8/23/19

Physics 8805: Learning from Data Lecture 2

Handouts: Extra copies of simple_sum_product_rule.ipynb
Copies of medical_example_by_Bayes.ipynb

Before class: Set up projector with laptop and notebooks for modules 0 and 1. Put up Canvas 8805 Modules.

Onboard:
- Give feedback!!
- Questions on Anaconda installation?
- Questions on Python or Jupyter notebooks?
- Questions on course logistics?
- Have you used git? Google "Furnstahl github 8805" for repository

Overview:
- We took a quick look at continuous pdfs, but before looking further (finishing ⑧), we'll put sum and product rules on the table.

- Do ⑨ and ⑯

- Class: fill out simple_sum_product_rule.ipynb in note book or hardcopy.
- Brief guidance on how you might fill in the table
  - f-string print(f'ratio = {ratio:.3f}') or npround(number, digits)
  - How do you use numpy arrays?
  - experts: write function to work with either ints or floats or numpy arrays.
- Recap of simple_sum_product_rule.
- medical_example_by_Bayes for homework.

- Return to exploring_pdfs.ipynb and finish ⑧
  identify these ↙
  Take-aways: Bayesian confidence intervals, various "point estimates", characteristics of different pdfs (eg. symmetry, heavy tails, ...), what "sampling" means, projected posteriors, ...

8/23/19

Bayesian updating via Bayes' Theorem

$$\underbrace{p(\theta|\text{data},I)}_{\text{posterior}} = \frac{\overbrace{p(\text{data}|\theta,I)}^{\text{likelihood}} \times \overbrace{p(\theta|I)}^{\text{prior}}}{p(\text{data}|I)}$$

θ: general vector of parameters

$p(\text{data}|I)$: data probability (or "fully marginalized likelihood" or evidence or ...)

$\Rightarrow$ more on denominator later, but often just a normalization.

I: information we have (or believe) about θ before we observe the data.

- The prior pdf is what information we have (or believe) about θ before we observe the data.
- The posterior pdf is our new pdf, given that we have observed the data.
- So Bayes' Theorem tells us how to update our expectations.
- * Coin tossing example to illustrate updating:
  Bayesian_updating_coinflip_interactive.ipynb

Storyline: We are observing successive flips of a coin (or any binary process). There is a definite true probability of getting heads, $(p_h)_{\text{true}}$ but we don't know what it is, other than it is between 0 and 1.

- We characterize our information about p_h as a pdf.
- Before any flips, we start with a preconceived notion of the probability, this is the prior pdf $p(p_h|I)$ [I is any info we have]
- With each flip of the coin, we gain additional information, so we update our expectation of p_h, so we find the posterior:

$$p(p_h | \underbrace{\#\text{tosses}}_{N}, \underbrace{\#\text{heads}}_{R}, I)$$

- note that outcome is discrete but p_h is continuous as $p_h \leq 1$.

8/23/19

Let's first play a bit with the simulation and then come back and think of the details.

- Note a few of the Python features (class structure)
  - class for data called Data. Easy compared to C++!
  - function to make a plot that is made repeatedly
  - elaborate widget ⇒ use as guide for making your own!
    (read from bottom up to understand the structure)

- User interface features
  - tabs to control parameters or look at documentation
  - set the true p_h by the slider
  - press "Next" to flip "jump" # of times. Reset to go back to beginning
  - plot shows updating from three different initial prior pdfs.

- Class: tell your neighbor how to interpret each of the priors.
  - uniform prior: any probability is equally likely. Is this uninformative?
  - informative prior: we have reason to believe the coin is fair.
  - anti-prior: Could be anything but most likely a two-headed or two-tailed coin.

What is minimal common info? ⇒ $0 \le p_h \le 1$ and $\int p(p_h)\,dp_h = 1$

- Things to try:
  - First one flip at a time. How do you understand the changes intuitively?
  - What happens with more and more tosses?
  - Try different values of the true p_h.

- What happens with enough data? All converge to narrow pdf including p_h true (but not model!)
- Which prior(s) get to the correct conclusion fastest for $p_h = .1, .9, .5$?
- Does it matter is you update after every toss or all at once?
  - We'll need to look at details!

- What about the difference between a particular ordering of data for R heads versus any order?

6/23/19

Suppose we had a fair coin $\Rightarrow p_h = 0.5$

$$\Rightarrow p(R \text{ heads out of } N \text{ tosses} \mid \text{fair coin}) = \binom{N}{R}(0.5)^R(0.5)^{N-R}$$

$$\Rightarrow p(R, N \mid p_h = 0.5)$$

combinations $= \frac{N!}{R!(N-R)!}$

Is the sum rule obeyed?

$$\sum_{R=0}^{N} p(R, N \mid p_h = \tfrac{1}{2}) = \sum_{R=0}^{N} \binom{N}{R}\left(\tfrac{1}{2}\right)^N = \left(\tfrac{1}{2}\right)^N \cdot 2^N = 1 \checkmark$$

[proof: $(x+y)^N = \sum_{R=0}^{N} \binom{N}{R} x^R y^{N-R} \xrightarrow{x=y=1} 2^N = \sum_{R=0}^{N} \binom{N}{R}$]

More general p_h! ($x = p_h$, $y = 1 - p_h$ $\Rightarrow$ sum rule works in general!)

$$p(R, N \mid p_h) = \binom{N}{R} p_h^R (1 - p_h)^{N-R}$$

But we want the other way around: $p(p_h \mid R, N)$

Bayes says: $p(p_h \mid R, N) \propto p(R, N \mid p_h) \cdot p(p_h)$

(denominator just normalizes, doesn't depend on p_h. So we don't need to distinguish particular ordering.)

Claim: Can do sequentially or all at once. When is this true? (When the tosses are independent)

What would happen if you tried to update using the same results over and over again?

8/23/19

- So how are we doing the calculation of the updated posterior?
  - In this case, we can do analytic calculations.

Case I: uniform (flat) prior

$$\Rightarrow p(p_h | R, N, I) = N\, p(R, N | p_h)\, \cancel{p(p_h)}^{1} = N\, p_h^R (1-p_h)^{N-R}$$

(suppress I)

But $\int_0^1 dp_h\, p(p_h | R, N) = 1 \Rightarrow N \dfrac{\Gamma(1+N-R)\Gamma(1+R)}{\Gamma(2+N)}$

[Recall Beta function $B(x,y) = \int_0^1 t^{x-1}(1-t)^{y-1}\,dt = \dfrac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)}$ for $\mathrm{Re}\, x, y > 0$

and $\Gamma(x) = (x-1)!$ for integer x]

$$\Rightarrow N = \frac{\Gamma(2+N)}{\Gamma(1+N-R)\Gamma(1+R)}$$ and we can trivially update.

Case II: conjugate prior: posterior has same form as prior.
Here if we pick a beta distribution as prior, it is conjugate.
From scipy.stats.beta documentation:

$$\text{i.e. } p(x|a,b) \to f(x,a,b) = \frac{\Gamma(a+b)x^{a-1}(1-x)^{b-1}}{\Gamma(a)\Gamma(b)} \Rightarrow \text{likelihood is } f(p_h, 1+R, 1+N-R)!$$

If prior $p(p_h|I) = f(p_h, \alpha, \beta)$, then normalized posterior

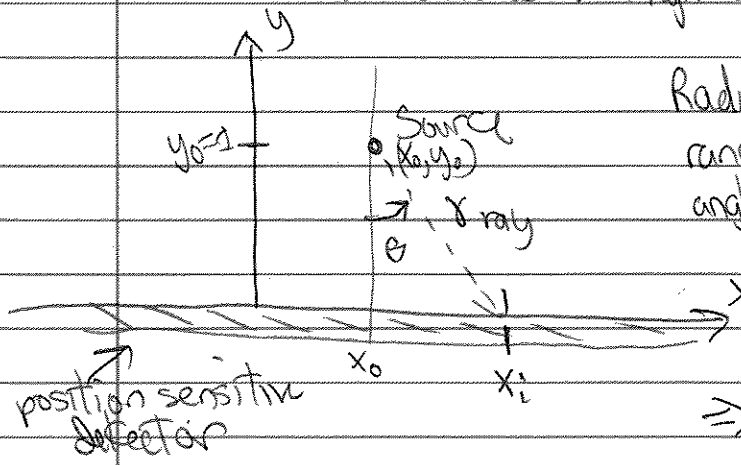
$$p(p_h|R,N) \propto p(R,N|p_h)\,p(p_h) = f(p_h, \alpha+R, \beta+N-R)$$

So update by $\alpha \to \alpha+R$, $\beta \to \beta+N-R$!

Check against code! Valuable to use conjugate prior if it is

8/23/19

First look at the radioactive lighthouse problem
⇒ radioactive_lighthouse_exercise.ipynb



Radioactive source emits gamma rays randomly in time but uniformly in angle. The source is at (x_0, y_0).

Gamma rays are detected on the x-axis and positions recorded
⇒ $x_1, x_2, x_3, \ldots x_N \Rightarrow \{x_k\}$

Goal: Given the $\{x_k\}$, estimate (x_0, y_0).
1st pass: Take $y_0=1$ as given, so only estimate x_0.

Naively, how would you estimate x_0 given $x_1, \ldots, x_N$? (Take average)
Let's compare the naive approach to a Bayesian approach.

Follow the notebook leading questions for the Bayesian estimate!