Introduction to Bayesian Modeling Tutorial

Adam Morris – Computational Social Cognition Bootcamp, July 2017
thatadammorris.com thatadammorris@gmail.com

Dear Dr. Watson:

How are things at Baker Street? It must be quite peaceful, what with Holmes on holiday. I don't know how you stand his rubbish violin playing all day.

Anyway, down to business. An urgent matter has come up. We need to make some inferences based on sparse data. Unfortunately, Holmes usually does this for us. Do you know of any other way to make such judgments?

If so, please come at your earliest convenience. The address is 4 Whitehall Place, at the corner of Scotland Yard.

With my deepest gratitude,
Lestrade

This tutorial is a practical introduction to the nitty-gritty of programming Bayesian models. It is targeted at social scientists who ultimately want to model human decision-making, but have never simulated Bayesian inference before. It will employ simple models in simple tasks, of the form used in much psychology and cognitive science. (It is not meant for, say, computer scientists in machine learning; I will brazenly ignore all technical details.)

I will assume that you have already learned the theoretical basics of Bayesian inference. You should know what a prior, likelihood, and posterior are. If these concepts are new to you, I recommend reading "Information Theory, Inference, and Learning" by David MacKay (skip to the chapters about Bayesian inference).

This tutorial will not teach you the theory of Bayesian inference. Rather, it will assist you in translating that theory into practice. In my own experience, that translation is often the most difficult part. This tutorial is an attempt to share whatever practical wisdom I've acquired on this matter.

One other important note: This tutorial uses MATLAB. The upside to using MATLAB (or, say, Python) is that, because it has no built-in inference engines, it forces you to write out all the details yourself – an extremely useful pedagogical exercise.

The downside is that, for *actual* complex Bayesian modeling, it's often easier to use languages that take care of the nitty-gritty for you. For such a language, I recommend Noah Goodman's WebPPL, and his excellent introduction on ProbMods.org.

Some advice

Before you dive into the Mysteries, I want to give you some advice. Programming Bayesian models is both simple and challenging. It is simple because it ultimately boils down to implementing a few laws of probability (Bayes' rule, the law of total probability, etc.). But it is challenging because all the terms in these equations end up blending together, and things can get jumbled quickly.

For that reason, before you begin programming, I highly recommend writing out everything with pencil and paper. Write out the variables, what your goal is, what you know, and how you can derive your goal from what you know. Here's an example:

What you should write out	Example
The variables (and the values they can take on)	C means whether the person smoked cigarettes, L means whether the person got lung cancer.
Your goal	Compute $Prob(C = 1 \mid L = 1)$, the probability that the person smoked given that they got lung cancer.
What you know	$Prob(L = 1 \mid C = 1) = 0.4$ – i.e. the probability that someone gets lung cancer given that they smoke. $Prob(L = 1 \mid C = 0) = 0.01$ – i.e. the probability that someone gets lung cancer given that they don't smoke. $Prob(C = 1) = 0.15$ – i.e. the prior probability that someone smokes.
How you can derive your goal from what you know	By Bayes' rule, $Prob(C = 1 \mid L = 1) = \frac{Prob(L=1 \mid C=1) * Prob(C=1)}{Prob(L=1 \mid C=1) * Prob(C=1) + Prob(L=1 \mid C=0) * Prob(C=0)}$.

Writing this all out will help you immensely.

Denominator troubles. In the above example, you might have expected the denominator of Bayes' rule to be written as $Prob(L = 1)$, not $Prob(L = 1 \mid C = 1) * Prob(C = 1) + Prob(L = 1 \mid C = 0) * Prob(C = 0)$. These are actually identical; if you're confused, Google the "law of total probability". To compute the denominator in Bayes'

rule, you just compute the *numerator* for each possible value of C, and then add them all together.

Why does this make sense? Think of the numerator in Bayes' rule as assigning a "score" to each possible value of C. To get the actual probability of that value of C, we have to figure out what percentage of the total score was assigned to that value. (Probabilities are between 0 and 1, remember!) So to do that, we compute the score for each value of C, add them all together to get a "total score", and then divide each individual value's score by that total. This is called *normalizing* the numerator.

You might think of this normalizing stuff as a kind of trivial annoyance; the denominator just doesn't seem that important. In some sense, you're right – all the interesting stuff is happening in the numerator.

The problem is that computing the denominator is actually the hardest part of applying Bayes' rule. (Much of the time, anyway.) In the cigarettes example, it seems simple because there's only two values of C that we have to score. But what if there were a hundred possible values of the variable we cared about? Or a thousand? Or infinite? (Imagine that C represented, not *whether* someone smoked cigarettes, but rather the *number* of cigarettes they'd ever smoked. How would you compute the denominator then?) As you'll see in Mystery 2, this is a challenge you'll have to overcome.

Pseudocode. I've written some pseudocode to help you get started. Suppose your goal (as it will be in Mystery 1) is to compute the posterior of some hypothesis given some observed data. Your code will look something like this:

```
write function to get the prior probability of any hypothesis:
 "prior(hypothesis) = ..."

write function to get the likelihood of any data set given any hypothesis:
 "likelihood(data, hypothesis) = ..."

write function to compute the numerator of Bayes' rule, for any data & hypothesis:
 "getNumerator(hypothesis, data) = prior(hypothesis) * likelihood(data, hypothesis)"


write function to compute the denominator of Bayes' rule, for any data:
 "getDenominator(data) = sum(getNumerator(hypothesis, data) over all hypotheses)"

do Bayes' rule for a particular hypothesis & data set:
hypothesis = 1, data = 0; (for example)
posterior = getNumerator(hypothesis, data) / getDenominator(data)
```

If we were following the cigarettes-and-cancer example above, then "hypothesis" means whether they smoked (i.e. $C = 1$ or 0) and "data" means whether they got lung cancer

(i.e. $L = 1$ or 0). The prior function might look like "if hypothesis = 1, then return .15; otherwise, return .85". And the likelihood function might look like "if hypothesis = 1 and data = 1, then return .4; otherwise…" Make sense?

By writing every part of Bayes' rule out in a different function, I'm being *very* general. You can, of course, combine some of these functions together, or do it all in one big calculation. (For instance, in my solution code, I typically only write out "getNumerator" and "getDenominator".) Writing the functions separately, however, will turn out to be handy when things get complicated in later Mysteries – and, if you're confused, it will help you see exactly what's going on.

Alright, get to it!