Convolution of piecewise defined functions

Indrek Mandre <indrek(at)mare.ee>

May 30, 2012

Abstract

I ran into the problem of calculating the PDF of a sum of values, each of different uniform distribution. As the PDF of the sum is the convolution of the individual PDF-s, I worked out the formulae for doing this. Alas, they turned out to be of no use to me, but maybe they'll be of assistance to you.

1 Convolution of two "one-piece" functions

Given two functions

$$F(x) = \begin{cases} f(x), & a_f < x < b_f; \\ 0, & \text{otherwise;} \end{cases} \tag{1}$$

$$G(x) = \begin{cases} g(x), & a_g < x < b_g; \\ 0, & \text{otherwise;} \end{cases} \tag{2}$$

where $b_f - a_f < b_g - a_g$ (we can suitably arrange the functions so as $F * G = G * F$). The convolution of F and G is then

$$(F * G)(x) = \int_{-\infty}^{\infty} F(y) G(x-y)\, dy = \tag{3}$$

$$= \int_{a_f}^{b_f} f(y) G(x-y)\, dy = \tag{4}$$

$$= \begin{cases} \int_{a_f}^{x-a_g} f(y) g(x-y)\, dy, & a_f + a_g < x < b_f + a_g; \\ \int_{a_f}^{b_f} f(y) g(x-y)\, dy, & b_f + a_g < x < a_f + b_g; \\ \int_{x-b_g}^{b_f} f(y) g(x-y)\, dy, & a_f + b_g < x < b_f + b_g; \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

2 Convolution of two generic piecewise functions

We are given two properly defined piecewise functions

$$F(x) = \begin{cases} f_1(x), & a_f^1 < x < b_f^1, \\ \cdots \\ f_N(x), & a_f^N < x < b_f^N, \\ 0, & \text{otherwise;} \end{cases} \tag{6}$$

$$G(x) = \begin{cases} g_1(x), & a_g^1 < x < b_g^1, \\ \cdots \\ g_M(x), & a_g^M < x < b_g^M, \\ 0, & \text{otherwise;} \end{cases} \tag{7}$$

where value intervals for individual functions don't overlap. We can rewrite $F(x)$ as a sum of "one-piece" functions:

$$F(x) = \sum_{i=1}^{N} F_i(x), \tag{8}$$

where

$$F_i(x) = \begin{cases} f_i(x), & a_f^i < x < b_f^i, \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

The same can be done for $G(x)$. Now taking the convolution of the two functions is a simple matter:

$$(F * G)(x) = \int_{-\infty}^{\infty} F(y) G(x-y)\, dy \tag{10}$$

$$= \sum_{i=1}^{N} \sum_{j=1}^{M} \int_{-\infty}^{\infty} F_i(y) G_j(x-y)\, dy, \tag{11}$$

$$= \sum_{i=1}^{N} \sum_{j=1}^{M} (F_i * G_j)(x), \tag{12}$$

where elements of the sum are calculated as described in the previous section.

3 Sample code

This is how one could use the convolution formulae in the Python/Sympy CAS.

```
import sympy
import sympy.abc

# Convolute two "one-piece" functions. Arguments F and G
# are tuples in form (h(x), a_h, b_h), where h(x) is
# the function and [a_h, b_h) is the range where the functions
# are non-zero.
def convolute_onepiece(x, F, G):
```

```
    f,a_f,b_f = F
    g,a_g,b_g = G
    # make sure ranges are in order, swap values if necessary
    if b_f - a_f > b_g - a_g:
      f, a_f, b_f, g, a_g, b_g = g, a_g, b_g, f, a_f, b_f
    y = sympy.Dummy('y')
    i = sympy.integrate(f.subs(x, y) * g.subs(x, x-y), y)
    return [
      (i.subs(y, x-a_g)-i.subs(y, a_f), a_f+a_g, b_f+a_g),
      (i.subs(y, b_f)-i.subs(y, a_f), b_f+a_g, a_f+b_g),
      (i.subs(y, b_f)-i.subs(y, x-b_g), a_f+b_g, b_f+b_g)]

# Two "flat" functions, uniform centered PDF-s.
F=(sympy.sympify(0.5), -1, 1)
G=(sympy.sympify(0.05), -10, 10)

print convolute_onepiece(sympy.abc.x, F, G)
```