question 2 views

Daily Challenge 25.3

~~(Due: Monday 3/25 at 12:00 noon Eastern)~~
(**Due: Tuesday 3/26 at 12:00 noon Eastern**)

A Bernoulli trial is a random variable with two possible outcomes, called "success" and "failure", and such that the probability of success is p (thus the probability of failure is $1-p$).

For instance, you can think of flipping an unfair coin which comes up heads with probability p as a single Bernoulli trial.

A classic problem in statistics is to *infer* the probability p of success in a Bernoulli trial by observing subsequent outcomes. Concretely, imagine flipping a (potentially) unfair coin repeatedly, and trying to guess what the "fairness value" p is.

We showed in the last two meetings that, using Bayes' rule, if we begin with a uniform prior $f_P(p) = \begin{cases} 1 & \text{if } 0 \le p \le 1 \\ 0 & \text{otherwise} \end{cases}$, then after observing N trials with h heads (or "successes") and $t = N - h$ tails ("failures"), our updated PDF is

$$\begin{aligned} f(p \mid H = h) &= \frac{\mathbb{P}(H = h \mid p) \cdot f_P(p)}{\mathbb{P}(H = h)} \\ &= \frac{\binom{N}{h} p^h (1-p)^{N-h} \cdot 1}{\int_0^1 \binom{N}{h} q^h (1-q)^{N-h} \, dq} \\ &= \frac{p^h (1-p)^t}{B(h+1, t+1)}, \end{aligned}$$

where in the last step we have recalled the definition of the beta function,

$$B(\alpha, \beta) = \int_0^1 u^{\alpha-1} (1-u)^{\beta-1} \, du.$$

We say that p follows a *beta distribution* with *shape parameters* $\alpha = h+1$ and $\beta = t+1$, and write $p \sim \text{Beta}(h+1, t+1)$ for short.

Note that the beta distribution is different from the beta function! The beta *function* is $B(\alpha, \beta) = \int_0^1 u^{\alpha-1}(1-u)^{\beta-1} \, du$, but the beta *distribution* is $f(x; \alpha, \beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)}$.

(**Part a**) Suppose we have already observed h_0 heads and t_0 tails, so our prior PDF is

$$f_P(p) = \frac{p^{h_0}(1-p)^{t_0}}{B(h_0+1, t_0+1)}.$$

Now we keep flipping and observe h' additional heads and t' additional tails. Use Bayes' theorem to compute our posterior PDF over p, and show that it is now

$$f_{P|H'}(p \mid h', t') = \frac{p^{h_0+h'}(1-p)^{t_0+t'}}{B(h_0+h'+1, t_0+t'+1)}.$$

[Hint: just plug into Bayes' theorem, but replace the uniform prior $f_P(p) = 1$ we had before with the beta prior. Your expression will involve a ratio of beta functions which cancel nicely.]

(**Part b**) Now open Python. First write a function that flips an unfair coin with probability p a total of N times and stores the results in a list.

```
def unfair_flip(p = 0.5, N=100):
    """
    Flips a coin with probability p of getting heads a total of N times
    """

    results_list = []

    ## Your code here

    return results_list
```

Hint: you can either use np.random.choice with the optional `p` argument, or if you want to implement it yourself, pick a random number uniformly between 0 and 1, then return tails if the number is bigger than p and heads otherwise.

(**Part c**) Now write a function that takes in a list of coin flip outcomes, and a time at which to compute your posterior PDF, and plots the PDF using the Bayesian update described above.

That is: your function takes in a list of "heads or tails" outcomes, counts the number h of heads and t of tails up to a certain time, and then plots the appropriate beta distribution.

```
def plot_posterior(outcomes, time):
    """
    Generates a plot of the Bayesian-updated posterior PDF over p after the given time
    """

    ## (1) Slice the list of outcomes up to the given time
    ## (2) Count the number of heads h and tails t in the sliced list
    ## (3) Define the appropriate posterior f(p) = ( p^{h} ( 1 - p )^{t} )/( B ( h + 1 , t + 1 )
    ## (4) Plot f(p) on [0,1] using matplotlib
```

Use scipy.special.beta for the beta function in the denominator.

(**Part d**) Choose a particular value of p and run a large number of flips, say 500.

Plot your posterior PDF at a few illustrative points:

- At time 0, the PDF is uniform (flat on $[0, 1]$).
- At time 1, you've seen only a single heads or tails, so the PDF is highly biased to either the left or right.
- At an intermediate time (a few dozen), the PDF should be a reasonable-width Gaussian close to the true p.
- At late times ($t \approx 500$), the PDF should have converged to a narrow Gaussian around the true p.

Push.

daily_challenge

Updated 19 days ago by Christian Ferko

the students' answer, *where students collectively construct a single answer*

woah

~ An instructor (Christian Ferko) endorsed this answer ~

Updated 18 days ago by Logan Pachulski

the instructors' answer, *where instructors collectively construct a single answer*

(**Part a**) As before, we used Bayes' theorem for a continuous variable (the probability p of heads) conditioned on a discrete outcome (the observations h' and t'), namely

$$\begin{aligned} f(p \mid H = h') &= \frac{\mathbb{P}(H = h' \mid p) \cdot f_P(p)}{\mathbb{P}(H = h')} \\ &= \frac{\mathbb{P}(H = h' \mid p) \cdot f_P(p)}{\int_0^1 \mathbb{P}(H = h' \mid q) \cdot f_P(q)\, dq}, \end{aligned}$$

where we have used the law of total probability in the denominator.

Let's make sure we understand all of the pieces.

1. The factor $\mathbb{P}(H = h' \mid p)$ in the numerator is the *conditional probability* of obtaining h' heads, given that the probability of heads on a flip is p. This is the usual Bernoulli distribution,

   $$\mathbb{P}(H = h' \mid p) = \binom{N'}{h'} p^{h'} (1-p)^{t'}$$

   where $N' = h' + t'$.

2. The expression $f_P(p)$ is our *prior probability distribution* for the probability p of a heads. Since we have assumed that we already flipped the coin N_0 times and got h_0 heads and t_0 tails, our prior is a beta distribution with shape parameters $\alpha = h_0 + 1$ and $\beta = t_0 + 1$:

   $$f_P(p) = \frac{p^{h_0}(1-p)^{t_0}}{B(h_0 + 1, t_0 + 1)},$$

3. In the denominator, we integrate over all possible values of q, but the object in the integrand is again the product of a Bernoulli likelihood and a beta distribution.

   $$\int_0^1 \mathbb{P}(H = h' \mid q) \cdot f_P(q)\, dq = \int_0^1 \underbrace{\binom{N'}{h'} q^{h'} (1-q)^{t'}}_{\mathbb{P}(H = h' \mid q)} \cdot \underbrace{\left(\frac{q^{h_0}(1-q)^{t_0}}{B(h_0 + 1, t_0 + 1)} \right)}_{f_P(q)} dq$$

   where note that we have replaced all p's with q's, since q is what appears in the arguments.

Putting these pieces together, our posterior is

$$f(p \mid H = h') = \left(\binom{N'}{h'} p^{h'} (1-p)^{t'} \right) \cdot \left(\frac{1}{\int_0^1 \binom{N'}{h'} q^{h'} (1-q)^{t'} \cdot \left(\frac{q^{h_0}(1-q)^{t_0}}{B(h_0+1, t_0+1)} \right) \, dq} \right) \cdot \left(\frac{p^{h_0}(1-p)^{t_0}}{B(h_0+1, t_0+1)} \right),$$

Although this looks horrible, it simplifies dramatically. The $\binom{N'}{h'}$ factors cancel top and bottom, while the $B(h_0 + t, t_0 + 1)$ in the denominator of the denominator is a constant that can be pulled out of the integral. Meanwhile, the powers of p and $(1-p)$ combine upstairs by power rules. This gives

$$\begin{aligned} f(p \mid H = h') &= \left(p^{h'+h_0}(1-p)^{t'+t_0} \right) \cdot \left(\frac{B(h_0+1, t_0+1)}{\int_0^1 q^{h'}(1-q)^{t'} \cdot q^{h_0}(1-q)^{t_0} \, dq} \right) \cdot \left(\frac{1}{B(h_0+1, t_0+1)} \right) \\ &= \left(p^{h'+h_0}(1-p)^{t'+t_0} \right) \cdot \left(\frac{B(h_0+1, t_0+1)}{B(h_0+h'+1, t_0+t'+1)} \right) \cdot \left(\frac{1}{B(h_0+1, t_0+1)} \right), \end{aligned}$$

where we have recognized the definition of the beta function in the denominator. But now the betas cancel in the second and third factors, leaving us with

$$\begin{aligned} f(p \mid H = h') &= \frac{p^{h'+h_0}(1-p)^{t'+t_0}}{B(h_0+h'+1, t_0+t'+1)} \\ &\sim \text{Beta}(h_0+h'+1, t_0+t'+1), \end{aligned}$$

which is the definition of the beta distribution with new shape parameters $\alpha = h_0 + h' + 1, \beta = t_0 + t' + 1$

Punchline: if you start with a beta distribution, Bayesian updating is easy! You only need to keep track of the total number of heads and tails, and then your posterior at any step is just $\text{Beta}(h+1, t+1)$

(**Parts b-d)** See the notebook here, also exported as a standalone document here. I've copied the relevant code below for convenience.

```
def unfair_flip(p = 0.5, N=100):
    """
    Flips a coin with probability p of getting heads a total of N times
    """

    results_list = [ np.random.choice([0, 1], p=[ 1-p, p ] ) for i in range(N)]

    return results_list

def plot_posterior(outcomes, time):
    """
    Generates a plot of the Bayesian-updated posterior PDF over p after the given time
    """

    sliced_outcomes = outcomes[:time]
    heads_count = sum(sliced_outcomes)
    tails_count = time - heads_count

    ## Sanity check:
    assert(tails_count == sum( [outcome==0 for outcome in sliced_outcomes ] ) )

    def my_posterior(p):
        return np.power(p, heads_count)*np.power(1-p, tails_count)/beta(heads_count + 1, tails_count + 1)

    ## Now generate the plot

    plot_x = np.linspace(0, 1, 500)
    plot_y = np.array( [my_posterior(x) for x in plot_x ] )

    plt.plot(plot_x, plot_y)
    plt.fill_between(plot_x, 0, plot_y, color="#aaaadd", alpha=0.5)

    return
```

Updated 19 days ago by Christian Ferko

followup discussions *for lingering questions and comments*