Funções e Operadores Fuzzy

Jorge Vitor Gonçalves de Souza

October 1, 2022

Implementação em python das funções de pertinência, operadores de complemento, interseção e união de conjuntos fuzzy realizado para a disciplina de Inteligência Computacional(IC).

Bibliotecas

Para implementação foram utilizadas as bibliotecas NumPy e math . Para geração dos gráficos foi utilizada a matplotlib .

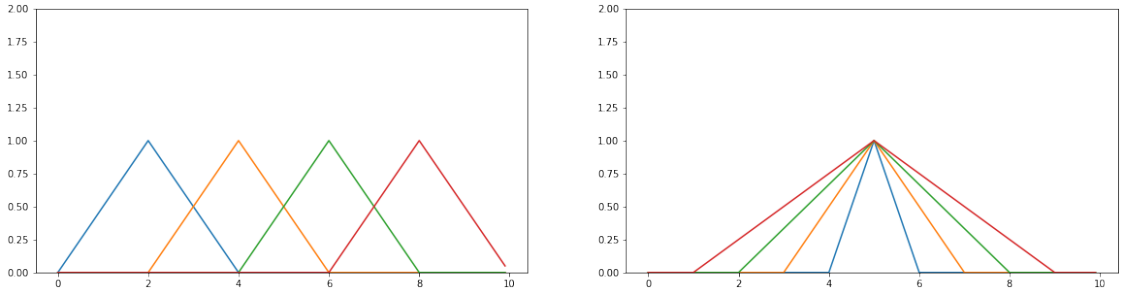
[1]:
```
import numpy as np
from math import e
import matplotlib.pyplot as plt
```

Funções de Pertinência

Triangular

[2]:
```
def triangular(i, a, m, b):
    y = list()
    for x in i:
        y.append(max(min((x - a)/(m - a), (b - x)/(b - m)), 0))
    return y

x = np.arange(0, 10, 0.1) #Declaração do eixo X

#plotagem do gráfico
plt.figure(figsize = ((20, 5)))
plt.subplot(1, 2, 1)
plt.ylim(0, 2)
plt.plot(x, triangular(x, 0, 2, 4))
plt.plot(x, triangular(x, 2, 4, 6))
plt.plot(x, triangular(x, 4, 6, 8))
plt.plot(x, triangular(x, 6, 8, 10))

plt.subplot(1, 2, 2)
plt.ylim(0, 2)
plt.plot(x, triangular(x, 4, 5, 6))
plt.plot(x, triangular(x, 3, 5, 7))
plt.plot(x, triangular(x, 2, 5, 8))
plt.plot(x, triangular(x, 1, 5, 9))
```

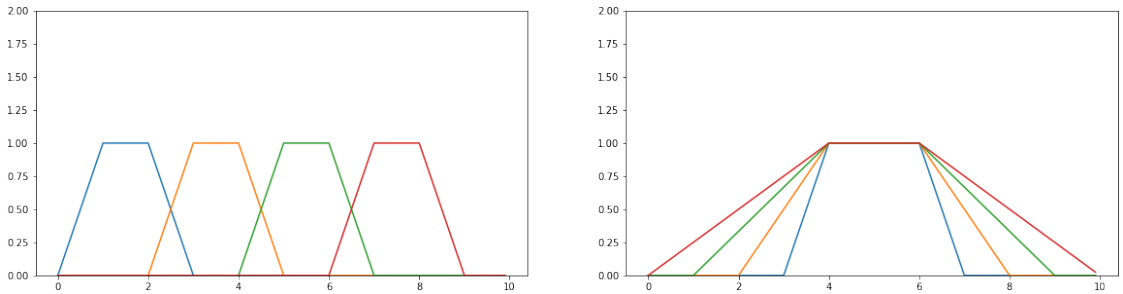
[2]: [<matplotlib.lines.Line2D at 0x7f3e224795b0>]



Trapezoidal

[3]:
```
def trapezoidal(i, a, m, n, b):
    y = list()
    for x in i:
        y.append(max(min((x - a)/(m - a), 1,(b - x)/(b - n)), 0))
    return y

#plotagem do gráfico
plt.figure(figsize = ((20, 5)))
plt.subplot(1, 2, 1)
plt.ylim(0, 2)
plt.plot(x, trapezoidal(x, 0, 1, 2, 3))
plt.plot(x, trapezoidal(x, 2, 3, 4, 5))
plt.plot(x, trapezoidal(x, 4, 5, 6, 7))
plt.plot(x, trapezoidal(x, 6, 7, 8, 9))

plt.subplot(1, 2, 2)
plt.ylim(0, 2)
plt.plot(x, trapezoidal(x, 3, 4, 6, 7))
plt.plot(x, trapezoidal(x, 2, 4, 6, 8))
plt.plot(x, trapezoidal(x, 1, 4, 6, 9))
plt.plot(x, trapezoidal(x, 0, 4, 6, 10))
```

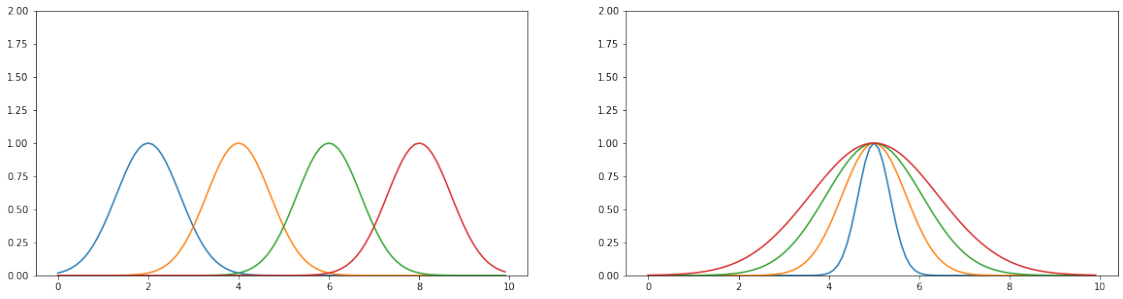
[3]: [<matplotlib.lines.Line2D at 0x7f3e223e5250>]



Gaussiana

[4]:
```
def gaussiana(i, k ,m):
    y = list()
    k = k/2
    for x in i:
        y.append(e**((-(x - m)**2)/(k**2)))
    return y

#plotagem do gráfico
plt.figure(figsize = ((20, 5)))
plt.subplot(1, 2, 1)
plt.ylim(0, 2)
plt.plot(x, gaussiana(x, 2, 2))
plt.plot(x, gaussiana(x, 2, 4))
plt.plot(x, gaussiana(x, 2, 6))
plt.plot(x, gaussiana(x, 2, 8))

plt.subplot(1, 2, 2)
plt.ylim(0, 2)
plt.plot(x, gaussiana(x, 1, 5))
plt.plot(x, gaussiana(x, 2, 5))
plt.plot(x, gaussiana(x, 3, 5))
plt.plot(x, gaussiana(x, 4, 5))
```

[4]:
```
[<matplotlib.lines.Line2D at 0x7f3e222c0d00>]
```



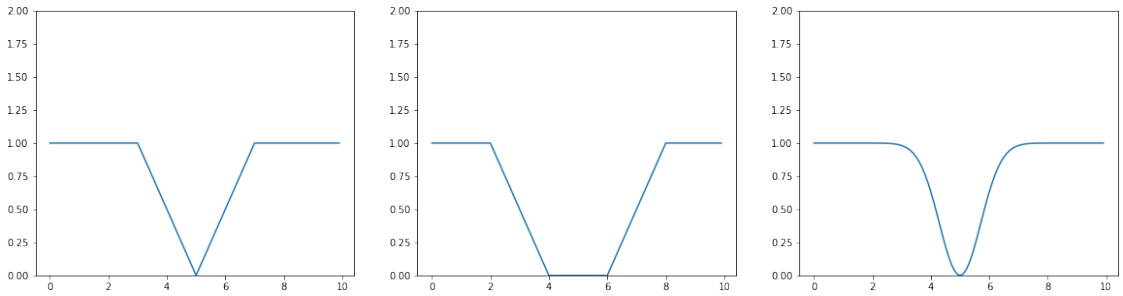
Operadores de Complemento

Zadeh

[5]:
```
def zadeh(a):
    y = list()
    for i in np.arange(0, len(a), 1):
```

```
        y.append(1 - a[i])
    return y

plt.figure(figsize = ((20, 5)))

plt.subplot(1, 3, 1)
plt.ylim(0, 2)
plt.plot(x, zadeh(triangular(x, 3, 5, 7)))

plt.subplot(1, 3, 2)
plt.ylim(0, 2)
plt.plot(x, zadeh(trapezoidal(x, 2, 4, 6, 8)))

plt.subplot(1, 3, 3)
plt.ylim(0, 2)
plt.plot(x, zadeh(gaussiana(x, 2, 5)))
```

[5]: `[<matplotlib.lines.Line2D at 0x7f3e21fee8e0>]`

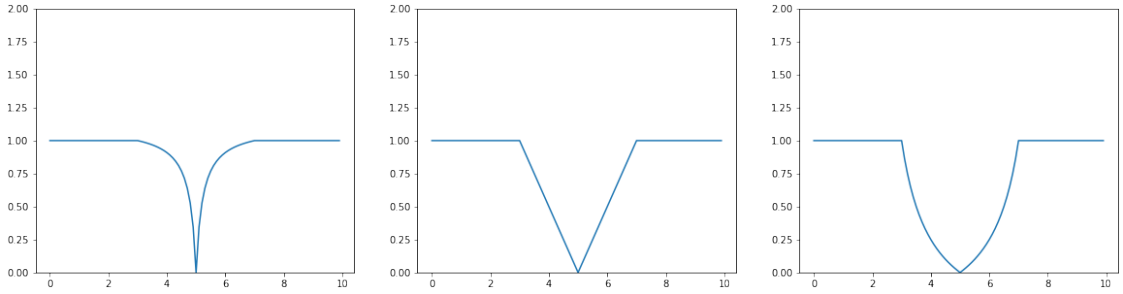


Sugeno

[6]:
```
def sugeno(a, s):
    y = list()
    for i in np.arange(0, len(a), 1):
        y.append((1 - a[i])/(1 + s*a[i]))
    return y

plt.figure(figsize = ((20, 5)))

plt.subplot(1, 3, 1)
plt.ylim(0, 2)
plt.plot(x, sugeno(triangular(x, 3, 5, 7), -0.9)) # -1 < s < 0

plt.subplot(1, 3, 2)
plt.ylim(0, 2)
plt.plot(x, sugeno(triangular(x, 3 , 5, 7), 0))   # s = 0
```

```
plt.subplot(1, 3, 3)
plt.ylim(0, 2)
plt.plot(x, sugeno(triangular(x, 3, 5, 7), 2))     # s > 0
```

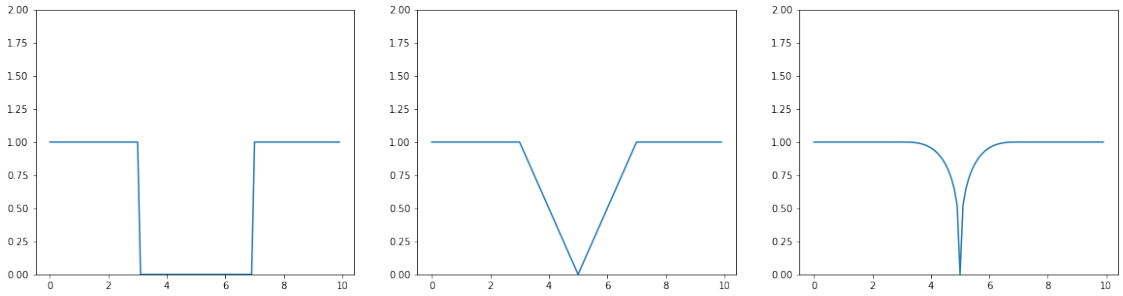
[6]: [<matplotlib.lines.Line2D at 0x7f3e21e3dfd0>]



Yager

[7]:
```
def yager(a, w):
    y = list()
    for i in np.arange(0, len(a), 1):
        y.append((1 - a[i]**w)**(1/w))
    return y

plt.figure(figsize = ((20, 5)))

plt.subplot(1, 3, 1)
plt.ylim(0, 2)
plt.plot(x, yager(triangular(x, 3, 5, 7), 0.001)) # w = 0.001

plt.subplot(1, 3, 2)
plt.ylim(0, 2)
plt.plot(x, yager(triangular(x, 3, 5, 7), 1))     # w = 1

plt.subplot(1, 3, 3)
plt.ylim(0, 2)
plt.plot(x, yager(triangular(x, 3, 5, 7), 3))     # w = 3
```

[7]: [<matplotlib.lines.Line2D at 0x7f3e21cf4370>]

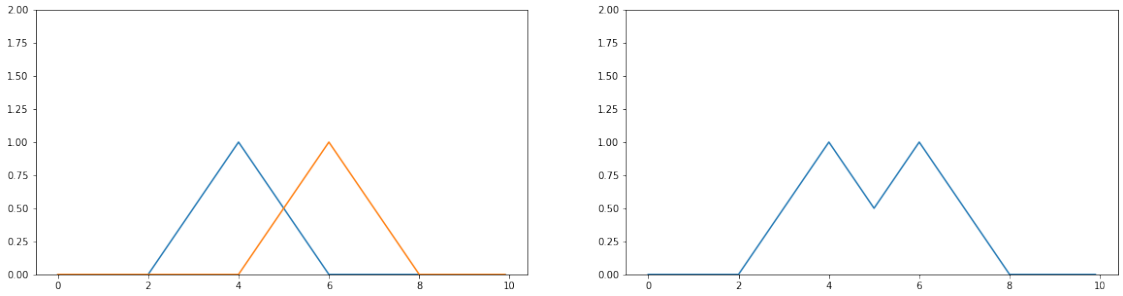


Operadores de União

Máximo

[8]:
```
def maximo(a, b):
    y = list()
    for i in np.arange(0, len(a), 1):
        y.append(max(a[i], b[i]))
    return y

plt.figure(figsize = ((20, 5)))

plt.subplot(1, 2, 1)
plt.ylim(0, 2)
plt.plot(x, triangular(x, 2, 4, 6))
plt.plot(x, triangular(x, 4, 6, 8))

plt.subplot(1, 2, 2)
plt.ylim(0, 2)
plt.plot(x, maximo(triangular(x, 2, 4, 6), triangular(x, 4, 6, 8)))
```

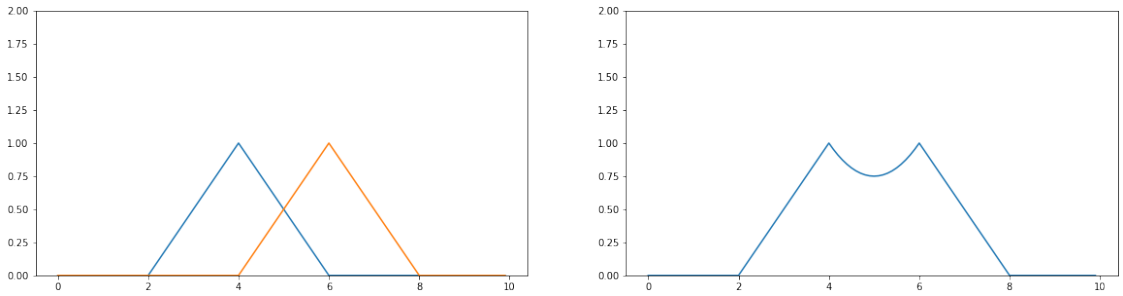
[8]:
```
[<matplotlib.lines.Line2D at 0x7f3e21c50880>]
```



Soma Probabilística

[9]:
```
def somaProbabilistica(a, b):
    y = list()
    for i in np.arange(0, len(a), 1):
        y.append((a[i] + b[i]) - (a[i]*b[i]))
    return y

plt.figure(figsize = ((20, 5)))

plt.subplot(1, 2, 1)
plt.ylim(0, 2)
plt.plot(x, triangular(x, 2, 4, 6))
plt.plot(x, triangular(x, 4, 6, 8))

plt.subplot(1, 2, 2)
plt.ylim(0, 2)
plt.plot(x, somaProbabilistica(triangular(x, 2, 4, 6), triangular(x, 4, 6, 8)))
```

[9]:
```
[<matplotlib.lines.Line2D at 0x7f3e21ba8070>]
```

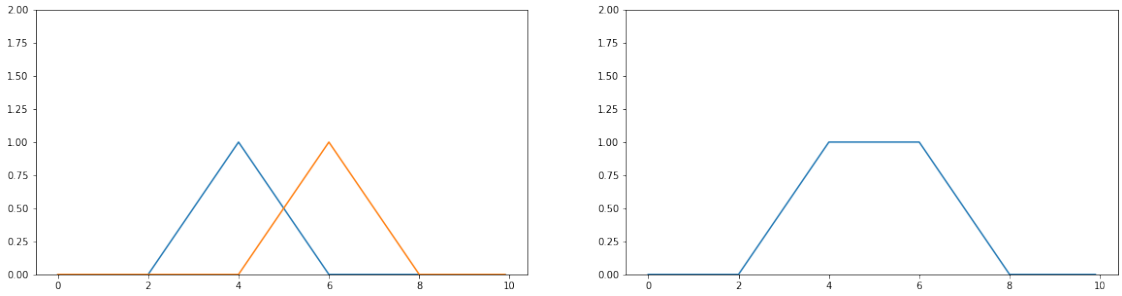


Soma Limitada

[10]:
```
def somaLimitada(a, b):
    y = list()
    for i in np.arange(0, len(a), 1):
        y.append(min(a[i]+b[i],1))
    return y

plt.figure(figsize = ((20, 5)))

plt.subplot(1, 2, 1)
plt.ylim(0, 2)
plt.plot(x, triangular(x, 2, 4, 6))
plt.plot(x, triangular(x, 4, 6, 8))

plt.subplot(1, 2, 2)
plt.ylim(0, 2)
```

```
plt.plot(x, somaLimitada(triangular(x, 2, 4, 6), triangular(x, 4, 6, 8)))
```

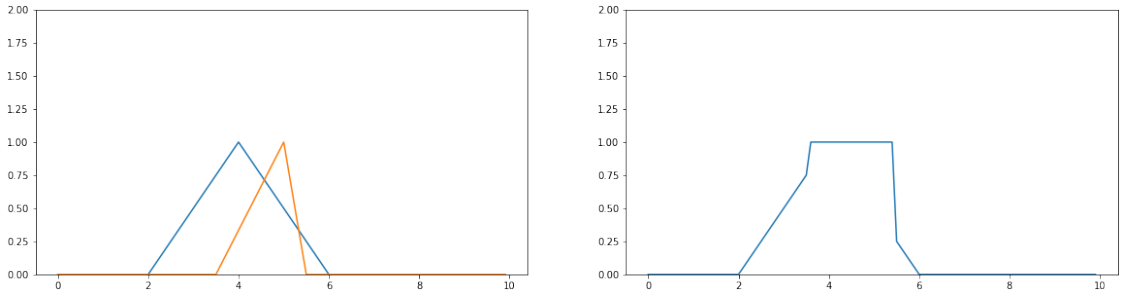
[10]: [<matplotlib.lines.Line2D at 0x7f3e21a740a0>]



Soma Drástica

[11]:
```
def somaDrastica(a, b):
    y = list()
    for i in np.arange(0, len(a), 1):
        if b[i] == 0:
            y.append(a[i])
        elif a[i] == 0:
            y.append(b[i])
        else:
            y.append(1)
    return y

plt.figure(figsize = ((20, 5)))
plt.subplot(1, 2, 1)
plt.ylim(0, 2)
plt.plot(x, triangular(x, 2, 4, 6))
plt.plot(x, triangular(x, 3.5, 5, 5.5))

plt.subplot(1, 2, 2)
plt.ylim(0, 2)
plt.plot(x, somaDrastica(triangular(x, 2, 4, 6), triangular(x, 3.5, 5, 5.5)))
```

[11]: [<matplotlib.lines.Line2D at 0x7f3e219b4c10>]

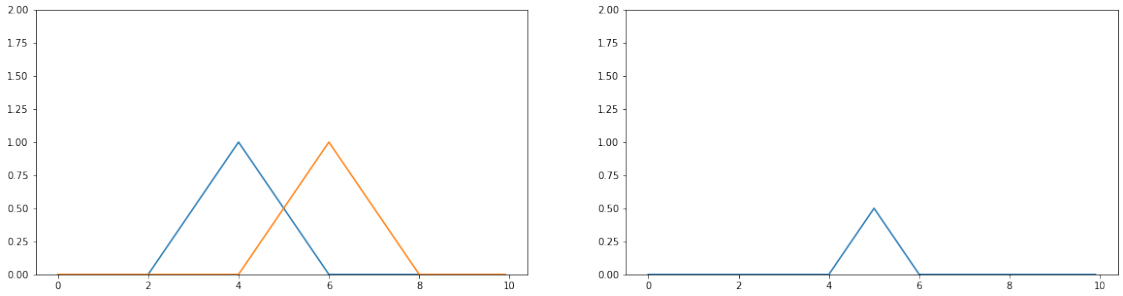


Operadores de Interseção

Mínimo

[12]:
```
def minimo(a, b):
    y = list()
    for i in np.arange(0, len(a), 1):
        y.append(min(a[i], b[i]))
    return y

plt.figure(figsize = ((20, 5)))

plt.subplot(1, 2, 1)
plt.ylim(0, 2)
plt.plot(x, triangular(x, 2, 4, 6))
plt.plot(x, triangular(x, 4, 6, 8))

plt.subplot(1, 2, 2)
plt.ylim(0, 2)
plt.plot(x, minimo(triangular(x, 2, 4, 6), triangular(x, 4, 6, 8)))
```

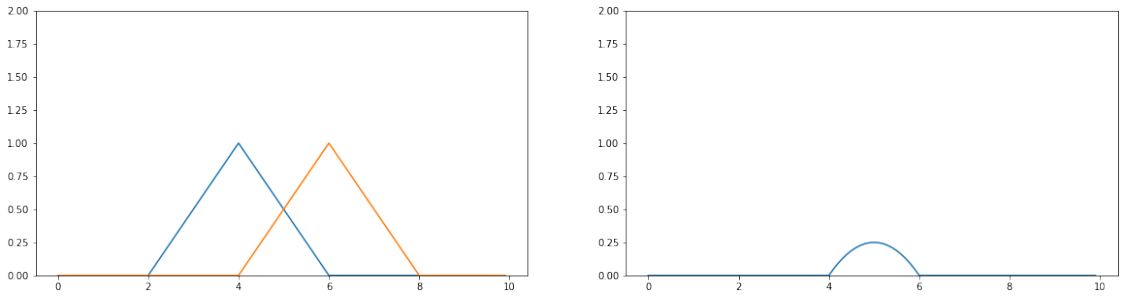
[12]:
```
[<matplotlib.lines.Line2D at 0x7f3e2190c040>]
```



Produto

[13]:
```
def produto(a, b):
    y = list()
    for i in np.arange(0, len(a), 1):
        y.append(a[i]*b[i])
    return y

plt.figure(figsize = ((20, 5)))

plt.subplot(1, 2, 1)
plt.ylim(0, 2)
plt.plot(x, triangular(x, 2, 4, 6))
plt.plot(x, triangular(x, 4, 6, 8))

plt.subplot(1, 2, 2)
plt.ylim(0, 2)
plt.plot(x, produto(triangular(x, 2, 4, 6), triangular(x, 4, 6, 8)))
```

[13]:
```
[<matplotlib.lines.Line2D at 0x7f3e2184fa00>]
```

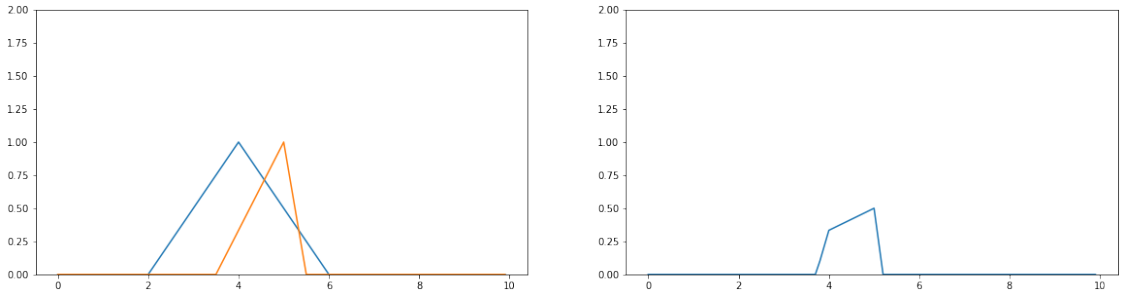


Produto Limitado

[14]:
```
def produtoLimitado(a, b):
    y = list()
    for i in np.arange(0, len(a), 1):
        y.append(max(0, (a[i]+b[i])-1))
    return y

plt.figure(figsize = ((20, 5)))

plt.subplot(1, 2, 1)
plt.ylim(0, 2)
plt.plot(x, triangular(x, 2, 4, 6))
plt.plot(x, triangular(x, 3.5, 5, 5.5))

plt.subplot(1, 2, 2)
plt.ylim(0, 2)
```

```
plt.plot(x, produtoLimitado(triangular(x, 2, 4, 6), triangular(x, 3.5, 5, 5.5)))
```

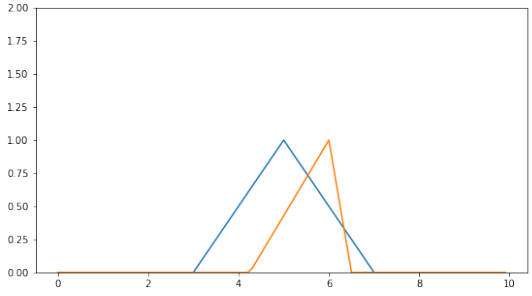
[14]: [<matplotlib.lines.Line2D at 0x7f3e217a2370>]

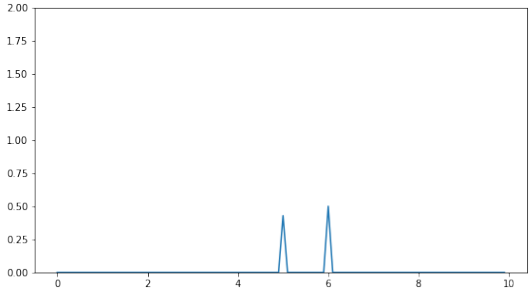


Produto Drástico

[15]:
```
def produtoDrastico(a, b):
    y = list()
    for i in np.arange(0, len(a), 1):
        if a[i] == 1:
            y.append(b[i])
        elif b[i] == 1:
            y.append(a[i])
        else:
            y.append(0)
    return y

plt.figure(figsize = ((20, 5)))

plt.subplot(1, 2, 1)
plt.ylim(0, 2)
plt.plot(x, triangular(x, 3, 5, 7))
plt.plot(x, triangular(x, 4.25, 6, 6.5))

plt.subplot(1, 2, 2)
plt.ylim(0, 2)
plt.plot(x, produtoDrastico(triangular(x, 3, 5, 7), triangular(x, 4.25, 6, 6.5)))
```

[15]: [<matplotlib.lines.Line2D at 0x7f3e216f20a0>]


2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
2
4
6
8
10


2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
2
4
6
8
10