

Atividade_3

September 25, 2018

1 4. Relações Fuzzy

```
In [1]: from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter
from matplotlib.collections import PolyCollection
from matplotlib.colors import colorConverter

import numpy as np
```

1.1 Criação dos conjuntos Fuzzy A e B

```
In [71]: # definição dos conjuntos fuzzy - representação por NUMPY ARRAY
A = np.array([[1,0], [2,0.2], [3,0.4], [4,0.6], [5,0.8], [6,1], [7,0.8], [8,0.6], [9,0.4], [10,0.2], [11,0], [12,0]])
B = np.array([[5,0], [6,0.2], [7,0.4], [8,0.6], [9,0.8], [10,1], [11,0.8], [12,0.6], [13,0.4], [14,0.2], [15,0], [16,0]])

In [70]: # função para plotagem dos conjuntos 2d

def plotar_conjunto_2d(cojuntos, labels, title=None):

    color = ['--bo', '--go', '--ro', '--yo', '--po']

    fig, ax = plt.subplots()

    for i in range(cojuntos.shape[0]):
        plt.plot(cojuntos[i,:,0], cojuntos[i,:,1], color[i], label=labels[i])

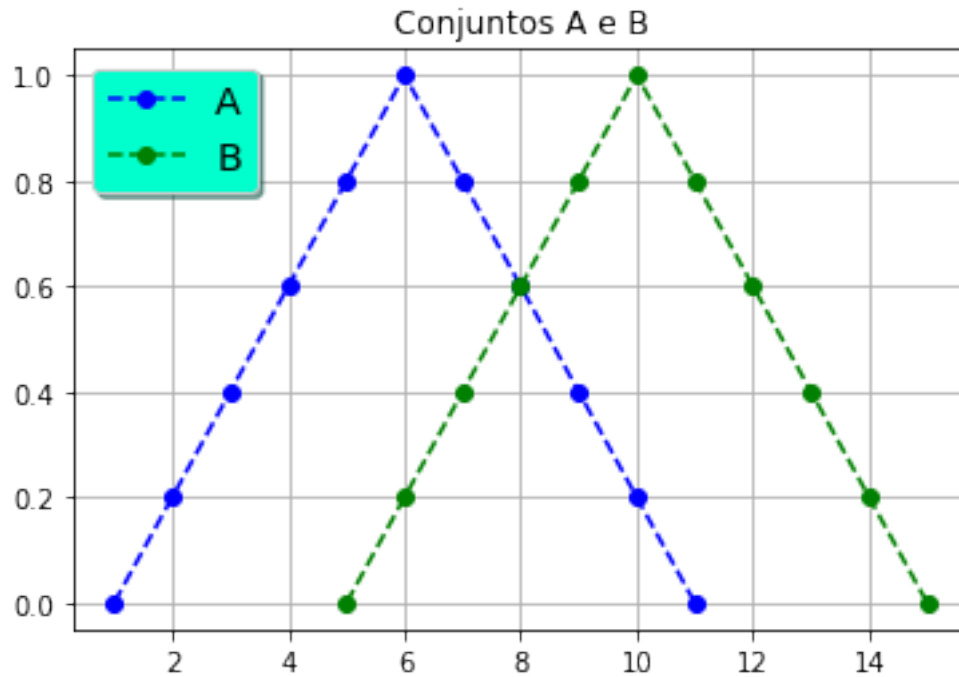
    legend = ax.legend(loc='upper left', shadow=True, fontsize='x-large')

    # Put a nicer background color on the legend.
    legend.get_frame().set_facecolor('#00FFCC')
    plt.title(title)
    plt.grid(True)
```

1.1.1 Representação 2D de A e B

```
In [72]: # conjunto dos conjuntos
conjuntos = np.array([A,B], dtype=np.float32)

# plot sets
plotar_conjunto_2d(conjuntos, ['A','B'], 'Conjuntos A e B')
```



1.2 Produto cartesiano

```
In [73]: def produto_cartesiano(A,B):
        """
        1. faz o produto carteziano entre o valor das cordenadas
        2. adiciona o minimo valor da pertinencia a coordenada Z
        """
        produto_cartesiano = np.zeros([A.shape[0],B.shape[0],3], dtype=np.float64)

        for i in range(A.shape[0]):
            for j in range(B.shape[0]):

                produto_cartesiano[i][j][0] = A[i][0]
                produto_cartesiano[i][j][1] = B[j][0]
                produto_cartesiano[i][j][2] = min(A[i][1],B[j][1])

        return produto_cartesiano
```

```
In [90]: # obtenho um array 3d a partir dos meus conjuntos fuzzy A e B
new_set_3d = produto_cartesiano(A,B)
```

1.2.1 Representação 3D de A e B

```
In [91]: def plot_conjuntos_3D(X,Y,Z):

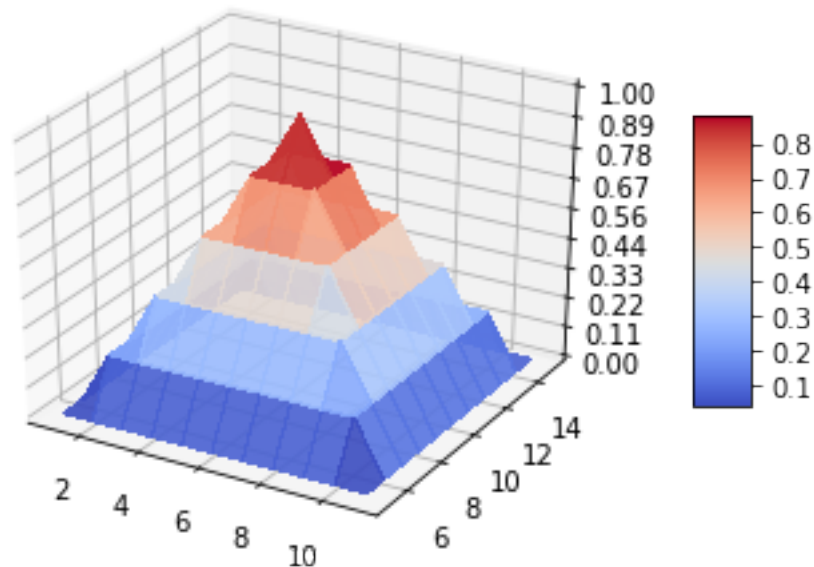
    fig = plt.figure()
    ax = fig.gca(projection='3d')

    # Plot the surface.
    surf = ax.plot_surface(X, Y, Z, alpha=0.9, cmap=cm.coolwarm, linewidth=0, antialiased=True)

    # Customize the z axis.
    # ax.set_zlim(-1.01, 1.01)
    ax.zaxis.set_major_locator(LinearLocator(10))
    ax.zaxis.set_major_formatter(FormatStrFormatter('%.02f'))

    # Add a color bar which maps values to colors.
    fig.colorbar(surf, shrink=0.5, aspect=5)
```

```
In [92]: plot_conjuntos_3D(new_set_3d[:, :, 0], new_set_3d[:, :, 1], new_set_3d[:, :, 2])
```



1.3 Projeção

In [98]: `def plotar_conjunto_projetado(X, Y, Z):`

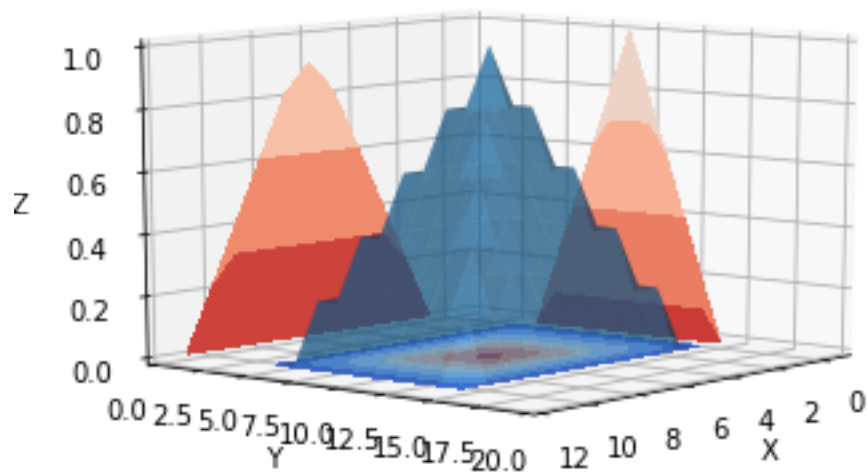
```
    fig = plt.figure()
    ax = fig.gca(projection='3d')

    ax.plot_surface(X, Y, Z, alpha=0.5) #,

    cset = ax.contourf(X, Y, Z, zdir='z', offset=0, cmap=cm.coolwarm)
    cset = ax.contourf(X, Y, Z, zdir='x', offset=0, cmap=cm.coolwarm)
    cset = ax.contourf(X, Y, Z, zdir='y', offset=0, cmap=cm.coolwarm)

    ax.set_xlabel('X')
    ax.set_xlim(-1, 12)
    ax.set_ylabel('Y')
    ax.set_ylim(0, 20)
    ax.set_zlabel('Z')
    ax.view_init(10, 40)
    ax.set_zlim(0, 1)
```

In [99]: `plotar_conjunto_projetado(np.copy(new_set_3d[:, :, 0]),
 np.copy(new_set_3d[:, :, 1]),
 np.copy(new_set_3d[:, :, 2]))`



In [93]: `def plotar_projecao(X, Y, Z):`

```

fig = plt.figure()
ax = fig.gca(projection='3d')

for i in range(X.shape[0]):
    for j in range(X.shape[1]):
        Z[i][j] = max(A[i][j], B[j][j])

ax.plot_surface(X, Y, Z, alpha=0.5) #,

#cset = ax.contourf(X, Y, Z, zdir='z', offset=0, cmap=cm.coolwarm)
cset = ax.contourf(X, Y, Z, zdir='x', offset=0, cmap=cm.coolwarm)
cset = ax.contourf(X, Y, Z, zdir='y', offset=0, cmap=cm.coolwarm)

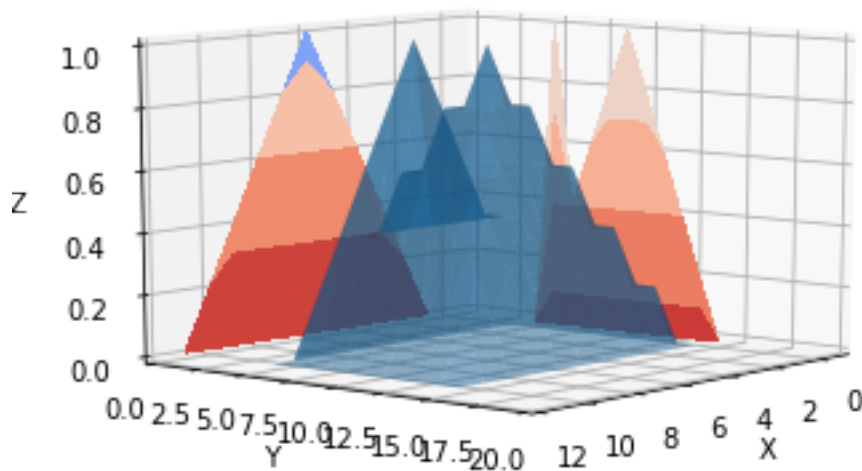
ax.set_xlabel('X')
ax.set_xlim(-1, 12)
ax.set_ylabel('Y')
ax.set_ylim(0, 20)
ax.set_zlabel('Z')
ax.view_init(10, 40)
ax.set_zlim(0, 1)

```

```

In [100]: # obtendo as novas coordenadas selecionado a maior pertinência
plotar_projecao(np.copy(new_set_3d[:, :, 0]),
                np.copy(new_set_3d[:, :, 1]),
                np.copy(new_set_3d[:, :, 2]))

```



1.4 Extensão cilíndrica

In [95]: `def plotar_projecao_cil_1(X, Y, Z):`

```
    fig = plt.figure()
    ax = fig.gca(projection='3d')

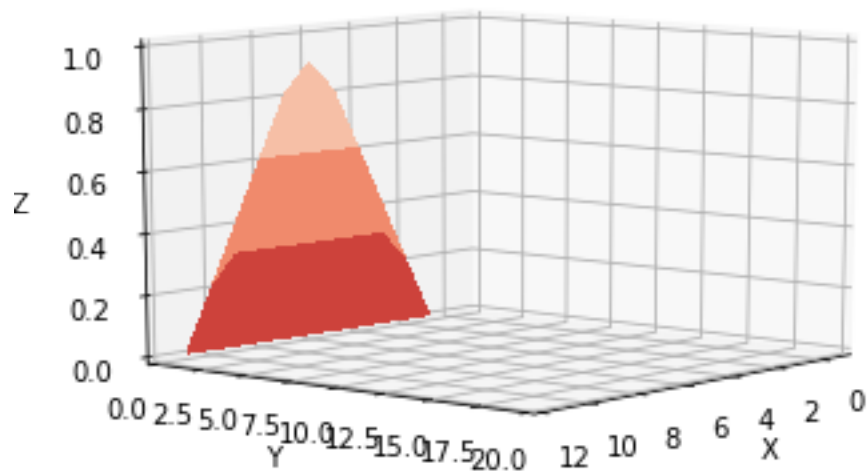
    #ax.plot_surface(X, Y, Z, alpha=0.5) #,

    #cset = ax.contourf(X, Y, Z, zdir='z', offset=0, cmap=cm.coolwarm)
    #cset = ax.contourf(X, Y, Z, zdir='x', offset=0, cmap=cm.coolwarm)
    cset = ax.contourf(X, Y, Z, zdir='y', offset=0, cmap=cm.coolwarm)

    ax.set_xlabel('X')
    ax.set_xlim(-1, 12)
    ax.set_ylabel('Y')
    ax.set_ylim(0, 20)
    ax.set_zlabel('Z')
    ax.view_init(10, 40)
    ax.set_zlim(0, 1)
```

1.4.1 Conjunto Fuzzy A no eixo X

In [96]: `plotar_projecao_cil_1(new_set_3d[:, :, 0], new_set_3d[:, :, 1], new_set_3d[:, :, 2])`



2 Referências

- [create 3-dimensional arrays](#)
- [Pot 3d WITH MATPLOTLIB](#)