Chapter 4

Modeling and Reasoning with Bayesian Networks

Our goal in this chapter is to use Bayesian networks for addressing a number of reasoning problems that arise in a variety of applications. In the process of considering these problems, we will develop a repertoire of modeling techniques which have a much wider range of application than is considered in this chapter.

4.1 Introduction

The application areas we will consider in this chapter include diagnosis, bioinformatics, channel coding, and commonsense reasoning. For each one of these applications, we will state a specific reasoning problem which can be addressed by posing a formal query with respect to a Bayesian network. We will discuss the process of constructing the required network, and then identify the specific queries that need to be applied.

There are at least four general types of queries which can be posed with respect to a Bayesian network. Which specific query to use and in what situation is not always trivial, and some of the queries are guaranteed to be equivalent under certain conditions. We define these query types formally in Section 4.2 and then discuss them and their relationships in more detail when we go over the various applications in Section 4.3.

The construction of a Bayesian network involves three major steps. First, we must decide on the set of relevant variables and their possible values. Next, we must build the network structure by connecting the variables into a DAG. Finally, we must define the CPT for each network variable. The last step is the quantitative part of this construction process, and can be the most involved in certain situations. Two of the key issues that come up here is the potentially large size of CPTs, and the significance of the specific numbers used to populate them. We present techniques for dealing with the first issue in Section 4.4, and for dealing with the second issue in Section 4.5.

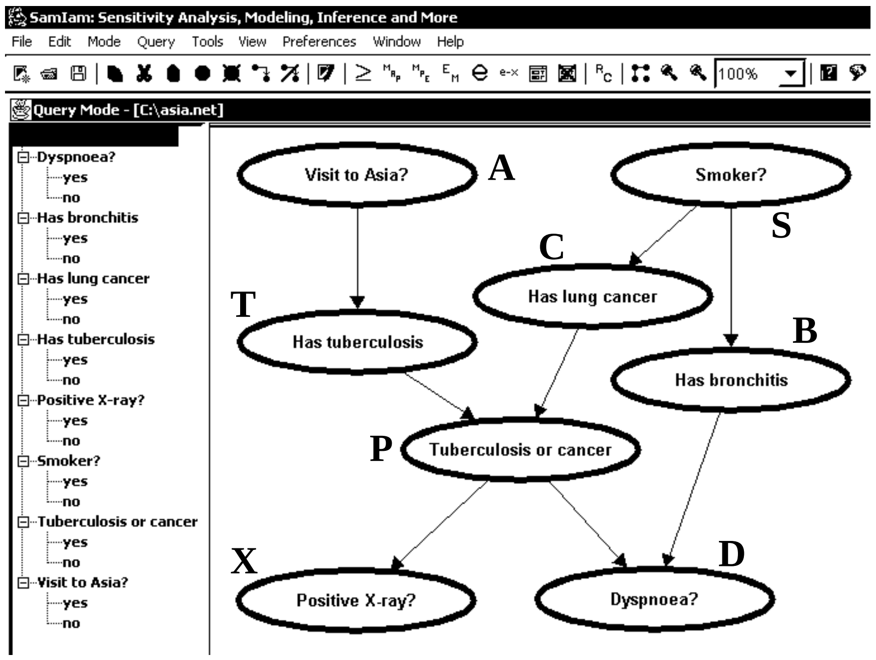


Figure 4.1: A screen shot of the Asia network from SamIam.

4.2 Reasoning with Bayesian Networks

Consider Figure 4.1, which depicts a screen shot of the SamIam system for reasoning with Bayesian networks.[1] The figure shows a Bayesian network, known as Asia, for modeling beliefs about certain diseases and their relationship to whether the patient is a smoker and whether they visited Asia. We will be using this network and the SamIam system to illustrate the type of queries that one can pose to a Bayesian network.

4.2.1 Probability of Evidence

One of the simplest queries that one can pose to a Bayesian network is to ask for the probability of some variable instantiation e, $\mathsf{Pr}(\mathsf{e})$, where Pr is the probability distribution induced by the network. For example, in the Asia network, we may be interested in knowing the probability that the patient has a positive X–ray, but no dyspnoea, $\mathsf{Pr}(X{=}\mathsf{yes}, D{=}\mathsf{no})$. This can be done in the SamIam system by selecting the corresponding values of these variables in the left panel of Figure 4.1, leading SamIam to return a probability of about 3.96%. The variables $\mathbf{E} = \{X, D\}$ are called *evidence variables* in this case, and the query

[1]SamIam is available from *http://reasoning.cs.ucla.edu/samiam/downloads.php*.

$\mathsf{Pr}(\mathbf{e})$ is known as a *probability–of–evidence* query, although it refers to a very specific type of evidence corresponding to the instantiation of some variables.

There are other types of evidence, beyond variable instantiations. In fact, any propositional sentence constructed from evidence variables does constitute a piece of evidence which we may want to compute the probability of. For example, we may want to know the probability that the patient has either a positive X–ray or dyspnoea, X=yes $\vee$ D=yes. Bayesian network tools, such as SamIam, do not usually provide direct support for computing the probability of arbitrary pieces of evidence, but such probabilities can be computed indirectly using one of the following techniques:

- *The Case–Analysis Method:* We know by case analysis that:

  $$\mathsf{Pr}(X{=}\mathsf{yes} \vee D{=}\mathsf{yes}) = \mathsf{Pr}(X{=}\mathsf{yes}, D{=}\mathsf{yes}) + \mathsf{Pr}(X{=}\mathsf{yes}, D{=}\mathsf{no}) + \mathsf{Pr}(X{=}\mathsf{no}, D{=}\mathsf{yes}),$$

  where each term on the righthand side is a probability of some instantiation. Hence, we can reduce the probability of an arbitrary piece of evidence α about variables $\mathbf{E}$ into the probabilities of some instantiations $\mathbf{e}$. This can always be done, but is only practical when the number of evidence variables $\mathbf{E}$ is relatively small, otherwise, the number of instantiations $\mathbf{e}$ may be quite large.

- *The Auxiliary–Node Method:* We can add an auxiliary node E to the network, declare nodes X and D as the parents of E, and then adopt the following CPT for E:[2]

x	d	e	$\mathsf{Pr}(e\|x,d)$
yes	yes	yes	1
yes	no	yes	1
no	yes	yes	1
no	no	yes	0

  Given this CPT, the event E=yes is then equivalent to X=yes $\vee$ D=yes and, hence, we can compute the probability of the latter by computing the probability of the former.

The Auxiliary–Node method can also suffer from an exponential blow up, since the size of used CPT is exponential in the number of evidence variables. This type of CPT, however, is quite special as it only contains probabilities equal to 0 or 1. When a CPT satisfies this property, we say that it is *deterministic.* We also refer to the corresponding node as a *deterministic node.* We shall present in Section 4.4 some techniques for representing deterministic CPTs which do not suffer from this exponential blowup, therefore, establishing the Auxiliary–Node method as a more practical alternative to the one based on case analysis.

We note here that in most of the literature on Bayesian network inference, the term "evidence" is almost always used to mean an instantiation of some variables. Since any arbitrary piece of evidence can be modelled using an instantiation, we will also stick to this usage unless stated otherwise.

[2]We have omitted redundant rows from the given CPT.

4.2.2 Prior and Posterior Marginals

If probability–of–evidence queries are one of the simplest, then *posterior–marginal queries* are one of the most common. We will first explain what is meant by the terms "posterior" and "marginal" and then explain this common class of queries.

Marginals

Given a joint probability distribution Pr over variables $X_1, \ldots, X_n$, the *marginal distribution* of Pr over a subset of these variables, say $X_1, \ldots, X_m$, $m \leq n$, is a distribution over the instantiations of variables $X_1, \ldots, X_m$:

$$\mathsf{Pr}(x_1, \ldots, x_m) = \sum_{x_{m+1}, \ldots, x_n} \mathsf{Pr}(x_1, \ldots, x_n).$$

That is, the marginal distribution can be viewed as a *projection* of the joint distribution on the potentially smaller set of variables $X_1, \ldots, X_m$. In fact, most often, the set of variables $X_1, \ldots, X_m$ is small enough to allow an explicit representation of the marginal distribution in tabular form (which is usually not feasible for the joint distribution).

Hence, to compute the marginal distribution over variables $\mathbf{X}$ in a Bayesian network, is to compute the marginal distribution of Pr over variables $\mathbf{X}$, where Pr is the joint probability distribution induced by the network. Moreover, when the marginal distribution is computed given some evidence $\mathbf{e}$,

$$\mathsf{Pr}(x_1, \ldots, x_m | \mathbf{e}) = \sum_{x_{m+1}, \ldots, x_n} \mathsf{Pr}(x_1, \ldots, x_n | \mathbf{e}),$$

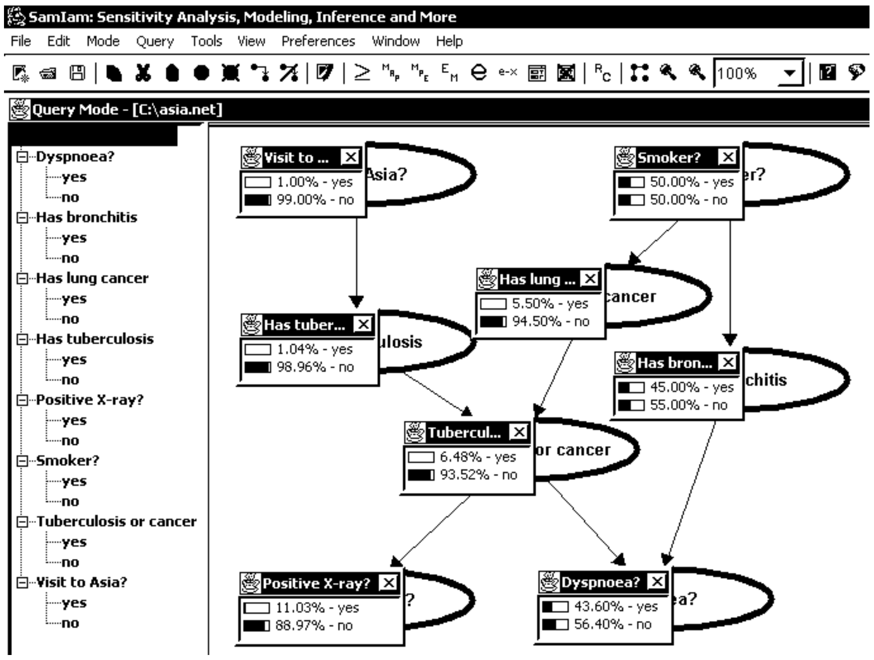
it is known as a *posterior marginal.* This is to be contrasted with the marginal distribution given no evidence, which is known as a *prior marginal.*

Figure 4.2 depicts a screen shot of SamIam, where the prior marginals are shown for every variable in the network. Figure 4.3 depicts another screen shot of SamIam, where posterior marginals are shown for every variable given that the patient has a positive X–ray but no dyspnoea, $\mathbf{e}$: X=yes, D=no. The small windows containing marginals in Figures 4.2 and 4.3 are known as *monitors,* and are quite common in tools for reasoning with Bayesian networks. According to these monitors, we have the following prior and posterior marginals for lung cancer, C, respectively:

C	$\mathsf{Pr}(c)$
yes	5.50%
no	94.50%

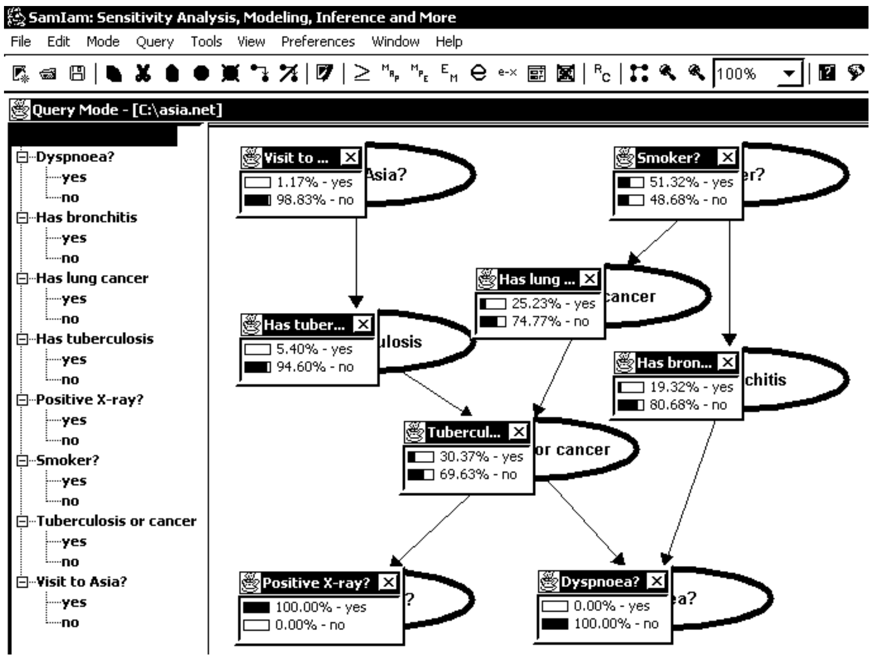
C	$\mathsf{Pr}(c\|\mathbf{e})$
yes	25.23%
no	74.77%

Note that all marginals in Figures 4.2 and 4.3 are over single variables. The graphical user interfaces (GUIs) of most available tools support only marginals over single variables, even though the algorithms underlying these tools are capable of computing other types of marginals. For example, many of these


SamIam: Sensitivity Analysis, Modeling, Inference and More
File Edit Mode Query Tools View Preferences Window Help
100%
Query Mode - [C:\asia.net]
Dyspnoea? yes no
Has bronchitis yes no
Has lung cancer yes no
Has tuberculosis yes no
Positive X-ray? yes no
Smoker? yes no
Tuberculosis or cancer yes no
Visit to Asia? yes no

Visit to ... 1.00% - yes 99.00% - no
Smoker? 50.00% - yes 50.00% - no
Has lung ... 5.50% - yes 94.50% - no
Has tuber... 1.04% - yes 98.96% - no
Has bron... 45.00% - yes 55.00% - no
Tubercul... 6.48% - yes 93.52% - no
Positive X-ray? 11.03% - yes 88.97% - no
Dyspnoea? 43.60% - yes 56.40% - no

Figure 4.2: Prior marginals in the Asia network.


SamIam: Sensitivity Analysis, Modeling, Inference and More
File Edit Mode Query Tools View Preferences Window Help
100%
Query Mode - [C:\asia.net]
Dyspnoea? yes no
Has bronchitis yes no
Has lung cancer yes no
Has tuberculosis yes no
Positive X-ray? yes no
Smoker? yes no
Tuberculosis or cancer yes no
Visit to Asia? yes no

Visit to ... 1.17% - yes 98.83% - no
Smoker? 51.32% - yes 48.68% - no
Has lung ... 25.23% - yes 74.77% - no
Has tuber... 5.40% - yes 94.60% - no
Has bron... 19.32% - yes 80.68% - no
Tubercul... 30.37% - yes 69.63% - no
Positive X-ray? 100.00% - yes 0.00% - no
Dyspnoea? 0.00% - yes 100.00% - no

Figure 4.3: Posterior marginals in the Asia network given a positive X–ray and no dyspnoea.

algorithms—which we discuss in later chapters—will readily compute marginals over families, where the *family* of a variable consists of the variable and its parents in the network. Family marginals turn out to be very important for parameter estimation algorithms, yet they are rarely used by end–users, which is why they are not usually supported by the GUIs of common tools for Bayesian network reasoning.

Virtual Evidence

Most GUIs do not provide direct support for accommodating soft evidence either, with the expectation that users will utilize the method of virtual evidence for this purpose. Suppose for example that we receive soft evidence which *doubles* the odds of a positive X–ray or dyspnoea, X=yes $\vee$ D=yes. We showed in the previous section that this disjunction can be represented explicitly in the network using the auxiliary variable E. We can also represent the soft evidence explicitly by adding another auxiliary variable V, as shown in Figure 4.4, where the strength of this soft evidence is captured by the CPT of variable V. All we have to do is ensure that

$$\frac{\mathsf{Pr}(V{=}\mathsf{yes}|E{=}\mathsf{yes})}{\mathsf{Pr}(V{=}\mathsf{yes}|E{=}\mathsf{no})} = 2,$$

where the number 2 is the Bayes factor as discussed in Chapter 2. One choice for the CPT of variable V is then:[3]

e	v	$\mathsf{Pr}(v\|e)$
yes	yes	.8
no	yes	.4

We can then accommodate the soft evidence by setting the value of auxiliary variable V to yes as shown in Figure 4.5. Note the prior and posterior marginals over variable E, which are shown in Figure 4.4 and 4.5, respectively:

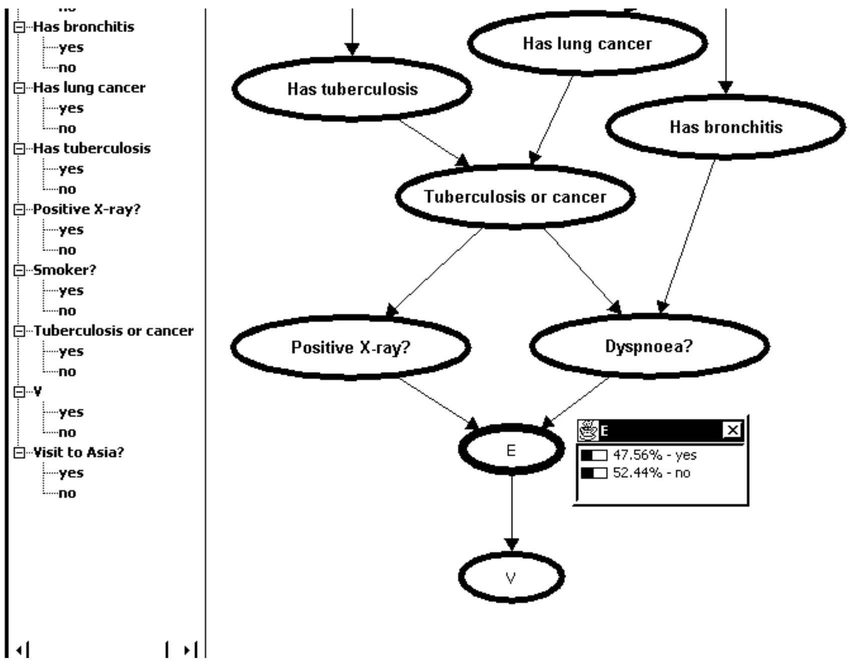
E	$\mathsf{Pr}(e)$
yes	47.56%
no	52.44%

E	$\mathsf{Pr}(e\|V{=}\mathsf{yes})$
yes	64.46%
no	35.54%

The ratio of odds is then:

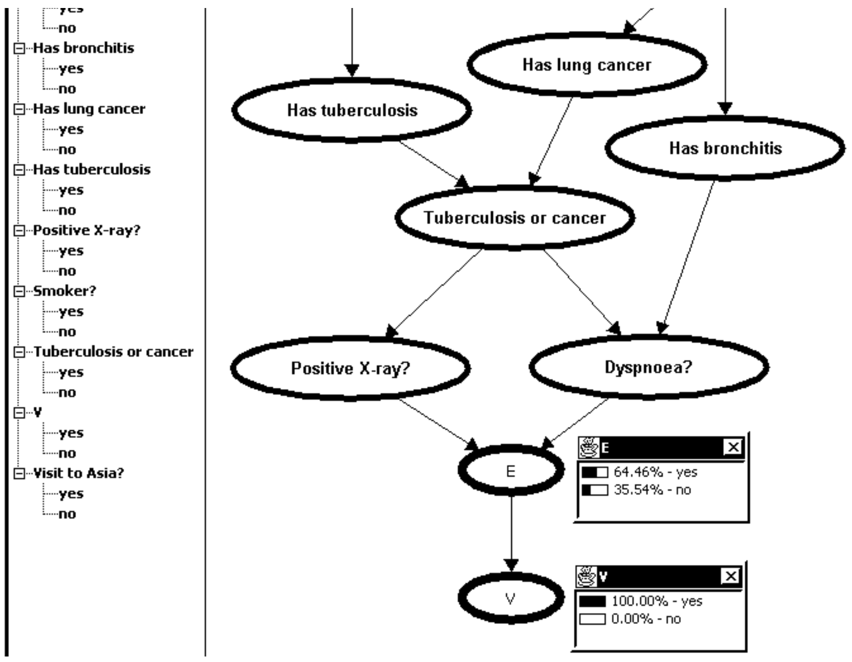
$$\frac{\mathsf{O}(E{=}\mathsf{yes}|V{=}\mathsf{yes})}{\mathsf{O}(E{=}\mathsf{yes})} = \frac{64.46/35.54}{47.56/52.44} \approx 2.$$

Hence, the hard evidence V=yes leads to doubling the odds of E=yes as expected. This ratio is not exactly 2 because by default, SamIam displays marginals only with a two–decimal precision.[4]



E
47.56% - yes
52.44% - no

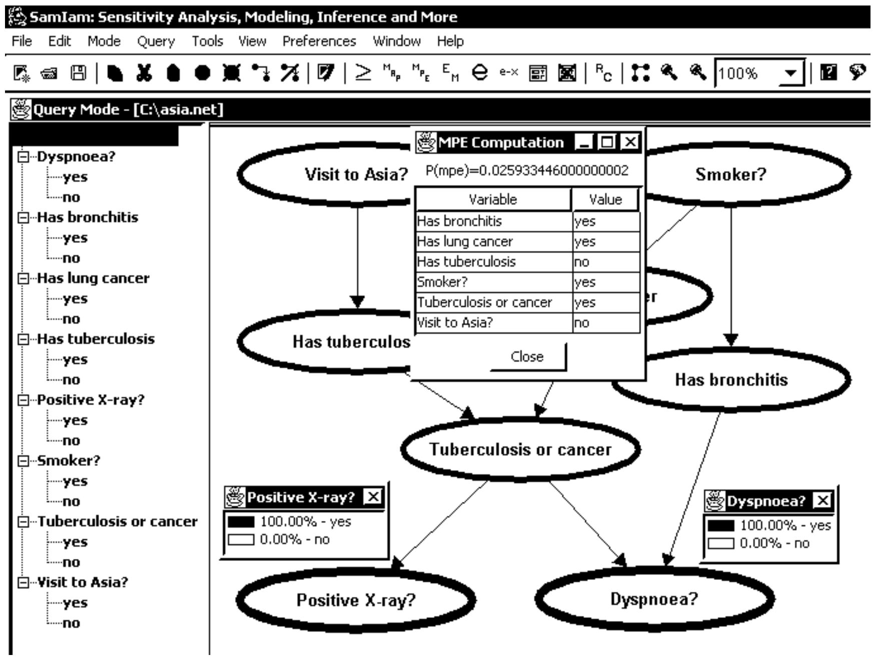
Figure 4.4: Representing soft evidence on variable E using auxiliary variable V.



E
64.46% - yes
35.54% - no

V
100.00% - yes
0.00% - no

Figure 4.5: Asserting soft evidence on variable E by setting the value of auxiliary variable V.



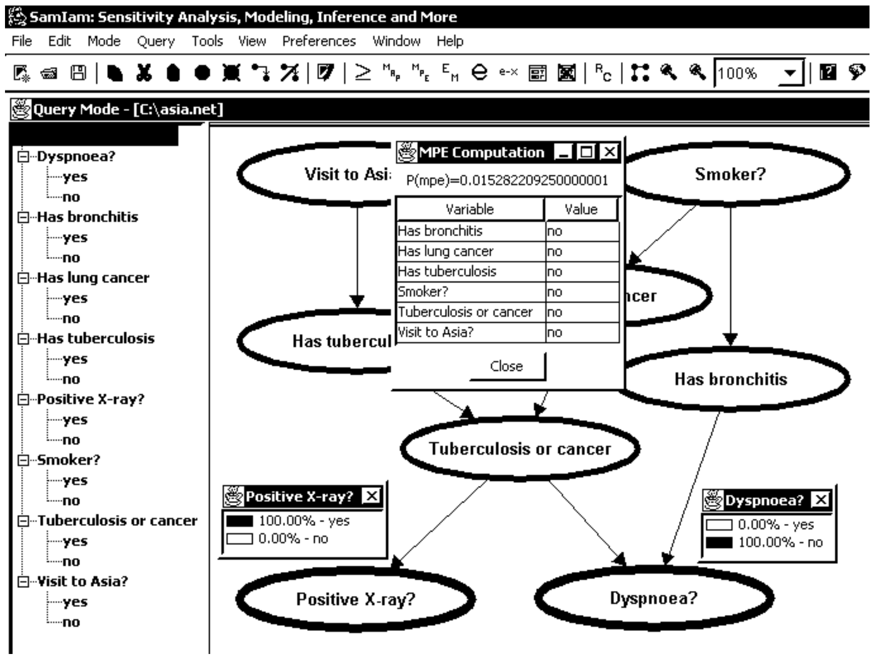
Positive X-ray?
■ 100.00% - yes
□ 0.00% - no

Dyspnoea?
■ 100.00% - yes
□ 0.00% - no

P(mpe)=0.025933446000000002

Variable	Value
Has bronchitis	yes
Has lung cancer	yes
Has tuberculosis	no
Smoker?	yes
Tuberculosis or cancer	yes
Visit to Asia?	no

Figure 4.6: Computing the most probable explanation (MPE) given a positive X–ray and dyspnoea.



Positive X-ray?
■ 100.00% - yes
□ 0.00% - no

Dyspnoea?
□ 0.00% - yes
■ 100.00% - no

P(mpe)=0.015282209250000001

Variable	Value
Has bronchitis	no
Has lung cancer	no
Has tuberculosis	no
Smoker?	no
Tuberculosis or cancer	no
Visit to Asia?	no

Figure 4.7: Computing the most probable explanation (MPE) given a positive X–ray and no dyspnoea.

4.2.3 Most Probable Explanation: MPE

We now turn to another important class of queries with respect to Bayesian networks, for computing the most probable explanation. The goal here is not to compute any particular probabilities, but instead to identify the most probable instantiation of network variables given some evidence. Specifically, if $X_1, \ldots, X_n$ are all network variables, and if $\mathbf{e}$ is the given evidence, the goal then is to identify an instantiation $x_1, \ldots, x_n$ for which the probability $\mathsf{Pr}(x_1, \ldots, x_n|\mathbf{e})$ is maximal. Such an instantiation $x_1, \ldots, x_n$ will be called a *most probable explanation (MPE)* given evidence $\mathbf{e}$.

Consider Figure 4.6 which depicts a screen shot of SamIam after having computed the MPE given a patient with positive X–ray and dyspnoea. According to the result of this query, the most likely scenario is one in which the patient made no visit to Asia; is a smoker; has lung cancer and bronchitis; but no tuberculosis.

It is important to note here that a most probable explanation cannot be obtained directly from posterior marginals. That is, if $x_1, \ldots, x_n$ is an instantiation which is obtained by choosing each value x_i so as to maximizes the probability $\mathsf{Pr}(x_i|\mathbf{e})$, then $x_1, \ldots, x_n$ is not necessarily a most probable explanation. Consider the posterior marginals in Figure 4.3 for an example. If we choose for each variable the value with maximal probability, we get an explanation in which the patient is a smoker:

$$\alpha:\ A{=}\mathsf{no},\ S{=}\mathsf{yes},\ T{=}\mathsf{no},\ C{=}\mathsf{no},\ B{=}\mathsf{no},\ P{=}\mathsf{no},\ X{=}\mathsf{yes},\ D{=}\mathsf{no},$$

which has a probability of $\approx 20.03\%$ given the evidence $\mathbf{e}$: $X{=}\mathsf{yes}$, $D{=}\mathsf{no}$. The most probable explanation given by Figure 4.7, however, is one in which the patient is not a smoker:

$$\alpha:\ A{=}\mathsf{no},\ S{=}\mathsf{no},\ T{=}\mathsf{no},\ C{=}\mathsf{no},\ B{=}\mathsf{no},\ P{=}\mathsf{no},\ X{=}\mathsf{yes},\ D{=}\mathsf{no},$$

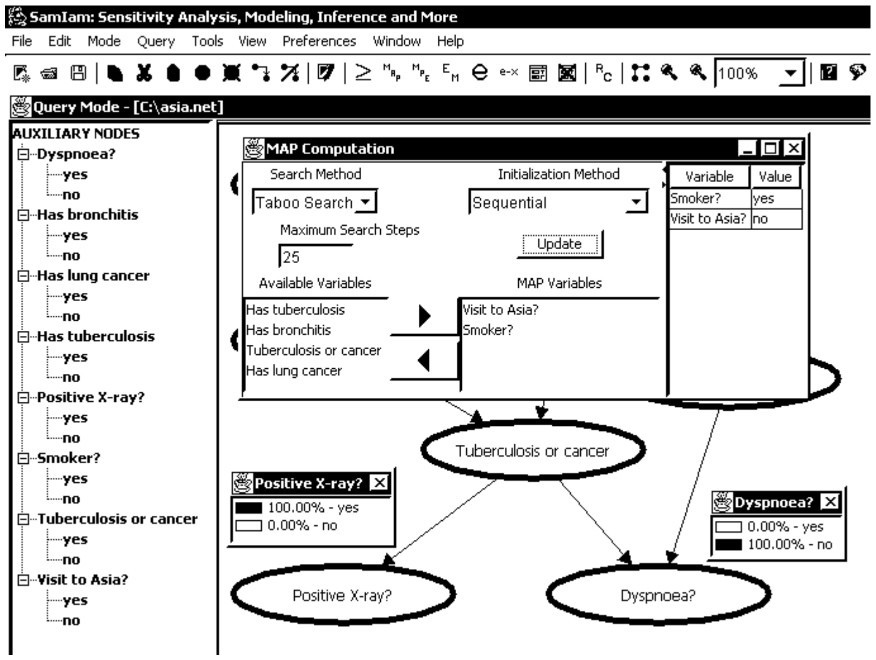
which has a probability of $\approx 38.57\%$ given evidence $\mathbf{e}$: $X{=}\mathsf{yes}$, $D{=}\mathsf{no}$. Note that SamIam displays the probability of the MPE instantiation α, $\mathsf{Pr}(\alpha) \approx 1.53\%$, instead of $\mathsf{Pr}(\alpha|\mathbf{e}) \approx 38.57\%$.

4.2.4 Maximum a Posteriori Hypothesis: MAP

The most probable explanation query is a special case of a more general class of queries for finding the most probable instantiation for a subset of network variables. Specifically, suppose that the set of all network variables is $\mathbf{X}$ and let $\mathbf{M}$ be a subset of these variables. Given some evidence $\mathbf{e}$, our goal is then to find an instantiation $\mathbf{m}$ of variables $\mathbf{M}$, for which the probability $\mathsf{Pr}(\mathbf{m}|\mathbf{e})$ is maximal. Any instantiation $\mathbf{m}$ which satisfies the previous property is known as a *maximum a posteriori hypothesis (MAP).* Moreover, the variables in $\mathbf{M}$ are known as *MAP variables.* Clearly, MPE is a special case of MAP, when the

[3]Again, we are suppressing the redundant rows in this CPT.

[4]Although this can be changed using the preferences menu of SamIam.



Search Method: Taboo Search
Initialization Method: Sequential
Maximum Search Steps: 25

Available Variables: Has tuberculosis, Has bronchitis, Tuberculosis or cancer, Has lung cancer

MAP Variables: Visit to Asia?, Smoker?

Variable	Value
Smoker?	yes
Visit to Asia?	no

Positive X-ray?
■ 100.00% - yes
□ 0.00% - no

Dyspnoea?
□ 0.00% - yes
■ 100.00% - no

Figure 4.8: Computing the maximum a posteriori hypothesis (MAP) given a positive X–ray and no dyspnoea.

MAP variables include all network variables. One reason why a distinction is made between MAP and MPE, however, is that MPE is much easier to compute algorithmically, an issue which we shall explain in later chapters.

Consider Figure 4.8 for an example of MAP. Here, we have a patient with a positive X–ray and no dyspnoea, so the evidence is $X=\textsf{yes}$, $D=\textsf{no}$. The MAP variables are $\mathbf{M} = \{A, S\}$, so we want to know the most likely instantiation of these variables given the evidence. According to Figure 4.8, the instantiation

$$A=\textsf{no},\ S=\textsf{yes},$$

is a MAP, which happens to have a probability of $\approx 50.74\%$ given the evidence.

Computing MAP is very difficult in general, and we will formally explain why in later chapters. This difficulty is one reason why one finds very little support for this type of queries in tools for Bayesian network inference. In fact, the MAP algorithm in SamIam is not exact. That is, although it tries to find an instantiation with a maximal probability, it is not guaranteed to return one—the answer given above is correct though!

A common method for approximating MAP is to compute an MPE and then project the result on the MAP variables. We stress, however, that this is only an approximation scheme as it may return an instantiation of the MAP variables, which is not maximally probable. Consider again the MPE example from Figure 4.7, which gives the following most probable instantiation:

$$A{=}\mathsf{no},\ S{=}\mathsf{no},\ T{=}\mathsf{no},\ C{=}\mathsf{no},\ B{=}\mathsf{no},\ P{=}\mathsf{no},\ X{=}\mathsf{yes},\ D{=}\mathsf{no},$$

under evidence $X{=}\mathsf{yes}$, $D{=}\mathsf{no}$. Projecting this MPE on the variables $\mathbf{M} = \{A, S\}$, we get the instantiation,

$$A{=}\mathsf{no},\ S{=}\mathsf{no},$$

which has a probability $\approx 48.09\%$ given the evidence. This instantiation is clearly not a MAP as we found a more probable instantiation earlier (that is, $A{=}\mathsf{no}$, $S{=}\mathsf{yes}$ with a probability of about 50.74%).

There is a relatively general class of situations in which the solution to a MAP query can be obtained immediately from an MPE solution by projecting it on the MAP variables. To formally define this class of situations, suppose that $\mathbf{E}$ contains evidence variables, $\mathbf{M}$ contains MAP variables, and let $\mathbf{Y}$ contain all other variables. The condition is that there is at most one instantiation $\mathbf{y}$ of variables $\mathbf{Y}$ which is compatible with any particular instantiations $\mathbf{m}$ and $\mathbf{e}$ of variables $\mathbf{M}$ and $\mathbf{E}$, respectively.[5] We will later discuss two classes of applications where this condition can be satisfied: diagnosis of digital circuits and channel coding.

4.3 Modeling with Bayesian Networks

We will discuss in this section a number of reasoning problems that arise in real–world applications and then show how each can be addressed by first modeling the problem using a Bayesian network, and then posing one of the queries defined in the previous section. For each of the considered application areas, we will state a concrete reasoning problem and develop a specific solution. The construction method of our networks will be made general enough so it can be easily carried over to other instances of the problem.

Before we proceed with this modeling exercise, we need to state some general modeling principles that we will adhere to in all our examples. Specifically, each Bayesian network will be constructed in three consecutive steps.

Step 1: Define the network variables and their values. We will partition network variables into three types: query, evidence, and intermediary variables. A *query variable* is one which we need to ask questions about, such as compute its posterior marginal. An *evidence variable* is one which we may need to assert evidence about. Finally, an *intermediary variable* is neither query nor evidence and is meant to aid the modeling process by detailing the relationship

[5]Given this condition, there is a one–to–one correspondence between instantiations of variables $\mathbf{Y} \cup \mathbf{M} \cup \mathbf{E}$ and instantiations of variables $\mathbf{M} \cup \mathbf{E}$ that have non–zero probability. Moreover, both types of instantiations have equal probabilities since $\mathsf{Pr}(\mathbf{y}|\mathbf{m}, \mathbf{e}) = 1$ and, hence, $\mathsf{Pr}(\mathbf{y}, \mathbf{m}, \mathbf{e}) = \mathsf{Pr}(\mathbf{y}|\mathbf{m}, \mathbf{e})\mathsf{Pr}(\mathbf{m}, \mathbf{e}) = \mathsf{Pr}(\mathbf{m}, \mathbf{e})$ for any instantiation $\mathbf{y}, \mathbf{m}, \mathbf{e}$ where $\mathsf{Pr}(\mathbf{y}, \mathbf{m}, \mathbf{e}) \neq 0$. This means that optimization in the space of instantiations of variables $\mathbf{Y} \cup \mathbf{M}$ is equivalent to the optimization in the space of instantiations of variables $\mathbf{M}$.

between evidence and query variables. Query and evidence variables are usually immediately determined from the problem statement. Intermediary variables are less obvious to determine and can depend on subtle modeling decisions. We will, however, provide some specific rules for when intermediary variables are necessary and when they can be done without. Determining the values of variables may also not be that obvious and will be dealt with in the context of specific examples.

Step 2: Define the network structure. This entails the determination of network edges. In all of our examples, we will be guided by a causal interpretation of network structure. Hence, the determination of the network structure will then be reduced to answering the following question about each network variable X: what is the set of variables $\mathbf{P}$ that we regard as the direct causes of X? Given the informality of this method of structure construction, we will verify the resulting network by confirming that we agree with the independence statements that it declares.

Step 3: Define the CPTs. The difficulty and objectivity of this step varies considerably from one problem to another. We will consider some problems where the CPTs are determined completely from the problem statement by objective considerations. We will also consider other problems in which the CPTs are a reflection of subjective beliefs, to the point where our specification of them is relatively arbitrary. For this class of problems, the techniques of Section 4.5 are especially useful as they provide tools for assessing the impact of CPT parameters on the overall conclusions drawn by the developed network.

4.3.1 Diagnosis I: Model from Expert

The first modeling example we shall consider is from medical diagnosis. Consider the following commonplace medical information:

> The flu is an acute disease characterized by fever, body aches and pains, and can be associated with chilling and a sore throat. The cold is a bodily disorder popularly associated with chilling and can cause a sore throat. Tonsillitis is inflammation of the tonsils which leads to a sore throat and can be associated with fever.

Our goal here is to develop a Bayesian network to capture this knowledge and then use it to diagnose the condition of a patient suffering from some of the symptoms mentioned above.

Our first step is to identify network variables, which as we mentioned earlier fall into three categories: query, evidence, and intermediary. To determine query variables, we need to identify those events that we need to ask questions about. In this case, we need to know whether the patient has a flu, a cold, or tonsillitis, which suggests three corresponding variables for this purpose; see the left of Figure 4.9. To determine evidence variables, we need to identify those events about which we can collect information. These correspond to the different symptoms that a patient can exhibit: chilling, body ache and pain, sore throat, and fever, which again leads to four corresponding variables as shown on the left

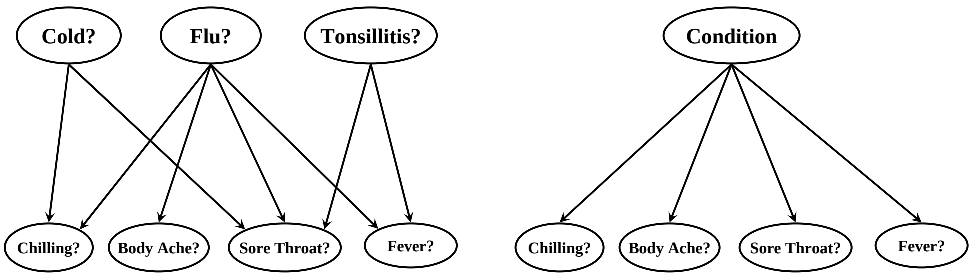


Figure 4.9: Two Bayesian network structures for medical diagnosis.

of Figure 4.9. The medical information given above does not seem to suggest any intermediary variables, so we will not include any. The values of each of the identified variables can be simply one of two, true or false, although more refined information may suggest different degrees of body ache. Determining the network structure is relatively straightforward: there are no causes for the different conditions, and the cause of each symptom is immediate from the given information.

It is tempting though to have a different network structure for this problem, which is shown on the right of Figure 4.9. In this case, we decided to have one variable "Condition," which has multiple values: normal, cold, flu, and tonsillitis. This network is clearly simpler than the first one, and is an instance of a very common structure known as *Naive Bayes,* which was the preferred structure for early reasoning systems based on Bayesian networks, as it is very easy to reason with—more on this in later chapters.

There is, however, quite a bit of difference between the two structures in Figure 4.9, since the Naive Bayes structure makes a key commitment known as the *single–fault* assumption. Specifically, it assumes that only one condition can exist in the patient at any time since the multiple values of the condition variable are exclusive of each other. This single–fault assumption has implications which are inconsistent with the medical information given above. For example, it implies that if the patient is known to have a cold, then whether or not he has a fever does not influence our belief in him suffering from a sore throat. This follows because fever and sore throat are d–separated by the condition variable in the Naive Bayes structure. This is incorrect though since a fever will increase our belief in tonsillitis, which will then increase our belief in a sore throat. Note that fever and sore throat are not d–separated by the cold variable in the structure on the left of Figure 4.9.

Here are some undesirable practical implications of this modeling inaccuracy. First, if the only evidence we have is body ache, then the probability of flu will go up in both networks. But this will also lead to dropping the probabilities of cold and tonsillitis in the Naive Bayes structure, yet these probabilities will remain the same in the other network since both cold and tonsillitis are d–separated from body ache. Second, if all we know is that the patient has no fever, then the belief in cold will increase in the Naive Bayes structure, while it remains unchanged in the other structure since cold is d–separated from fever.

We now turn to the specification of CPTs for the developed network structure. The main point here is that the CPTs for this network fall into one of two categories: the CPTs for the various conditions, and the ones for the symptoms. Specifically, the CPT for a condition, such as tonsillitis, must provide the belief in developing tonsillitis by a person about whom we have no knowledge of any symptoms. The CPT for a symptom, such as chilling, must provide the belief in this symptom under the four possible conditions: no cold and no flu, cold and no flu, no cold and flu, cold and flu. The probabilities needed for specifying these CPTs are usually obtained from a medical expert, who supplies this information based on known medical statistics or subjective beliefs gained through practical experience.

Another key method for specifying the CPTs of this and similar networks is by estimating them directly from medical records of previous patients. These records may look as follows:

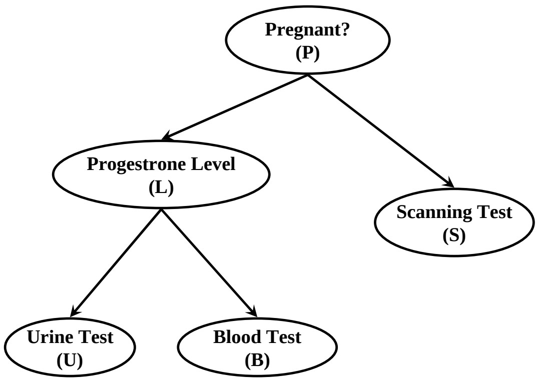
Case	*Cold*?	*Flu*?	*Tonsillitis*?	*Chilling*?	*Bodyache*?	*Sorethroat*?	*Fever*?
1	true	false	?	true	false	false	false
2	false	true	false	true	true	false	true
3	?	?	true	false	?	true	false
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Each row in the above table represents a medical case of a particular patient, where a ? indicates the unavailability of corresponding data for that patient. Many of the tools for Bayesian network inference can take a table such as the one given above and then generate a parameterization Θ of the given network structure, which maximizes the likelihood of seeing the given cases. In particular, if each case is represented by event d_i, then such tools will generate a parameterization Θ which leads to a probability distribution Pr that attempts to maximize the following quantity:

$$\prod_i \mathsf{Pr}(d_i).$$

Each term $\mathsf{Pr}(d_i)$ in the above product represents the probability of seeing the case d_i, and the product itself represents the probability of seeing all the cases (assuming that the cases are independent).

We close this section by noting that a diagnostic problem for a particular patient corresponds to a set of symptoms, which represent the known evidence. The goal is then to compute the most probable combination of conditions given the evidence. This can be solved by posing a MAP query to the network, with the MAP variables being cold, flu, and tonsillitis. If the evidence covers all of the four symptoms, the MAP query will then reduce to an MPE query.



P	θ_p
yes	.87

P	S	$\theta_{s\|p}$
yes	−ve	.10
no	+ve	.01

P	L	$\theta_{l\|p}$
yes	undetectable	.10
no	detectable	.01

L	B	$\theta_{b\|l}$
detectable	−ve	.30
undetectable	+ve	.10

L	U	$\theta_{u\|l}$
detectable	−ve	.20
undetectable	+ve	.10

Figure 4.10: A Bayesian network for detecting pregnancy based on three tests. Redundant CPT rows have been omitted.

4.3.2 Diagnosis II: Model from Expert

We now consider another problem from medicine which will serve to illustrate some major issues that arise when constructing Bayesian networks. The problem is a slight variation on a similar one introduced by Jensen:

> A few weeks after inseminating a cow, we have three possible tests to confirm pregnancy. The first is a scanning test which has a false positive of 1% and a false negative of 10%. The second is a blood test, which detects progesterone with a false positive of 10% and a false negative of 30%. The third test is a urine test, which also detects progesterone with a false positive of 10% and a false negative of 20%. The probability of a detectable progesterone level is 90% given pregnancy, and 1% given no pregnancy. The probability that insemination will impregnate a cow is 87%.

Our task here is to build a Bayesian network and use it to compute the probability of pregnancy given the results of some of these pregnancy tests.

The information given above suggests the following variables:

- One query variable to represent pregnancy (P).

- Three evidence variables to represent the results of various tests: scanning test (S), blood test (B), and urine test (U).
- One intermediary variable to represent progesterone level (L).

Moreover, common understanding of causal relationships in this domain suggests the causal structure in Figure 4.10, where pregnancy is a direct cause of both the scanning test and progesterone level, which in turn is the direct cause of the blood and urine tests. Some of the independencies implied by the constructed structure are given below:

- The blood and urine tests are independent, given the progesterone level.
- The scanning test is independent of the blood and urine tests, given the status of pregnancy.

Note, however, that blood and urine tests are not independent, even if we know the status of pregnancy, since the result of one test will affect our belief in progesterone level, which will then affect our belief in the second test's outcome. The CPTs for this problem can be specified directly from the problem statement as shown in Figure 4.10.

Suppose now that we inseminate a cow, wait for a few weeks, and then perform the three tests which all come out negative. Hence, the evidence we have is:

$$\mathbf{e}:\ S{=}\text{–ve},\ B{=}\text{–ve},\ U{=}\text{–ve}.$$

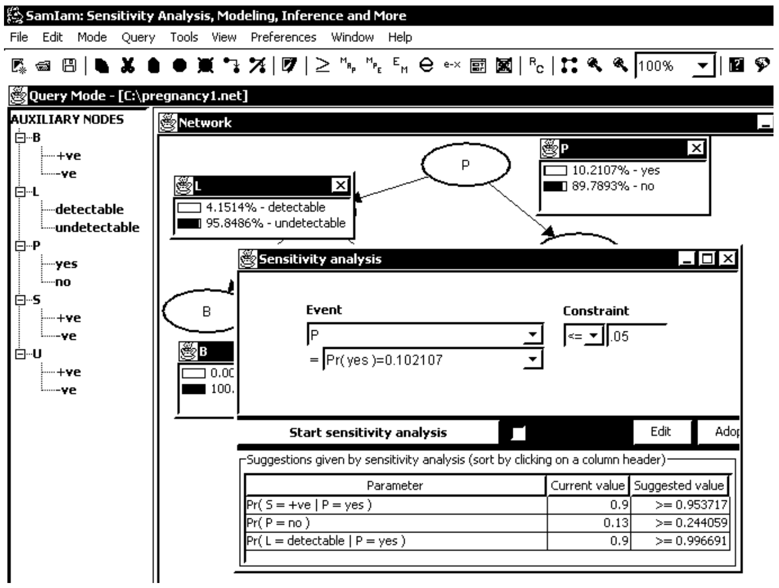
If we compute the posterior marginal for pregnancy given this evidence, we get:

P	$\Pr(p\vert\mathbf{e})$
yes	10.21%
no	89.79%

Note that even though the probability of pregnancy is reduced from 87% to 10.21%, it is still relatively high given that all three tests came out negative. In fact, the most conclusive test in this case is the scanning test, and the probability of pregnancy given that it comes out negative is as high as 40.22%.

4.3.3 Sensitivity Analysis

Suppose now that a farmer is not too happy with this and would like three negative tests to drop the probability of pregnancy to no more than 5%. Moreover, the farmer is willing to buy different test kits for this purpose, but needs to know the false positive and negative rates of the new tests, which would ensure the above constraint. This is a problem of *sensitivity analysis* in which we try to understand the relationship between the parameters of a Bayesian network, and the conclusions drawn based on the network. The SamIam tool does indeed provide functionality to conduct such an analysis. In particular, we can ask SamIam the following questions:


SamIam: Sensitivity Analysis, Modeling, Inference and More
File Edit Mode Query Tools View Preferences Window Help
100%
Query Mode - [C:\pregnancy1.net]
AUXILIARY NODES
B
+ve
-ve
L
detectable
undetectable
P
yes
no
S
+ve
-ve
U
+ve
-ve
Network
P
L
B
Sensitivity analysis
Start sensitivity analysis
Edit
Adop

P: 10.2107% - yes; 89.7893% - no

L: 4.1514% - detectable; 95.8486% - undetectable

B: 0.00; 100.

Event: P = Pr(yes)=0.102107

Constraint: <= .05

Suggestions given by sensitivity analysis (sort by clicking on a column header)

Parameter	Current value	Suggested value
Pr(S = +ve | P = yes)	0.9	>= 0.953717
Pr(P = no)	0.13	>= 0.244059
Pr(L = detectable | P = yes)	0.9	>= 0.996691

Figure 4.11: Sensitivity analysis in SamIam: what are the parameter changes which are necessary to ensure that the probability of pregnancy is no more than 5% given three negative tests?

> Which network parameters do we have to change, and by how much, so as to ensure that the probability of pregnancy given three negative tests would be no more than 5%?

Figure 4.11 depicts a screen shot of SamIam in which the above question is posed together with results that SamIam reports back. According to these results, there are three possible changes, each of which is guaranteed to satisfy the above constraint:

1. If the false negative rate for the scanning test were about 4.63% instead of 10%.

2. If the probability of pregnancy given insemination were about 75.59% instead of 83%.

3. If the probability of a detectable progesterone level given pregnancy were about 99.67% instead of 90%.

The last two changes are not feasible since the farmer does not intend to change the insemination procedure, nor does he control the progesterone level. Hence, he is left with only one option: replace the scanning test by another one that has a lower false negative rate.

What is interesting about the above results of sensitivity analysis is that they imply that improving the blood and urine tests cannot help. That is, the inherent uncertainty in progesterone level given pregnancy is such that even a perfect blood or urine test cannot help us in reaching the confidence level we want. These tests, however, would become relevant if we were less ambitious. For example, if our goal is to drop the probability of pregnancy to no more than 8% (instead of 5%), then SamIam identifies the following additional possibilities:

- The false negative for the blood test should be no more than about 12.32% instead of 30%.
- The false negative for the urine test should be no more than about 8.22% instead of 20%.

As is clear from the above example, sensitivity analysis is an important mode of analysis when developing Bayesian networks. We will discuss sensitivity analysis in some depth in later chapters, where we will show that it can be performed quite efficiently since its computational complexity is similar to that of computing posterior marginals.

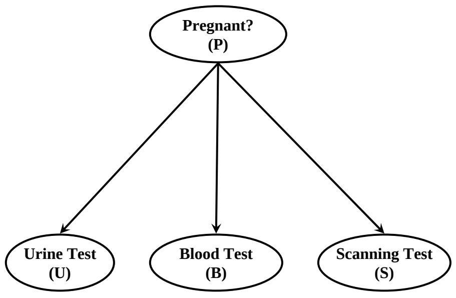
4.3.4 Network Granularity

The pregnancy problem provides an opportunity to discuss one of the central issues that arises when building Bayesian networks (and models in general): How fine grained should the network be?

Specifically, consider again the network in Figure 4.10 and note that progesterone level (L) is neither a query variable nor an evidence variable. That is, we cannot observe the value of this variable, nor are we interested in making inferences about it. The question then is: Why do we need to include it in the network?

Progesterone level is an intermediary variable, which helps in modeling the relationship between the blood and urine tests on the one hand, and pregnancy on the other. It is therefore a modeling convenience as it would be harder to build the model without including it explicitly. For example, the supplier of these tests may have only provided their false positive and negative rates with respect to progesterone level, and we may have obtained the numbers relating progesterone to pregnancy from another source. Hence, the inclusion of this intermediary variable in the network helps in integrating these two pieces of information in a modular way. But now that we have the network in Figure 4.10, we are able to compute the following quantities:

$$\begin{aligned}
\mathsf{Pr}(B{=}\mathsf{-ve}|P{=}\mathsf{yes}) &= 36\% \\
\mathsf{Pr}(B{=}\mathsf{+ve}|P{=}\mathsf{no}) &= 10.6\% \\
\mathsf{Pr}(U{=}\mathsf{-ve}|P{=}\mathsf{yes}) &= 27\% \\
\mathsf{Pr}(U{=}\mathsf{+ve}|P{=}\mathsf{no}) &= 10.7\%
\end{aligned}$$



P	θ_p
yes	.87

P	S	$\theta_{s\|p}$
yes	−ve	.10
no	+ve	.01

P	B	$\theta_{b\|p}$
yes	−ve	.36
no	+ve	.106

P	U	$\theta_{u\|p}$
yes	−ve	.27
no	+ve	.107

Figure 4.12: A Bayesian network for detecting pregnancy based on three tests. Redundant CPT rows have been omitted.

which allow us to build the network in Figure 4.12, where the progesterone level is no longer represented explicitly. The question now is whether this simpler network is equivalent to the original one from the viewpoint of answering queries.

By examining the two structures, one can immediately detect a major discrepancy: The simpler network in Figure 4.12 finds the blood and urine tests independent given pregnancy, while the original one in Figure 4.10 does not. One practical implication of this difference is that two negative blood and urine tests will count more in ruling out a pregnancy in the simpler network than they would in the original one. Specifically, the probability of pregnancy given these two negative tests is 45.09% in the simpler network, while it is 53.96% in the original one. Similarly, two positive tests will count more in establishing a pregnancy in Figure 4.12 than they would in Figure 4.10. The difference is not as dramatic in this case though: 99.61% in the simpler network versus 99.54% in the original one. The reason for this is that in the original network, two positive tests lead to a probability of 99.51% for detectable progesterone level, which implies that the two positive tests are *almost* independent in this case (they would be completely independent if the progesterone level was known for sure). Two negative tests, however, lead to a probability of only 21.22%, leaving much uncertainty about progesterone level, which provides more room for interaction between the two tests.

The moral of the previous example is that intermediary variables cannot be bypassed in certain cases as that may lead to changing the model in some undesirable ways. Here, the term “bypass” refers to the process of removing a variable, redirecting its parents to its children, and then updating the CPTs of these children (as they now have different parents). As it turns out though, one

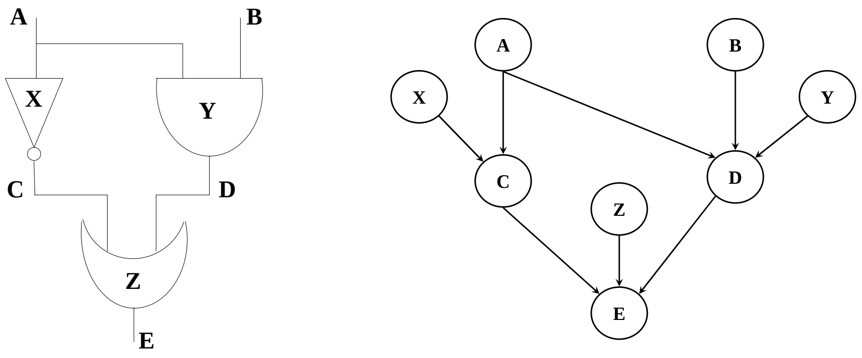


Figure 4.13: On the left, a digital circuit. On the right, a corresponding Bayesian network structure.

can identify a general case in which an intermediary variable can be bypassed without affecting *model accuracy,* which we define formally next. Suppose that Pr is the distribution induced by a Bayesian network, and let Pr' be the distribution induced by the new network after bypassing an intermediary variable. The bypass procedure does not affect model accuracy in case $\mathsf{Pr}(\mathbf{q}, \mathbf{e}) = \mathsf{Pr}'(\mathbf{q}, \mathbf{e})$ for all instantiations of query variables $\mathbf{Q}$ and evidence variables $\mathbf{E}$. This also implies that Pr an Pr' will agree on any other query formulated using these variables.

Suppose now that X is a variable that is neither a query variable nor an evidence variable. Then X can be bypassed in case it has a single child Y, as long as the CPT for variable Y is updated as follows:

$$\theta'_{y|\mathbf{uv}} = \sum_x \theta_{y|x\mathbf{v}} \theta_{x|\mathbf{u}}.$$

Here, $\mathbf{U}$ are the parents of variable X, and $\mathbf{V}$ are the other parents of variable Y. This bypass can be easily justified using the techniques we shall introduce in the next chapter. We will also see some concrete examples of it in some of the problems that we shall discuss later. We finally note that even though a variable may be bypassed without affecting the model accuracy, one may wish not to bypass it simply because the bypass procedure will lead to a large CPT. For example, if X and Y have five parents each, then Y will end up having nine parents after the bypass.

4.3.5 Diagnosis III: Model from Design

We now turn to another reasoning problem from the domain of diagnosis, which differs from previous problems in a fundamental way. Specifically, the Bayesian network model we shall develop for this problem is quite general to the point that it can be generated automatically by software for similar instances of the considered problem. The problem statement is as follows:

> Consider the digital circuit on the left of Figure 4.13. Given some values for the circuit primary inputs and output (test vector), our goal is to decide whether the circuit is behaving normally. If not, our goal is then to decide the most likely health states of its components.

The evidence variables for this problem are the primary inputs and output of the circuit, A, B and E, since we can observe the values of these variables. Given that our goal is to reason about the health of components, we need to include a query variable for each one of these components: X, Y and Z. That leaves the question of whether we need any intermediary variables. The obvious candidates for such variables are the states of internal wires, C and D. As it turns out, modeling the circuit becomes much easier if we do include these variables, especially if we are dealing with more realistic circuits that may have hundreds or thousands of components—it would be very difficult (if not infeasible) to model such circuits without explicitly representing the states of internal wires. The above choices lead to the variables shown in the structure to the right of Figure 4.13.

The edges of this structure are decided as follows. There are no direct causes for the health state of each component, hence, variables X, Y and Z have no parents—a more refined problem, however, may include other conditions, such as circuit power, which may end up being a direct cause of such variables. Similarly, there are no direct causes for the circuit primary inputs, A and B, which is why they have no parents in the given structure. Consider now variable D, which is the output of the and–gate. The direct causes of this variables are the gates inputs, A and B, and its state of health Y. Hence, these are the parents of D in the network structure. For the same reason, A and X are the parents of C, while C, D and Z are the parents of E. It should be clear that the developed structure generalizes easily to other circuits, regardless of the number and type of their gates. In fact, it generalizes to any system which is composed of *function blocks,* where the outputs of each block are determined by its inputs and its state of health—as long as the system does not contain feedback loops as that would lead to a cyclic structure.

To completely specify the Bayesian network, we need to specify its CPTs. We also have to decide on the values of different variables, which we avoided until now for a good reason. First, the values of variables representing circuit wires—whether primary inputs, outputs, or internal wires—are simply one of low or high. The choice of values for health variables is not as obvious though. The two choices are as follows. First, for each component, its health variable may only take one of two values, ok or faulty. The problem with this choice, however, is that the value faulty is too vague as a component may fail in a number of modes. For example, it is common to talk about stuck–at–zero faults in which the gate will generate a low output regardless of its inputs. Similarly, one can have stuck–at–one faults, or input–output–short faults in which an inverter would simply short its input to its output. From the viewpoint of precision, it is more appropriate to represent fault modes. Yet, this choice may put more demands on us when specifying the CPTs as we discuss next.

First, note that the CPTs for this structure fall in one of three classes: CPTs for variables representing primary inputs (A, B); CPTs for variables representing gate outputs (C, D, E); and CPTs for variables representing component health (X, Y, Z). We will consider each one of these next.

The CPTs for health variables depend on the values we choose for these variables. If we choose the values ok and faulty, then the CPT for each component, say X, would look like this:

X	θ_x
ok	.99
faulty	.01

Hence, if we have the *reliability* of each component, then all such CPTs are determined immediately. If we choose to represent fault modes, say stuckat0 and stuckat1, then the CPT would look like this:

X	θ_x
ok	.99
stuckat0	.005
stuckat1	.005

which implies that we know the probabilities of various fault modes. Clearly, we can assume that all fault modes are equally likely, in which case the component reliability will again be enough to specify the CPT.

The CPTs for component outputs are straightforward in case we represent fault modes, since the probability of each possible output is then guaranteed to be either 0 or 1, and can be determined directly from the gate's functionality. For example, following is the CPT for the inverter X (we are omitting redundant rows):

A	X	C	$\theta_{c\|a,x}$
high	ok	high	0
low	ok	high	1
high	stuckat0	high	0
low	stuckat0	high	0
high	stuckat1	high	1
low	stuckat1	high	1

The CPTs for the other two gates can be specified similarly. If we choose to have only two values, ok and faulty, for health variables, then we need to decide on the probabilities in case the gate is faulty:

A	X	C	$\theta_{c\|a,x}$
high	ok	high	0
low	ok	high	1
high	faulty	high	?
low	faulty	high	?

It is common to use a probability of .50 in the above case, and we show later that this choice is equivalent in a precise sense to the earlier choice of assigning equal probabilities to fault modes.

We now move to the CPTs for primary inputs, such as A, which require that we specify tables such as this one:

A	θ_a
high	.5
low	.5

We assumed here that a high input at A is equally likely to a low input. This appears arbitrary at first, but the good news is that the choice for these CPTs does not matter for the class of queries we are interested in. That is, if our goal is to compute the probability of some health state x, y, z given some test vector a, b, e, then this probability is independent of the probabilities $\mathsf{Pr}(a)$ and $\mathsf{Pr}(b)$ which can be chosen arbitrarily in this case. To prove this in general, let the primary inputs be $\mathbf{I}$, the primary outputs be $\mathbf{O}$, and the health variables be $\mathbf{H}$. First, note that in a Bayesian network, a probability of the form $\mathsf{Pr}(\mathbf{q}|\mathbf{e})$ does not depend on the CPTs for variables $\mathbf{E}$ if these variables are roots in the network. We also have:

$$\begin{aligned}
\mathsf{Pr}(\mathbf{h}|\mathbf{i},\mathbf{o}) &= \frac{\mathsf{Pr}(\mathbf{h},\mathbf{i},\mathbf{o})}{\mathsf{Pr}(\mathbf{i},\mathbf{o})} \quad \text{by Bayes conditioning} \\
&= \frac{\mathsf{Pr}(\mathbf{o}|\mathbf{i},\mathbf{h})\mathsf{Pr}(\mathbf{i}|\mathbf{h})\mathsf{Pr}(\mathbf{h})}{\mathsf{Pr}(\mathbf{o}|\mathbf{i})\mathsf{Pr}(\mathbf{i})} \quad \text{by chain rule} \\
&= \frac{\mathsf{Pr}(\mathbf{o}|\mathbf{i},\mathbf{h})\mathsf{Pr}(\mathbf{i})\mathsf{Pr}(\mathbf{h})}{\mathsf{Pr}(\mathbf{o}|\mathbf{i})\mathsf{Pr}(\mathbf{i})} \quad \text{since } \mathbf{I} \text{ and } \mathbf{H} \text{ are d–separated} \\
&= \frac{\mathsf{Pr}(\mathbf{o}|\mathbf{i},\mathbf{h})\mathsf{Pr}(\mathbf{h})}{\mathsf{Pr}(\mathbf{o}|\mathbf{i})}.
\end{aligned}$$

Since both $\mathbf{I}$ and $\mathbf{H}$ are roots, $\mathsf{Pr}(\mathbf{o}|\mathbf{i},\mathbf{h})$ and $\mathsf{Pr}(\mathbf{o}|\mathbf{i})$ do not depend on the CPTs for primary inputs $\mathbf{I}$. Moreover, $\mathsf{Pr}(\mathbf{h})$ depends only on the CPTs for health variables $\mathbf{H}$ since these variables are independent of each other.

Note, however, that if our goal is to use the Bayesian network to predict the probability that a certain wire, say E, is high, then the CPTs for primary inputs matter considerably. But if this is our goal, then it would be reasonable to expect that we have some distribution on the primary inputs, otherwise, we do not have enough information to answer the query of interest.

Fault Modes Revisited

We now return to the two choices we considered when modeling component faults. According to the first choice, the health of each component, say X, had only two values, ok and faulty, which leads to the following CPTs for the component health and output:

X	θ_x
ok	.99
faulty	.01

A	X	C	$\theta_{c\|a,x}$
high	ok	high	0
low	ok	high	1
high	faulty	high	.5
low	faulty	high	.5

According to the second choice, the health of each component had three values, ok, stuckat0 and stuckat1, leading to the following CPTs:

X	θ_x
ok	.99
stuckat0	.005
stuckat1	.005

A	X	C	$\theta_{c\|a,x}$
high	ok	high	0
low	ok	high	1
high	stuckat0	high	0
low	stuckat0	high	0
high	stuckat1	high	1
low	stuckat1	high	1

We will now show that these two choices are equivalent in a precise sense, except that the second choice is preferable computationally.

First, the equivalence between the above choices is in the following sense. Since each health variable has a single child (see Figure 4.13), then we can eliminate these variables as suggested in Section 4.3.4. In particular, if we eliminate variable X, then its single child C will have only one parent A and the following CPT:

$$\mathsf{Pr}(c|a) = \sum_x \mathsf{Pr}(c|a,x)\mathsf{Pr}(x).$$

If we eliminate the health variable X, assuming it has values ok and faulty, we get the following CPT for C:

A	C	$\theta_{c\|a}$
high	high	$.005 = (0 * .99) + (.5 * .01)$
low	high	$.995 = (1 * .99) + (.5 * .01)$

Similarly, if we eliminate the health variable X, assuming it has values ok, stuckat0 and stuckat1, we get the following CPT for C:

A	C	$\theta_{c\|a}$
high	high	$.005 = (0 * .99) + (0 * .005) + (1 * .005)$
low	high	$.995 = (1 * .99) + (0 * .005) + (1 * .005)$

Hence, the two CPTs for variable C are the same. This would also be the case if we eliminate health variables Y and Z, leading to equivalent CPTs for each of variables D and E. What this means is that the two Bayesian networks for the circuit in Figure 4.13 are equivalent in the following sense: Any query which involves only the wires A, B, C, D and E is guaranteed to have the same answer with respect to each network.

There is one key difference between the two networks, however, which has a major computational implication. The network with fault modes satisfies the following crucial property: given the values of health variables (X, Y, Z), and given the values of input/output variables (A, B, E), there is at most one instantiation of the remaining variables (C, D) which is consistent with these values. What this means is that MAP queries on this network, where MAP variables are X, Y, Z, and evidence variables are A, B, E, can be reduced to MPE queries by simply projecting the result of an MPE query on the MAP variables X, Y and Z. This is not true, however, for the network with no fault modes, which prohibits us from reducing MAP computations to MPE computations.

To consider an example, let us assume that we have the following test vector:

$$\mathbf{e}: A{=}\mathsf{high},\ B{=}\mathsf{high},\ E{=}\mathsf{low}.$$

And suppose that we compute MAP and MPE under this observation with respect to both networks. According to the network with fault modes, we get:

Query	X	Y	Z	A	B	C	D	E
MPE given **e**	ok	stuckat0	ok	high	high	low	low	low
MAP given **e**	ok	stuckat0	ok					

Hence, we could have obtained this MAP result by simply projecting the MPE solution on health variables, which is guaranteed to always work for this class of networks. If we compute these queries with respect to the other network though, we get:

Query	X	Y	Z	A	B	C	D	E
MPE given **e**	ok	ok	faulty	high	high	low	low	low
MAP given **e**	ok	faulty	ok					

Hence, we could not have obtained this MAP result by projecting the MPE solution, which is not surprising since this is not possible in general.

We close this section by stressing that the above observations apply to any digital circuit and its corresponding Bayesian networks which are constructed as suggested above.

Integrating Time

We now turn to an extension of the diagnosis problem we considered thus far, in which we assume that we have two test vectors instead of only one. For example, we may apply two high inputs to the circuit and observe an abnormal low output, leading us to blame the and–gate (Y) as we did earlier. If we look at this example carefully, however, we find that the or–gate (Z) is a close candidate to blame in this case. To improve our confidence level, we perform another test. Specifically, apply two low inputs to the circuit and observe another abnormal low output. Our goal now is to find the most probable health state of the circuit given these two test vectors.

The key point to realize in this more elaborate problem is that we now have six evidence variables instead of only three, as we need to capture the second test vector. This leads to three additional evidence variables A', B' and E'. The same applies to the intermediary variables, leading to two additional variables C' and D', which are needed to relate the elements of the second test vector; see Figure 4.14. Whether we need to include additional health variables depends on whether the health of a component stays the same during each of the two tests. If we want to allow for the possibility of intermittent faults, where the health

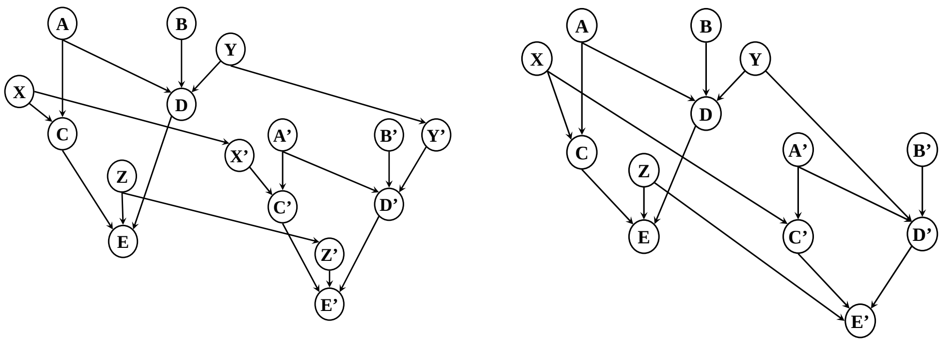


Figure 4.14: Two Bayesian network structures for diagnosing a circuit using two test vectors.

of a component can change from one test to another, then we need to include additional health variables X', Y', and Z' as shown on the left of Figure 4.14. Otherwise, the original health variables are sufficient, leading to the structure on the right of Figure 4.14.

If we decide to allow intermittent faults, then we need a new class of CPTs to completely specify the resulting structure on the left of Figure 4.14. In particular, for each component, say X, we now need to specify a CPT such as this one:

X	X'	$\theta_{x'\|x}$
ok	ok	.99
ok	faulty	.01
faulty	ok	.001
faulty	faulty	.999

which specifies a *persistence* model for the health of various components. For example, the above table says that there is a 1% chance that a healthy component would become faulty, and there is a .1% chance that a faulty component would become healthy again (intermittent fault).

4.3.6 Channel Coding

Consider the following problem from the domain of channel coding:

> We need to send four bits U_1, U_2, U_3 and U_4 from a source S to a destination D over a noisy channel, where there is a 1% chance that a bit will be inverted before it gets to the destination. To improve the reliability of this process, we will add three redundant bits X_1, X_2 and X_3 to the message, where X_1 is the XOR of U_1 and U_3, X_2 is the XOR of U_2 and U_4, and X_3 is the XOR of U_1 and U_4. Given that we received a message containing seven bits at destination D, our goal is to restore the message generated at the source S.

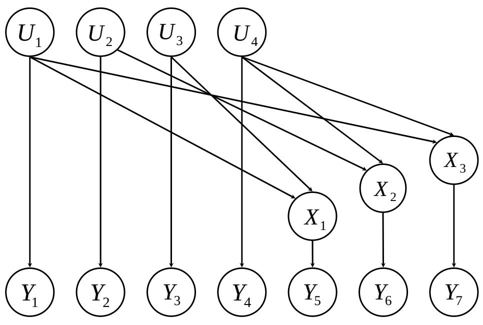


Figure 4.15: A Bayesian network structure for modeling a coding problem.

In channel coding terminology, the bits $U_1, \ldots, U_4$ are known as *information bits,* $X_1, \ldots, X_3$ are known as *redundant bits,* and $U_1, \ldots, U_4, X_1, \ldots, X_3$ is known as the *code word,* or *channel input.* Moreover, the message received at the destination $Y_1, \ldots, Y_7$ is known as the *channel output.* Our goal then is to restore the channel input given some channel output.

As we have seen in previous examples, query and evidence variables are usually determined immediately from the problem statement. In this case, evidence variables are $Y_1, \ldots, Y_7$ and they represent the bits received at destination D. Moreover, query variables are $U_1, \ldots, U_4$ and they represent the bits originating at source S. One can also include the redundant bits $X_1, \ldots, X_3$ in query variables, or view them as constituting the set of intermediary variables. We shall see later that this choice does not matter much, so we will include these bits in the query variables.

The causal structure for this problem is shown in Figure 4.15 where edges have been determined based on a basic understanding of causality in this domain. Specifically, the direct causes of each redundant bit are its two corresponding information bits, and the direct cause of each bit on the channel output is the corresponding bit on the channel input. Information bits have no parents since they are not directly caused by any of the other bits.

There are three CPT types in the problem. First, the CPT for each redundant bit, say X_1, is given as follows:

U_1	U_3	X_1	$\theta_{x_1\|u_1,u_3}$
1	1	1	0
1	0	1	1
0	1	1	1
0	0	1	0

This CPT simply captures the functional relationship between X_1 and the corresponding information bits U_1 and U_3. That is, $\mathsf{Pr}(x_1|u_1, u_3) = 1$ iff $x_1 = u_1 \oplus u_3$.

The CPT for a channel output bit, say Y_1, is as follows:

U_1	Y_1	$\theta_{y_1\|u_1}$
1	0	.01
0	1	.01

This CPT is then capturing the simple noise model given in the problem statement, but we later discuss a more realistic noise model based on Gaussian distributions.

Finally, the CPTs for information bits, such as U_1, capture the distribution of messages sent out from the source S. Assuming a uniform distribution, we can then use the following CPT:

U_1	θ_{u_1}
1	.5
0	.5

MAP or Posterior–Marginal (PM) Queries?

Now that we have completely specified a Bayesian network for this problem, we need to decide on the specific query to pose. Our goal is to restore the channel input given channel output, but there are two ways for achieving this:

1. Compute a MAP for the channel input $U_1, \ldots, U_4, X_1, \ldots, X_3$ given channel output $Y_1, \ldots, Y_7$.[6]

2. Compute the PM for each bit U_i/X_i in the channel input, given channel output $Y_1, \ldots, Y_7$, and then select the value of U_i/X_i which is most probable.

To understand the tradeoff between these two queries, suppose that we have a *decoder,* which returns some channel input $u_1, \ldots, u_4, x_1, \ldots, x_3$ whenever it is given a channel output $y_1, \ldots, y_7$. Suppose further that our goal is to evaluate the performance of this decoder on a given set of channel inputs $\alpha_1, \ldots, \alpha_n$. There are two possible quality measures that one can use for this purpose:

- *Word Error Rate (WER):* We can send each one of these inputs α_i into the channel, collect the corresponding output β_i, feed it into the decoder, and finally collect the decoder output γ_i. We then compare each decoder output γ_i with corresponding channel input α_i. Let m be the number of channel inputs that are recovered incorrectly by the decoder, i.e., there is a mismatch on some bit between α_i and γ_i. The word error rate for that decoder is then m/n.

[6]This is actually an MPE query since MAP variables include all variables except those for evidence. Even if the MAP variables contain $U_1, \ldots, U_4$ only, this MAP query can be obtained by projecting an MPE on the information bits $U_1, \ldots, U_4$, since the redundant bits $X_1, \ldots, X_3$ are functionally determined by the information bits $U_1, \ldots, U_4$.

- *Bit Error Rate (BER):* We perform the same experiment as above, except that when comparing the decoder output γ_i with the original channel input α_i, we count the number of bits on which they disagree. Let k be the number of disagreed upon bits, and let l be the total number of bits sent into the channel. The bit error rate for the decoder is then k/l.

Given these performance measures, the choice between MAP versus PM queries becomes a choice between the above performance measures. In particular, decoders based on MAP queries are optimizing WER, while decoders based on PM queries are optimizing BER. Suppose for example that the information bits represent pixels in an image, which is to be viewed and interpreted by a human when it arrives at the destination. Optimizing BER appears to be a reasonable choice in this case.

Noise Model

The problem statement above assumed a simple noise model, according to which a channel input bit u_i is received as a channel output bit y_i, where $y_i \neq u_i$ with 1% probability. In practice, however, channel output is assumed to be continuous, and noise is modeled by a Gaussian distribution. Specifically, let Z_i be a continuous variable representing the channel output for input bit U_i. Noise is then specified by two Gaussian density functions $N_0(z_i)$ and $N_1(z_i)$, which represent the uncertainty in channel output Z_i given the input bits U_i=0 and U_i=1, respectively.

This more sophisticated noise model can be implemented by interpreting the continuous channel output z_i as a soft evidence on the input bit U_i=1 with a Bayes factor of $N_1(z_i)/N_0(z_i)$ (or, equivalently, as a soft evidence on the input bit U_i=0 with a Bayes factor of $N_0(z_i)/N_1(z_i)$). The answer we obtain from this construction is equivalent to the answer we obtain from a Bayesian network with continuous variables Z_i. We do not prove this here though since we did not discuss the semantics of Bayesian networks with continuous variables.

4.3.7 Commonsense Knowledge

Consider the following commonsense reasoning problem, which is a variation on a problem proposed by Charniak:

> When SamBot goes home at night, he wants to know if his family is home before he tries the doors. (Perhaps the most convenient door to enter is double locked when nobody is home). Often when SamBot́s wife leaves the house she turns on an outdoor light. However, she sometimes turns on this light if she is expecting a guest. Also, SamBot́s family has a dog. When nobody is home, the dog is in the back yard. The same is true if the dog has bowel trouble. Finally, if the dog is in the back yard, SamBot will probably hear her barking, but sometimes he can be confused by other dogs barking. SamBot is equipped with two sensors: a light-sensor for detecting outdoor

> lights and a sound-sensor for detecting the barking of dogs. Both of these sensors are not completely reliable and can break. Moreover, they both require SAMBOTś battery to be in good condition.

Our goal is then to build a Bayesian network that SAMBOT will use to reason about the above situation. Specifically, given sensory input, SAMBOT needs to compute his beliefs in whether his family is home and whether any of his hardware is broken.

This problem is less structured than any of the problems we looked at so far. The choice of evidence and query variables remain relatively obvious though. We only have two evidence variables in this case:

LightSensor: Is SAMBOTś light sensor detecting a light (outdoor light)?

SoundSensor: Is SAMBOTś sound sensor detecting a sound (barking)?

We also have four query variables:

FamilyHome: Is SAMBOTś family (wife) home?

LightSensorBroken: Is SAMBOTś light sensor broken?

SoundSensorBroken: Is SAMBOTś sound sensor broken?

Battery: Is SAMBOTś Battery in good condition?

Finally, we have six intermediary variables:

ExpectingCompany: Is SAMBOTś family (wife) expecting company?

OutdoorLight: Is the outdoor light on?

DogOutside: Is SAMBOTś dog outside?

DogBarking: Is some dog barking?

DogBowel: Is SAMBOTś dog having bowel problems?

OtherBarking: Are other dog's barking?

The causal structure corresponding to the above choices is shown in Figure 4.16. Note that all intermediary variables, except for ExpectingCompany, have a single child each. Hence, they can be easily bypassed using the technique we discussed in Section 4.3.4.

Parameterizing this structure can be done based on a combination of sources:

- Statistical information, such as reliabilities of sensors and battery.
- Subjective beliefs relating to how often the wife goes out, guests are expected, the dog has bowel trouble, etc.
- Objective beliefs regarding the functionality of sensors.

One can also imagine the robot recording his experiences each evening and then constructing a data table similar to the one we discussed in Section 4.3.1, which can then be used to learn the network parameters automatically.

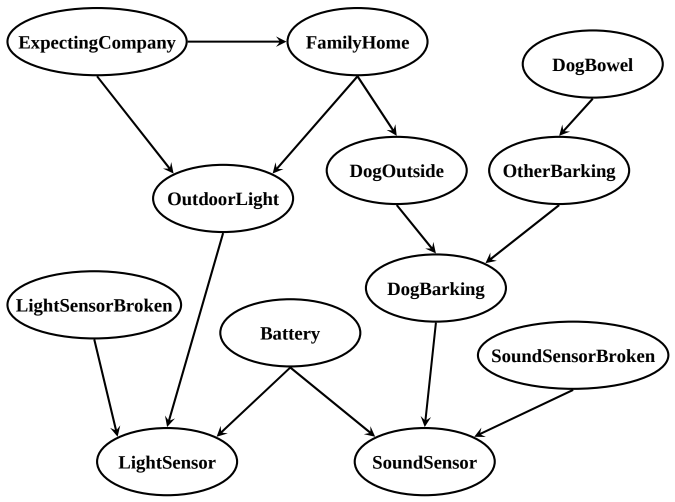


Figure 4.16: A Bayesian network structure for the SamBot problem.

4.4 Dealing with Large CPTs

One of the major issues that arise when building Bayesian network models is the potentially large size of CPTs. Suppose, for example, that we have a variable E with parents $C_1, \ldots, C_n$, and suppose further that each one of these variables has only two values. We then need 2^n independent parameters to specify the CPT for variable E completely. The following table gives a concrete feel of this CPT size for different values of n:

Number of Parents: n	Parameter Count: 2^n
2	4
3	8
6	64
10	1024
20	$1,048,576$
30	$1,073,741,824$

Hence, both modeling and computational problems will arise as the number of parents n gets larger, but the modeling problem will obviously manifest first. Specifically, a table with 1024 rows is rarely a concern from a computational viewpoint, but imagine trying to commit a medical expert to specifying 1024 numbers in order to quantify the relationship between headache and ten different medical conditions that may cause it.

There are two different types of solutions to this problem of large CPTs, which we discuss next.

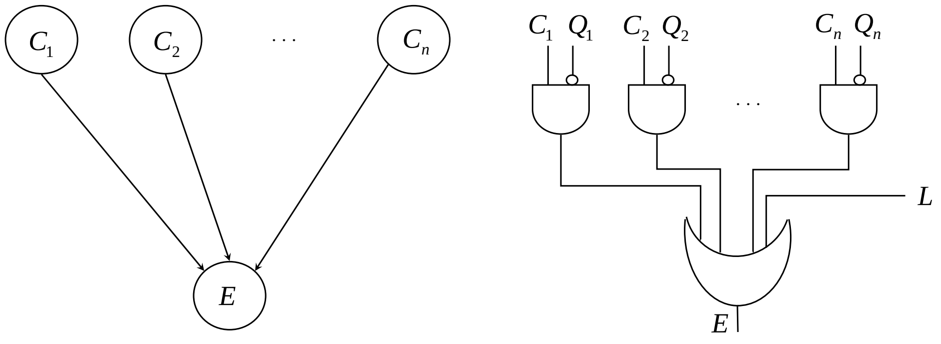


Figure 4.17: On the left, a node E with n parents, leading to a CPT whose size is exponential in n. On the right, a circuit diagram illustrating noisy–or semantics for the CPT on the left.

4.4.1 Canonical Models of Local Structure

The first approach for dealing with large CPTs is to try to develop a *micro model* which details the relationship between the parents $C_1, \ldots, C_n$ and their common child E. The goal here is to reveal the local structure of this relationship in order to specify it using a smaller number of parameters than 2^n.

One of the most successful micro models for this purpose is known as the *noisy–or* model, and is depicted in Figure 4.17. To understand this model, it is best to interpret parents $C_1, \ldots, C_n$ as causes, and variable E as their common effect. The intuition here is that each cause C_i is capable of establishing the effect E on its own, regardless of other causes, except under some unusual circumstances which are summarized by the *suppressor* variable Q_i. That is, when the suppressor Q_i of cause C_i is active, cause C_i is no longer able to establish E. Moreover, the *leak* variable L is meant to represent all other causes of E which were not modelled explicitly. Hence, even when none of the causes C_i is active, the effect E may still be established by the leak variable L. Given this interpretation of the noisy–or model, one would then expect the probabilities of suppressors and leak to be usually small in practice.

The noisy–or model in Figure 4.17 can then be specified using $n+1$ parameters, which is remarkable from a modeling viewpoint. For example, to model the relationship between headache and ten different conditions that may cause it, all we need is the following numbers:

- $\theta_{q_i} = \mathsf{Pr}(Q_i{=}\mathsf{active})$: the probability that the suppressor of cause C_i is active. That is, the percentage of cases under which condition C_i does not cause a headache.
- $\theta_l = \mathsf{Pr}(L{=}\mathsf{active})$: the probability that the leak variable is active. That is, the percentage of cases under which the patient will have a headache even though he does not suffer from any of the modelled conditions.

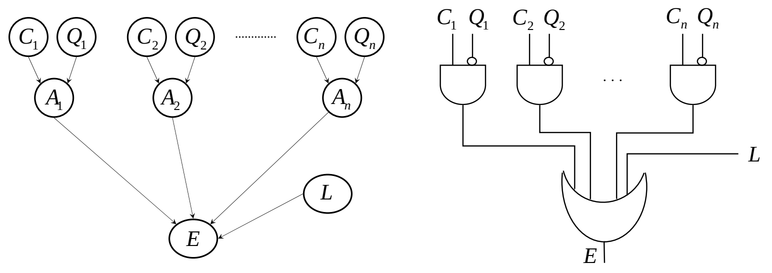


Figure 4.18: A Bayesian network structure corresponding to a noisy–or model.

What is even more remarkable about the noisy–or model is that it contains enough information to completely specify the conditional probability of variable E given any instantiation α of the parents $C_1, \ldots, C_n$. That is, we can use the noisy–or model to completely specify the CPT for variable E. To show this, let I_α be the indices of causes that are active in α. For example, if

$$\alpha: \; C_1\text{=active}, \; C_2\text{=active}, \; C_3\text{=passive}, \; C_4\text{=passive}, \; C_5\text{=active},$$

then I_α is the set containing indices $1, 2$ and 5. Using this notation, we have:

$$\mathsf{Pr}(E\text{=passive}|\alpha) = (1 - \theta_l) \prod_{i \in I_\alpha} \theta_{q_i} \tag{4.1}$$

From this equation, we also get $\mathsf{Pr}(E\text{=active}|\alpha) = 1 - \mathsf{Pr}(E\text{=passive}|\alpha)$. Hence, the full CPT for variable E, with its 2^n parameters, can be induced from the $n + 1$ parameters associated with the noisy–or model.

One can derive Equation 4.1 in a number of ways. The more intuitive derivation is that given the status α of causes $C_1, \ldots, C_n$, the effect E will be passive only if the leak was passive and all suppressors Q_i, for $i \in I_\alpha$, were active. Since the leak and suppressors are assumed to be independent, the probability of that happening is simply given by Equation 4.1.

Another way to derive Equation 4.1 is to build a *micro Bayesian network* as given in Figure 4.18, which explicitly represents the noisy–or model. This network will have $3n + 2$ variables, where the CPTs for all variables, except causes $C_1, \ldots, C_n$, are determined from the noisy–or model. Note here that variables $L, Q_1, \ldots, Q_n$ has a single child each. Hence, we can eliminate each of them as discussed in Section 4.3.4, while updating the CPT for variable E after eliminating each variable. We can then eliminate variables $A_1, \ldots, A_n$, since each of these variables has a single child E. If we eliminate all variables, we get the CPT given by Equation 4.1.

To consider a concrete example of noisy–or models, let us revisit the medical diagnosis problem from Section 4.3.1. Sore throat (S) has three causes in this problem: cold (C), flu (F) and tonsillitis (T). If we assume that S is related to its causes by a noisy–or model, we can then specify the CPT for S by the following four probabilities:

C1	C2	C3	C4	Pr(E=1)
1	1	1	1	**0.0**
1	1	1	0	**0.0**
1	1	0	1	**0.0**
1	1	0	0	**0.0**
1	0	1	1	**0.0**
1	0	1	0	**0.0**
1	0	0	1	**0.0**
1	0	0	0	**0.0**
0	1	1	1	**0.9**
0	1	1	0	**0.9**
0	1	0	1	**0.9**
0	1	0	0	**0.9**
0	0	1	1	**0.3**
0	0	1	0	**0.3**
0	0	0	1	**0.6**
0	0	0	0	**0.8**

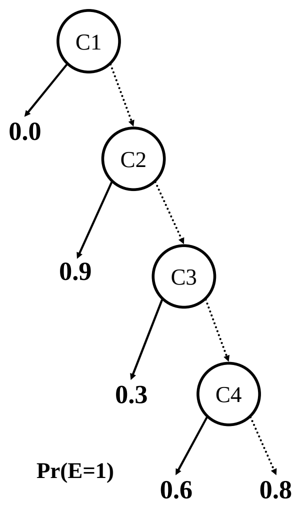


Figure 4.19: A CPT and its corresponding decision tree representation. Solid edges represent 1–values and dotted edges represent 0–values.

- The suppressor probability for cold, say .15.
- The suppressor probability for flu, say, .01.
- The suppressor probability for tonsillitis, say .05.
- The leak probability, say .02.

The CPT for sore throat is then determined completely as follows:

C	F	T	S	$\theta_{s\mid c,f,t}$	Equation 4.1
true	true	true	true	0.9999265	$1-(1-.02)(.15)(.01)(.05)$
true	true	false	true	0.99853	$1-(1-.02)(.15)(.01)$
true	false	true	true	0.99265	$1-(1-.02)(.15)(.05)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	
false	false	false	true	.02	$1-(1-.02)$

4.4.2 Other Representations of CPTs

The noisy–or model is only one of several other models for local structure. Each one of these models is based on some assumption about the way parents $C_1, \ldots, C_n$ interact with their common child E. If the assumption corresponds to reality, then one can use these models for local structure. Otherwise, the resulting Bayesian network will be an inaccurate model of that reality.

Most often, we have some local structure in the relationship between a node and its parents, but that structure does not fit nicely into any of the existing canonical models such as noisy–or. Consider the CPT in Figure 4.19 for an example. Here, we have a node with four parents and a CPT that exhibits a considerable amount of structure. For example, the probability of $E{=}1$ given $C_1{=}1$ is 0.0 regardless of the values assumed by parents $C_2, \ldots, C_4$. Moreover, given that $C_1{=}0$ and $C_2{=}1$, the probability of $E{=}1$ is 0.9 regardless of the values of other two parents. Even with all of this local structure, the CPT of this nodes does not correspond to the assumptions underlying a noisy–or model and, hence, cannot be generated by such a model.

For this kind of irregular structure, there are several non–tabular representations that are not necessarily exponential in the number of parents. We discuss some of these representations below.

Decision Trees

One of the more popular representations for this purpose is the decision tree, an example of which is shown in Figure 4.19. The basic idea here is that we start at the root of the tree, and then branch downward at each node depending on the value of the variable attached to that node. The decision tree in Figure 4.19 represents the probability of $E{=}1$ under every possible instantiation of the parents $C_1, \ldots, C_4$, except that these instantiations are not represented explicitly. For example, if $C_1{=}1$, then the probability of $E{=}1$ is immediately decided to be 0.0, as shown on the very left of this decision tree. Obviously, the decision tree can have a size which is linear in the number of parents if there is enough structure in the CPT. But it is important to stress that the decision tree may also be exponential in size if no such structure exists in the given CPT.

If–Then Rules

A CPT for variable E can be represented using a set of if–then rules of the form:

$$\text{If } \alpha_i \text{ then } \mathsf{Pr}(e) = p_i,$$

where α_i is a propositional sentence constructed from the parents of variable E. For example, the CPT in Figure 4.19 can be represented using the following rules:

If $C_1{=}1$	then	$\mathsf{Pr}(E{=}1) = 0.0$
If $C_1{=}0 \wedge C_2{=}1$	then	$\mathsf{Pr}(E{=}1) = 0.9$
If $C_1{=}0 \wedge C_2{=}0 \wedge C_3{=}1$	then	$\mathsf{Pr}(E{=}1) = 0.3$
If $C_1{=}0 \wedge C_2{=}0 \wedge C_3{=}0 \wedge C_4{=}1$	then	$\mathsf{Pr}(E{=}1) = 0.6$
If $C_1{=}0 \wedge C_2{=}0 \wedge C_3{=}0 \wedge C_4{=}0$	then	$\mathsf{Pr}(E{=}1) = 0.8$

For the rule–based representation to be complete and consistent, the set of rules must satisfy a number of conditions. First, for every instantiation e of the variable E, the set of rules, If α_i then $\mathsf{Pr}(e) = p_i$, must be such that their premises α_i are mutually exclusive and exhaustive. This ensures that rules will not conflict with each other and that they will cover every possible instantiation of the parents. We also need to have one set of rules for each value e of variable E, except possibly for one. These two conditions will ensure that every CPT parameter is implied by the given set of rules. Again, the rule–based representation may be very efficient in cases where the CPT has a lot of structure, yet may be of exponential size when no such structure exists.

Deterministic CPTs

A deterministic CPT is one in which every probability is either 0 or 1. These CPTs are very common in practice and we have seen a number of them in Sections 4.3.5 and 4.3.6.

Suppose now that we have a deterministic CPT for variable E with values $e_1, \ldots, e_m$. For each one of these values e_i, and for each parent instantiation α, we must then have $\mathsf{Pr}(e_i|\alpha) = 1$ or $\mathsf{Pr}(e_i|\alpha) = 0$. Hence, we can represent this CPT compactly by a set of propositional sentences,

$$\Gamma_i \quad \Longrightarrow \quad E{=}e_i,$$

where the set of sentences Γ_i must be mutually exclusive. These sentences are interpreted as follows:

$$\mathsf{Pr}(e_i|\alpha) = \begin{cases} 1 & \text{if parent instantiation } \alpha \text{ is consistent with } \Gamma_i; \\ 0 & \text{otherwise.} \end{cases}$$

Consider for example the following deterministic CPT from Sections 4.3.5:

A	X	C	$\theta_{c\|a,x}$
high	ok	high	0
low	ok	high	1
high	stuckat0	high	0
low	stuckat0	high	0
high	stuckat1	high	1
low	stuckat1	high	1

We can represent this CPT as follows:

$$\begin{array}{lcl} (X{=}\mathsf{ok} \wedge A{=}\mathsf{high}) \vee X{=}\mathsf{stuckat0} & \Longrightarrow & C{=}\mathsf{low} \\ (X{=}\mathsf{ok} \wedge A{=}\mathsf{low}) \vee X{=}\mathsf{stuckat1} & \Longrightarrow & C{=}\mathsf{high} \end{array}$$

This representation is very effective in general, especially when the number of parents is quite large.

Expanding CPT representations

We close this section with a word of caution on how the above representations of CPTs are sometimes used by Bayesian network tools. Specifically, many of these tools will *expand* these representations into their corresponding CPTs before they perform inference. In such a case, these representations are only being utilized in addressing the modeling problem since the size of expanded CPT is still exponential in the number of parents. The reason why these tools

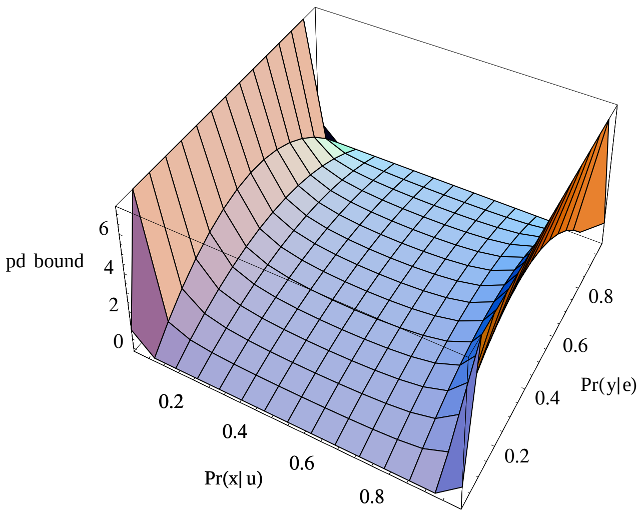


Figure 4.20: An upper bound on the partial derivative $|\partial \mathsf{Pr}(\mathbf{y}|\mathbf{e})/\partial \tau_{x|\mathbf{u}}|$ as a function of the query value $\mathsf{Pr}(\mathbf{y}|\mathbf{e})$ and the parameter value $\mathsf{Pr}(x|\mathbf{u})$.

perform this expansion before inference is that many algorithms for inference in Bayesian networks require a tabular representation of CPTs as they cannot operate on the above representations directly. We will elaborate on this issue further when we discuss algorithms in future chapters.

4.5 The Significance of Network Parameters

The parameters of a Bayesian network can be viewed as *local beliefs* since a parameter $\theta_{x|\mathbf{u}}$ represents our belief in one variable X given the state of other variables $\mathbf{U}$ that are very closely related to it (direct causes). On the other hand, the results of queries posed to a Bayesian network can be viewed as *global beliefs* since the result of a query $\mathsf{Pr}(\mathbf{y}|\mathbf{e})$ represents our belief in some variables $\mathbf{Y}$ given the state of other variables $\mathbf{E}$ that can be distantly related to them (i.e., their indirect causes, indirect effects, or any combination of these). One of the more practical issues when reasoning with Bayesian networks is that of understanding the way in which these global beliefs are impacted by the local beliefs. That is, we often need to understand the relationship between the result of a query $\mathsf{Pr}(\mathbf{y}|\mathbf{e})$ and the value of some network parameter $\theta_{x|\mathbf{u}}$. We will present next some known relationships between these two quantities which can provide valuable insights when building Bayesian network models.

Suppose that X is a variable that has two values, x and $\overline{x}$, and a set of parents $\mathbf{U}$. Obviously enough, we must have

$$\theta_{x|\mathbf{u}} + \theta_{\overline{x}|\mathbf{u}} = 1$$

for any parent instantiation $\mathbf{u}$. Therefore, if we change either of these parameters, we must also change the other parameter to ensure that their sum continues to be 1. Now let $\tau_{x|\mathbf{u}}$ be a *meta parameter* such that:

$$\begin{aligned} \theta_{x|\mathbf{u}} &= \tau_{x|\mathbf{u}} \\ \theta_{\overline{x}|\mathbf{u}} &= 1 - \tau_{x|\mathbf{u}}. \end{aligned}$$

By changing the meta parameter $\tau_{x|\mathbf{u}}$, we are then simultaneously changing both parameters $\theta_{x|\mathbf{u}}$ and $\theta_{\overline{x}|\mathbf{u}}$ in a consistent way. Given this new tool, let us consider the following result, which provides a bound on the partial derivative of query $\mathsf{Pr}(\mathbf{y}|\mathbf{e})$ with respect to meta parameter $\tau_{x|\mathbf{u}}$:

$$\left|\frac{\partial \mathsf{Pr}(\mathbf{y} \mid \mathbf{e})}{\partial \tau_{x|\mathbf{u}}}\right| \leq \frac{\mathsf{Pr}(\mathbf{y} \mid \mathbf{e})(1 - \mathsf{Pr}(\mathbf{y} \mid \mathbf{e}))}{\mathsf{Pr}(x \mid \mathbf{u})(1 - \mathsf{Pr}(x \mid \mathbf{u}))}. \tag{4.2}$$

Note that the above bound is independent of the Bayesian network under consideration; that is, it applies to any Bayesian network regardless of its structure and parameterization. The plot of this bound against the current value of the meta parameter, $\Pr(x|\mathbf{u})$, and the current value of the query, $\Pr(\mathbf{y}|\mathbf{e})$, is shown in Figure 4.20. A number of observations are in order about this plot:

1. The bound approaches infinity for extreme values of parameter $\Pr(x|\mathbf{u})$, and attains its smallest value when $\Pr(x|\mathbf{u}) = .5$. Therefore, changing extreme parameters can have a much more dramatic impact on queries than changing non–extreme parameters.

2. The bound approaches 0 for extreme values of query $\Pr(\mathbf{y}|\mathbf{e})$, and attains its highest value when $\Pr(\mathbf{y}|\mathbf{e}) = .5$. Therefore, changing parameters will have a much less impact on extreme queries than it will have on non–extreme ones.

This means that queries $\mathsf{Pr}(\mathbf{y}|\mathbf{e})$ that have extreme values tend to be robust against small changes in network parameters, where robustness is understood as a small change in the absolute value of query value. This also means that one has to be careful when changing extreme parameters as they can be very influential. Therefore, the worst situation from a robustness viewpoint materializes if one has extreme parameters and non–extreme queries. Moreover, the best situation from a robustness viewpoint materializes if one has non–extreme parameters and extreme queries.

The bound in 4.2 can be used to show an even more specific relationship between parameters and queries. Specifically, let $O(x|\mathbf{u})$ denote the odds of variable X given its parents $\mathbf{Y}$:

$$O(x|\mathbf{u}) = \mathsf{Pr}(x|\mathbf{u})/(1 - \mathsf{Pr}(x|\mathbf{u})),$$

and let $O(\mathbf{y}|\mathbf{e})$ denote the odds of variables $\mathbf{Y}$ given evidence $\mathbf{E}$:

$$O(\mathbf{y}|\mathbf{e}) = \mathsf{Pr}(\mathbf{y}|\mathbf{e})/(1 - \mathsf{Pr}(\mathbf{y}|\mathbf{e})).$$

Let $O'(x|\mathbf{u})$ and $O'(\mathbf{y}|\mathbf{e})$ denote these odds after having applied an arbitrary change to the meta parameter $\tau_{x|\mathbf{u}}$. We then have:

$$|\ln(O'(\mathbf{y}|\mathbf{e})) - \ln(O(\mathbf{y}|\mathbf{e}))| \leq \; |\ln(O'(x|\mathbf{u})) - \ln(O(x|\mathbf{u}))|. \tag{4.3}$$

The above inequality allows us to bound the amount of change in a query value using only the amount of change we applied to a parameter, without requiring any information about the Bayesian network under consideration. This inequality can be very useful in practice as it allows one to assess the impact of a parameter change on some query very efficiently (in constant time).

Consider for example the screen shot in Figure 4.3 on Page 5, where the evidence $\mathbf{e}$ indicates a patient with positive X–ray and no dyspnoea. The belief in this patient visiting Asia, $\mathsf{Pr}(A{=}\mathsf{yes}|\mathbf{e})$, is about 1.17% in this case. Moreover, the belief in him having cancer, $\mathsf{Pr}(C{=}\mathsf{yes}|\mathbf{e})$, is about 25.23%. Consider now the parameter $\theta_{C=\mathsf{yes}|S=\mathsf{yes}}$, which represents our local belief in cancer given smoking. This parameter is currently set to 10% and we would like to change its value to 5%. Our interest here is in assessing the impact of this change on the probability of having visited Asia. Using the bound in 4.3, we have:

$$|\ln(\frac{p}{1-p}) - \ln(\frac{1.17}{98.83})| \leq \; |\ln(\frac{5}{95}) - \ln(\frac{10}{90})|,$$

where $p = \mathsf{Pr}'(A{=}\mathsf{yes}|\mathbf{e})$ is the new probability in a visit to Asia after having changed the parameter $\theta_{C=\mathsf{yes}|S=\mathsf{yes}}$ from 10% to 5%. Solving for p, we get:

$$.56\% \leq \mathsf{Pr}'(A{=}\mathsf{yes}|\mathbf{e}) \leq 2.44\%$$

If we actually change the parameter and perform exact inference, we find that the exact value of $\mathsf{Pr}'(A{=}\mathsf{yes}|\mathbf{e})$ is 1.19%, which is within the bound as expected.

We can use the same technique to bound the change in our belief in cancer after the same parameter change, which gives:

$$13.78\% \leq \mathsf{Pr}'(C{=}\mathsf{yes}|\mathbf{e}) \leq 41.60\%$$

Note that the bound is much wider in this case, since the query under consideration is much less extreme.

Consider now the parameter $\theta_{B=\mathsf{yes}|S=\mathsf{yes}}$ which represents our local belief in bronchitis given smoking. This parameter is currently set to 60% and we would like to reduce it to 50%. We now have the following bounds for query change:

$$.78\% \leq \mathsf{Pr}'(A{=}\mathsf{yes}|\mathbf{e}) \leq 1.74\%$$

$$18.36\% \leq \mathsf{Pr}'(C{=}\mathsf{yes}|\mathbf{e}) \leq 33.61\%$$

These bounds illustrate that a bigger change in a less extreme parameter can lead to a tighter bound on global belief change.

We can also use the bound in 4.3 to compute the permissible amount of parameter change which would guarantee a certain bound on the query change. Specifically, suppose that we have a query whose current value is q, and suppose that we are about to change a parameter whose current value is p. To ensure that the new query value stays $\geq q'$, where $q' < q$, the amount of permissible parameter change, δ, is bounded by

$$\frac{p(1-p)(Q-1)}{(1-p)+pQ} \leq \delta \leq \frac{p(1-p)(1-Q)}{p+(1-p)Q}, \tag{4.4}$$

where

$$Q \overset{def}{=} \frac{q'}{1-q'} / \frac{q}{1-q}.$$

Moreover, to ensure that the new query value stays $\leq q'$, where $q' > q$, the amount of permissible parameter change, δ, is bounded by

$$\frac{p(1-p)(1-Q)}{p+(1-p)Q} \leq \delta \leq \frac{p(1-p)(Q-1)}{(1-p)+pQ}. \tag{4.5}$$

Figure 4.21 plots the bound in 4.4 for different initial values, q, of the query. The figure shows the permissible amount of change δ that can be applied to an arbitrary parameter, whose current value is p, while ensuring that the query value will not lessen by more than σ%: $q' = q - q\sigma$.

We close this section by emphasizing that the bound in 4.3 and its derivatives assume that the parameter we are changing concerns a variable with only two values. These bounds do have generalizations to non–binary variables, but we defer the discussion of these generalization until later chapters where we discuss sensitivity analysis in greater technical depth.

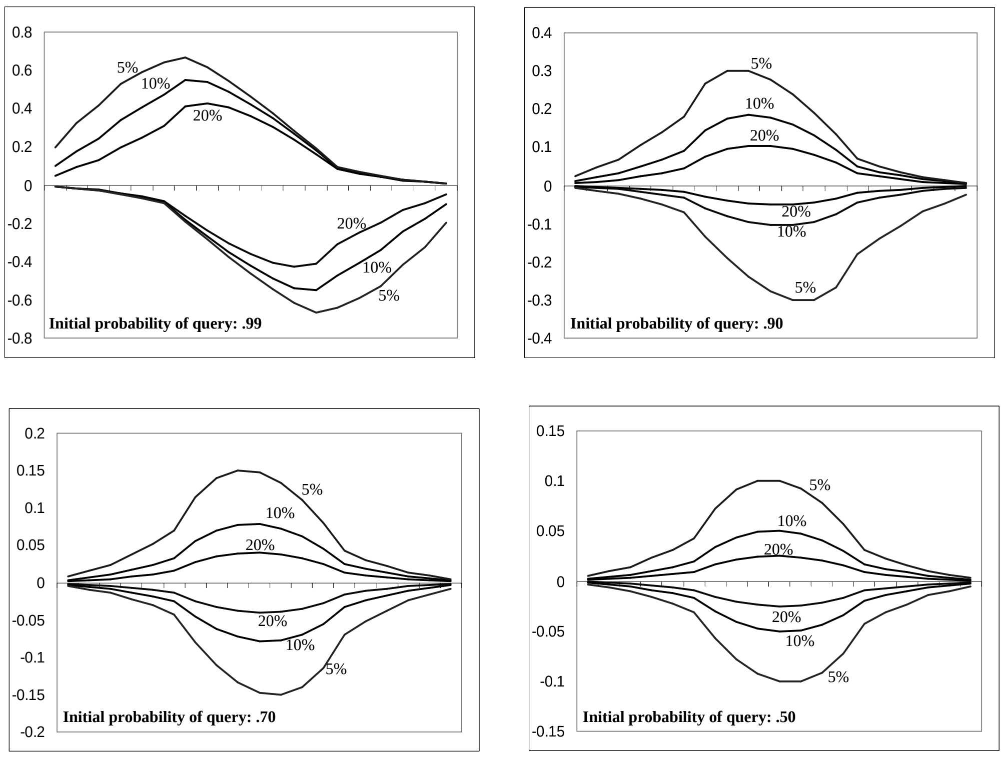


Figure 4.21: The y–axis of each curve represents the maximum and minimum amount of change δ that can be applied to a parameter, while ensuring that the query value will not lessen by more than $\sigma\%$, for $\sigma = 5\%$, 10%, and 20%. The x–axis of each curve represents the initial value of the parameter we are changing.