question 2 views

Daily Challenge 24.7

~~(Due: Tuesday 3/19 at 12:00 noon Eastern)~~
Pushed: (**Due: Wednesday 3/20 at 12:00 noon Eastern**)

This exercise will give you some more practice with the convolution, in addition to exposing you to some functional programming ideas in Python.

(1) Problem: implementing the discrete convolution.

As we saw, the convolution of two probability distributions f and g is defined by

$$(f * g)(z) = \int_{-\infty}^{\infty} f(x)g(z-x)\,dx.$$

We know that all probability distributions have total area 1, which means that they eventually go to zero at large positive x and large negative x. The set of x values where a probability distribution $f(x)$ is nonzero is called the **support** of f.

For this problem, we will assume that all probability distributions are supported on some interval $[-S, S]$. That means $f(x) = 0$ if $x > S$ or $x < -S$. If this is true, we can cut off the integral to

$$(f * g)(z) = \int_{-S}^{S} f(x)g(z-x)\,dx,$$

since the integrand is zero elsewhere.

How could we implement a discrete version of this convolution on a computer? First chop up the range $[-S, S]$ into a bunch of points x_i:

$$-S = x_0 < x_1 < \cdots < x_N = S.$$

It's easiest if we assume the spacing Δx is the same between any pair of adjacent points in the partition: that is, $x_i - x_{i-1} = \Delta x$.

Then approximate the integral with a Riemann sum:

$$(f * g)(z) \approx \sum_{i=0}^{N} f(x_i)g(z-x_i)\,\Delta x.$$

Now you'll implement this in Python

(**Part a**) Write a function `convolve(f , g, support=10)` which returns the convolution of two Python functions f and g, assuming they both go to zero outside the support.

I should emphasize that you are writing a function which takes in two functions and returns another function.

The skeleton of your code might look like this.

```
def convolve(f, g, support=10):
    '''
    Returns the convolution of two functions f and g

    We assume f(x) = 0 and g(x) = 0 if x>support or x<-support

    Inputs:
        f: a Python function that maps floats to floats and returns zero outside the support interval
        g: a Python function that maps floats to floats and returns zero outside the support interval
        support: the number S such that f and g vanish outside [-S, S]

    Returns:
        a new Python function which maps floats to floats and gives the convolution of f and g
    '''

    def h(z):
        ## Chop up the interval [-support, support] into a bunch of pieces x_i,
        ## compute the sum described above, then return the value of that sum

    return h
```

Hint: for chopping up the interval, I recommend using numpy's linspace function.

(**Part b**) Let's test out your convolution. We know that convolving two uniform distributions should give a triangle.

So run the following code in a cell:

```
import matplotlib.pyplot as plt, numpy as np

def uniform(x):
    """
    Uniform distribution: returns 1 if 0<=x<=1 and 0 otherwise
    """
    if 0<=x<=1:
        return 1
    else:
        return 0

h2 = convolve(uniform,uniform,support=1)

x = np.linspace(0, 2, 100)
y = np.array([h2(i) for i in x])

plt.plot(x,y)
```

If your convolution function is correct, it should look like a triangle.

(**Part c**) We also said that, if you convolve the uniform with itself more times, it looks more Gaussian.

Now run this cell:

```
h3 = convolve(h,uniform,support=2)

x = np.linspace(0, 3, 100)
y = np.array([h3(i) for i in x])

plt.plot(x,y)
```

This is the plot of the convolution of three uniforms. What does it look like? How close to Gaussian does it seem after adding just 3 random variables?

(**Part d**) We also know that convolving two exponentials gives something that looks schematically like Maxwell-Boltzmann.

1. Define a Python function `exponential(x, lam=1)` which returns $\lambda e^{-\lambda x}$ if $x > 0$ and returns zero if $x < 0$.
2. Convolve this function with itself, choosing an appropriate support.
3. Plot the convolution as we did above. Is the shape as you expected?

(Note that you cannot use `lambda` as a variable name in Python since this keyword is reserved for lambda functions.)

Upload your notebook to Github when done.

daily_challenge

Updated 25 days ago by Christian Ferko

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

woah

Updated 18 days ago by Logan Pachulski

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

My function looks like this:

```
def convolve(f, g, support=100):
    '''
    Returns the convolution of two functions f and g

    We assume f(x) = 0 and g(x) = 0 if x>support or x<-support

    Inputs:
```

```
        f: a Python function that maps floats to floats and returns zero outside the support interval
        g: a Python function that maps floats to floats and returns zero outside the support interval
        support: the number S such that f and g vanish outside [-S, S]

    Returns:
        a new Python function which maps floats to floats and gives the convolution of f and g
    '''

    partition = np.linspace(-support, support, 100*support)
    delta_x = partition[1] - partition[0]

    def h(z):
        return sum([f(x)*g(z-x)*delta_x for x in partition])

    return h
```

The notebook with my plots is convolution.ipynb.

Updated 28 days ago by Christian Ferko

followup discussions *for lingering questions and comments*