SAMPLE CHAPTER

Practical Probabilistic Programming

Avi Pfeffer

FOREWORD BY Stuart Russell

MANNING

Practical Probabilistic Programming

by Avi Pfeffer

Chapter 9

brief contents

9 The three rules of probabilistic inference

This chapter covers

- Three important rules for working with probabilistic models:
  - The chain rule, which lets you build complex models out of simple components
  - The total probability rule, which lets you simplify a complex probabilistic model to answer simple queries
  - Bayes' rule, with which you can draw conclusions about causes from observations of their effects
- The basics of Bayesian modeling, including how to estimate model parameters from data and use them to predict future cases

In part 2 of this book, you learned all about writing probabilistic programs for a variety of applications. You know that probabilistic programming systems use inference algorithms operating on these programs to answer queries, given evidence. How do they do that? That's what this part of the book is all about. It's important that you know about this, so you can design models and choose algorithms that support fast and accurate inference.

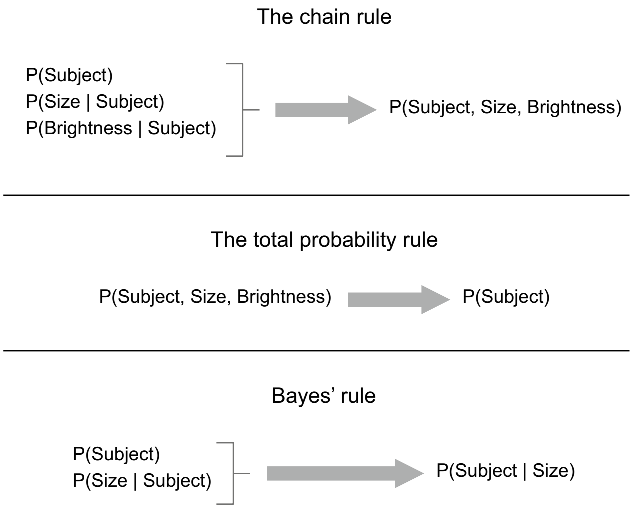


Figure 9.1 Inputs and output of each of the three rules of probabilistic inference. The chain rule lets you turn a set of conditional probability distributions into a joint probability distribution. The total probability rule lets you take a joint probability distribution over a set of variables and produce a distribution over a single variable. Bayes' rule lets you "invert" a conditional probability distribution over an effect, given a cause, into a conditional probability distribution over the cause, given the effect.

This chapter begins with the basics of inference: the three rules of probabilistic inference. The inputs and output of each of the three rules is summarized in figure 9.1:

- First you'll learn about the chain rule, which lets you go from simple (local conditional probability distributions over individual variables) to complex (a full joint probability distribution over all variables).
- The total probability rule, described in section 9.2, goes from complex (a full joint distribution) back to simple (a distribution over a single variable).
- Finally, Bayes' rule, described in section 9.3, is probably the most famous rule of inference. Bayes' rule lets you "flip" the direction of the dependencies, turning a conditional distribution over an effect, given a cause, into a distribution over a cause, given an effect. Bayes' rule is essential to incorporating evidence, which is often an observation of an effect, and inferring a cause.

These three rules of inference can be used to answer queries.

Before diving into the new material, let's recap some definitions from chapter 4 that you need in this chapter:

- *Possible worlds*—All states you consider possible
- *Probability distribution*—An assignment of a probability between 0 and 1 to each possible world, such that all of the probabilities add up to 1
- *Prior probability distribution*—The probability distribution before seeing any evidence
- *Conditioning on the evidence*—The process of applying evidence to a probability distribution

- *Posterior probability distribution*—The probability distribution after seeing the evidence; the result of conditioning
- *Conditional probability distribution*—Rule that specifies a probability distribution over one variable for every combination of values of some other variables
- *Normalizing*—The process of proportionally adjusting a set of numbers so they add up to 1

NOTE For each of the rules, a sidebar presents generic mathematical definitions. These are useful if you want a deeper understanding; and if you're comfortable with the mathematical notation, this more abstract discussion can help cement the principles. If not, feel free to skip these sidebars. The main thing is that you understand why and how the rule is used.

9.1 The chain rule: building joint distributions from conditional probability distributions

As you may recall, chapter 4 covered how a probabilistic model defines a probability distribution over possible worlds, as well as the ingredients of probabilistic models: variables, dependencies, functional forms, and numerical parameters. I hinted that the chain rule is the essential mechanism that turns these ingredients into a probability distribution over possible worlds. I promised you a full discussion of the chain rule in part 3, and now it's time for that discussion.

How does the chain rule define a probability distribution over possible worlds? In other words, *how does it specify a number between 0 and 1 for each possible world?* Let's revisit our Rembrandt example from chapter 4. Let's start with the variables Subject, Size, and Brightness, and assume you're given the dependency model where Size and Brightness both depend on Subject but not on each other. You're also given a specification of a probability distribution over Subject; a CPD of Size, given Subject; and a CPD of Brightness, given Subject. These ingredients are summarized in figure 9.2. (For Subject, which doesn't depend on anything, I use the same CPD table notation as the other variables, except that there are no conditioning variables and only one row.)

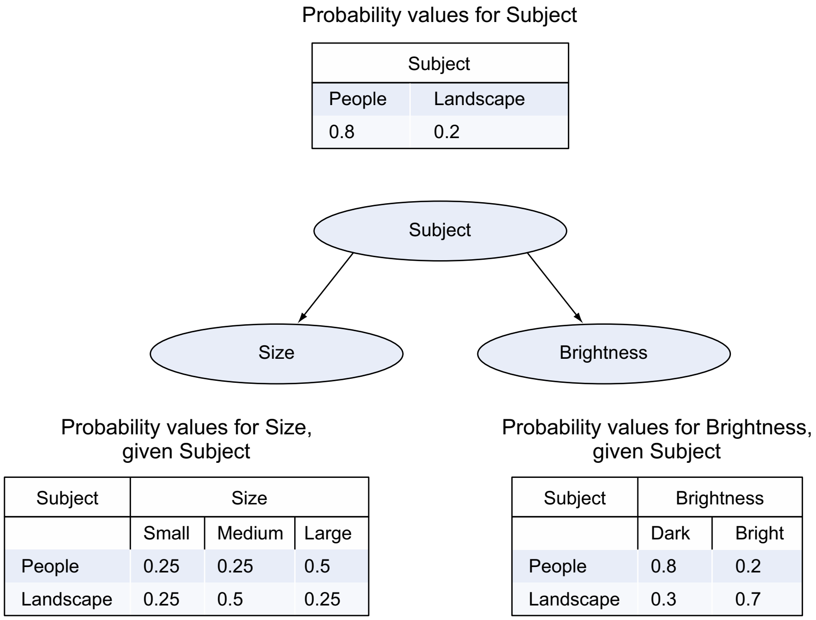
Now, a possible world specifies a value for each of the variables Subject, Size, and Brightness. *How do you get the probability of a possible world?* For example, what's P(Subject = People, Size = Large, Brightness = Dark)? According to the chain rule, this is simple. *You find the correct entries in the CPD tables and multiply them together.* So, in this case,

P(Subject = People, Size = Large, Brightness = Dark) =

P(Subject = People) × P(Size = Large | Subject = People) × P(Brightness = Dark | Subject = People) =

0.8 × 0.5 × 0.8 =

0.32



Probability values for Subject

Subject	
People	Landscape
0.8	0.2

Probability values for Size, given Subject

Subject	Size		
	Small	Medium	Large
People	0.25	0.25	0.5
Landscape	0.25	0.5	0.25

Probability values for Brightness, given Subject

Subject	Brightness	
	Dark	Bright
People	0.8	0.2
Landscape	0.3	0.7

Figure 9.2 Bayesian network structure and CPDs for the chain rule example

You can use the same formula for all possible values of Subject, Size, and Brightness, and get the result shown in table 9.1. This result is called a *joint probability distribution* over Subject, Size, and Brightness, because it specifies the probability of each joint value of these three variables.

Table 9.1 Joint probability distribution resulting from applying the chain rule to the CPDs in figure 9.1. You multiply P(Subject) by P(Size | Subject) and P(Brightness | Subject). The probabilities add up to 1.

Subject	Size	Brightness	Probability
People	Small	Dark	0.8 × 0.25 × 0.8 = 0.16
People	Small	Bright	0.8 × 0.25 × 0.2 = 0.04
People	Medium	Dark	0.8 × 0.25 × 0.8 = 0.16
People	Medium	Bright	0.8 × 0.25 × 0.2 = 0.04
People	Large	Dark	0.8 × 0.5 × 0.8 = 0.32
People	Large	Bright	0.8 × 0.5 × 0.2 = 0.08
Landscape	Small	Dark	0.2 × 0.25 × 0.3 = 0.015
Landscape	Small	Bright	0.2 × 0.25 × 0.7 = 0.035
Landscape	Medium	Dark	0.2 × 0.5 × 0.3 = 0.03

Table 9.1 Joint probability distribution resulting from applying the chain rule to the CPDs in figure 9.1. You multiply P(Subject) by P(Size | Subject) and P(Brightness | Subject). The probabilities add up to 1.

Subject	Size	Brightness	Probability
Landscape	Medium	Bright	0.2 × 0.5 × 0.7 = 0.07
Landscape	Large	Dark	0.2 × 0.25 × 0.3 = 0.015
Landscape	Large	Bright	0.2 × 0.25 × 0.7 = 0.035

The truth is, I've cheated a little bit. The standard chain rule for three variables says that for the third variable, you need to condition its probability on both the first two variables. So rather than

P(Subject = People, Size = Large, Brightness = Dark) =

P(Subject = People) × P(Size = Large | Subject = People) × P(Brightness = Dark | Subject = People)

you should be computing

P(Subject = People, Size = Large, Brightness = Dark) =

P(Subject = People) × P(Size = Large | Subject = People) × P(Brightness = Dark | Subject = People, Size = Large)

That would be the officially correct statement of the chain rule. But I'm taking advantage of specific knowledge I have about the dependencies, namely that Brightness doesn't depend on Size, only Subject. Brightness is conditionally independent of Size, given Subject:

P(Brightness = Dark | Subject = People, Size = Large) =

P(Brightness = Dark | Subject = People)

So I can legitimately simplify the chain rule the way I have. Anytime you have a Bayesian network and want to use the chain rule to define the full probability distribution over all variables, you can *always* simplify the rule so that each variable depends only on its parents in the network. On the other hand, if Brightness wasn't conditionally independent of Size, given Subject, you'd have to use the longer form. Bayesian networks and the chain rule go hand in hand. A Bayesian network specifies exactly the form of the chain rule to use in building up the joint distribution.

That's all there is to the chain rule, a simple but crucial rule in probabilistic modeling. The chain rule is essential to understanding not only Bayesian networks but also generative models in general. Because probabilistic programs are encodings of generative models, now that you understand the chain rule, you have a fundamental understanding of the probabilistic model defined by a probabilistic program.

The generic chain rule

This chain rule is a generic principle that applies to any dependency model and any set of CPDs for the variables, in whatever functional form. As long as each variable has a CPD that specifies a probability distribution over its values for any possible values of the variables it depends on, you can get the probability of any joint assignment to all variables by multiplying the correct numbers in the CPDs.

In mathematical notation, you start with two variables X and Y, such that Y depends on X. You're given $P(X)$, a probability distribution over the values of X, and $P(Y \mid X)$, the CPD of Y given X. The chain rule takes these two ingredients $P(X)$ and $P(Y \mid X)$, and turns them into a probability distribution $P(X,Y)$ over X and Y jointly. For every possible value x of X and y of Y, the chain rule is defined by this simple formula:

$$P(X = x, Y = y) = P(X = x)P(Y = y \mid X = x)$$

> **NOTATION ALERT** It's standard practice to use uppercase letters like X and Y to represent variables, and lowercase letters like x and y for values.

You have a convenient way to indicate that this formula holds for *every* value x and y:

$$P(X,y) = P(X)P(Y \mid X)$$

This easy-to-remember formula is shorthand for many formulas about specific values x and y.

What if you have more than two variables? The chain rule generalizes to any number of variables. Suppose you have variables X_1, X_2, …, X_n. In the standard statement of the chain rule, you don't make any independence assumptions, so each variable depends on all variables that precede it. The full chain rule, using shorthand notation, is as follows:

$$P(X_1,X_2,\ldots X_n) = P(X_1)P(X_2 \mid X_1)P(X_3 \mid X_1,X_2)\ldots P(X_n \mid X_1,X_2,\ldots X_{n-1})$$

Let's see what this formula says. It says that to get the joint probability distribution over X_1, X_2,…X_n, you start with X_1 and get its probability, and then you look at X_2, which depends on X_1, and get its appropriate probability out of its CPD. Then you get the probability of X_3, which depends on X_1 and X_2, from its CPD, continuing recursively until finally, you get the appropriate probability of X_n, which depends on all previous variables, from its CPD. This formula, by the way, is the reason for the name *chain rule.* You compute a joint probability distribution from a chain of conditional distributions.

Our multivariable chain rule formula didn't make any assumptions about dependencies, especially not independence relationships. Adding independence information can significantly simplify the formula. Instead of including all previous variables on the right-hand side of the "|" for a given variable, you need to include only the variables it directly depends on in the dependency model. For example, let's consider the three variables Subject, Size, and Brightness. If you were to follow formula (3), you'd get

P(Subject, Size, Brightness) = P(Subject) P(Size | Subject) P(Brightness | Subject, Size)

(continued)

But according to the network, Brightness doesn't depend on Size, only Subject. So you can simplify this formula to

P(Subject, Size, Brightness) = P(Subject) P(Size | Subject) P(Brightness | Subject)

And indeed, this is the formula used to compute table 9.1.

9.2 The total probability rule: getting simple query results from a joint distribution

The chain rule lets you build a joint distribution out of simple CPDs, say, a joint distribution over Subject, Size, and Brightness. Typically, your query will be about a particular variable or small number of variables. For example, you might want to infer the identity of a painter based on observations of a painting. Suppose you have a joint distribution over all of the variables. How do you get a probability distribution over a single variable? The principle is simple: the probability of any value of the variable is equal to the sum of probabilities of all joint assignments to all variables that are consistent with that value.

You already saw this basic principle in chapter 4: *the probability of any fact is the sum of the probabilities of possible worlds consistent with that fact*. So to get the probability that Subject = Landscape, you add the probabilities of all possible worlds consistent with Subject = Landscape. Because each world consists of a joint assignment of values to all variables, including Subject, you look for the worlds in which the value assigned to Subject is Landscape. This simple principle usually goes by the fancy name of the *law of total probability*, but I prefer to call it the more mundane *total probability rule*.

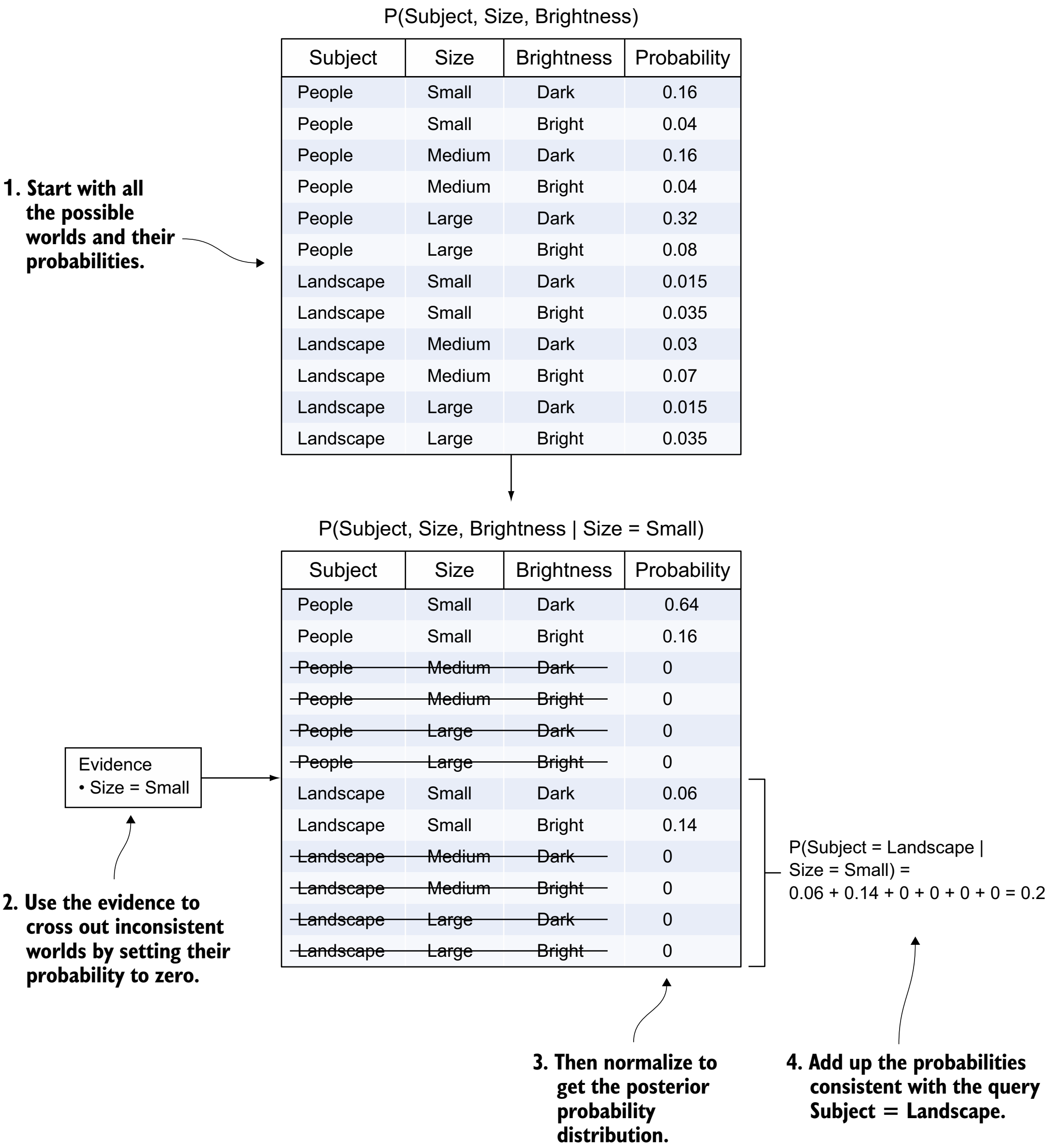
The use of the total probability rule is illustrated in figure 9.3. You start with the prior probability distribution shown in the top of the figure. You then condition on the evidence that Size = Small to obtain the posterior distribution in the middle. You use the usual two steps: first, you cross out all assignments of values to the variables inconsistent with the evidence that Size = Small, and then you normalize the remaining probabilities so that they sum to 1. On the bottom of the figure, you use the total probability rule to compute the probability that the painting is a landscape, given the evidence. You add the probabilities of all rows in which the value of the Subject variable is Landscape.

Notice how the posterior probability of a row in which the value of Size is anything other than Small is 0. This is always the case, because you cross out worlds inconsistent with the evidence and set their probability to 0. So, in fact,

P(Subject = Landscape, Brightness = Dark | Size = Small)

is equal to

P(Subject = Landscape, Brightness = Dark, Size = Small | Size = Small)



P(Subject, Size, Brightness)

Subject	Size	Brightness	Probability
People	Small	Dark	0.16
People	Small	Bright	0.04
People	Medium	Dark	0.16
People	Medium	Bright	0.04
People	Large	Dark	0.32
People	Large	Bright	0.08
Landscape	Small	Dark	0.015
Landscape	Small	Bright	0.035
Landscape	Medium	Dark	0.03
Landscape	Medium	Bright	0.07
Landscape	Large	Dark	0.015
Landscape	Large	Bright	0.035

1. Start with all the possible worlds and their probabilities.

P(Subject, Size, Brightness | Size = Small)

Subject	Size	Brightness	Probability
People	Small	Dark	0.64
People	Small	Bright	0.16
~~People~~	~~Medium~~	~~Dark~~	0
~~People~~	~~Medium~~	~~Bright~~	0
~~People~~	~~Large~~	~~Dark~~	0
~~People~~	~~Large~~	~~Bright~~	0
Landscape	Small	Dark	0.06
Landscape	Small	Bright	0.14
~~Landscape~~	~~Medium~~	~~Dark~~	0
~~Landscape~~	~~Medium~~	~~Bright~~	0
~~Landscape~~	~~Large~~	~~Dark~~	0
~~Landscape~~	~~Large~~	~~Bright~~	0

Evidence
• Size = Small

P(Subject = Landscape | Size = Small) = 0.06 + 0.14 + 0 + 0 + 0 + 0 = 0.2

2. Use the evidence to cross out inconsistent worlds by setting their probability to zero.

3. Then normalize to get the posterior probability distribution.

4. Add up the probabilities consistent with the query Subject = Landscape.

Figure 9.3 Using the total probability rule to answer a query. The total probability of a value of a variable is the sum of probabilities of all joint assignments consistent with the value.

You can see that P(Subject = Landscape | Size = Small), which is the sum of two rows in the middle table in figure 9.3, can be expressed by the following summation:

$$P(\text{Subject} = \text{Landscape} \mid \text{Size} = \text{Small}) =$$

$$P(\text{Subject} = \text{Landscape}, \text{Brightness} = \text{Dark} \mid \text{Size} = \text{Small}) + P(\text{Subject} = \text{Landscape}, \text{Brightness} = \text{Bright} \mid \text{Size} = \text{Small})$$

A concise way of writing this summation uses the Greek letter Σ, which is the standard mathematical notation for addition:

$$P(\text{Subject} = \text{Landscape} \mid \text{Size} = \text{Small}) =$$

$$\Sigma_b\, P(\text{Subject} = \text{Landscape}, \text{Brightness} = b \mid \text{Size} = \text{Small}) \quad (1)$$

On the right-hand side of this equation, b stands for any possible value of Brightness, and Σ_b means you add the following terms for all possible values of Brightness. We say we're "summing out" Brightness. Now, formula (1) holds for any possible values of Subject and Size, so you can use the snappy shorthand from the previous section:

$$P(\text{Subject} \mid \text{Size}) = \Sigma_b\, P(\text{Subject}, \text{Brightness} = \text{b} \mid \text{Size})$$

The generic total probability rule

Now that you've seen the total probability rule applied to our example, you're ready to see the general mathematical definition. It's the same simple principle, but the notation is a little messier. You have a joint probability distribution over a set of variables, and you want to sum out some of those variables to get a distribution over the other variables. For example, you might have a joint distribution over Color, Brightness, Width, and Height. You want a distribution over Color and Brightness and to sum out Width and Height. Now, your joint distribution over all of the variables may be conditioned on some other set of variables, such as Rembrandt and Subject. To keep the formulas short, you'll use the initial of each variable. Also, let's assume that the possible values of Width and Height are small and large. According to the total probability rule

$$P(C = \text{yellow}, B = \text{bright} \mid R = \text{true}, S = \text{landscape}) =$$

$$P(C = \text{yellow}, B = \text{bright}, W = \text{small}, H = \text{small} \mid R = \text{true}, S = \text{landscape}) +$$

$$P(C = \text{yellow}, B = \text{bright}, W = \text{small}, H = \text{large} \mid R = \text{true}, S = \text{landscape}) +$$

$$P(C = \text{yellow}, B = \text{bright}, W = \text{large}, H = \text{small} \mid R = \text{true}, S = \text{landscape}) +$$

$$P(C = \text{yellow}, B = \text{bright}, W = \text{large}, H = \text{large} \mid R = \text{true}, S = \text{landscape})$$

(continued)

You can write this in mathematical notation. Let's call the variables that you want the distribution $X_1,\ldots,X_n$, and the variables to be summed out $Y_1,\ldots,Y_m$. Let's call the variables you're conditioning on $Z_1,\ldots,Z_l$. The total probability rule says that for any values $x_1,\ldots,x_n$ of $X_1,\ldots,X_n$ and $z_1,\ldots,z_l$ of $Z_1,\ldots,Z_l$:

$$\mathrm{P}(X_1 = x_1,\ldots,X_n = x_n \mid Z_1 = z_1,\ldots,Z_1 = z_l) =$$

$$\Sigma_{y1}\ \Sigma_{y2}\ldots\Sigma_{y3}\ \mathrm{P}(X_1 = x_1,\ldots,X_n = x_n,Y_1 = y_1,\ldots,Y_m = y_m \mid Z_1 = z_1,\ldots,Z_1 = z_1)$$

All this is saying is that to get the conditional probability that the target variables $X_1,\ldots,X_n$ have values $x_1,\ldots,x_n$, you take the sum of all cases in the full conditional distribution over all variables in which the values of the target variables match the values $x_1,\ldots,x_n$.

Because this formula holds for all values $x_1,\ldots,x_n$ and $z_1,\ldots,z_l$, you can use the same shorthand as in section 2.1 and write

$$\mathrm{P}(X_1,\ldots,X_n \mid Z_1,\ldots,Z_1) = \Sigma_{y1}\ \Sigma_{y2}\ldots\Sigma_{ym}\ \mathrm{P}(X_1,\ldots,X_n,\ Y_1 = y_1,\ldots,Y_m = y_m \mid Z_1,\ldots,Z_1)$$

Another notational trick makes the total probability rule an easy formula to remember. If you have a set of variables $X_1,\ldots,X_n$, you can summarize them all with a boldface X. So X is shorthand for $X_1,\ldots X_n$. Likewise, you can use a bold lowercase x as shorthand for the values $x_1,\ldots,x_n$.

> **NOTATION ALERT** It's common to use nonbold, italic letters like X and x for individual variables or values, and boldface X and x for sets of variables or values.

So for specific values x and z, you have

$$\mathrm{P}(\boldsymbol{X} = \boldsymbol{x} \mid \boldsymbol{Z} = \boldsymbol{z}) = \Sigma_y\,\mathrm{P}(\boldsymbol{X} = \boldsymbol{x},\boldsymbol{Y} = \boldsymbol{y} \mid \boldsymbol{Z} = \boldsymbol{z})$$

Generalizing over all values x and z, you finally get the pithy formula

$$\mathrm{P}(\boldsymbol{X} \mid \boldsymbol{Z}) = \Sigma_y\mathrm{P}(\boldsymbol{X},\boldsymbol{Y} = \boldsymbol{y} \mid \boldsymbol{Z})$$

This summarizes the total probability rule.

There's a technical term you might encounter when you start with a joint distribution over a set of variables and you sum out some of the variables to get a probability distribution over the remaining variables. This resulting distribution is called the *marginal distribution* over the remaining variables, and the process of summing out variables to get the marginal distribution over other variables is called *marginalization*. Most typically, you sum out all but one of the variables and end up with the marginal distribution over a single variable.

Now you've covered two of the three rules of probabilistic inference. Let's turn our attention to the last, and possibly the most interesting one.

9.3 Bayes' rule: inferring causes from effects

The final piece of the puzzle in reasoning about probabilistic models is Bayes' rule, named after Rev. Thomas Bayes, an eighteenth-century mathematician who first discovered how to infer knowledge about causes from observations about effects. *Bayes' rule lets you compute the conditional probability of a cause, given its effect, by combining the prior probability of the cause (before you know anything about the effect) and the probability of the effect, given the cause.*

9.3.1 Understanding, cause, effect, and inference

Bayes' rule is related to the notions of cause and effect, which in turn are related to the dependencies in your model. In an ordinary program, when one variable *X* uses the value of another variable *Y* in its definition, changing the value of *Y* can result in a change to *X*. So, in a sense, *Y* is a cause of *X*. In the same way, if you're building a probabilistic model where *X* depends on *Y*, *Y* is often a cause of *X*. For example, consider Subject and Brightness. You modeled Brightness as depending on Subject, and typically a painter might decide what type of painting to paint before deciding how bright it should be. So in this sense, Subject is a cause of Brightness.

I'm using the word *cause* a little loosely here. A more accurate description is to say that you're modeling the *generative process* of the data. In this process, you imagine the painter first choosing the subject, and then based on that, choosing the brightness. So the painter first generates a value for the Subject variable, which then gets passed to the generation of the value of the Size variable. When a model follows a generative process, you loosely use the words *cause* and *effect* when the value of one variable is being used by another.

Figure 9.4 shows a slightly more elaborate example of a generative process, described by a Bayesian network. In this example, the first variable that gets generated is whether the painting is by Rembrandt or not, because the identity of the painter influences everything about the painting. Then the painter chooses the Subject, which in turn helps determine the Size and Brightness. The reason Size depends on both Rembrandt and Subject is that landscapes by different painters might tend to have different sizes; similarly for Brightness.

The right-hand side of figure 9.4 makes an important point. Although the generative process follows the arrows in the model, inference about the model can go in any direction. In fact, in this example, our goal is to decide whether the painting is a Rembrandt, so the inference will go in the opposite direction from the generative process. I've emphasized this point throughout the book; the direction of the arrows in the network isn't necessarily the direction in which you expect to do inference. Don't let the way you typically reason about a domain (for example, "I look at the painting's brightness to decide who the artist is") guide the way you structure the network. Instead, think about the generative process. In most cases, following the generative process results in the simplest and clearest model. You can infer in any direction you want.

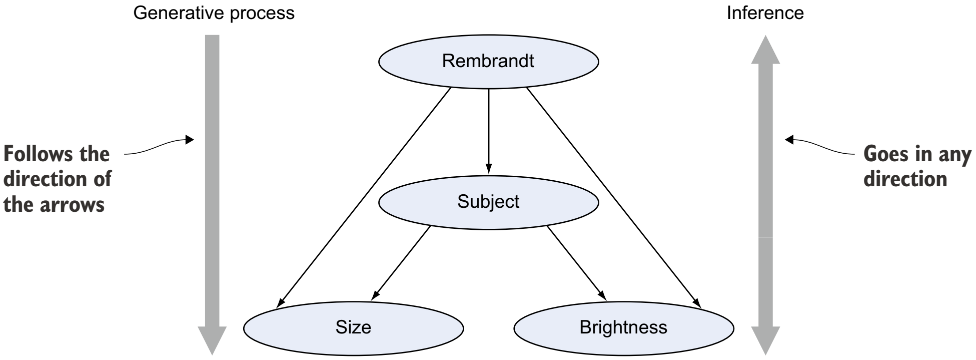
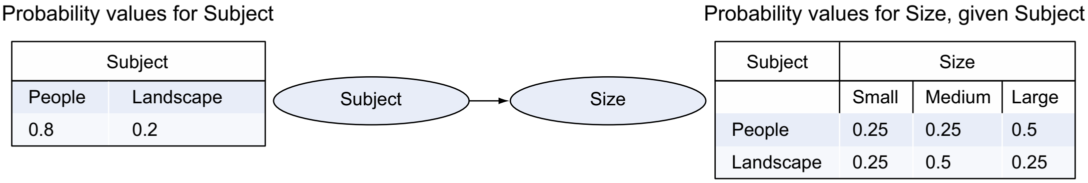


Figure 9.4 Network arrows often follow the generative process, but inference can be in any direction.

Okay, so I've said you can infer in the opposite direction from the arrows in the network. How do you do that? Bayes' rule is the answer! Let's look at the two-variable example in figure 9.5. Here, the network follows the natural generative process in which Subject determines Size. You're given, as ingredients, P(Subject) and P(Size | Subject). First, let's think about inference in the forward direction, following the generative process. Suppose you observe that Subject = Landscape, and you want to query the posterior probability of Size. You can get it directly from P(Size | Subject). If you want to infer an effect from evidence about a cause, you have that information immediately available.



Probability values for Subject

Subject	
People	Landscape
0.8	0.2

Probability values for Size, given Subject

Subject	Size		
	Small	Medium	Large
People	0.25	0.25	0.5
Landscape	0.25	0.5	0.25

Figure 9.5 Two-variable model for the Bayes' rule example

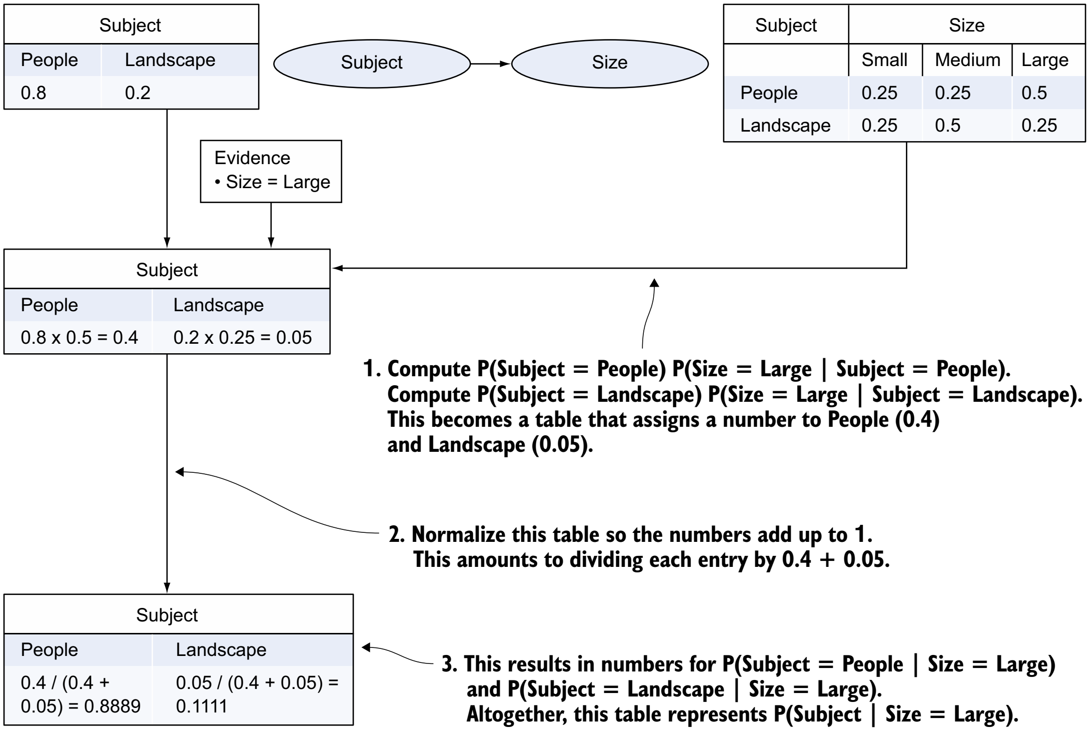
But often you observe evidence about an effect, and you want to infer something about a possible cause of that effect. You want to invert the model, because you want to get P(Subject | Size), which is the probability of the cause, given the effect. Bayes' rule makes this possible.

9.3.2 Bayes' rule in practice

The operation of Bayes' rule is simple. I'll show how it works first and then explain each of the steps. The full process is illustrated in figure 9.6. You start with the model from figure 9.5. Then you observe evidence that Size = Large. You want to compute a posterior probability distribution over Subject given this evidence—you want to compute P(Subject | Size = Large). Here's what you do:

1. Calculate P(Subject = People) P(Size = Large | Subject = People) = 0.8 × 0.5 = 0.4, and P(Subject = Landscape) P(Size = Large | Subject = Landscape) = 0.2 × 0.25 = 0.05. These numbers are shown in the middle table of figure 9.6.
2. Normalize this table to get the answer you want. The normalizing factor is 0.4 + 0.05 = 0.45. So, P(Subject = People | Size = Large) = 0.4 / 0.45 = 0.8889, and P(Subject = Landscape | Size = Large) = 0.05 / 0.45 = 0.1111. This answer is shown in the bottom table of figure 9.6.

Now, why does this work? You have the ingredients for this process in the chain rule and the total probability rule. You're going to construct the joint probability distribution from the CPD ingredients by using the chain rule, as you learned in section 9.1. Then you'll apply the chain rule again, this time in the opposite direction.



Subject	
People	Landscape
0.8	0.2

Subject	Size		
	Small	Medium	Large
People	0.25	0.25	0.5
Landscape	0.25	0.5	0.25

Evidence
• Size = Large

Subject	
People	Landscape
0.8 x 0.5 = 0.4	0.2 x 0.25 = 0.05

**1. Compute P(Subject = People) P(Size = Large | Subject = People).
Compute P(Subject = Landscape) P(Size = Large | Subject = Landscape).
This becomes a table that assigns a number to People (0.4)
and Landscape (0.05).**

**2. Normalize this table so the numbers add up to 1.
This amounts to dividing each entry by 0.4 + 0.05.**

Subject	
People	Landscape
0.4 / (0.4 + 0.05) = 0.8889	0.05 / (0.4 + 0.05) = 0.1111

**3. This results in numbers for P(Subject = People | Size = Large)
and P(Subject = Landscape | Size = Large).
Altogether, this table represents P(Subject | Size = Large).**

Figure 9.6 Bayes' rule in operation

Finally, you'll use the total probability rule to complete the calculation. Here are the steps:

1. Take P(Subject) and P(Size | Subject) and apply the chain rule to get P(Subject, Size) = P(Subject) P(Size | Subject).
2. Using the chain rule, but breaking things up in the inverse direction, you can write P(Size, Subject) = P(Size) P(Subject | Size).
3. Because P(Subject, Size) and P(Size, Subject) are equal, you can put 1 and 2 together to get P(Size) P(Subject | Size) = P(Subject) P(Size | Subject).
4. Divide by P(Size) on both sides of this equation to get

$$\mathrm{P(Subject|Size)} = \frac{\mathrm{P(Subject)P(Size|Subject)}}{\mathrm{P(Size)}}$$

You now have the answer to your query, P(Subject | Size) on the left-hand side. This is the formula typically referred to as Bayes' rule, but it's not yet in usable form, because it includes P(Size), which you don't have, so there's one more step.

1. Use the total probability rule and the chain rule to express P(Size) in terms you know. First, use the total probability rule to write P(Size) = Σ_s P(Subject = *s*, Size). Then, use the chain rule to write P(Subject = *s*, Size) = P(Subject = *s*) P(Size | Subject = *s*). Finally, combine those to get P(Size) = Σ_s P(Subject = *s*) P(Size | Subject = *s*).
2. You get your final answer:

$$\mathrm{P(Subject|Size = Large)} = \frac{\mathrm{P(Subject)P(Size = Large|Subject)}}{\Sigma_s \mathrm{P(Subject} = s\mathrm{)P(Size = Large|Subject} = s\mathrm{)}}$$

You can see how this answer relates to the two steps illustrated in figure 9.6. The first step computes the numerator P(Subject) P(Size = Large | Subject) for each of the two possible values of Subject. Now, take a look at the denominator. You add P(Subject = *s*) P(Size | Subject = *s*) for each possible value *s* of Subject. But this is just the quantity you computed in the first step for each value of Subject. The denominator adds together all of the quantities you computed in the first step. So you need to divide each of these quantities by their total. This is another way of saying that you normalize those quantities, which you do in step 2.

Although Bayes' rule is simple, there's more to learn about it. Bayes' rule provides the basis for the Bayesian modeling framework, which is the subject of the next section. That section also goes into more depth on how Bayes' rule works, and provides a sidebar on the generic Bayes' rule.

9.4 Bayesian modeling

Bayes' rule provides the basis for a general approach to modeling, in which you infer knowledge about causes from observations of their effects, and then apply that knowledge to other potential effects.

This section demonstrates Bayesian modeling by using the coin-toss scenario you first encountered in chapter 2. Based on the results of 100 coin tosses (the effects of the model), you'll do the following:

- Use Bayes' rule to infer the bias of the coin (the cause of the effects)
- Demonstrate several methods to predict the result of the 101st coin toss
  - The maximum a posteriori (MAP) method
  - The maximum likelihood estimation (MLE) method
  - The full Bayesian method

Figure 9.7 reproduces the Bayesian network for the example where you're trying to predict the toss of the 101st coin, based on the outcome of the first 100 tosses. You have three variables: the Bias of the coin, the NumberOfHeads in the first 100 tosses, and the outcome of Toss_{101}. Bias is generated first, and then all of the coin tosses depend on Bias. If Bias is known, the coin tosses are independent. Remember, when I say that Bias is generated first, I'm describing the generative process, not that Bias is *known* first. This is yet another example indicating that the order in which variables are generated isn't necessarily the order of inference. In our example, Bias is generated first, but inference goes from NumberOfHeads to Bias.

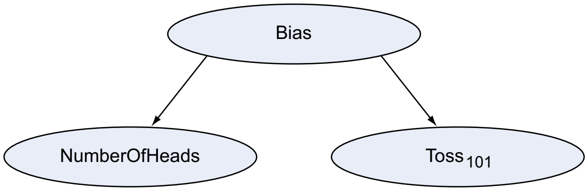


Figure 9.7 Bayesian network for the coin-toss example

You're using the beta-binomial model, so Bias is characterized by a beta distribution, whereas NumberOfHeads is characterized by a binomial distribution that depends on Bias. As a reminder:

- The binomial variable characterizes the number of times a random process comes out a certain way, out of a total number of attempts. In our example, a binomial is used to characterize the number of times a coin toss comes out heads. A binomial variable is parameterized by the probability that each attempt comes out the right way.
- This probability is the bias of the coin. If you knew the bias of the coin, this could be a specific value. But in this scenario, you don't know the bias, and you're trying to estimate it based on the outcomes of the coin tosses. Therefore,

> you model the bias by using a random variable. Specifically, you use the beta distribution, which is a continuous distribution, to model this bias. For a continuous distribution, you use a *probability density function* (PDF) instead of specifying the probability of each value. A beta distribution has two parameters, α and β. α can intuitively be understood as representing the number of heads you've previously seen, plus one. Similarly for β and tails. As mentioned in chapter 4, you use the beta distribution because it works well with the binomial. You'll see why in this section.

The outcome of any future coin toss is given by a `Flip`, in which the probability it comes out heads is equal to Bias. As is implied by the Bayesian network, the future coin toss depends directly only on the bias. If the bias is known, the other coin tosses don't add any information. But if the bias is unknown, the first 100 coin tosses provide information about the bias that can then be used to predict the 101st coin toss.

9.4.1 Estimating the bias of a coin

How do you use this model to predict a future coin toss based on the outcome of the first 100? This is where Bayesian modeling comes in. In Bayesian modeling, you can use Bayes' rule to infer a posterior probability distribution over the bias from observing the number of tosses that came out heads. You can then use this posterior distribution to predict the next toss.

This process is shown in figure 9.8. If you observe thousands of tosses and 40% of them come out heads, you might infer that the bias is probably close to 0.4. If you don't have as many tosses, the inference will be less confident. These inferences are the direct result of applying Bayes' rule. Getting back to our example, if you see 63 heads in 100 coin tosses, you can compute a posterior distribution over Bias given that NumberOfHeads = 63, and then use this to predict Toss_{101}.

To achieve this, you start with a prior distribution for Bias. The beta distribution is characterized by two parameters, α and β. Let's call the parameters of the prior beta

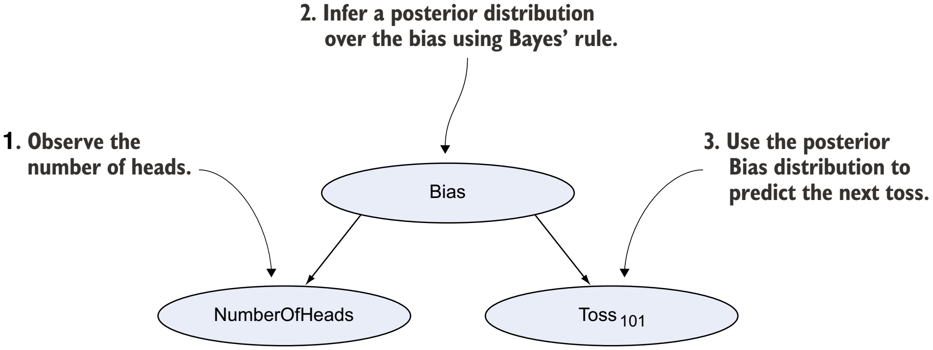

2. Infer a posterior distribution over the bias using Bayes' rule.
1. Observe the number of heads.
3. Use the posterior Bias distribution to predict the next toss.
Bias
NumberOfHeads
Toss 101

Figure 9.8 Order of inference in the biased coin example

distribution for Bias α_0 and β_0. Remember from chapter 2 that α_0 and β_0 represent the number of imaginary heads and tails you've seen prior to observing any real tosses, plus one. To get the posterior distribution, you add the actual number of heads and tails to those imaginary numbers. For example, suppose you start with beta(2, 5). This means that you imagine having seen 1 head and 4 tails (because α_0 is the number of imagined heads plus one, and similarly for β_0). You then observe 63 heads and 37 tails. The posterior distribution over the bias is given by beta(65, 42). If you call the parameters of the posterior beta distribution α_1 and β_1, you have the simple formula

$\alpha_1 = \alpha_0 +$ number of observed heads

$\beta_1 = \beta_0 +$ number of observed tails

NOTE In practice, you don't have to make these calculations yourself. A probabilistic programming system's algorithms will take care of everything for you. You specify that you want to use a beta-binomial model, and it will make all necessary calculations. But it's important that you understand the principles behind how the systems work, which is why you're spending time on it here.

Figure 9.9 shows this beta(65, 42) distribution, superimposed on the original beta(2, 5). You can see a couple of things. First, the peak of the distribution has moved to the right, because the fraction of heads in the actual observations (63 out of 100) is more than in the imaginary observations you started with (1 out of 5). Second, the peak has become sharper. Because you have 100 additional observations, you're much more confident in your assessment of the bias.

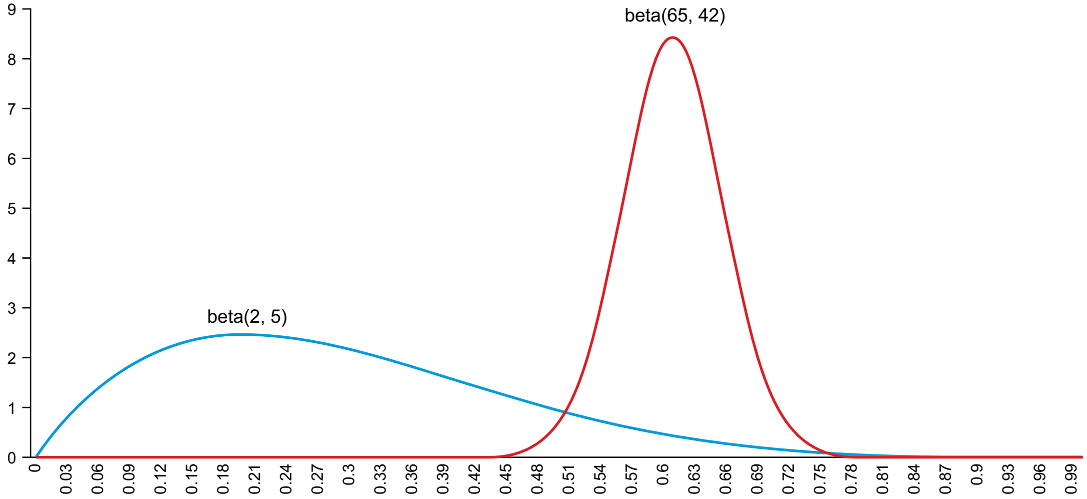


Figure 9.9 Inferring the bias of the coin from a sequence of observations. Here, you've observed 63 heads and 37 tails and added them to the alpha and beta parameters. The posterior PDF beta(65, 42) is superimposed over the prior PDF beta(2, 5).

This simple formula of adding outcomes to create the new beta-binomial model you got for the posterior distribution is the result of applying Bayes' rule. In applying Bayes' rule to the coin-toss example, you deal with three quantities:

- p(Bias = *b*): The prior probability density of the value *b* of Bias. (A lowercase p is used in this notation and below to emphasize that the quantity is a probability density, not a probability.)
- P(NumberOfHeads = 63 | Bias = *b*): The probability of observing that NumberOfHeads = 63, given that the value of Bias is *b*. This probability is known as the *likelihood* of *b* given the data.
- p(Bias = *b* | NumberOfHeads = 63): The posterior probability density of the value *b* of the Bias.

Because this example deals with a continuous variable (the bias), it's slightly more complicated than the example in section 9.3.2 about the painting. I'll repeat the conclusion of that example here, so you can see how it also works for the biased coin example. In section 9.3.2, you got the following expression for the probability distribution over the subject of the painting, given evidence about the size:

$$\text{P(Subject|Size = Large)} = \frac{\text{P(Subject)P(Size = Large|Subject)}}{\Sigma_s \text{P(Subject} = s\text{)P(Size = Large|Subject} = s\text{)}}$$

Let's focus on the denominator. It adds the values of the numerator for all possible values of Subject. It's a normalizing factor that ensures that (a) the left-hand side is always proportional to the numerator on the right-hand side, and (b) the left-hand side values sum to 1. You summarize this formula by using the following notation:

$$\text{P(Subject | Size = Large)} \propto \text{P(Subject)P(Size = Large | Subject)}$$

The symbol ∝ means that the left-hand side *is proportional to* the right-hand side, where the constant of proportionality is $1/\Sigma_s$ P(Subject = *s*) P(Size = Large | Subject = *s*). The left-hand side is the posterior probability distribution over Subject. The first term on the right-hand side is the prior distribution. The second term, the probability of observing specific data about the Size for a given value of Subject, is the likelihood. Therefore, the preceding formula can be summarized as follows:

$$\text{Posterior} \propto \text{Prior x Likelihood}$$

This formula is broken down in figure 9.10. If there's one formula you should remember about Bayesian modeling, it's this one. Although you saw it specifically for the painting example, it's a general principle that holds for applications of Bayes' rule. To get the actual posterior of the particular value *b*, you compute the right-hand side of this equation for every possible value of *b*, and add all of the results to get the total B.

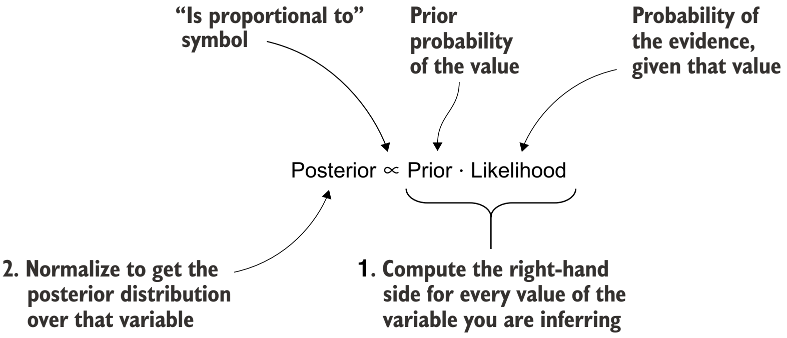

"Is proportional to" symbol
Prior probability of the value
Probability of the evidence, given that value
Posterior ∝ Prior · Likelihood
2. Normalize to get the posterior distribution over that variable
1. Compute the right-hand side for every value of the variable you are inferring

Figure 9.10 Structure of the Bayesian modeling formula

This total B is the normalizing factor. You then divide the Prior times the Likelihood by B to get the Posterior.

If you're talking about a continuous variable, as in our example, this normalizing process can be difficult, because it requires integrating over all possible values of *b*. Returning to our coin-tossing example, Bayes' rule says that

$$\mathrm{p}(\mathrm{Bias} = b|\mathrm{NumberOfHeads} = 63) = \frac{\mathrm{P}(\mathrm{Bias} = b)\mathrm{P}(\mathrm{NumberOfHeads} = 63|\mathrm{Bias} = b)}{\int_0^1 \mathrm{P}(\mathrm{Bias} = x)\mathrm{P}(\mathrm{NumberOfHeads} = 63|\mathrm{Bias} = x)\,dx}$$

Using our "proportional to" notation, you can rewrite this as follows:

$$\mathrm{p}(\mathrm{Bias} = b \mid \mathrm{NumberOfHeads} = 63) \propto \mathrm{P}(\mathrm{Bias} = b)\mathrm{P}(\mathrm{NumberOfHeads} = 63 \mid \mathrm{Bias} = b)$$

Once again, the posterior is proportional to the prior times the likelihood. Although this last equation is simple, it does hide an integral that can be difficult to estimate. Fortunately, in the case of the beta-binomial model, a simple solution to this equation exists, which you've already seen at the beginning of this section. You add the number of observed successes and failures to the parameters of the beta distribution. This is why the beta and binomial work well together. If you take any arbitrary continuous distribution and try to pair it with the binomial, you'll end up with an integration problem that doesn't have an easy solution. But when you pair a beta with the binomial, you get an easy answer.

Working with probabilistic programming systems, you'll never have to compute these integrals yourself. Probabilistic programming systems can often use approximation algorithms to deal with these difficult integration problems, so you're not restricted to working with functional forms that fit together particularly well. Nevertheless, when you have such a form available to you, it's best to use it.

NOTE In chapter 6, you first encountered the technical term *conjugate prior* to describe a prior distribution that works well with a distribution that depends on the parameter. Technically, this means that the posterior distribution has the same form as the prior distribution. Using this term, the beta distribution is the conjugate prior of the binomial, because the posterior distribution over the parameter is also a beta distribution. When you have a conjugate prior distribution, the integration in Bayes' rule has an easy solution. This is why conjugate distributions are used frequently in Bayesian statistics. But when using probabilistic programming, you're not limited to conjugate distributions.

The generic Bayes' rule

Now that you've learned more about Bayes' rule, in particular the proportionality relationship, it's time to explain the generic Bayes' rule. As with the total probability rule, Bayes' rule can be generalized to any number of variables, and can include conditioning variables. Following the notation of section 9.2, you have three sets of variables: $X_1,\ldots,X_n$ (the "causes"), $Y_1,\ldots,Y_m$ (the "effects"), and $Z_1,\ldots,Z_l$ (the conditioning variables). You're given $P(X_1,\ldots,X_n \mid Z_1,\ldots,Z_l)$, the prior probability of the causes, conditioned on the conditioning variables, and $P(Y_1,\ldots,Y_n \mid X_1,\ldots,X_m, Z_1,\ldots,Z_l)$, the conditional probability of the effects given the causes, again conditioned on the conditioning variables. You want the probability of the causes, given the effects, once again conditioned on the conditioning variables. This is $P(X_1,\ldots,X_n \mid Y_1,\ldots,Y_m, Z_1,\ldots,Z_l)$. Bayes' rule says that

$$P(X_1,\ldots,X_n \mid Y_1,\ldots,Y_m, Z_1,\ldots,Z_l) = \frac{P(X_1,\ldots,X_n \mid Z_1,\ldots,Z_l)P(Y_1,\ldots,Y_n \mid X_1,\ldots,X_m, Z_1,\ldots,Z_l)}{\Sigma_{x_1}\ldots\Sigma_{x_n} P(X_1 = x_1,\ldots,X_n = x_n \mid Z_1,\ldots,Z_l)P(Y_1,\ldots,Y_n \mid X_1 = x_1,\ldots,X_n = x_n, Z_1,\ldots,Z_l)}$$

I promised that in this section, I'd make the notation for this formula simpler. Because the denominator is the normalizing factor, you can use our "is proportional to" shorthand to make the equation much easier to understand:

$$P(X_1,\ldots,X_n \mid Y_1,\ldots,Y_m, Z_1,\ldots,Z_l) \propto P(X_1,\ldots,X_n \mid Z_1,\ldots,Z_l)P(Y_1,\ldots,Y_n \mid X_1,\ldots,X_m, Z_1,\ldots,Z_l)$$

This is the same as our Posterior $\propto$ Prior × Likelihood equation except that the posterior is a joint distribution over multiple cause variables, the likelihood also considers multiple effect variables, and other variables (the Z variables) influence the causes and effects.

Finally, recall that you can use boldface letters like X, Y, and Z for sets of variables. Bayes' rule can then be summarized in the succinct formula

$$P(X \mid Y,Z) \propto P(X \mid Z)P(Y \mid X,Z)$$

where X refers to all causes, Y refers to all effects, and Z refers to all conditioning variables. This pithy formula is the best way to remember the generic Bayes' rule.

Now you've learned how to estimate Bias. The next step is to use it to predict Toss_{101}.

9.4.2 Predicting the next coin toss

Okay, you've gotten a posterior distribution over Bias in the form of a beta distribution. How do you predict the next coin toss? There are three common ways to do this, all of which turn out to be simple for the beta-binomial model. As mentioned earlier, they are as follows:

- The maximum a posteriori (MAP) method
- The maximum likelihood estimation (MLE) method
- The full Bayesian method

You'll look at each method in turn.

Using the maximum a posteriori method

In the first method, called *maximum a posteriori* (*MAP*) *estimation*, you compute the value of Bias that has the highest posterior probability density. This value, which maximizes the prior times the likelihood, is called the *most likely value* of the Bias. You then use this value of the Bias to predict the next coin toss.

The MAP process is described in figure 9.11. The first step is to compute a posterior distribution over the Bias by using the approach of the previous section. You start with a prior of beta(2, 5), observe 63 heads and 37 tails, and obtain a posterior of beta(65, 42). In the next step, you compute the value of the Bias that's the peak of beta(65, 42). Looking back at figure 9.9, this is the point on the x-axis for which the value of beta(65, 42) is highest. In other words, you want the mode of beta(65, 42). It turns out there's a simple formula for this:

$$\text{mode}(\text{beta}(\alpha,\beta)) = \frac{\alpha - 1}{\alpha + \beta - 2}$$

In our example, the mode is equal to (65 – 1)/(65 + 42 – 2), which is approximately 0.6095. Now you assume that Bias is equal to 0.6095, and compute the probability that Toss_{101} is heads, given the data of 63 heads and 37 tails. The functional form for Toss_{101} says that the probability that the toss comes out heads is equal to the value of Bias, which you assumed is 0.6095. So your answer is 0.6095.

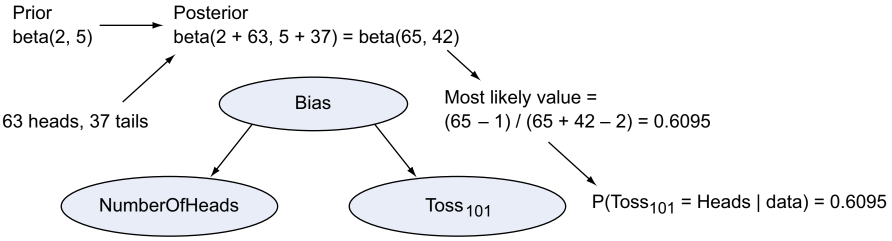


Figure 9.11 Predicting the next coin flip using the MAP method

Using the maximum likelihood estimation

The second method is a commonly used special case of the MAP estimation process called *maximum likelihood estimation* (MLE). In MLE, you choose the parameter values that "fit the data" the best, without regard to any prior. The MLE method is sometimes considered non-Bayesian, but it also fits into the Bayesian framework if you assume that every possible value of Bias has the same prior. So the formula

$$\text{Posterior} \propto \text{Prior} \times \text{Likelihood}$$

collapses to

$$\text{Posterior} \propto \text{Likelihood}$$

Therefore, the most likely value of the posterior is the value that maximizes the likelihood, hence the name *maximum likelihood estimation.*

The maximum likelihood method is illustrated in figure 9.12. This is similar to the MAP method shown in figure 9.11, except that you start with a prior of beta(1, 1), which assigns the same probability density to every value between 0 and 1. If you recall that the parameters of the prior are the imaginary number of heads and tails you've seen, plus one, you'll see that this prior represents the case where you don't imagine having seen any heads or tails. You then go through the same sequence of calculations, resulting in a prediction of 0.63. This result is no coincidence. You observed 63 out of 100 tosses resulting in heads. The value of the Bias that's most consistent with these observations is that there's exactly a 0.63 chance of any coin toss resulting in heads. So you see that the maximum likelihood estimate chooses the parameter value that best fits with the data, whereas the MAP estimate counterbalances the data with the prior.

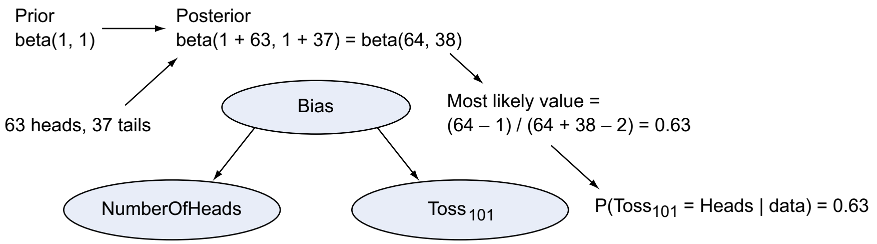


Figure 9.12 Predicting the next coin toss using the maximum likelihood method

Using the full Bayesian method

The third approach to predicting the next coin toss is sometimes called the *full Bayesian method*, because instead of estimating a single value of the Bias, it uses the full posterior distribution over Bias. The process is illustrated in figure 9.13. It starts in the same way as the MAP method, by computing the posterior distribution over Bias. To use this distribution to predict Toss_{101}, you use the formula for $P(\text{Toss}_{101} = \text{Heads} \mid \text{Data})$ shown in the figure. This formula is derived by applying the total probability rule and the chain rule. The main thing to notice is that it involves integration, because Bias is a continuous variable. Just as for estimating the posterior parameter value, this integration can be difficult to work with. But the beta-binomial model is again easy. It turns out that if the posterior is $\text{beta}(\alpha_1, \beta_1)$, then the probability that the next toss is heads is

$$\frac{\alpha_1}{\alpha_1 + \beta_1}$$

So in our example, the probability of heads is 65 / (65 + 42) = 0.6075. And, to close the loop, this simple formula for the probability of heads is why you add 1 to the count of heads and tails in the parameters of the beta distribution: so that you end up with a simple formula for the probability of the next coin toss coming up heads.

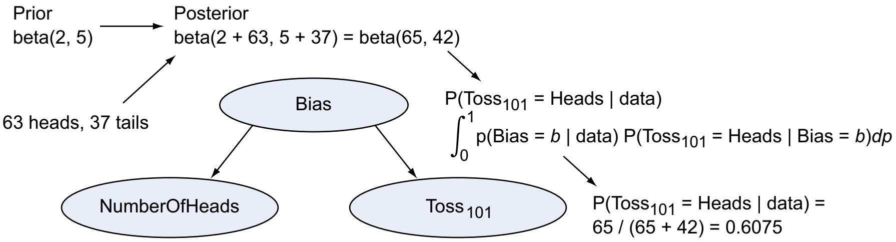

Prior
beta(2, 5)
Posterior
beta(2 + 63, 5 + 37) = beta(65, 42)
63 heads, 37 tails
Bias
$P(\text{Toss}_{101} = \text{Heads} \mid \text{data})$
$\int_0^1 p(\text{Bias} = b \mid \text{data})\, P(\text{Toss}_{101} = \text{Heads} \mid \text{Bias} = b)dp$
NumberOfHeads
Toss_{101}
$P(\text{Toss}_{101} = \text{Heads} \mid \text{data}) =$
65 / (65 + 42) = 0.6075

Figure 9.13 Predicting the next coin toss using the full Bayesian method

Comparing the methods

Having seen these three methods, let's compare them:

- *The MLE method* provides the best fit to the data, but is also liable to overfit the data. Overfitting is a problem in machine learning whereby the learner fits the pattern found in the data too closely, in a way that's unable to be generalized. This can especially be a problem with only a few coin tosses. For example, if there are only 10 coin tosses and 7 of them come out heads, should you immediately conclude that the bias is 0.7? Even a fair coin will come out heads 7 times out of 10 a fair percentage of times, so the coin tosses don't provide conclusive evidence that the coin isn't fair.

  The MLE method has two advantages that make it popular. First, it tends to be relatively efficient, because it doesn't require integrating over all parameter values to predict the next instance. Second, it doesn't require specifying a prior, which can be difficult when you don't have any basis for one. Nevertheless, the susceptibility to overfitting can be a significant problem with this method.

- *The MAP method* can be a good compromise. Including a prior can serve two purposes. One is to encode prior beliefs that you have. The other is to counteract overfitting. For example, if you start with a beta(11, 11) prior, you aren't biasing the results toward heads or tails in any way, but the effect of the data will be dampened by adding 10 imaginary heads and tails to the result. To see this, suppose you toss the coin 10 times and 7 of them come up heads. Remember that a beta(11, 11) prior means that you've seen 10 imaginary heads and 10 imaginary tails. Adding 7 more heads and 3 more tails gives you 17 heads and 13 tails in total. So the MAP estimate for the bias is 17 / (17 + 13) = 17/30 ≈ 0.5667. You can also see this from the formula for the mode of a beta distribution given earlier, which is

  $$\frac{\alpha - 1}{\alpha + \beta - 2}$$

  With seven heads and three tails, the posterior is beta(18, 14), so the mode is 17/30. Even though 70% of your data is heads, your posterior belief in heads is still only slightly more than 0.5, and a lot less than 0.7 for the MLE method. In addition to being able to counter overfitting, the MAP method is also relatively efficient, because it doesn't require integrating over all parameter values. But it does require specifying a prior, which can be difficult.
- *The full Bayesian approach*, where feasible, can be superior to the other approaches, because it uses the full distribution. In particular, when the mode of the distribution isn't representative of the full distribution, the other approaches can be misleading. For a beta distribution, this isn't a serious issue; the MAP and full Bayesian predictions are close to each other in our example. Specifically, with a beta(11, 11) prior and seven observed heads and three observed tails, you get a beta(18, 14) posterior. The Bayesian estimate of the probability that the next toss will be 18 / (18 + 14) = 18/32 = 0.5625, or just slightly less than the MAP estimate. For other distributions, especially those with multiple peaks, however, the full Bayesian approach can produce significantly better estimates than the MAP approach. Even the MAP approach, which uses a prior, will settle on one of the peaks, and completely ignore an important part of the distribution. But the Bayesian approach is more difficult to execute computationally.

Probabilistic programming systems vary in the range of approaches they support. Most typically, they support full Bayesian reasoning. Because full Bayesian reasoning often requires integration, these systems use approximation algorithms. Some probabilistic programming systems also support maximum likelihood and MAP estimation for specific models, which can be more computationally efficient. In particular, Figaro provides both full Bayesian and MAP algorithms. Chapter 12 shows you how to use these approaches practically in Figaro.

So now you know the basic rules of inference, and you understand how Bayesian modeling uses Bayes' rule to learn from data and use the learned knowledge for future predictions. In the forthcoming chapters, you'll learn specific algorithms for inference. Two main families of inference algorithms are used in probabilistic programming: factored algorithms and sampling algorithms. These two families are the subjects of the next two chapters.

9.5 Summary

- The chain rule lets you take the conditional probability distributions of individual variables and construct a joint probabilistic model over all variables.
- The total probability rule lets you take a joint probabilistic model over a set of variables and reduce it to get a probability distribution over individual variables.
- The network arrows in a probabilistic model typically follow the process by which the data is generated, but inference in the model can go in any direction. Bayes' rule lets you do this.
- Bayesian modeling uses Bayes' rule to infer causes from observations of their effects, and uses those inferences to predict future outcomes.
- In Bayesian inference, the posterior probability of a value of a variable is proportional to the prior probability of the value times the likelihood of the value, which is the probability of the evidence given the value.
- In the MAP estimation approach, the most likely posterior value of a parameter is used to predict future instances.
- In the MLE approach, the prior is ignored, and the parameter value that maximizes the likelihood is used for prediction. This is the simplest approach but can overfit the data.
- In the full Bayesian approach, the full posterior probability distribution over the parameter value is used to predict future instances. This is the most accurate approach but can be computationally difficult.

9.6 Exercises

Solutions to selected exercises are available online at www.manning.com/books/practical-probabilistic-programming.

1 Consider the detailed printer model from the printer diagnosis example, shown in the Bayesian network in figure 5.11 (in chapter 5). Consider the following case:

- Printer Power Button On = true
- Toner Level = low
- Toner Low Indicator On = false
- Paper Flow = smooth
- Paper Jam Indicator On = false
- Printer State = poor

- a Write the probability of this case using the full chain rule, where each variable is conditioned on all preceding variables.
- b Simplify this expression by taking into account independence relationships in the network.
- c Write an expression for the joint probability distribution that applies in a general way to all cases, without specifying specific values of variables.

2 For the network in exercise 1:

- a Write an expression for the probability that Printer Power Button On = true.
- b Write an expression for the probability that Printer Power Button On = true and Printer State = poor.

3 Assume that 1 in 40 million US citizens become president of the United States.

- a Assume that 50% of presidents are left-handed, compared to 10% of the general population. What is the probability someone became the president of the United States, given that he or she is left-handed?
- b Now assume that 15% of US presidents went to Harvard, compared to 1 in 2,000 for the general population. What is the probability that someone became the president of the United States, given that he or she went to Harvard?
- c Assuming left-handedness and going to Harvard are conditionally independent, given whether someone became president, what's the probability that someone became the president of the United States, given that he or she is left-handed and went to Harvard?

PROGRAMMING/DATA SCIENCE

Practical Probabilistic Programming

Avi Pfeffer

The data you accumulate about your customers, products, and website users can not only help you interpret your past, it can help you predict your future! Probabilistic programming uses code to draw probabilistic inferences from data. By applying specialized algorithms, your programs assign degrees of probability to conclusions. This means you can forecast future events like sales trends, computer system failures, experimental outcomes, and many other critical concerns.

Practical Probabilistic Programming introduces the working programmer to probabilistic programming. In this book, you'll immediately work on practical examples like building a spam filter, diagnosing computer system data problems, and recovering digital images. You'll discover probabilistic inference, where algorithms help make extended predictions about issues like social media usage. Along the way, you'll learn to use functional-style programming for text analysis, object-oriented models to predict social phenomena like the spread of tweets, and open universe models to gauge real-life social media usage. The book also has chapters on how probabilistic models can help in decision making and modeling of dynamic systems.

What's Inside

- Introduction to probabilistic modeling
- Writing probabilistic programs in Figaro
- Building Bayesian networks
- Predicting product lifecycles
- Decision-making algorithms

This book assumes no prior exposure to probabilistic programming. Knowledge of Scala is helpful.

Avi Pfeffer is the principal developer of the Figaro language for probabilistic programming.

To download their free eBook in PDF, ePub, and Kindle formats, owners of this book should visit www.manning.com/books/practical-probabilistic-programming

“An important step in moving probabilistic programming from research laboratories out into the real world.”

—From the Foreword by Stuart Russell, UC Berkeley

“Clear examples and down-to-earth explanations of a difficult and complex topic.”

—Mark Elston, Advantest America

“Coherent, practical, and accessible. A fantastic hands-on book on probabilistic programming with Scala.”

—Kostas Passadis, IPTO

“Probabilistic programming is complex! Avi makes the subject straightforward and intuitive to learn.”

—Earl Bingham, Eyelock

Free eBook

SEE INSERT

MANNING

$59.99 / Can $68.99 [INCLUDING eBOOK]

ISBN 13: 978-1-61729-233-0
ISBN 10: 1-61729-233-8

55999

9 781617 292330