Detección de animales

José Salvador Rico Mercado

21 de septiembre de 2014

Agradecimientos: Tere :)

Índice General

1 Bla, bla, bla

Aquí hay algunos ejemplos de lo chido que está el notebook y cómo se ve en el pdf. Lo chido es que ya no hay que preocuparse por el .tex ni nada de eso. Lo único que hay que poner aquí son las citas bibliográficas porque obviamente esas no se ponen solas jaja. Checa este libro generado en IPython: Probabilistic Programming and Bayesian Methods. Está chingón!

Dame tus críticas!

1.1 The Philosophy of Bayesian Inference

> You are a skilled programmer, but bugs still slip into your code. After a particularly difficult implementation of an algorithm, you decide to test your code on a trivial example. It passes. You test the code on a harder problem. It passes once again. And it passes the next, *even more difficult*, test too! You are starting to believe that there may be no bugs in this code... [1]

1.1.1 The Bayesian state of mind

Consider the following examples demonstrating the relationship between individual beliefs and probabilities:

- I flip a coin, and we both guess the result. We would both agree, assuming the coin is fair, that the probability of Heads is 1/2. Assume, then, that I peek at the coin. Now I know for certain what the result is: I assign probability 1.0 to either Heads or Tails (whichever it is). Now what is *your* belief that the coin is Heads? My knowledge of the outcome has not changed the coin's results. Thus we assign different probabilities to the result.

- Your code either has a bug in it or not, but we do not know for certain which is true, though we have a belief about the presence or absence of a bug.

- A medical patient is exhibiting symptoms x, y and z. There are a number of diseases that could be causing all of them, but only a single disease is present. A doctor has beliefs about which disease, but a second doctor may have slightly different beliefs. [2]

For example, consider the posterior probabilities (read: posterior beliefs) of the above examples, after observing some evidence X:

1. $P(A)$: the coin has a 50 percent chance of being Heads. $P(A|X)$: You look at the coin, observe a Heads has landed, denote this information X, and trivially assign probability 1.0 to Heads and 0.0 to Tails.

2. $P(A)$: This big, complex code likely has a bug in it. $P(A|X)$: The code passed all X tests; there still might be a bug, but its presence is less likely now. [3]

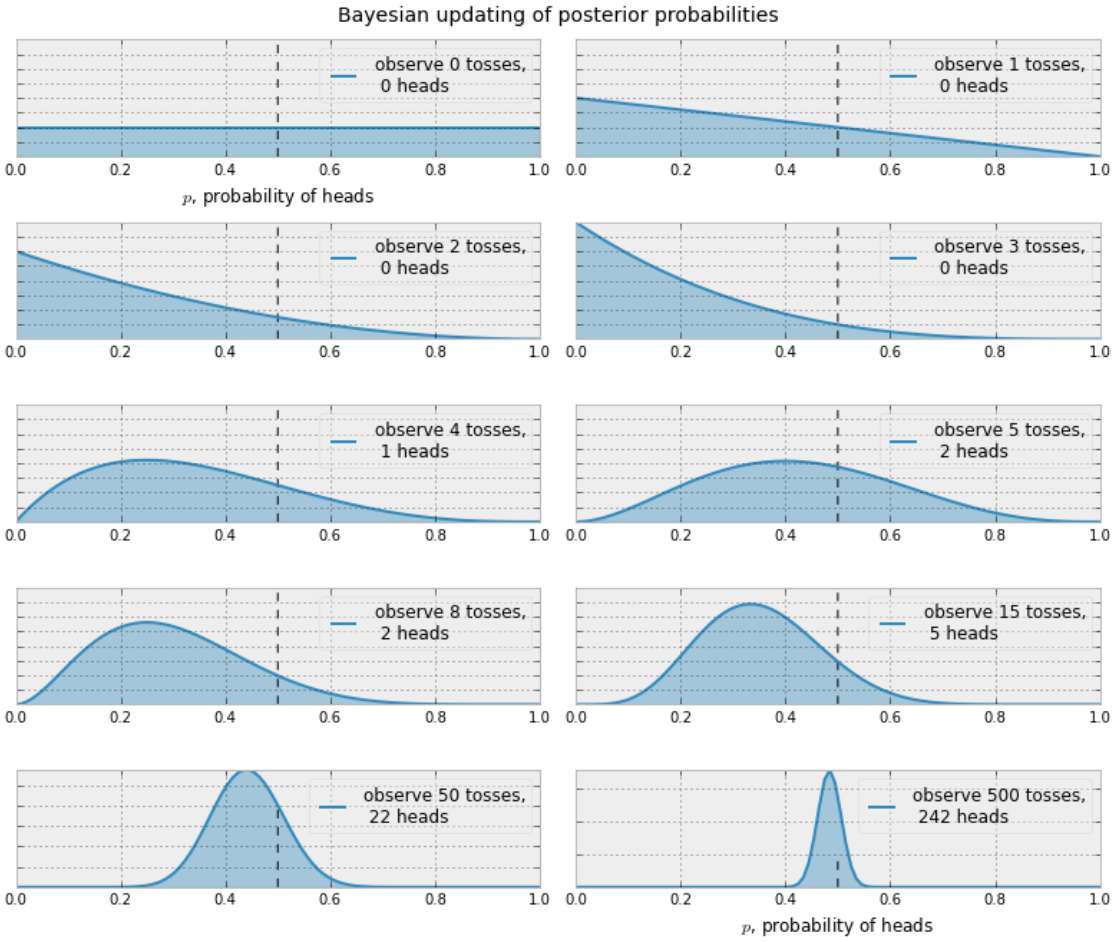
3. $P(A)$: The patient could have any number of diseases. $P(A|X)$: Performing a blood test generated evidence X, ruling out some of the possible diseases from consideration. [4]

1.1.2 Our Bayesian framework

Secondly, we observe our evidence [5, 6, 7]

```
In [5]: """
        The book uses a custom matplotlibrc file, which provides the unique styles for
        matplotlib plots. If executing this book, and you wish to use the book's
        styling, provided are two options:
            1. Overwrite your own matplotlibrc file with the rc-file provided in the
               book's styles/ dir. See http://matplotlib.org/users/customizing.html
            2. Also in the styles is  bmh_matplotlibrc.json file. This can be used to
               update the styles in only this notebook. Try running the following code:

                import json, matplotlib
                s = json.load( open("../styles/bmh_matplotlibrc.json") )
```

```
        matplotlib.rcParams.update(s)

"""

# The code below can be passed over, as it is currently not important, plus it
# uses advanced topics we have not covered yet. LOOK AT PICTURE, MICHAEL!
%matplotlib inline
from IPython.core.pylabtools import figsize
import numpy as np
from matplotlib import pyplot as plt
figsize(11, 9)

import scipy.stats as stats

dist = stats.beta
n_trials = [0, 1, 2, 3, 4, 5, 8, 15, 50, 500]
data = stats.bernoulli.rvs(0.5, size=n_trials[-1])
x = np.linspace(0, 1, 100)

# For the already prepared, I'm using Binomial's conj. prior.
for k, N in enumerate(n_trials):
    sx = plt.subplot(len(n_trials) / 2, 2, k + 1)
    plt.xlabel("$p$, probability of heads") \
        if k in [0, len(n_trials) - 1] else None
    plt.setp(sx.get_yticklabels(), visible=False)
    heads = data[:N].sum()
    y = dist.pdf(x, 1 + heads, 1 + N - heads)
    plt.plot(x, y, label="observe %d tosses,\n %d heads" % (N, heads))
    plt.fill_between(x, 0, y, color="#348ABD", alpha=0.4)
    plt.vlines(0.5, 0, 4, color="k", linestyles="--", lw=1)

    leg = plt.legend()
    leg.get_frame().set_alpha(0.4)
    plt.autoscale(tight=True)


plt.suptitle("Bayesian updating of posterior probabilities",
             y=1.02,
             fontsize=14)

plt.tight_layout()
```



2 Bla, bla, bla

2.1 No es necesario todo el código

```
In [14]: # Habrá codigo que sea muy largo y no tan importante y lo podemos poner en el directorio
         %run imprime.py
```

```
A esto me refiero con no poner todo el codigo en el documento. Este archivo puede estar en el directorio
```

2.1.1 Se pueden importar imágenes

```
In [3]: from IPython.display import Image
        Image('http://www.funnyjunksite.com/pictures/funnypics/animals/monkey/funny_monkey_picture_92.jp
```

```
Out[3]:
```

2.2 Esto de Cython está bastante chido. Con poco esfuerzo corre mucho más rápido.

```
In [5]: %load_ext cythonmagic

In [6]: def f(x):
            return x**2-x

        def integrate_f(a, b, N):
            s = 0; dx = (b-a)/N
            for i in range(N):
                s += f(a+i*dx)
            return s*dx

In [7]: %%cython
        cdef double fcy(double x) except? -2:
            return x**2-x

        def integrate_fcy(double a,double b,int N):
            cdef int i
            cdef double s, dx
            s = 0; dx = (b-a)/N
            for i in range(N):
                s += fcy(a+i*dx)
            return s*dx

In [8]: %timeit integrate_f(0, 1, 100)
        %timeit integrate_fcy(0, 1, 100)
```

```
10000 loops, best of 3: 44.2 µs per loop
1000000 loops, best of 3: 747 ns per loop
```

Bibliografía

[1] Tony Hey, Stewart Tansley, and Kristin Tolle, editors. *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research, Redmond, Washington, 2009.

[2] O. Heffernan. 'Climategate' scientist speaks out. *Nature*, 463(7283):860, 2010.

[3] J. Couzin-Frankel. Cancer research. As questions grow, Duke halts trials, launches investigation. *Science*, 329(5992):614–5, 2010.

[4] F. Pérez and B. E. Granger. IPython: a System for Interactive Scientific Computing. *Computing in Science & Engineering*, 9(3):21–29, May 2007. URL: http://ipython.org.

[5] F. Pérez, B. E. Granger, and J. D. Hunter. Python: an ecosystem for scientific computing. *Computing in Science & Engineering*, 13(2):13–21, 2011.

[6] Frederic Brochu, Ulrik Egede, J. Elmsheuser, K. Harrison, R. W. L. Jones, H. C. Lee, Dietrich Liko, A. Maier, Jakub T. Moscicki, A. Muraru, GlenÑ. Patrick, Katarina Pajchel, W. Reece, B. H. Samset, M. W. Slater, A. Soroko, C. L. Tan, and Daniel C. Vanderster. Ganga: a tool for computational-task management and easy access to grid resources. *CoRR*, abs/0902.2685, 2009.

[7] Science Software Branch at the Space Telescope Science Institute. Space Telescope Science Institute stsci_python. URL: http://www.stsci.edu/institute/software_hardware/pyraf/stsci_python.