Bayesian Statistics in Python

Think Bayes

O'REILLY®

Allen B. Downey

Math/Statistics/Python

Think Bayes

If you know how to program with Python, and know a little about probability, you're ready to tackle Bayesian statistics. This book shows you how to solve statistical problems with Python code instead of mathematical notation, and use discrete probability distributions instead of continuous mathematics. Once you get the math out of the way, the Bayesian fundamentals will become clearer, and you'll begin to apply these techniques to real world problems.

Bayesian statistical methods are becoming more common and more important, but not many resources are available to help beginners. Based on undergraduate classes taught by author Allen Downey, this book's computational approach helps you get a solid start.

- Use your existing programming skills to learn and understand Bayesian statistics
- Work with problems involving estimation, prediction, decision analysis, evidence, and hypothesis testing
- Get started with simple examples, using coins, M&Ms, Dungeons & Dragons dice, paintball, and hockey
- Learn computational methods for solving real world problems, such as interpreting SAT scores, simulating kidney tumors, and modeling the human microbiome

Allen Downey is a Professor of Computer Science at Olin College of Engineering. With a Ph.D. in Computer Science from U.C. Berkeley, he has taught computer science at Wellesley College, Colby College, and U.C. Berkeley. He is also the author of *Think Stats* and *Think Python* (both O'Reilly).

US $29.99 CAN $31.99
ISBN: 978-1-449-37378-7

Think Bayes

Allen B. Downey

Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo O'REILLY®

Think Bayes

by Allen B. Downey

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://safaribooksonline.com*). For more information, contact our corporate/institutional sales department: 800-998-9938 or *corporate@oreilly.com*.

Editors: Mike Loukides and Ann Spencer
Production Editor: Melanie Yarbrough
Proofreader: Jasmine Kwityn
Indexer: Allen Downey

Cover Designer: Randy Comer
Interior Designer: David Futato
Illustrator: Rebecca Demarest

September 2013: First Edition

Revision History for the First Edition:

2013-09-10: First release

2014-02-10: Second release

2014-08-22: Third release

See *http://oreilly.com/catalog/errata.csp?isbn=9781449370787* for release details.

ISBN: 978-1-449-37078-7

[LSI]

Table of Contents

Preface

My theory, which is mine

The premise of this book, and the other books in the *Think X* series, is that if you know how to program, you can use that skill to learn other topics.

Most books on Bayesian statistics use mathematical notation and present ideas in terms of mathematical concepts like calculus. This book uses Python code instead of math, and discrete approximations instead of continuous mathematics. As a result, what would be an integral in a math book becomes a summation, and most operations on probability distributions are simple loops.

I think this presentation is easier to understand, at least for people with programming skills. It is also more general, because when we make modeling decisions, we can choose the most appropriate model without worrying too much about whether the model lends itself to conventional analysis.

Also, it provides a smooth development path from simple examples to real-world problems. Chapter 3 is a good example. It starts with a simple example involving dice, one of the staples of basic probability. From there it proceeds in small steps to the locomotive problem, which I borrowed from Mosteller's *Fifty Challenging Problems in Probability with Solutions*, and from there to the German tank problem, a famously successful application of Bayesian methods during World War II.

Modeling and approximation

Most chapters in this book are motivated by a real-world problem, so they involve some degree of modeling. Before we can apply Bayesian methods (or any other analysis), we have to make decisions about which parts of the real-world system to include in the model and which details we can abstract away.

For example, in Chapter 7, the motivating problem is to predict the winner of a hockey game. I model goal-scoring as a Poisson process, which implies that a goal is equally

likely at any point in the game. That is not exactly true, but it is probably a good enough model for most purposes.

In Chapter 12 the motivating problem is interpreting SAT scores (the SAT is a standardized test used for college admissions in the United States). I start with a simple model that assumes that all SAT questions are equally difficult, but in fact the designers of the SAT deliberately include some questions that are relatively easy and some that are relatively hard. I present a second model that accounts for this aspect of the design, and show that it doesn't have a big effect on the results after all.

I think it is important to include modeling as an explicit part of problem solving because it reminds us to think about modeling errors (that is, errors due to simplifications and assumptions of the model).

Many of the methods in this book are based on discrete distributions, which makes some people worry about numerical errors. But for real-world problems, numerical errors are almost always smaller than modeling errors.

Furthermore, the discrete approach often allows better modeling decisions, and I would rather have an approximate solution to a good model than an exact solution to a bad model.

On the other hand, continuous methods sometimes yield performance advantages—for example by replacing a linear- or quadratic-time computation with a constant-time solution.

So I recommend a general process with these steps:

1. While you are exploring a problem, start with simple models and implement them in code that is clear, readable, and demonstrably correct. Focus your attention on good modeling decisions, not optimization.
2. Once you have a simple model working, identify the biggest sources of error. You might need to increase the number of values in a discrete approximation, or increase the number of iterations in a Monte Carlo simulation, or add details to the model.
3. If the performance of your solution is good enough for your application, you might not have to do any optimization. But if you do, there are two approaches to consider. You can review your code and look for optimizations; for example, if you cache previously computed results you might be able to avoid redundant computation. Or you can look for analytic methods that yield computational shortcuts.

One benefit of this process is that Steps 1 and 2 tend to be fast, so you can explore several alternative models before investing heavily in any of them.

Another benefit is that if you get to Step 3, you will be starting with a reference implementation that is likely to be correct, which you can use for regression testing (that is, checking that the optimized code yields the same results, at least approximately).

Working with the code

Many of the examples in this book use classes and functions defined in `thinkbayes.py`. You can download this module from *http://thinkbayes.com/thinkbayes.py*.

Most chapters contain references to code you can download from *http://thinkbayes.com*. Some of those files have dependencies you will also have to download. I suggest you keep all of these files in the same directory so they can import each other without changing the Python search path.

You can download these files one at a time as you need them, or you can download them all at once from *http://thinkbayes.com/thinkbayes_code.zip*. This file also contains the data files used by some of the programs. When you unzip it, it creates a directory named `thinkbayes_code` that contains all the code used in this book.

Or, if you are a Git user, you can get all of the files at once by forking and cloning this repository: *https://github.com/AllenDowney/ThinkBayes*.

One of the modules I use is `thinkplot.py`, which provides wrappers for some of the functions in `pyplot`. To use it, you need to install `matplotlib`. If you don't already have it, check your package manager to see if it is available. Otherwise you can get download instructions from *http://matplotlib.org*.

Finally, some programs in this book use NumPy and SciPy, which are available from *http://numpy.org* and *http://scipy.org*.

Code style

Experienced Python programmers will notice that the code in this book does not comply with PEP 8, which is the most common style guide for Python (*http://www.python.org/dev/peps/pep-0008/*).

Specifically, PEP 8 calls for lowercase function names with underscores between words, `like_this`. In this book and the accompanying code, function and method names begin with a capital letter and use camel case, `LikeThis`.

I broke this rule because I developed some of the code while I was a Visiting Scientist at Google, so I followed the Google style guide, which deviates from PEP 8 in a few places. Once I got used to Google style, I found that I liked it. And at this point, it would be too much trouble to change.

Also on the topic of style, I write "Bayes's theorem" with an *s* after the apostrophe, which is preferred in some style guides and deprecated in others. I don't have a strong preference. I had to choose one, and this is the one I chose.

And finally one typographical note: throughout the book, I use PMF and CDF for the mathematical concept of a probability mass function or cumulative distribution function, and Pmf and Cdf to refer to the Python objects I use to represent them.

Prerequisites

There are several excellent modules for doing Bayesian statistics in Python, including `pymc` and OpenBUGS. I chose not to use them for this book because you need a fair amount of background knowledge to get started with these modules, and I want to keep the prerequisites minimal. If you know Python and a little bit about probability, you are ready to start this book.

Chapter 1 is about probability and Bayes's theorem; it has no code. Chapter 2 introduces `Pmf`, a thinly disguised Python dictionary I use to represent a probability mass function (PMF). Then Chapter 3 introduces `Suite`, a kind of Pmf that provides a framework for doing Bayesian updates. And that's just about all there is to it.

Well, almost. In some of the later chapters, I use analytic distributions including the Gaussian (normal) distribution, the exponential and Poisson distributions, and the beta distribution. In Chapter 15 I break out the less-common Dirichlet distribution, but I explain it as I go along. If you are not familiar with these distributions, you can read about them on Wikipedia. You could also read the companion to this book, *Think Stats*, or an introductory statistics book (although I'm afraid most of them take a mathematical approach that is not particularly helpful for practical purposes).

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic
: Indicates new terms, URLs, email addresses, filenames, and file extensions.

`Constant width`
: Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords.

`Constant width bold`
: Shows commands or other text that should be typed literally by the user.

`Constant width italic`
: Shows text that should be replaced with user-supplied values or by values determined by context.

This icon signifies a tip, suggestion, or general note.

This icon indicates a warning or caution.

Safari® Books Online

Safari Books Online (*www.safaribooksonline.com*) is an on-demand digital library that delivers expert content in both book and video form from the world's leading authors in technology and business.

Technology professionals, software developers, web designers, and business and creative professionals use Safari Books Online as their primary resource for research, problem solving, learning, and certification training.

Safari Books Online offers a range of plans and pricing for enterprise, government, education, and individuals. Members have access to thousands of books, training videos, and prepublication manuscripts in one fully searchable database from publishers like O'Reilly Media, Prentice Hall Professional, Addison-Wesley Professional, Microsoft Press, Sams, Que, Peachpit Press, Focal Press, Cisco Press, John Wiley & Sons, Syngress, Morgan Kaufmann, IBM Redbooks, Packt, Adobe Press, FT Press, Apress, Manning, New Riders, McGraw-Hill, Jones & Bartlett, Course Technology, and hundreds more. For more information about Safari Books Online, please visit us online.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at *http://oreil.ly/think-bayes*.

To comment or ask technical questions about this book, send email to *bookquestions@oreilly.com*.

For more information about our books, courses, conferences, and news, see our website at *http://www.oreilly.com*.

Find us on Facebook: *http://facebook.com/oreilly*

Follow us on Twitter: *http://twitter.com/oreillymedia*

Watch us on YouTube: *http://www.youtube.com/oreillymedia*

Contributor List

If you have a suggestion or correction, please send email to *downey@allendowney.com*. If I make a change based on your feedback, I will add you to the contributor list (unless you ask to be omitted).

If you include at least part of the sentence the error appears in, that makes it easy for me to search. Page and section numbers are fine, too, but not as easy to work with. Thanks!

- First, I have to acknowledge David MacKay's excellent book, *Information Theory, Inference, and Learning Algorithms*, which is where I first came to understand Bayesian methods. With his permission, I use several problems from his book as examples.
- This book also benefited from my interactions with Sanjoy Mahajan, especially in fall 2012, when I audited his class on Bayesian Inference at Olin College.
- I wrote parts of this book during project nights with the Boston Python User Group, so I would like to thank them for their company and pizza.
- Jonathan Edwards sent in the first typo.
- George Purkins found a markup error.
- Olivier Yiptong sent several helpful suggestions.
- Yuriy Pasichnyk found several errors.
- Kristopher Overholt sent a long list of corrections and suggestions.
- Robert Marcus found a misplaced *i*.
- Max Hailperin suggested a clarification in Chapter 1.
- Markus Dobler pointed out that drawing cookies from a bowl with replacement is an unrealistic scenario.

- Tom Pollard and Paul A. Giannaros spotted a version problem with some of the numbers in the train example.
- Ram Limbu found a typo and suggested a clarification.
- In spring 2013, students in my class, Computational Bayesian Statistics, made many helpful corrections and suggestions: Kai Austin, Claire Barnes, Kari Bender, Rachel Boy, Kat Mendoza, Arjun Iyer, Ben Kroop, Nathan Lintz, Kyle McConnaughay, Alec Radford, Brendan Ritter, and Evan Simpson.
- Greg Marra and Matt Aasted helped me clarify the discussion of *The Price is Right* problem.
- Marcus Ogren pointed out that the original statement of the locomotive problem was ambiguous.
- Jasmine Kwityn and Dan Fauxsmith at O'Reilly Media proofread the book and found many opportunities for improvement.

CHAPTER 1

Bayes's Theorem

Conditional probability

The fundamental idea behind all Bayesian statistics is Bayes's theorem, which is surprisingly easy to derive, provided that you understand conditional probability. So we'll start with probability, then conditional probability, then Bayes's theorem, and on to Bayesian statistics.

A probability is a number between 0 and 1 (including both) that represents a degree of belief in a fact or prediction. The value 1 represents certainty that a fact is true, or that a prediction will come true. The value 0 represents certainty that the fact is false.

Intermediate values represent degrees of certainty. The value 0.5, often written as 50%, means that a predicted outcome is as likely to happen as not. For example, the probability that a tossed coin lands face up is very close to 50%.

A conditional probability is a probability based on some background information. For example, I want to know the probability that I will have a heart attack in the next year. According to the CDC, "Every year about 785,000 Americans have a first coronary attack (*http://www.cdc.gov/heartdisease/facts.htm*)."

The U.S. population is about 311 million, so the probability that a randomly chosen American will have a heart attack in the next year is roughly 0.3%.

But I am not a randomly chosen American. Epidemiologists have identified many factors that affect the risk of heart attacks; depending on those factors, my risk might be higher or lower than average.

I am male, 45 years old, and I have borderline high cholesterol. Those factors increase my chances. However, I have low blood pressure and I don't smoke, and those factors decrease my chances.

Plugging everything into the online calculator at *http://cvdrisk.nhlbi.nih.gov/calculator.asp*, I find that my risk of a heart attack in the next year is about 0.2%, less than the national average. That value is a conditional probability, because it is based on a number of factors that make up my "condition."

The usual notation for conditional probability is $p(A|B)$, which is the probability of A given that B is true. In this example, A represents the prediction that I will have a heart attack in the next year, and B is the set of conditions I listed.

Conjoint probability

Conjoint probability is a fancy way to say the probability that two things are true. I write $p(A \text{ and } B)$ to mean the probability that A and B are both true.

If you learned about probability in the context of coin tosses and dice, you might have learned the following formula:

$$p(A \text{ and } B) = p(A)\ p(B) \qquad \text{WARNING: not always true}$$

For example, if I toss two coins, and A means the first coin lands face up, and B means the second coin lands face up, then $p(A) = p(B) = 0.5$, and sure enough, $p(A \text{ and } B) = p(A)\ p(B) = 0.25$.

But this formula only works because in this case A and B are independent; that is, knowing the outcome of the first event does not change the probability of the second. Or, more formally, $p(B|A) = p(B)$.

Here is a different example where the events are not independent. Suppose that A means that it rains today and B means that it rains tomorrow. If I know that it rained today, it is more likely that it will rain tomorrow, so $p(B|A) > p(B)$.

In general, the probability of a conjunction is

$$p(A \text{ and } B) = p(A)\ p(B|A)$$

for any A and B. So if the chance of rain on any given day is 0.5, the chance of rain on two consecutive days is not 0.25, but probably a bit higher.

The cookie problem

We'll get to Bayes's theorem soon, but I want to motivate it with an example called the cookie problem.[1] Suppose there are two bowls of cookies. Bowl 1 contains 30 vanilla cookies and 10 chocolate cookies. Bowl 2 contains 20 of each.

Now suppose you choose one of the bowls at random and, without looking, select a cookie at random. The cookie is vanilla. What is the probability that it came from Bowl 1?

This is a conditional probability; we want $\mathrm{p}(\text{Bowl } 1 \mid \text{vanilla})$, but it is not obvious how to compute it. If I asked a different question—the probability of a vanilla cookie given Bowl 1—it would be easy:

$$\mathrm{p}(\text{vanilla} \mid \text{Bowl } 1) = 3/4$$

Sadly, $\mathrm{p}(A \mid B)$ is *not* the same as $\mathrm{p}(B \mid A)$, but there is a way to get from one to the other: Bayes's theorem.

Bayes's theorem

At this point we have everything we need to derive Bayes's theorem. We'll start with the observation that conjunction is commutative; that is

$$\mathrm{p}(A \text{ and } B) = \mathrm{p}(B \text{ and } A)$$

for any events A and B.

Next, we write the probability of a conjunction:

$$\mathrm{p}(A \text{ and } B) = \mathrm{p}(A)\ \mathrm{p}(B \mid A)$$

Since we have not said anything about what A and B mean, they are interchangeable. Interchanging them yields

$$\mathrm{p}(B \text{ and } A) = \mathrm{p}(B)\ \mathrm{p}(A \mid B)$$

That's all we need. Pulling those pieces together, we get

$$\mathrm{p}(B)\ \mathrm{p}(A \mid B) = \mathrm{p}(A)\ \mathrm{p}(B \mid A)$$

1. Based on an example from *http://en.wikipedia.org/wiki/Bayes'_theorem* that is no longer there.

Which means there are two ways to compute the conjunction. If you have $p(A)$, you multiply by the conditional probability $p(B|A)$. Or you can do it the other way around; if you know $p(B)$, you multiply by $p(A|B)$. Either way you should get the same thing.

Finally we can divide through by $p(B)$:

$$p(A|B) = \frac{p(A)\ p(B|A)}{p(B)}$$

And that's Bayes's theorem! It might not look like much, but it turns out to be surprisingly powerful.

For example, we can use it to solve the cookie problem. I'll write B_1 for the hypothesis that the cookie came from Bowl 1 and V for the vanilla cookie. Plugging in Bayes's theorem we get

$$p(B_1|V) = \frac{p(B_1)\ p(V|B_1)}{p(V)}$$

The term on the left is what we want: the probability of Bowl 1, given that we chose a vanilla cookie. The terms on the right are:

- $p(B_1)$: This is the probability that we chose Bowl 1, unconditioned by what kind of cookie we got. Since the problem says we chose a bowl at random, we can assume $p(B_1) = 1/2$.
- $p(V|B_1)$: This is the probability of getting a vanilla cookie from Bowl 1, which is 3/4.
- $p(V)$: This is the probability of drawing a vanilla cookie from either bowl. Since we had an equal chance of choosing either bowl and the bowls contain the same number of cookies, we had the same chance of choosing any cookie. Between the two bowls there are 50 vanilla and 30 chocolate cookies, so $p(V) = 5/8$.

Putting it together, we have

$$p(B_1|V) = \frac{(1/2)\ (3/4)}{5/8}$$

which reduces to 3/5. So the vanilla cookie is evidence in favor of the hypothesis that we chose Bowl 1, because vanilla cookies are more likely to come from Bowl 1.

This example demonstrates one use of Bayes's theorem: it provides a strategy to get from $p(B|A)$ to $p(A|B)$. This strategy is useful in cases, like the cookie problem, where it is

easier to compute the terms on the right side of Bayes's theorem than the term on the left.

The diachronic interpretation

There is another way to think of Bayes's theorem: it gives us a way to update the probability of a hypothesis, H, in light of some body of data, D.

This way of thinking about Bayes's theorem is called the **diachronic interpretation**. "Diachronic" means that something is happening over time; in this case the probability of the hypotheses changes, over time, as we see new data.

Rewriting Bayes's theorem with H and D yields:

$$\mathrm{p}(H|D) = \frac{\mathrm{p}(H)\ \mathrm{p}(D|H)}{\mathrm{p}(D)}$$

In this interpretation, each term has a name:

- $\mathrm{p}(H)$ is the probability of the hypothesis before we see the data, called the prior probability, or just **prior**.
- $\mathrm{p}(H|D)$ is what we want to compute, the probability of the hypothesis after we see the data, called the **posterior**.
- $\mathrm{p}(D|H)$ is the probability of the data under the hypothesis, called the **likelihood**.
- $\mathrm{p}(D)$ is the probability of the data under any hypothesis, called the **normalizing constant**.

Sometimes we can compute the prior based on background information. For example, the cookie problem specifies that we choose a bowl at random with equal probability.

In other cases the prior is subjective; that is, reasonable people might disagree, either because they use different background information or because they interpret the same information differently.

The likelihood is usually the easiest part to compute. In the cookie problem, if we know which bowl the cookie came from, we find the probability of a vanilla cookie by counting.

The normalizing constant can be tricky. It is supposed to be the probability of seeing the data under any hypothesis at all, but in the most general case it is hard to nail down what that means.

Most often we simplify things by specifying a set of hypotheses that are

Mutually exclusive:
: At most one hypothesis in the set can be true, and

Collectively exhaustive:
There are no other possibilities; at least one of the hypotheses has to be true.

I use the word **suite** for a set of hypotheses that has these properties.

In the cookie problem, there are only two hypotheses—the cookie came from Bowl 1 or Bowl 2—and they are mutually exclusive and collectively exhaustive.

In that case we can compute $\mathrm{p}(D)$ using the law of total probability, which says that if there are two exclusive ways that something might happen, you can add up the probabilities like this:

$$\mathrm{p}(D) = \mathrm{p}(B_1)\ \mathrm{p}(D|B_1) + \mathrm{p}(B_2)\ \mathrm{p}(D|B_2)$$

Plugging in the values from the cookie problem, we have

$$\mathrm{p}(D) = (1/2)\ (3/4) + (1/2)\ (1/2) = 5/8$$

which is what we computed earlier by mentally combining the two bowls.

The M&M problem

M&M's are small candy-coated chocolates that come in a variety of colors. Mars, Inc., which makes M&M's, changes the mixture of colors from time to time.

In 1995, they introduced blue M&M's. Before then, the color mix in a bag of plain M&M's was 30% Brown, 20% Yellow, 20% Red, 10% Green, 10% Orange, 10% Tan. Afterward it was 24% Blue , 20% Green, 16% Orange, 14% Yellow, 13% Red, 13% Brown.

Suppose a friend of mine has two bags of M&M's, and he tells me that one is from 1994 and one from 1996. He won't tell me which is which, but he gives me one M&M from each bag. One is yellow and one is green. What is the probability that the yellow one came from the 1994 bag?

This problem is similar to the cookie problem, with the twist that I draw one sample from each bowl/bag. This problem also gives me a chance to demonstrate the table method, which is useful for solving problems like this on paper. In the next chapter we will solve them computationally.

The first step is to enumerate the hypotheses. The bag the yellow M&M came from I'll call Bag 1; I'll call the other Bag 2. So the hypotheses are:

- A: Bag 1 is from 1994, which implies that Bag 2 is from 1996.
- B: Bag 1 is from 1996 and Bag 2 from 1994.