the form *some-bn.x* refers to a variable in a belief network on the same host as the network in which the reference occurs. If the belief

network *some-bn* is not already loaded, then it is loaded from the file `some-bn.riso` on the local filesystem, and the variable x is sought within it. A qualified variable name of the form *some-host/some-bn.x* refers to a variable in a belief network on the same or a different host. The belief network *some-bn* is located by connecting to a daemon listening on a specified port on *some-host*, and x is sought within the belief network. The hostname can be a fully-qualified symbolic Internet address, although one need specify only enough to locate the host. The address can include a port number, e.g. *cedar.colorado.edu:2099*.

A.1.4 *Extending the class* `Variable`

A new property of variables is introduced by deriving a new class from `Variable`. The grammar shown in Table A.2 contains just a few basic properties for variables: name, type (continuous or discrete), a list of parents, and the description of a conditional distribution. These are enough to support some basic computations, but additional properties will be needed for many purposes. New properties of variables can easily be accomodated by the following scheme: To represent a variable with extended properties, a new class is derived from `Variable`. The parser for the `Variable` class is applied to the input stream, and control is transferred (by throwing an exception) to the parser for the new class whenever a keyword unknown to `Variable` appears in the input stream. This approach allows backward compatibility of types of variables, since a program using the new variable class can read variables of the original class as well.

At present, it is planned that the class `Variable` will make certain assumptions about default values for unspecified attributes. If the type is not specified, it is assumed to be continuous. If no parents are specified, there are assumed to be none. If no distribution is specified, none is assumed; probability calculations cannot be carried out, but assessments of dependence and independence are still meaningful. A class which extends `Variable` can substitute other default values for the ones mentioned here.

A.2 Additional remarks

There appear to be only two proposals for a belief network format not tied to any particular software package, namely the Belief Network Interchange Format (BNIF) [3] and the "Interchange Format" proposed by Fabio Cozman [21]. The latter is a simplified version of the BNIF, and like the the BNIF, it is proposed as a public belief network format to promote the exchange of belief networks, especially on the Internet.

Some of the features of the RISO belief network grammar are implemented in some way by an existing belief network format. Continuous variables are allowed by many formats. Temporal dependencies are allowed by the Netica format [4]. Scope rules are also mentioned in the Netica grammar, but there is a note (production 40 in the Netica grammar) to the effect that the scope rules are not implemented. The "Bayesian inference using Gibbs sampling" (BUGS) language [35] does not allow definition of arbitrary probability distributions, but it does provide a wide variety of built-in distributions.

Appendix B

RISO COMMUNICATIONS ARCHITECTURE

This appendix contains some notes on the manner in which belief networks communicate with each other, whether from one host to another or among belief networks on a single host. The major communication problem is the transmission of π- and λ-messages, but several other lesser problems are also handled by RISO, such as locating belief networks by name, invalidating partial results, and downloading class files. The details of the communication scheme are not too important, and in an "industrial strength" implementation the communications architecture would likely be heavily revised.

Communications medium. The RISO communications medium is the Internet. Probability distributions and likelihood functions are passed as blocks of data over socket connections. The sockets are handled by Java's remote procedure call mechanism, called Remote Method Invocation (RMI).

Belief network descriptions. RISO belief network descriptions are plain text. A description can be loaded from the local filesystem or sent as a string across a socket.

The belief network registry. Names of belief networks and contexts are kept in a externally accessible list called the "registry," which is maintained by the RMI software. Each host has a registry for the belief networks and contexts it is running. The registry stores each name with a reference (essentially a pointer) to the corresponding program object. From any other host, one can look up a name in the registry and obtain the reference. The registry listens for incoming connections on a well-known socket, usually 1099.

Obtaining a remote reference. When a child wants to connect to a remote parent (i.e., a parent in another belief network, on the same host or a different one), the child looks up

the parent's belief network name in the parent's host's registry, and uses the reference so obtained. The required information can be obtained from the long form of the parent's name:

birch.colorado.edu:1088/test-monitor.status

Here the host is *birch.colorado.edu*, the port on which the registry listens is 1088, the parent's belief network is *test-monitor*, and the parent variable is *status*. If not specified, the host defaults to the same host as the child, the port defaults to 1099, and the belief network defaults to the same belief network as the child.

If the child locates the parent's host but the parent belief network is not running, the child requests that the parent belief network be loaded from the local filesystem; this will causes any other belief networks which are farther upstream to be loaded as well. This scheme is reminiscent of the resolution of function dependencies when loading ordinary function libraries.

Function calls using a remote reference. The reference obtained from the registry can be used to form function calls. The marshalling and unmarshalling of function arguments in handled by RMI. Function arguments are converted into socket messages by the caller and sent to the host holding the reference. The appropriate function is called, and the return value is converted into a socket message and sent back to the caller.

This arrangement makes it possible to write programs which access already-running belief networks, even on different hosts. Such "afterthought" programming can be used to implement new operations (e.g., computation of entropy) that are not yet included in RISO.

Callbacks to remote programs. A program outside a belief network can request that it be notified (by RMI) when the posterior or another quantity (π, λ, or a π- or λ-message) is computed for a specified variable. The observer's callback will also be called when the posterior, etc., is cleared.

This scheme can be useful for "afterthought" programs which plot or display information related to a variable, as the evidence for a variable changes one way or another.

Invalidating partial results. Requests to set, clear, or change evidence within a belief network cause special messages to propagate through the belief network and any other belief networks reachable by d-connection to the modified variable. These messages are "invalid π-message," sent to child variables, and "invalid λ-message," sent to parents. Upon receipt of one of these messages, the corresponding message is cleared, and the posterior of the variable is cleared. In keeping with the general "lazy inference" policy, the posterior is not recomputed until there is a request for it.

Coping with lost connections. A belief network might die for any one of several reasons — the host crashes, the context is killed, the name of the belief network is removed from the registry, or the belief network is marked "stale." If a child notices that its parent is no longer reachable, it will try to reconnect to the parent. If the reconnection fails, the child uses the parent's prior in place of a π-message. (If no prior is available, all queries on the child will simply fail.) Using the prior is equivalent to assuming there is no evidence available from the parent.

On the other hand, if a parent notices that a child is unreachable, it simply removes the child from its list of children. Since no evidence is available from the child, it is effectively the same as a "non-informative" lambda message.

Downloading class files. The complete RISO software need be installed at only one site. (For now, that site is *civil.colorado.edu*.) Compiled code is downloaded to other sites as follows. A very short application called the "stub" is copied to each site. The stub is executed with the name of a RISO program as an argument. The stub calls the "`main`" function of the program, and to resolve this function call the runtime environment copies the required code from the site on which it is located (called the "codebase") over to the stub's host. Any further function calls are resolved in a similar manner, by copying over the appropriate code from the codebase. RISO belief networks may be composed of types which are present on the codebase — when the belief network is instantiated, the compiled code for any types not found on the stub host are downloaded.

This scheme has two substantial advantages. First, RISO need be installed on only one site, and other sites automatically use the most recent version of the software.

(The code is not cached on the stub host, but must be copied anew every time a RISO program is executed.) Second, the stub can prohibit the RISO program it executes from performing sensitive operations, such as reading and writing the local filesystem. This increases the security of the system, although to be remotely accessible, socket connections must be allowed.

Parallel message requests. A certain amount of parallel computation is possible in RISO's polytree inference algorithm. When a variable requires π-messages, RISO sends out requests for π-messages to all parents, then waits for the parents to send the messages. Likewise, λ-message requests are sent to all children, then RISO waits for the children to send messages. If the parents are on different hosts, the messages can be computed in parallel; likewise with the children.

Security. Little attention has been given to security. A major improvement would be the use of encrypted socket transmissions.

The registry on a given host is accessible from any other host. Some form of authorization and authentication should be enforced. Even a policy as primitive as that used to protect web pages would help.

Appendix C

π- AND λ-MESSAGES FOR SOME DISTRIBUTIONS

For some distributions, the integrations necessary to calculate π- and λ-messages can be carried out symbolically, and it is generally much faster to compute the symbolic result than to compute a numerical approximation. For some other distributions, an exact result is not known, but a close approximation can be computed symbolically. RISO checks the types of the partial results which are required for a given calculation, and if there is a corresponding symbolic result in this collection, that result is returned and a numerical approximation thereby avoided. The details of RISO's type-matching scheme are discussed in §5.4.

In this appendix, we will review a catalog of symbolic results, both exact and approximate, for a variety of distributions. These results are organized by the type of the result — posterior distribution, π_X, λ_X, π-message, and λ-message. There are doubtless many other special cases which could be added to this collection; indeed, RISO is structured especially to make it easy to formulate and make use of results for additional special cases.

For some kinds of calculations, there are helpers which accept mixtures of Gaussians, but no helper which accepts Gaussian distributions. If some incoming partial results are Gaussian, and no appropriate helper is found, each Gaussian is converted to a one-component mixture and a helper is again sought. For example, there is no helper to compute π_X which accepts one π-message of type Gaussian and one of type mixture of Gaussians; in this case, the Gaussian is converted to a one-component mixture and a helper which accepts two π-messages of type mixture of Gaussians is invoked.

Some cases of general interest are not covered in this appendix. Particularly important is the case of computing π_X with a mixture of conditional Gaussians with mixture of Gaussians π-messages. The mixture of conditional Gaussians distribution might be

derived from a mixture of multivariate Gaussians, as in the example of $RH \,|\, T$ in §6.7. The mixture of conditional Gaussians can also be considered a "mixture of experts" architecture, with each "expert" a linear regression [40]. It turns out that computing π_X for this case is difficult because the mixture proportions vary from one region of the parent space to another; this leads to algebraic difficulties. Given the expressive power of the mixture of conditional Gaussians, an exact rule or fast approximation for π_X and other quantities would be very useful.

C.1 A note on post-processing of mixture distributions

Some of the formulas stated in the following sections return results which are mixture distributions, often mixtures of Gaussians. Some heuristics are applied to these mixtures to make them easier to handle in further calculations. The heuristics are (i) flattening, (ii) pruning, and (iii) type conversions.

(i) Flatten. If any component p_i of a mixture p is itself a mixture, remove p_i from p and add each component p_{ij} to p with mixing component $\alpha_i \cdot \alpha_{ij}$.

(ii) Prune. Remove any component with mixing proportion less than a threshold, typically 0.0005. Ref. [6] describes a number of other pruning heuristics.

(iii) Convert types. If any component is a Gaussian with zero variance, replace it with a Gaussian delta function. (These types are formally equivalent, but represented in software by different types.)

C.2 Symbolic results for posterior distributions

The definition of the posterior distribution $p_{X|\mathbf{e}}$ of a variable X conditioned on some evidence $\mathbf{e}$ is given by Eq. 5.1.

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

EXACT RESULTS

C.2.1 *Both π_X and λ_X are discrete.*

In this case the computation of the posterior is very easy. The posterior is computed directly from the definition, with the constant of proportionality equal to

$$\sum_{x=0}^{\#X-1} \pi_X(x)\ \lambda_X(x) \tag{C.2}$$

where $\#X$ is the cardinality (assumed finite) of X.

C.2.2 *Discrete π_X and arbitrary λ_X.*

This case covers likelihood functions of a discrete variable which are expressed as integrals or other general functions, and not as look-up tables. The computation is carried out just as in the preceding section.

C.2.3 *Both π_X and λ_X are Gaussian.*

The posterior in this case is also Gaussian, with variance

$$\sigma_X^2 = 1/(1/\sigma_\pi^2 + 1/\sigma_\lambda^2) \tag{C.3}$$

and mean

$$\mu_X = \left(\frac{\mu_\pi}{\sigma_\pi^2} + \frac{\mu_\lambda}{\sigma_\lambda^2}\right) \times \sigma_X^2 \tag{C.4}$$

with obvious notation.

C.2.4 *Both π_X and λ_X are mixtures of Gaussians.*

The computation described in §C.4.3 is carried out with π_X and λ_X as the multiplicands. The result is pruned by throwing out components with a mass which is smaller than a threshold, and the pruned mixture is returned.

APPROXIMATE RESULTS

C.2.5 *Both π_X and λ_X are general mixtures.*

First π_X and λ_X are flattened. Let $\pi_X[i],\ i = 1, \ldots, n_\pi$ and $\lambda_X[j],\ j = 1, \ldots, n_\lambda$ be the components of π_X and λ_X (after flattening). The result is a mixture with $n_\pi \cdot n_\lambda$

components. Each component of the result is computed according to these rules:

- If $\lambda_X[j]$ is noninformative, the result component is $\pi[i]$.
- If $\pi[i]$ and $\lambda[j]$ are both Gaussian, the result component is a Gaussian with mean and variance given by Eqs. C.4 and C.3.
- If neither of the preceding two cases apply, the result component is a spline density approximation, constructed according to the description in §C.2.6.

In each case, the mass of the result component is the product of the masses assigned to $\pi_X[i]$ and $\lambda_X[j]$.

C.2.6 *Both π_X and λ_X are arbitrary.*

A monotone cublic spline (Appendix D) representation of the posterior is constructed. The support of the spline is taken as the intersection of the effective support of π_X and λ_X, or as the effective support of π_X alone if the effective support of λ_X cannot be determined. The number of knots is determined by computing, with an adaptive quadrature algorithm (QAGS, in QUADPACK at `www.netlib.org`), a numerical approximation of the normalizing integral

$$\int_{x\in I} \pi_X(x)\,\lambda_X(x)\,dx$$

where I is the support of the spline; the numerical calculation will require the evaluation of the integrand at some number of points in I, and when the calculation is complete these are taken as the knots of the spline. The spline is normalized so that its integral is unity, and the approximation is assumed to be zero outside of I.

C.3 Symbolic results for π_X calculations

The definition of π_X is given by Eq. 5.2.

$$\begin{aligned}\pi_X(x) &= p_{X|\mathbf{e}_X^+}(x) \\ &= \int du_1 \cdots \int du_m\; q_X(x,u_1,\ldots,u_m)\;\pi_{U_1,X}(u_1)\;\cdots\;\pi_{U_m,X}(u_m) \quad \text{(C.5)}\end{aligned}$$

This integration can be carried out for a number of special cases, including several of the form "the distribution of some function of such-and-such variables is thus."

EXACT RESULTS

C.3.1 *Identity with one arbitrary parent.*

In this case the conditional distribution of the child X is a delta function placed on the given value of the parent U. (Such a distribution is useful for constructing "shadow" variables in a belief network.) Thus we have

$$\pi_X(x) = \pi_{U,X}(x) \tag{C.6}$$

C.3.2 *Sum of Gaussian variables.*

The sum of Gaussian variables is a special case of a linear combination of Gaussian variables, as described in §C.3.4, taking the coefficients $a_1, \ldots, a_m$ all equal to 1.

C.3.3 *Sum of mixtures of Gaussian variables.*

The sum of variables with mixture of Gaussians distributions is a special case of the linear combination described in §C.3.5, taking the coefficients $a_1, \ldots, a_m$ all equal to 1.

C.3.4 *Linear combination of Gaussian variables.*

A linear combination

$$a_1 X_1 + a_2 X_2 + \cdots + a_m X_m \tag{C.7}$$

of Gaussian variables is again a Gaussian variable, with mean

$$a_1 \mu_1 + a_2 \mu_2 + \cdots + a_m \mu_m \tag{C.8}$$

and variance

$$a_1^2 \sigma_1^2 + a_2^2 \sigma_2^2 + \cdots + a_m^2 \sigma_m^2 \tag{C.9}$$

A sum of Gaussian variables (§C.3.2) is a special case.

C.3.5 *Linear combination of mixtures of Gaussian variables.*

The distribution of a linear combination (Eq. C.7) of variables with mixtures of Gaussians distributions is again a mixture of Gaussians. Let the mixing proportion, mean, and variance of the k'th component of the j'th π-message be denoted α_{jk}, μ_{jk}, and σ^2_{jk}, respectively. Let n_j denote the number of components of the j'th π-message. Then the result is a mixture of Gaussians with $n_1 n_2 \cdots n_m$ components, where m is the number of parents, each component corresponding to a combination $(k_1, k_2, \ldots, k_m)$ of components of the π-messages. The mixing proportion of the component corresponding to each such combination is the product of mixing proportions,

$$\alpha_{1,k_1} \cdot \alpha_{2,k_2} \cdots \alpha_{m,k_m} \tag{C.10}$$

and the mean of the component is a linear combination of the means,

$$a_1\, \mu_{1,k_1} + a_2\, \mu_{2,k_2} + \cdots + a_m\, \mu_{m,k_m} \tag{C.11}$$

and the variance of the component is a linear combination of the variances,

$$a_1^2\, \sigma^2_{1,k_1} + a_2^2\, \sigma^2_{2,k_2} + \cdots + a_m^2\, \sigma^2_{m,k_m} \tag{C.12}$$

C.3.6 *Product of lognormal variables.*

As the sum of Gaussian variables is again Gaussian, so the product of lognormal variables is again lognormal. Let $\mu_1, \ldots, \mu_m$ and $\sigma_1^2, \ldots, \sigma_m^2$ denote the parameters of the π-messages. Then the product has the parameters

$$\mu = \mu_1 + \cdots + \mu_m, \tag{C.13}$$

and

$$\sigma^2 = \sigma_1^2 + \cdots + \sigma_m^2 \tag{C.14}$$

C.3.7 *Ratio of lognormal variables.*

As the difference of Gaussian variables is again Gaussian, so the ratio of lognormal variables is again lognormal. A ratio variable has two parents; it is assumed the first is

the numerator and the second is the denominator. Let μ_1, σ_1 be the parameters of the numerator, and let μ_2, σ_2 be the parameters of the denominator. Then the ratio has the parameters

$$\mu = \mu_1 - \mu_2 \tag{C.15}$$

and

$$\sigma^2 = \sigma_1^2 + \sigma_2^2 \tag{C.16}$$

C.3.8 *Maximum of arbitrary distributions.*

The cumulative distribution function (c.d.f.) of the maximum of some set of distributions is just the product of the c.d.f. of each distribution in the set, and the density function is just the derivative of that product. So we have for the c.d.f. of π_X

$$\pi\text{-cdf}_X(x) = \pi\text{-cdf}_{U_1,X}(x) \cdots \pi\text{-cdf}_{U_m,X}(x) \tag{C.17}$$

where $\pi\text{-cdf}_{U_k,X}(x)$ is written in place of $\int_{-\infty}^{x} \pi_{U_k,X}(t)\, dt$. For the density, we have

$$\pi_X(x) = \sum_{j=1}^{m} \pi_{U_j,X}(x) \prod_{\substack{k=1 \\ k \neq j}}^{m} \pi\text{-cdf}_{U_k,X}(x) \tag{C.18}$$

C.3.9 *Minimum of arbitrary distributions.*

The complement of the c.d.f. of the minimum of a set of distributions is the product of the complement of the c.d.f. of each distribution. So we have for the c.d.f. of π_X

$$\pi\text{-cdf}_X(x) = 1 - (1 - \pi\text{-cdf}_{U_1,X}(x)) \cdots (1 - \pi\text{-cdf}_{U_m,X}(x)) \tag{C.19}$$

and for the density,

$$\pi_X(c) = \sum_{j=1}^{m} \pi_{U_j,X}(x) \prod_{\substack{k=1 \\ k \neq j}}^{m} (1 - \pi\text{-cdf}_{U_k,X}(x)) \tag{C.20}$$

C.3.10 *Disjunction of binary variables.*

The probability of a disjunction of independent binary variables is conveniently computed by noting that

$$\begin{aligned} \Pr(S_1 \vee S_2) &= \Pr(S_1) + \Pr(S_2) - \Pr(S_1)\ \Pr(S_2) \\ &= \Pr(S_2) + \Pr(S_1) \cdot (1 - \Pr(S_2)) \end{aligned} \tag{C.21}$$

from which we obtain the recursive formula

$$\begin{aligned} \Pr(S_1 \vee S_2 \vee \cdots \vee S_m) &= \Pr(S_m) + \Pr(S_1 \vee S_2 \vee \cdots \vee S_{m-1})\,(1 - \Pr(S_m)) \\ &= \Pr(S_m) + p_{m-1} \cdot (1 - \Pr(S_m)) \end{aligned} \tag{C.22}$$

writing p_k for $\Pr(S_1 \vee \cdots \vee S_k)$. Taking $p_0 = 0$, we can compute $p_1, p_2, \ldots, p_m$ in turn, and return p_m as the desired result.

This formula can be applied to discrete variables with more than two states by identifying the zero'th state as 0 and lumping all the other states together in 1. The result is still binary.

C.3.11 *Exclusive-or of binary variables.*

To obtain the probability of the exclusive-or of binary variables, express the exclusive-or in terms of conjunction and disjunction as follows:

$$\begin{aligned} \Pr(S_1 \oplus S_2) &= \Pr((S_1 \wedge \neg S_2) \vee (\neg S_1 \wedge S_2)) \\ &= \Pr(S_1 \wedge \neg S_2) + \Pr(\neg S_1 \wedge S_2) \\ &= \Pr(S_1)(1 - \Pr(S_2)) + (1 - \Pr(S_1))\Pr(S_2) \\ &= \Pr(S_2) + \Pr(S_1)(1 - 2\Pr(S_2)) \end{aligned} \tag{C.23}$$

The second equality follows from the first because $(S_1 \wedge \neg S_2) \wedge (\neg S_1 \wedge S_2)$ is identically false. Since exclusive-or is associative, we can compute $S_1 \oplus S_2 \oplus \cdots \oplus S_m$ recursively, with

$$\Pr(S_1 \oplus S_2 \oplus \cdots \oplus S_m) = \Pr(S_m) + \Pr(S_1 \oplus \cdots \oplus S_{m-1})(1 - 2\Pr(S_m)) \tag{C.24}$$

$$= \Pr(S_m) + p_{m-1} \cdot (1 - 2\Pr(S_m)) \tag{C.25}$$

writing p_k for $\Pr(S_1 \oplus \cdots \oplus S_k)$. Taking $p_0 = 0$, we compute $p_1, p_2, \ldots, p_m$ in turn, and return p_m as the desired result.

This formula can be applied to discrete variables with more than two states by identifying the zero'th state as 0 and lumping all the other states together in 1. The result is still binary.

C.3.12 *"Exactly one" of binary variables.*

The proposition "exactly one of $S_1, \ldots, S_m$" can be expressed as

$$\text{exactly one of } S_1, \ldots, S_m = \bigvee_{j=1}^{m} S_j \wedge \bigwedge_{\substack{k=1 \\ k \neq j}}^{m} \neg S_k \tag{C.26}$$

Since the terms in the disjunction are mutually exclusive, and the terms in the conjunction are assumed independent, the probability of "exactly one of..." is a simple calculation.

$$\Pr(\text{ exactly one of } S_1, \ldots, S_m \text{ }) = \sum_{j=1}^{m} \Pr(S_j) \prod_{\substack{k=1 \\ k \neq j}}^{m} (1 - \Pr(S_k)) \tag{C.27}$$

The Elvis Presley belief network described in Chapter 1 makes use of a variable which is "exactly one" of several binary variables.

C.3.13 *Indexed distribution with discrete π-messages.*

An "indexed distribution" is the name given (in RISO) to a conditional distribution with some discrete parents and some continuous parents. One can imagine that the discrete parents index a set of distributions conditioned on the continuous parents alone. The child distributions can have any types.

As a special case, when all the parents are discrete, the indexed distributions are unconditional. Then π_X for the child is just a mixture distribution, with each component corresponding to one of the indexed distributions, and its mixing proportion equal to a product of probabilities from the π-messages. Letting i_k be an index for the k'th π-message, and letting Q be the set of child distributions, then the mixture components

are

$$Q[i_1, \ldots, i_m] \tag{C.28}$$

and the mixing proportions are

$$\pi_{U_1,X}[i_1] \times \cdots \times \pi_{U_m,X}[i_m] \tag{C.29}$$

Since the child distributions need not all have the same type, this mixture is not generally any common type such as a mixture of Gaussians.

C.3.14 *Conditional Gaussian with Gaussian π-messages.*

If the child has a conditional Gaussian distribution with its conditional mean a linear combination of the parent values $u_1, \ldots, u_m$,

$$b + a_1 u_1 + a_2 u_2 + \cdots + a_m u_m$$

and constant conditional variance σ^2, then π_X is a Gaussian distribution with mean

$$b + a_1 \mu_1 + a_2 \mu_2 + \cdots + a_m \mu_m \tag{C.30}$$

and variance

$$\sigma^2 + a_1^2 \sigma_1^2 + a_2^2 \sigma_2^2 + \cdots + a_m^2 \sigma_m^2 \tag{C.31}$$

Here the mean and variance of the k'th π-message are denoted μ_k and σ_k^2, respectively.

C.3.15 *Conditional discrete with discrete parents.*

In this case, π_X is computed directly from the definition (Eq. 5.2 or C.5). The integrals are replaced by summations:

$$\pi_X(j) \;=\; \sum_{i_1} \cdots \sum_{i_m} q_X(j, i_1, \ldots, i_m)\, \pi_{U_1,X}(i_1) \;\cdots\; \pi_{U_m,X}(i_m) \tag{C.32}$$

where q_X is written as a shorthand for $p_{X|U_1,\ldots,U_m}$, the index j ranges from 0 to $\#X - 1$, and the indices i_k ranges from 0 to $\#U_k - 1$.

C.3.16 [*] *Autoregressive model with Gaussian parent.*

If the π-messages for ρ and σ are not delta functions, so the problem doesn't fall into the special case, the general π_X handler described in §C.3.26 is invoked.

APPROXIMATE RESULTS

C.3.17 *Sum of arbitrary distributions.*

In general, the density of a sum of variables is the convolution of the densities of the summands,

$$p_{x+y}(t) = \int_{-\infty}^{\infty} p_x(t-u)\, p_y(u)\, du$$

The convolution is approximated by discretizing the summands, computing the discrete Fourier transform of each variable, multiplying the Fourier transforms, and computing the inverse transform of the result. Especially if there are more than two summands, this is generally much faster than directly computing the convolution integral, since there are "fast" algorithms (see, e.g., Ref. [10]) which compute the discrete Fourier transform in time proportional to $N \log N$, where N is the number of sampling points. A trick (described in Figure 10-9 of Ref. [10]) is employed to compute the discrete Fourier transform of two summands at a time.

Every summand must be discretized with the same step size; changing the step size is equivalent to rescaling a variable, and the sum of rescaled variables is not related in any obvious way to the sum of the corresponding unscaled variables. The discretization is constructed by finding an effective support for each summand, and setting step size according to the support interval which is shortest; this ensures that all summands are sampled adequately. The number of grid points for each summand is

$$N_i = \left\lceil \frac{b_i - a_i}{\Delta x} \right\rceil$$

where a_i and b_i are the left and right endpoints of the effective support of the i'th summand, and Δx is the step size (same for all summands). The number of grid points in the discrete Fourier transform of the sum is equal to

$$N_1 + N_2 + \cdots + N_m - (m-1)$$

which could be large if some summand has an effective support which is much longer than the shortest effective support of any summand.

The discretization of the i'th summand is taken as

$$p_{ij} = p_i(a_i + j\Delta x), \qquad j = 0, \ldots, N_i - 1$$

That is, each density is simply sampled at the grid points. It might be more accurate to let p_{ij} equal the mass of the density function p_i on the interval $[a_i + j\Delta x, a_i + (j+1)\Delta x]$, but that has not yet been implemented in RISO.

The result of the discrete convolution is represented as a continuous density function by constructing a monotone cubic spline function (Appendix D) which has knots at the grid points. The function values are scaled so that the spline integrates to unity on the interval $[a_\Sigma, b_\Sigma]$, where a_Σ and b_Σ are the endpoints of the support of the sum,

$$a_\Sigma = \sum_{i=1}^{m} a_i, \qquad b_\Sigma = \sum_{i=1}^{m} b_i$$

C.3.18 *Product of distributions supported on* $(0, +\infty)$.

This case is handled by taking the logarithm of each of the multiplicands, computing the sum of the log-transformed variables, and taking the exponential of the result. The sum is computed as described in §C.3.17.

The density of a log-transformed variable is just

$$p_{\log x}(t) = p_x(\exp t)\ \exp t$$

while the density of an exponential-transformed variable is

$$p_{\exp x}(t) = p_x(\log t)\ \frac{1}{t}$$

Each multiplicand is log-transformed, then discretized. The convolution of the log-transformed multiplicands is computed, and a discrete approximation (with non-uniform step size) to the density of the product is then

$$x_i = \exp y_i, \qquad p_i = q_i / x_i$$

where y_i is a grid point in the log-transform space and q_i is the corresponding density of the summation of log-transformed variables.

A monotone cubic spline (Appendix D) is constructed from the pairs (x_i, p_i).

C.3.19 *Ratio of distributions supported on* $(0, +\infty)$.

We take the logarithm of the numerator and denominator. The log-transformed denominator is reflected about the origin before convolving with the log-transformed numerator. Then the result is transformed back to the original units. The details are just the same as in §C.3.18.

C.3.20 *Functional relation with arbitrary π-messages.*

The number of π-messages which are not delta functions is counted. The action taken depends on the number of non-delta π-messages.

All π-messages are delta functions. In this case, the integral (Eq. C.5) collapses to a function evaluation:

$$\pi_X(x) = \delta(x - F(u_1, \ldots, u_m)) \tag{C.33}$$

with each u_k the support point of the corresponding delta function $\pi_{U_k,X}$.

One π-message is non-delta. If there is exactly one π-message which is non-delta, the integral (Eq. C.5) becomes a one-dimensional integration. Without loss of generality, assume the non-delta π-message is $\pi_{U_1,X}$. The functional relation is effectively a function of one variable, $f(u) = F(u, u_2, \ldots, u_m)$. So the integral to be evaluated is

$$\int \delta(v - f(u))\, \pi_{U_1,X}(v)\, dv \tag{C.34}$$

This can be evaluated by a standard change of variables [62, § 5.2] to yield

$$\pi_X(x) = \sum_i \frac{\pi_{U_1,X}(u_i)}{|f'(u_i)|} \tag{C.35}$$

where the summation is over the preimages u_i of x, that is, $x = f(u_i)$ for each i. The preimages u_i are found by evaluating f at a large number of points in the effective

support S of $\pi_{U_1,X}$ and looking for sign changes of $x - f(u)$; an interval containing a sign change is refined by bisection to yield an approximation to the preimage u_i.

Eq. C.35 is evaluated at a number of points in the image $f(S)$ of the effective support S of $\pi_{U_1,X}$, and a spline approximation is constructed.

More than one π-message is non-delta. If there are N non-delta π-messages with $N > 1$, the integration (Eq. C.5) is expressed as an $(N-1)$-dimensional integration, with Eq. C.35 as the integrand. To make the integration easier, the π-message with greatest variance is handled by Eq. C.35; without loss of generality, let us assume that it is $\pi_{U_1,X}$. Thus the required computation is

$$\pi_X(x) = \int dU_2 \cdots \int dU_m \; \pi_{U_2,X}(U_2) \cdots \pi_{U_m,X} \; \sum_i \frac{\pi_{U_1,X}(u_{1,i})}{|\frac{\partial F}{\partial U_1}(u_{1,i}, U_2, \ldots, U_m)|} \tag{C.36}$$

where the preimages of x, for given $U_2, \ldots, U_m$, are denoted $u_{1,i}$. These preimages are found as described under the heading "One π-message is non-delta," above; the search is still one-dimensional. The integration over $U_2, \ldots, U_m$ is carried out by quasi Monte Carlo (§5.6) if $N > 2$, and by QAGS if $N = 2$.

C.3.21 *Regression model with Gaussian inputs.*

A regression model, in RISO, is a real function of one or more inputs, with some constant-variance noise assumed on the output. The distribution of the output is approximated by linearizing the function at $(\mu_1, \ldots, \mu_m)$,

$$F(x) \approx F(\mu) + \nabla F(\mu) \cdot (x - \mu)$$

where μ_i is the mean of the i'th π-message; for brevity, x is written for $(x_1, \ldots, x_m)$, likewise μ for $(\mu_1, \ldots, \mu_m)$. Then an exact expression for the distribution of the linearized output is computed. Let $F(x_1, \ldots, x_m)$ be the regression function, let σ_F^2 be the noise variance, and let μ_i, σ_i^2 be the parameters of the i'th π-message. Then the distribution of the linearized output is Gaussian, with mean

$$F(\mu_1, \ldots, \mu_m) \tag{C.37}$$

and variance

$$\sigma_F^2 + \sum_{i=1}^{m} \left(\frac{\partial F}{\partial x_i}(\mu_i) \right)^2 \sigma_i^2 \tag{C.38}$$

This is just a special case of the well-known result that for x Gaussian-distribution with mean μ and covariance matrix Σ, the matrix product Bx is again Gaussian, with mean $B\mu$ and covariance $B\Sigma B'$. See, for example, Chapter VIII of Ref. [81].

If the function F is strongly nonlinear, it would be more accurate to replace each Gaussian bump in the input with a Gaussian mixture of perhaps 3, 5, or many components. The components would be spread over the effective support of the bump, perhaps placed at μ, $\mu \pm \sigma$, $\mu \pm 2\sigma$, etc. The resulting approximation of the output distribution would be a Gaussian mixture which better captures the nonlinear effect of F. However, this idea has not yet been implemented in RISO.

C.3.22 *Regression model with mixtures of Gaussians inputs.*

The regression function is linearized as in §C.3.21, and an exact result is calculated for the linearized function. The distribution of the output is now a Gaussian mixture, with one component for each combination of the components of the input distributions, so there are $n_1 \cdot n_2 \cdots n_m$ altogether. The mixing proportion of each component is

$$\alpha_{1,i_1} \cdot \alpha_{2,i_2} \cdots \alpha_{m,i_m} \tag{C.39}$$

where the index i_j ranges over the components of the j'th π-message. The mean of each component is

$$F(\mu_{1,i_1}, \mu_{2,i_2}, \ldots, \mu_{m,i_m}) \tag{C.40}$$

and the corresponding variance is

$$\sigma_F^2 + \sum_{j=1}^{m} \left(\frac{\partial F}{\partial x_j}(\mu_{j,i_j}) \right)^2 \sigma_j^2 \tag{C.41}$$

Note that this approximation requires one linearization per component in the output distribution.

C.3.23 *Regression model with arbitrary inputs.*

If some incoming π-messages are not Gaussian or mixtures of Gaussians, those messages are approximated as mixtures of Gaussians, and the formulas in §C.3.22 are applied. The mixture of Gaussian approximations are computed by the algorithm described in §5.5.

C.3.24 *Classifier with arbitrary inputs.*

A classifier is a conditional probability model for a discrete child with discrete or continuous parents. For this type of model, π_X is computed directly from its definition; this requires an m-dimensional numerical integration. If $m > 1$, the integration algorithm is a quasi Monte Carlo algorithm; if $m = 1$, the integration algorithm is QAGS, an adaptive quadrature algorithm based on a 21-point Gauss-Kronrod rule. See §5.6 for details on integration algorithms used in RISO.

Conceptually, it may be preferable to specify the discrete variable as a parent, and the other variables as children, and to use the classifier to compute λ_X instead of π_X. However, that scheme has not been implemented in RISO.

C.3.25 *Indexed distribution with arbitrary parents.*

In this case, π_X is a mixture, with the number of components equal to the number of combinations of states of the index variables. Let the n index variables be $U_{j_1}, \ldots, U_{j_n}$. For each combination $(i_{j_1}, \ldots, i_{j_n})$ of index variables, there is one mixture component, with weight equal to the product of probabilities of the index variables,

$$\pi_{U_{j_1},X}[i_{j_1}] \times \cdots \times \pi_{U_{j_n},X}[i_{j_n}] \tag{C.42}$$

If the distribution q_X of the child X given the continuous parents $U \setminus \{U_{j_1}, \ldots, U_{j_n}\}$ is unconditional, then q_X is the mixture component. Otherwise, a π_X helper is sought, according to the types of the π-messages associated with the continuous parents, and the π_X computed by the helper is the mixture component.

C.3.26 *Arbitrary conditional distribution with arbitrary parents.*

In this case, no specific results are known, so π_X is computed directly from the definition (Eq. C.5). See §5.6 for details on integration algorithms.

C.4 Symbolic results for λ_X calculations

The definition of λ_X is given as Eq. 5.3.

$$\lambda_X(x) \propto p_{\mathbf{e}_X^-|X}(x) = \prod_{j=1}^{n} \lambda_{Y_j,X}(x) \tag{C.43}$$

EXACT RESULTS

C.4.1 *All λ-messages are discrete.*

In this case λ_X is again discrete. Let $p_{ij}, j = 0, 1, 2, \ldots, \#X - 1$, denote the probability table of the i'th λ-messages. Then the j'th element of the λ_X probability table is just

$$\lambda_X(j) \propto p_{1,j} \cdot p_{2,j} \cdots p_{m,j} \tag{C.44}$$

In this equation, the constant of proportionality is such that $\sum_j \lambda_X(j) = 1$. Strictly speaking, this normalization is optional, since likelihood functions are determined only up to a constant factor.

C.4.2 *All λ-messages are Gaussian.*

Let μ_i, σ_i^2 be the parameters of the λ-messages. Let

$$A = \sum_{i=1}^{m} \frac{1}{\sigma_i^2} \tag{C.45}$$

$$B = \sum_{i=1}^{m} \frac{\mu_i}{\sigma_i^2} \tag{C.46}$$

Then the λ-message is again Gaussian, with mean B/A and variance $1/A$.

C.4.3 *All λ-messages are mixtures of Gaussians.*

Let $p_1, \ldots, p_n$ be mixture distributions, each of which has some mixing parameters α_{kl} and component distributions p_{kl}. The density of each distribution is given by

$$p_k(x) = \sum_{l=1}^{N_k} \alpha_{kl}\, p_{kl}(x) \tag{C.47}$$

Then the product of these distributions is

$$\begin{aligned} \prod_{k=1}^{n} p_k(x) &= \prod_{k=1}^{n} \sum_{l=1}^{N_k} \alpha_{kl}\, p_{kl}(x) \\ &= \sum_{l_1=1}^{N_1} \cdots \sum_{l_n=1}^{N_n} \prod_{k=1}^{n} \alpha_{k,l_k}\, p_{k,l_k}(x) \end{aligned} \tag{C.48}$$

This shows that the result is again a mixture, with the number of components equal to the product $N_1 \cdots N_n$ of the numbers of components of each multiplicand, and mixing parameters equal to products $\alpha_{1,l_1} \cdots \alpha_{n,l_n}$ of the mixing parameters of the multiplicands. Each product $p_{1,l_1}(x) \cdots p_{n,l_n}(x)$ is evaluated according to Eqs. C.45 and C.46.

C.4.4 *All λ-messages are general mixtures.*

λ-messages which are mixtures may contain noninformative components, which originate from conditional distributions which are unconditional for some values of indexing parents. For example, a typical model for a malfunctioning temperature sensor assumes that the sensor output is not a function of the actual temperature.

If all λ-messages are general mixtures, then λ_X is also a general mixture. There is one component of λ_X for each combination of components of the λ-messages, and the type of the λ_X component depends on the types of the components of the λ-messages. A few special cases are detected, and one general case covers all other combinations. In the following, let n_i be the number of components of $\lambda_{Y_i,X}$, the i'th λ-message. Let $(j_1, \ldots, j_m)$ index a combination of components of the λ-messages, with $1 \le j_i \le n_i$, and let $\lambda_{Y_i,X}[j_i]$ be the j_i'th component of the i'th λ-message.

(i) If all the components $\lambda_{Y_i,X}[j_i]$ are noninformative, then the component of λ_X is also noninformative.

(ii) If all components $\lambda_{Y_i,X}[j_i]$ are noninformative except one, the component of λ_X is a copy of that sole informative component.

(iii) If all informative components $\lambda_{Y_i,X}[j_i]$ are Gaussian, the component of λ_X is also Gaussian, with mean and variance as stated in §C.4.2.

(iv) If none of the first three cases applies, the component of λ_X is represented explicitly as a product of the components $\lambda_{Y_i,X}[j_i]$. That is, a copy of each component is stored, and when the density function of the result needs to be computed, the density function of each $\lambda_{Y_i,X}[j_i]$ is computed and the product is returned. It is possible that storing a numerical approximation to the product (e.g., a spline approximation) might be a better idea.

As elsewhere, it would be a good idea to develop formulas for additional special cases — in this context, special cases of products of density functions.

APPROXIMATE RESULTS

C.4.5 *All λ-messages are arbitrary distributions.*

If X is discrete, the range over which λ_X is defined is known: the range is just $0, 1, 2, \ldots, \#X - 1$. To simplify further calculations, λ_X is evaluated for each value in its range, and the table of probability values so generated is stored, so that no further evaluations are needed. If λ_X becomes a partial result in another calculation (e.g., a posterior or π-message calculation), λ_X is represented in the calculation by the probability table.

If X is continuous, no evaluation or approximation is attempted, and λ_X is explicitly represented as a product of λ-messages. That is, a list of the messages is kept, and when λ_X needs to be evaluated, each λ-message is evaluated and the product of the results is returned.

C.5 Symbolic results for π-messages

The definition of the π-message π_{X,Y_k} from a parent X to its child Y_k is given by Eq. 5.4.

$$\pi_{X,Y_k}(x) = p_{X|\mathbf{e}\setminus\mathbf{e}^-_{X,Y}}(x) \propto \pi_X(x) \prod_{\substack{j=1 \\ j\neq k}}^{n} \lambda_{Y_j,X}(x) \tag{C.49}$$

From this definition one can see that a π-message is identical to the calculation of the posterior, with one of the λ-messages omitted. Thus we need only compute a likelihood function using one of the rules in §C.4 (omitting one λ-message), and then compute the product of π_X with the likelihood using a rule from §C.2.

C.6 Symbolic results for λ-messages

The definition of the λ-message λ_{X,U_k} from a child X to its parent U_k is given by Eq. 5.5.

$$\begin{aligned} \lambda_{X,U_k}(u_k) &= p_{\mathbf{e}\setminus\mathbf{e}^+_{X,U_k}|U_k}(u_k) \\ &\propto \int dx \int du_1 \cdots \int du_{k-1} \int du_{k+1} \cdots \int du_m \\ &\quad \times \lambda_X(x)\, q_X(x, u_1, \ldots, u_m) \prod_{\substack{j=1 \\ j\neq k}}^{m} \pi_{U_j,X}(u_j) \end{aligned} \tag{C.50}$$

EXACT RESULTS

C.6.1 *Conditional discrete with discrete likelihood and discrete π-messages.*

The result is a discrete probability distribution. A table of probabilities is computed by direct summation, with the i'th element

$$\lambda_{X,U_k}[i] \propto \sum_{j_X=0}^{\#X-1} \lambda_X[j_X] \sum_{j_1=0}^{\#U_1-1} \cdots \sum_{j_{k-1}=0}^{\#U_{k-1}-1} \sum_{j_{k+1}=0}^{\#U_{k+1}-1} \cdots \sum_{j_m=0}^{\#U_m-1} q_X[j_X, j_1, \ldots, j_m] \prod_{\substack{l=1 \\ l\neq k}}^{m} \pi_{U_l,X}[j_l] \tag{C.51}$$

The constant of proportionality is such that $\sum_i \lambda_{X,U_k}[i] = 1$. Strictly speaking, this normalization is optional, since likelihood functions are determined only up to a constant factor.

C.6.2 *Conditional Gaussian with Gaussian likelihood and Gaussian π-messages.*

The result is a Gaussian distribution. Suppose the conditional distribution of X given its parents $U = (U_1, \ldots, U_m)$ has conditional mean given by

$$\mu_{X|U} = A \cdot U + B,$$

and constant conditional variance $\sigma^2_{X|U}$. Let

$$\mu_{X,U_k} = \frac{\mu_\lambda - B}{A[k]} - \sum_{\substack{i=1 \\ i \neq k}} \frac{A[i]}{A[k]} \mu_i \tag{C.52}$$

$$\sigma_{X,U_k} = \frac{\sigma^2_\lambda + \sigma^2_{X|U}}{A[k]^2} + \sum_{\substack{i=1 \\ i \neq k}} \frac{A[i]^2}{A[k]^2} \sigma^2_i \tag{C.53}$$

Then the λ-message is a Gaussian distribution with mean $\mu_{X,Uk}$ and variance σ^2_{X,U_k}.

C.6.3 *Conditional discrete with discrete likelihood and no π-messages.*

The result is a discrete probability distribution. A table of probabilities is computed by direct summation, with the i'th element

$$\lambda_{X,U_k}[i] \propto \sum_{j_X=0}^{\#X-1} \lambda_X[j_X]\ q_X[j_X, i] \tag{C.54}$$

The constant of proportionality is such that $\sum_i \lambda_{X,U_k}[i] = 1$.

C.6.4 [*] *Autoregressive model with Gaussian likelihood and Gaussian π-messages.*

The present (September, 1999) implementation of RISO can handle a special case: Gaussian likelihood, and the π-messages for the correlation coefficient and the variance of additive noise are delta functions. In this special case, the λ-message is a Gaussian, with mean

$$\frac{\mu_\lambda}{\rho} \tag{C.55}$$

and variance

$$\frac{\sigma^2_\lambda + \sigma^2}{\rho^2} \tag{C.56}$$

where ρ and σ are the correlation and the noise variance, respectively, and μ_λ and σ_λ are the mean and variance of λ_X.

If the π-messages are not delta functions, so the problem doesn't fall into the special case, the general λ-message handler described in §C.6.10 is invoked.

C.6.5 *Indexed distribution, variable is evidence, and no π-messages.*

In this case, there is exactly one parent, say I, and I is discrete, so the λ-message is discrete. A table is constructed, with the i'th element equal to

$$p_{X|I}(x, i) \tag{C.57}$$

where x denotes the evidence value and i ranges from 0 to $\#I - 1$.

C.6.6 *Indexed distribution, variable is evidence, and discrete π-messages.*

This case is very prevalent, as the case of a measured sensor variable with one parent being the sensor status and the other parent being the "actual" variable falls under this heading.

This case is subdivided into two subcases, depending on the types of the components of the indexed distribution. Let us write the conditional distribution as $p_{X|U,I}$, with U the continuous parent and I the discrete parent.

(i) If all components of $p_{X|U,I}$ are conditional Gaussian, the λ-message is a mixture of Gaussians. Suppose the i'th component of $p_{X|U,I}$ has conditional mean $A_i u + B_i$, where $i = 0, \ldots, N-1$ is the value of the discrete parent I, and u is the value of the continuous parent U, and the conditional variance is σ_i^2. Then there are N components in the λ-message, and the i'th component has mean and variance

$$\frac{x - B_i}{A_i} \tag{C.58}$$

$$\frac{\sigma_i^2}{A_i^2} \tag{C.59}$$

and mixing proportion

$$\frac{\pi_{I,X}[i]}{A_i} \tag{C.60}$$

(ii) If some component of $p_{X|U,I}$ is not a conditional Gaussian, the λ-message is a general mixture. Let N be the number of components of $p_{X|U,I}$. Then the λ-message has N components, and the i'th component has one of three types.

- If the i'th component of $p_{X|U,I}$ is conditional Gaussian, the corresponding component of the λ-message is Gaussian, as described in Item (i) above.
- If the i'th component of $p_{X|U,I}$ does not depend on U, the corresponding component of the λ-message is a constant function, with value 1, and the mixing proportion for this component is equal to

$$\pi_{I,X}[i] \; p_{X|U,I}[i](x) \tag{C.61}$$

- Otherwise, the i'th component of $p_{X|U,I}$ is a general conditional distribution, and the corresponding component of the λ-message is a representation of the integral

$$\int q_X(x,u,i)\, du \tag{C.62}$$

with mixing proportion

$$\pi_{I,X}[i] \tag{C.63}$$

In either case (i) or (ii), before returning the mixture, the mixing proportions are normalized to sum to 1; this step is not strictly necessary.

APPROXIMATE RESULTS

C.6.7 *Conditional discrete with arbitrary likelihood and no π-messages.*

If X is discrete, the range over which the λ-message is defined is known: the range is just $0, 1, 2, \ldots, \#X - 1$. To simplify further calculations, the λ-message is evaluated for each value in its range, and the table of probability values so generated is stored, so that no further evaluations are needed. The λ-message is represented in further calculations (e.g., a π-message) by the probability table.

C.6.8 *Functional relation with arbitrary likelihood and π-messages*

In this case, the λ-message can usefully be represented as an integration over a predictive distribution:

$$\begin{aligned}\lambda_{X,U_k}(\tilde{u}_k) \;\propto\; & \int dx \int du_1 \cdots \int du_m \\ & \times \lambda_X(x)\; q_X(x, u_1, \ldots, u_{k-1}, \tilde{u}_k, u_{k+1}, \ldots, u_m)\; \delta_{u_k}(\tilde{u}_k) \prod_{\substack{j=1 \\ j\neq k}}^{m} \pi_{U_j,X}(u_j) \\ = \; & \int dx\; \lambda_X(x)\; \tilde{\pi}_X(x, \tilde{u}_k) \end{aligned} \tag{C.64}$$

denoting the made-up predictive distribution as $\tilde{\pi}_X$, with

$$\begin{aligned}&\tilde{\pi}_X(x, \tilde{u}_k) = \\ &\int du_1 \cdots \int du_m q_X(x, u_1, \ldots, u_{k-1}, \tilde{u}_k, u_{k+1}, \ldots, u_m)\; \delta_{u_k}(\tilde{u}_k) \prod_{\substack{j=1 \\ j\neq k}}^{m} \pi_{U_j,X}(u_j)\end{aligned} \tag{C.65}$$

Eq. C.65 is calculated by substituting a delta function over the parent to which the λ-message is being sent in place of the π-message from that parent; §C.3.20 gives rules for handling this calculation. Note that $\tilde{\pi}_X$ depends on the argument $\tilde{u}_k$ of the λ-message, so for every $\tilde{u}_k$ the predictive distribution is recomputed. The outer integration over X in Eq. C.64 is carried out by the algorithm QAGS from QUADPACK.

C.6.9 *Arbitrary conditional distribution, variable is evidence, arbitrary π-messages.*

In this case, λ_X is a delta function, positioned on the value x, so the integral which defines the λ-message reduces to

$$\begin{aligned}\lambda_{x,u_k}(\tilde{u}_k) \;\propto\; & \int du_1 \cdots \int du_{k-1} \int du_{k+1} \cdots \int du_m \\ & \times q_x(x, u_1, \ldots, u_{k-1}, \tilde{u}_k, u_{k+1}, \ldots, u_m) \prod_{\substack{j=1 \\ j\neq k}}^{m} \pi_{U_j,X}(u_j) \\ = \; & \int du_1 \cdots \int du_m \\ & \times q_X(x, u_1, \ldots, u_{k-1}, \tilde{u}_k, u_{k+1}, \ldots, u_m)\; \delta_{u_k}(\tilde{u}_k) \prod_{\substack{j=1 \\ j\neq k}}^{m} \pi_{U_j,X}(u_j)\end{aligned} \tag{C.66}$$

which is formally identical to π_X constructed from a list of π-messages which includes $\delta_{u_k}(\tilde{u}_k)$ in place of a genuine π-message from U_k to X. Thus this λ-message is computed by completing the list of π-messages with a delta function for U_k, and then using one of the formulas in §C.3. Note that the delta function changes with every new argument u_k for which the λ-message is to be evaluated, so a new π_X is calculated for each value of u_k.

C.6.10 *Arbitrary conditional distribution, arbitrary likelihood, arbitrary π-messages.*

A direct representation of §C.66 is constructed from the conditional distribution, likelihood, and π-messages. A numerical integration is carried out for each value of u_k for which λ_{X,U_k} is to be evaluated. In the present version (September, 1999) of the RISO software, the integration over the parents is carried out separately from the 1-dimensional integration over X. If there are two or more π-messages, the integration over $\{U_1, \ldots, U_m\} \backslash U_k$ is implemented as quasi Monte Carlo integration; if there is only one π-message, an ordinary Monte Carlo integration is calculated. (This is due to a characteristic of the algorithm, TOMS 659, employed to generate the low-discrepency sequence for QMC: it only works in two or more dimensions.) The integration over X is computed by a 21-point Gauss-Kronrod rule; it is not adaptive. It would probably be more efficient to carry out the integration over the child variable and the parents all at the same time.

Appendix D

NOTES ON MONOTONE CUBIC SPLINES

A set of knots $x_1 < x_2 < x_3 < \ldots < x_n$ is given. We wish to find an interpolating spline function which is nonnegative on the interval $[x_1, x_n]$. Assuming that the spline has a continuous first derivative, a monotone piecewise cubic spline is determined by the function value $f_i = f(x_i)$ and the value of the first derivative at $d_i = f'(x_i)$ at each knot x_i. Write the spline function and its derivative on the interval $[x_i, x_{i+1}]$ as

$$y_i(x) = \alpha_{i,0} + \alpha_{i,1} \cdot (x - x_i) + \alpha_{i,2} \cdot (x - x_i)^2 + \alpha_{i,3} \cdot (x - x_i)^3, \tag{D.1}$$

$$y_i'(x) = \alpha_{i,1} + 2\alpha_{i,2} \cdot (x - x_i) + 3\alpha_{i,3} \cdot (x - x_i)^2 \tag{D.2}$$

Matching function and derivative values at each end of the interval $[x_i, x_{i+1}]$,

$$y(x_i) = f_i, \qquad y'(x_i) = d_i, \tag{D.3}$$

$$y(x_{i+1}) = f_{i+1}, \qquad y'(x_{i+1}) = d_{i+1} \tag{D.4}$$

and solving for the coefficients $\alpha_{i,0}, \ldots, \alpha_{i,3}$, we find

$$\alpha_{i,0} = f_i, \tag{D.5}$$

$$\alpha_{i,1} = d_i, \tag{D.6}$$

$$\alpha_{i,2} = 3\frac{\Delta f_i}{(\Delta x_i)^2} - \frac{2d_i + d_{i+1}}{\Delta x_i}, \tag{D.7}$$

$$\alpha_{i,3} = -2\frac{\Delta f_i}{(\Delta x_i)^3} + \frac{d_i + d_{i+1}}{(\Delta x_i)^2} \tag{D.8}$$

where we have defined, for convenience,

$$\Delta x_i = x_{i+1} - x_i, \tag{D.9}$$

$$\Delta f_i = f_{i+1} - f_i \tag{D.10}$$

It remains to be seen how the derivatives at the knots are determined. Every choice of the d_i yields an interpolating cubic spline. For the purpose of representing probability

density functions, we need a spline which is everywhere nonnegative. According to Ref. [33], choosing

$$d_i = \begin{cases} \frac{S_{i-1}S_i}{\alpha S_i + (1-\alpha)S_{i-1}} & \text{if } S_{i-1}S_i > 0, \\ 0 & \text{otherwise} \end{cases} \tag{D.11}$$

with

$$S_i = \frac{\Delta f_i}{\Delta x_i} \tag{D.12}$$

$$\alpha = \frac{1}{3}\left(1 + \frac{\Delta x_i}{\Delta x_{i-1} + \Delta x_i}\right) \tag{D.13}$$

will yield a spline which stays within the limits $[\min_i f_i, \max_i f_i]$. Thus if all the f_i are nonnegative (as they are when they come from a probability density), the spline function will also be nonnegative. There are other choices of the d_i which have this property [33].

To complete the specification, it is assumed that $d_1 = d_n = 0$. It is assumed that the approximation is zero outside of the interval $[x_1, x_n]$; typically x_1 and x_n are assigned as the endpoints of the effective support of a probability density.

Appendix E

NOTES ON CONDITIONAL GAUSSIAN DISTRIBUTIONS

E.1 Definition of conditional Gaussian models

The relation between a Gaussian joint distribution and the distribution of some of the variables conditional on others is described by the following result [81, § VIII-9.3]. Let the vector X have a Gaussian distribution with mean μ and covariance Σ. Let us partition $X \in \mathbf{R}^{m+n}$ into $X^{(1)} \in \mathbf{R}^m$ and $X^{(2)} \in \mathbf{R}^n$, so $X = (X^{(1)}, X^{(2)})$, likewise partition μ as $(\mu^{(1)}, \mu^{(2)})$ and Σ as

$$\Sigma = \begin{pmatrix} \Sigma^{(11)} & \Sigma^{(12)} \\ \Sigma^{(21)} & \Sigma^{(22)} \end{pmatrix}$$

Then the distribution of $X^{(1)}$ given $X^{(2)}$ is again Gaussian, with mean

$$\mu^{(1|2)} = \mu^{(1)} + \Sigma^{(12)} \left(\Sigma^{(22)}\right)^{-1} \left(X^{(2)} - \mu^{(2)}\right) \tag{E.1}$$

and covariance

$$\Sigma^{(1|2)} = \Sigma^{(11)} - \Sigma^{(12)} \left(\Sigma^{(22)}\right)^{-1} \Sigma^{(21)} \tag{E.2}$$

Note that the mean of the conditional distribution is a linear combination of $X^{(2)}$ plus a constant term, while the covariance does not depend on $X^{(2)}$.

Another useful result (loc. cit.) concerns the marginal distribution of some variables in a joint Gaussian distribution. The marginal distribution of, say, $X^{(2)}$ is again Gaussian, with mean $\mu^{(2)}$ and covariance $\Sigma^{(22)}$.

Consider now the conditional distribution of $X^{(1)}$ given $X^{(2)}$ when their joint distribution is a mixture of joint Gaussian distributions. From the above results, we can show that this conditional distribution is a mixture of conditional Gaussians, with mixing parameters that depend on the context $X^{(2)}$. For convenience, write the Gaussian

density function as

$$g(x;\mu,\Sigma) = (2\pi)^{-k/2}\,(\det\Sigma)^{-1/2}\,\exp\left(-\frac{1}{2}(x-\mu)'\Sigma^{-1}(x-\mu)\right)$$

where k is the number of dimensions of x, not necessarily equal to $m+n$. Denote the joint mixture with N components as

$$p_X(x) = \sum_{j=1}^{N} p_{j,X}(x) = \sum_{j=1}^{N} \alpha_j\; g(x;\mu_j,\Sigma_j) \tag{E.3}$$

Since the integrations for marginalizing over $X^{(2)}$ distribute over the summation in Eq. E.3, the marginal distribution for $X^{(2)}$ is also a Gaussian mixture, with mixing coefficients equal to the original mixing coefficients:

$$p_{X^{(2)}}(x^{(2)}) = \sum_{j=1}^{N} \alpha_j\; p_{j,X^{(2)}}(x^{(2)}) = \sum_{j=1}^{N} \alpha_j\; g(x^{(2)};\mu_j^{(2)},\Sigma_j^{(22)}) \tag{E.4}$$

The conditional density of $X^{(1)}$ given $X^{(2)}$ is, as usual, the ratio of the joint density to the marginal density of $X^{(2)}$.

$$\begin{aligned}
p_{X^{(1)}|X^{(2)}}(x^{(1)},x^{(2)}) &= \frac{p_X(x^{(1)},x^{(2)})}{p_{X^{(2)}}(x^{(2)})} \\
&= \sum_{j=1}^{N} \frac{\alpha_j\; p_{j,X}(x)}{p_{X^{(2)}}(x^{(2)})} \\
&= \sum_{j=1}^{N} \frac{\alpha_j\; p_{j,X^{(2)}}(x^{(2)})}{p_{X^{(2)}}(x^{(2)})} \frac{p_{j,X}(x)}{p_{j,X^{(2)}}(x^{(2)})} \\
&= \sum_{j=1}^{N} \frac{\alpha_j\; p_{j,X^{(2)}}(x^{(2)})}{p_{X^{(2)}}(x^{(2)})}\, p_{j,X^{(1)}|X^{(2)}}(x^{(1)},x^{(2)}) \qquad \text{(E.5)} \\
&= \sum_{j=1}^{N} \frac{\alpha_j\; g(x^{(2)};\mu_j^{(2)},\Sigma_j^{(22)})}{p_{X^{(2)}}(x^{(2)})}\, g(x^{(1)};\mu_j^{(1|2)},\Sigma_j^{(1|2)}) \qquad \text{(E.6)}
\end{aligned}$$

with the conditional mean $\mu^{(1|2)}$ and conditional covariance $\Sigma^{(1|2)}$ defined in Eqs. E.1 and E.2, respectively. Eq. E.6 shows that the conditional distribution is a mixture of conditional Gaussians, and each component of the conditional mixture is just the conditional of the corresponding component in the joint mixture. The j'th mixing

parameter is equal to

$$\frac{\alpha_j \; p_{j,X^{(2)}}(x^{(2)})}{p_{X^{(2)}}(x^{(2)})} = \frac{\alpha_j \; g(x^{(2)}; \mu_j^{(2)}, \Sigma_j^{(22)})}{\sum_{i=1}^{N} \alpha_i \; g(x^{(2)}; \mu_i^{(2)}, \Sigma_i^{(22)})} \tag{E.7}$$

This quantity, which also appears in the expectation-maximization algorithm for fitting mixture parameters, is sometimes called the "responsibility" of the j'th mixture component (of the marginal distribution) for the datum $x^{(2)}$.

In summary, Eqs. E.4, E.5, and E.7 show that a conditional distribution of a joint mixture distribution is again a mixture, with each component equal to the conditional of the corresponding component of the joint distribution, and each mixing coefficient, a function of the variables $X^{(2)}$ on which we conditioning, equal to the responsibility of the corresponding component of the marginal distribution of $X^{(2)}$. These results hold for all types of mixture distributions, not just for Gaussian mixtures. In the latter case, the conditional mixture is a mixture of conditional Gaussians, and the marginal of any variables is also a mixture of Gaussians.

E.2 Computing π- and λ-messages for conditional Gaussians

If all π- and λ-messages are Gaussian densities and the relation of a variable to its parents is conditional Gaussian (linear in the mean), then exact results are easily obtained for posterior distributions, which are again Gaussian densities [63]. One might hope that exact results could be obtained if all the π- and λ-messages are Gaussian mixtures and relations are described by mixtures of conditional Gaussian densities, but unfortunately it appears that exact results cannot be computed for the predictive support, π_X, and the likelihood message, $\lambda_{X,U}$. The derivation of these functions requires an integration of the product of the conditional density with π- or λ-messages. Due to the presence of the variable mixing coefficient (Eq. E.7) in the conditional density, the integration is difficult; the variable mixing coefficient is not a Gaussian density function but a ratio of such functions.[1] I made an attempt to use Mathematica to compute π_X for a simple

[1] The exact results presented in Refs. [29, 30] stem from the assumption that the conditional distribution does not contain a ratio of Gaussian density functions. As the ratio serves to normalize the conditional density so that integrating over it w.r.t. X yields 1, it seems indefensible to omit the ratio.

example, but even Mathematica was unable to carry out the integration exactly.

Thus it appears that the computation of predictive support and likelihood messages will require numerical integrations. Although numerical integrations become time-consuming when the number of dimensions is greater than about 3, the integrands are well-behaved, at least. It can readily be demonstrated that the π_X and $\lambda_{X,U}$ functions can be expressed as repeated summations over an integral containing a single product of a conditional Gaussian density with some other Gaussian densities. These integrals can be evaluated by the same algorithms used for numerical integration elsewhere in RISO. The output is a Gaussian mixture which approximates (in the minimum cross-entropy sense) the π_X or $\lambda_{X,U}$ function.

If all π- and λ-messages are Gaussian mixtures, then the posterior, likelihood support, and predictive message functions can all be computed exactly, and the results are again Gaussian mixtures. It is easy to show at the parameters of the results are simple functions of the parameters of the π- and λ-messages. Thus only π_X or $\lambda_{X,U}$ functions need to be approximated in a belief network composed of mixtures of conditional Gaussians; the $\pi_{X,Y}$, λ_X, and posterior can be computed exactly.

Appendix F

MISCELLANEOUS FORMULAS FOR MUTUAL INFORMATION

F.1 An identity relating MI and average conditional MI

There is an identity, relating the mutual information of different groups of variables, which is useful in calculations. Suppose we have two variables X_1 and X_2 and we wish to compute the mutual information $MI((X_1, X_2), Y)$ of these two, considered components of an ordered set (X_1, X_2), with another variable Y. Let's agree to the usual abuse of notation, writing $p(X, Y)$, $p(X|Y)$, $p(Y)$, etc., instead of the more precise $p_{XY}(u, v)$, $p_{X|Y}(u, v)$, $p_Y(u)$, etc. From the definition of MI we have

$$\begin{aligned} MI((X_1, X_2), Y) &= \int dX_1 \int dX_2 \int dY\, p(X_1, X_2, Y) \log \frac{p(X_1, X_2|Y)}{p(X_1, X_2)} \\ &= \int dX_1 \int dX_2 \int dY\, p(X_2, Y|X_1)\, p(X_1) \log \frac{p(X_2|Y, X_1)}{p(X_2|X_1)} \\ &\quad + \int dX_1 \int dX_2 \int dY\, p(X_2, Y|X_1)\, p(X_1) \log \frac{p(X_1|Y)}{p(X_1)} \\ &= \int dX_1 \int dX_2 \int dY\, p(X_2, Y|X_1)\, p(X_1) \log \frac{p(X_2|Y, X_1)}{p(X_2|X_1)} \\ &\quad + \int dX_1 \int dY\, p(Y|X_1)\, p(X_1) \log \frac{p(X_1|Y)}{p(X_1)} \\ &= \overline{MI}(X_2, Y|X_1) + MI(X_1, Y) \end{aligned} \tag{F.1}$$

where $\overline{MI}$ is an average conditional mutual information — note that

$$\begin{aligned} \overline{MI}(X_2, Y|X_1) &= \int dX_1 \left(\int dX_2 \int dY p(X_2|Y, X_1)\, p(Y|X_1) \log \frac{p(X_2|Y, X_1)}{p(X_2|X_1)} \right) p(X_1) \\ &= \int dX_1\, MI(X_2, Y|X_1)\, p(X_1) \end{aligned} \tag{F.2}$$

If there are more than two X's involved, the above decomposition can be repeated; the result is

$$MI((X_1, \ldots, X_m), Y) = \sum_{k=1}^{m-1} \overline{MI}(X_{k+1}, Y|X_1, \ldots, X_k) + MI(X_1, Y) \tag{F.3}$$

Thus a mutual information of multiple variables can be computed as the sum of one or more average conditional mutual informations and one unconditional mutual information. As it happens, it is convenient to write an algorithm to compute the $\overline{MI}$ terms and add them up; this approach was used to compute the results described in §8.1.1.

F.2 Kullback-Leibler divergence between two Gaussian densities

The Kullback-Leibler divergence between two Gaussian densities $p_1(x) = g(x; \mu_1, \sigma_1)$ and $p_2(x) = g(x; \mu_2, \sigma_2)$ is just

$$\begin{aligned} KL(p_1, p_2) &= \int dx\, p_1(x)\, \log p_1(x) - \int dx\, p_1(x) \left(-\log \sigma_2 \sqrt{2\pi} - \frac{1}{2} \frac{(x-\mu_2)^2}{\sigma_2^2} \right) \\ &= \frac{1}{2} \left(\frac{\sigma_1^2}{\sigma_2^2} - 1 \right) + \frac{1}{2} \frac{(\mu_1 - \mu_2)^2}{\sigma_2^2} - \log \frac{\sigma_1}{\sigma_2} \end{aligned} \tag{F.4}$$

Since the exponentials in the densities p_1 and p_2 are to the base e, it is important that the logarithm in Eq. F.4 be the natural logarithm, and the result is in nats. One can, of course, divide the entire right-hand side by $\log 2$ to obtain a result in bits, but it is incorrect to simply substitute a logarithm to the base 2 for the natural logarithm in Eq. F.4.

Appendix G

A REMARK ON INVARIANT MEASURES OF DISCRETE-TIME SYSTEMS

There is an important equation in the study of nonlinear dynamical systems which can be derived from the propagation equations described in §5.3. A dynamical system in discrete time can be represented as a belief network as shown in Figure G.1. If the state variable y has a known value at time t, then the value at time $t+1$ is a deterministic function F,

$$y_{t+1} = F(y_t) \tag{G.1}$$

Thus for the conditional distribution of y_{t+1} given y_t we may write

$$p_{y_{t+1}|y_t}(y_{t+1}, y_t) = \delta(y_{t+1} - F(y_t)) \tag{G.2}$$

In itself, this conditional distribution is not particularly interesting. However, suppose that we do not know y_t with perfect accuracy; all we have is a distribution p_{y_t} which describes what we do know about y_t. Then we compute a distribution over y_{t+1} according to Eq. 5.2,

$$p_{y_{t+1}}(u) = \int \delta(u - F(v))\; p_{y_t}(v)\; dv \tag{G.3}$$

Figure G.1: A representation of a dynamical system as a directed graph. The transition from time t to $t+1$ is governed by a deterministic function F; the distribution of the state variable at $t+1$ is given by $\delta(y_{t+1} - F(y_t))$.

Any uncertainty in y_t is propagated downstream to $y_{t+2}, y_{t+3}, \ldots$ by repeating the same transformation,

$$\begin{aligned} p_{y_{t+2}}(u) &= \int \delta(u - F(v))\, p_{y_{t+1}}(v)\, dv \\ p_{y_{t+3}}(u) &= \int \delta(u - F(v))\, p_{y_{t+2}}(v)\, dv \\ p_{y_{t+4}}(u) &= \int \delta(u - F(v))\, p_{y_{t+3}}(v)\, dv \\ &\vdots \end{aligned}$$

For some kinds of dynamical systems, for example diffusion systems, the state becomes more and more widely dispersed with each time step. However, there are systems for which the state y reaches a limiting distribution, called the "invariant measure" for the system. The invariant measure p^* must satisfy

$$p^*(u) = \int \delta(u - F(v))\, p^*(v)\, dv \tag{G.4}$$

Eq. G.4 is called the Frobenius-Perron equation, and it plays a central role in the study of the ergodic properties of dynamical systems. From the general equations for the computation of π_y, we can see that solutions of the Frobenius-Perron equation can be characterized as π-messages which are invariant under the evolution F of the system. There is lots of fun to be had in the computation of solutions of Eq. G.4; see, for example, Ref. [61].

If a system has an indecomposable invariant measure, then the system is ergodic and time-averages coincide with space-averages. This is commonly interpreted as meaning that the system can be treated as random even if we know it is completely deterministic; as a typical case, whether or not there is such a thing as the "average climate of the earth" would follow from a demonstration that global weather is governed by equations which are "chaotic," in the generally accepted definition [61].

However, note that solutions of the Frobenius-Perron equation are descriptions of long-term behavior alone. In the short term, there may be more specific distributions (i.e., distributions with less entropy) which describe the state of the system, which tend

to decay (in the absence of new information) toward the invariant measure; see, for example, Figures 2 and 3 in Ref. [32]. For systems governed by differential equations, the evolution of a distribution over states is related to the local Lyapunov exponents of the system [44]. Short-term distributions can always be computed, at least in principle, by Eq. G.3, but numerical approximations will usually be necessary.