Lógica Fuzzy em Python: O Guia para Iniciantes - Agrupamento com Fuzzy c-means

- Agrupamento parcial difuso
- Um registro pode estar dentro de mais de um grupo
- É gerada uma probabilidade para cada grupo
- Artigo: https://www.sciencedirect.com/science/article/pii/0098300484900207

Importação das bibliotecas

In [14]:
```
import skfuzzy
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt
import numpy as np
```

Carregamento da base de dados

In [2]:
```
base = pd.read_csv('./credit_card_clients.csv', header = 1)
base.shape
```

Out[2]:
```
(30000, 25)
```

In [3]:
```
base
```

Out[3]:

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	...	BIL
0	1	20000	2	2	1	24	2	2	-1	-1	...	
1	2	120000	2	2	2	26	-1	2	0	0	...	
2	3	90000	2	2	2	34	0	0	0	0	...	
3	4	50000	2	2	1	37	0	0	0	0	...	
4	5	50000	1	2	1	57	-1	0	-1	0	...	
...	...	...	...	...	...	...	...	...	...	...	...	
29995	29996	220000	1	3	1	39	0	0	0	0	...	
29996	29997	150000	1	3	2	43	-1	-1	-1	-1	...	
29997	29998	30000	1	2	2	37	4	3	2	-1	...	
29998	29999	80000	1	3	1	41	1	-1	0	0	...	
29999	30000	50000	1	2	1	46	0	0	0	0	...	

30000 rows × 25 columns

In [4]:
```
base['BILL TOTAL'] = base['BILL_AMT1']+base['BILL_AMT2']+base['BILL_AMT3']+base['BILL_
```

In [5]:
```
base
```

Out[5]:

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	...	BIL
0	1	20000	2	2	1	24	2	2	-1	-1	...	
1	2	120000	2	2	2	26	-1	2	0	0	...	
2	3	90000	2	2	2	34	0	0	0	0	...	
3	4	50000	2	2	1	37	0	0	0	0	...	
4	5	50000	1	2	1	57	-1	0	-1	0	...	
...	...	...	...	...	...	...	...	...	...	...	...	
29995	29996	220000	1	3	1	39	0	0	0	0	...	
29996	29997	150000	1	3	2	43	-1	-1	-1	-1	...	
29997	29998	30000	1	2	2	37	4	3	2	-1	...	
29998	29999	80000	1	3	1	41	1	-1	0	0	...	
29999	30000	50000	1	2	1	46	0	0	0	0	...	

30000 rows × 26 columns

In [6]:
```
base['LIMIT_BAL'].mean()
```

Out[6]:
```
167484.32266666667
```

In [9]:
```
base['BILL TOTAL'].mean()
```

Out[9]:
```
269861.6712
```

In [10]:
```
x = base.iloc[:,[1,25]].values
x
```

Out[10]:
```
array([[ 20000,   7704],
       [120000,  17077],
       [ 90000, 101653],
       ...,
       [ 30000,  70496],
       [ 80000, 266611],
       [ 50000, 230874]], dtype=int64)
```

In [12]:
```
x.min(), x.max()
```

Out[12]:
```
(-336259, 5263883)
```

In [18]:
```
scaler = MinMaxScaler()
x = scaler.fit_transform(x)
x
```

Out[18]:
```
array([[0.01010101, 0.06142041],
       [0.11111111, 0.06309411],
       [0.08080808, 0.07819659],
       ...,
       [0.02020202, 0.07263298],
       [0.07070707, 0.10765263],
       [0.04040404, 0.10127118]])
```

In [19]:
```
x.min(), x.max()
```

Out[19]:
```
(0.0, 1.0)
```

In [20]:
```
x.shape
```

Out[20]:
```
(30000, 2)
```

Agrupamento com Fuzzy c-means

- Documentação: https://pythonhosted.org/scikit-fuzzy/api/skfuzzy.cluster.html

In [22]:
```
agrupamento = skfuzzy.cmeans(data=x.T, c=5, m= 2, error=0.005, maxiter= 100, init=None
agrupamento
```

Out[22]:
```
(array([[0.21564938, 0.07586243],
        [0.29391045, 0.27849328],
        [0.40472   , 0.08861346],
        [0.03728556, 0.0827495 ],
        [0.12504559, 0.13280708]]),
 array([[0.02532596, 0.18651485, 0.06865385, ..., 0.00988812, 0.04928087,
         0.01060539],
        [0.00842266, 0.02591939, 0.01459874, ..., 0.00322117, 0.01373448,
         0.00347588],
        [0.00687257, 0.02381706, 0.0118888 , ..., 0.00255099, 0.00969479,
         0.0025025 ],
        [0.90064552, 0.35443708, 0.65205761, ..., 0.95848441, 0.62464476,
         0.94265631],
        [0.05873329, 0.40931162, 0.252801  , ..., 0.02585531, 0.3026451 ,
         0.04075992]]),
 array([[0.27853138, 0.07041846, 0.26632244, ..., 0.19728777, 0.35899173,
         0.12940421],
        [0.11197204, 0.3185655 , 0.32369161, ..., 0.02792469, 0.02162268,
         0.04514666],
        [0.04213723, 0.40185071, 0.09952865, ..., 0.38211984, 0.2371949 ,
         0.612521  ],
        [0.34931367, 0.01116265, 0.01662628, ..., 0.10383458, 0.31049823,
         0.14206927],
        [0.21804567, 0.19800268, 0.29383103, ..., 0.28883312, 0.07169246,
         0.07085887]]),
 array([[0.2060551 , 0.10531515, 0.1348615 , ..., 0.19547404, 0.14838764,
         0.17707777],
        [0.35730719, 0.28251088, 0.29245749, ..., 0.34248324, 0.28108055,
         0.3093108 ],
        [0.39555482, 0.29471583, 0.32407938, ..., 0.38484991, 0.33455512,
         0.36453579],
        [0.03455329, 0.07639729, 0.04376002, ..., 0.01985425, 0.04167929,
         0.01878238],
        [0.13530821, 0.07109196, 0.07027989, ..., 0.12088464, 0.05987838,
         0.09032555]]),
 array([172.41908219, 130.59781473, 130.46483156, 129.40479847,
        122.64088182, 101.57439771,  81.41698493,  72.61221194,
         68.78407922,  66.18038295,  63.60640332,  61.61879676,
         60.57723351,  60.16812231,  60.02893989,  59.98014409,
         59.95954025,  59.94857195,  59.94167467,  59.93692258,
         59.93347728,  59.93088954,  59.92888644,  59.92729172,
         59.92598818,  59.92489671,  59.92396329,  59.92315062,
         59.92243268,  59.92179102,  59.92121238,  59.92068698,
         59.9202075 ,  59.91976826,  59.91936478,  59.9189934 ,
         59.91865108,  59.91833521,  59.91804353,  59.91777406,
         59.91752499,  59.91729474,  59.91708182,  59.91688491,
         59.91670278,  59.91653431,  59.91637846,  59.91623427,
         59.91610087,  59.91597745,  59.91586324,  59.91575756,
         59.91565976,  59.91556925,  59.91548549,  59.91540796,
         59.9153362 ,  59.91526978,  59.91520829,  59.91515137,
         59.91509867,  59.91504988,  59.91500471,  59.91496288,
         59.91492415,  59.91488828,  59.91485506,  59.9148243 ,
         59.91479581,  59.91476943,  59.91474498,  59.91472234,
         59.91470137,  59.91468195,  59.91466395,  59.91464727,
         59.91463183,  59.91461751,  59.91460425,  59.91459196,
         59.91458058,  59.91457003,  59.91456025,  59.91455119,
         59.9145428 ,  59.91453501,  59.9145278 ,  59.91452112,
         59.91451492,  59.91450918,  59.91450386,  59.91449893,
         59.91449436,  59.91449012,  59.91448619,  59.91448255,
         59.91447917,  59.91447604,  59.91447314]),
```

```
 99,
 0.6463813954806176)
```

In [24]:
```
previsoes_porcentagem = agrupamento[1]
previsoes_porcentagem.shape
```

Out[24]:
```
(5, 30000)
```

In [25]:
```
previsoes_porcentagem
```

Out[25]:
```
array([[0.02532596, 0.18651485, 0.06865385, ..., 0.00988812, 0.04928087,
        0.01060539],
       [0.00842266, 0.02591939, 0.01459874, ..., 0.00322117, 0.01373448,
        0.00347588],
       [0.00687257, 0.02381706, 0.0118888 , ..., 0.00255099, 0.00969479,
        0.0025025 ],
       [0.90064552, 0.35443708, 0.65205761, ..., 0.95848441, 0.62464476,
        0.94265631],
       [0.05873329, 0.40931162, 0.252801  , ..., 0.02585531, 0.3026451 ,
        0.04075992]])
```

In [26]:
```
previsoes_porcentagem[0][0],previsoes_porcentagem[1][0],previsoes_porcentagem[2][0],pr
```

Out[26]:
```
(0.025325959263659915,
 0.00842265749871542,
 0.006872572098128892,
 0.900645520659472,
 0.0587332904800238)
```

In [28]:
```
previsoes_porcentagem[0][0]+previsoes_porcentagem[1][0]+previsoes_porcentagem[2][0]+pr
```

Out[28]:
```
1.0
```

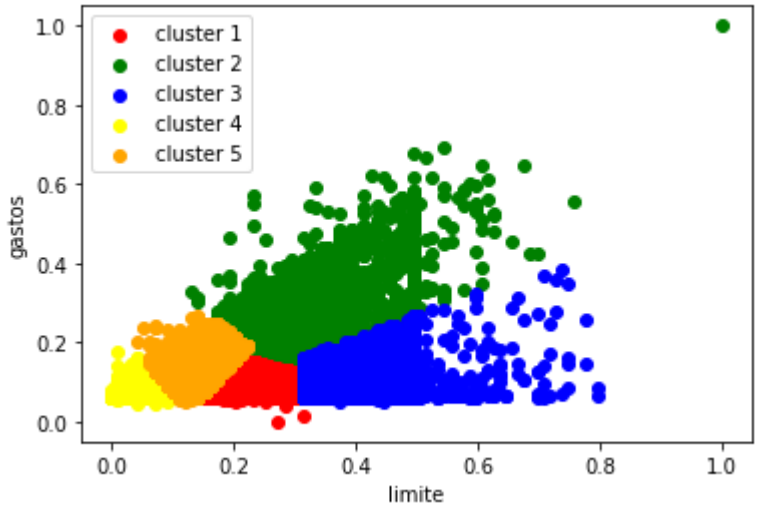
In [29]:
```
previsoes = previsoes_porcentagem.argmax(axis = 0)
previsoes
```

Out[29]:
```
array([3, 4, 3, ..., 3, 3, 3], dtype=int64)
```

In [30]:
```
np.unique(previsoes, return_counts=True)
```

Out[30]:
```
(array([0, 1, 2, 3, 4], dtype=int64),
 array([ 6549,  2236,  3446, 11341,  6428], dtype=int64))
```

In [32]:
```
plt.scatter(x[previsoes==0,0], x[previsoes == 0, 1], c = 'red', label = 'cluster 1')
plt.scatter(x[previsoes==1,0], x[previsoes == 1, 1], c = 'green', label = 'cluster 2')
plt.scatter(x[previsoes==2,0], x[previsoes == 2, 1], c = 'blue', label = 'cluster 3')
plt.scatter(x[previsoes==3,0], x[previsoes == 3, 1], c = 'yellow', label = 'cluster 4'
plt.scatter(x[previsoes==4,0], x[previsoes == 4, 1], c = 'orange', label = 'cluster 5'
plt.xlabel('limite')
plt.ylabel('gastos')
plt.legend();
```

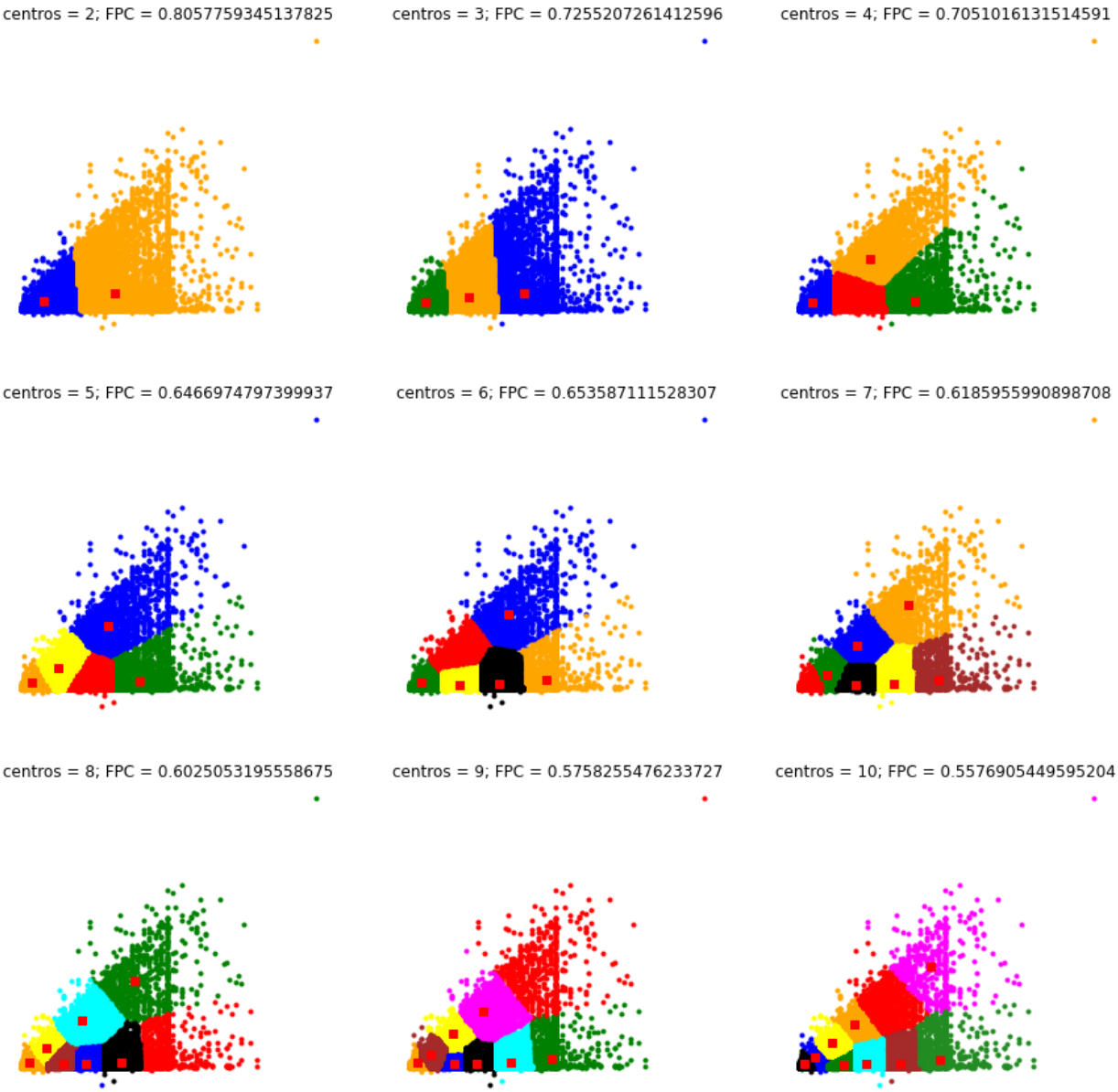


Escolha do número de grupos

In [53]:
```
colors = ['blue', 'orange', 'green', 'red', 'yellow', 'black', 'brown', 'cyan', 'mager
```

In [59]:
```
fig, axes = plt.subplots(3, 3,figsize = (15,15))
fpcs = []
for n_clusters, ax in enumerate(axes.reshape(-1), 2):
    centroides, previsoes, _, _, _, _, fpc = skfuzzy.cmeans(data=x.T, c=n_clusters, m=
    fpcs.append(fpc)

    previsoes = np.argmax(previsoes, axis=0)
    print(previsoes)

    for i in range(n_clusters):
        ax.plot(x[previsoes == i, 0], x[previsoes == i, 1], '.', color = colors[i])

    ax.set_title('centros = {}; FPC = {}'.format(n_clusters, fpc))
    ax.axis('off')

    for centro in centroides:
        ax.plot(centro[0], centro[1], 'rs')
```

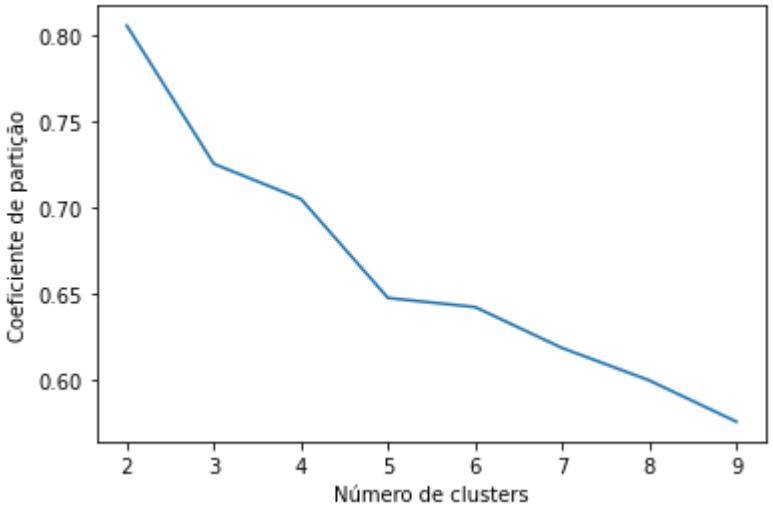
```
[0 0 0 ... 0 0 0]
[2 2 2 ... 2 2 2]
[0 0 0 ... 0 0 0]
[1 4 1 ... 1 1 1]
[2 4 2 ... 2 2 2]
[3 2 2 ... 3 2 3]
[1 6 4 ... 1 4 1]
[1 0 6 ... 1 6 1]
[5 2 0 ... 5 0 0]
```



In [47]:
```
fpcs
```

Out[47]:
```
[0.8057747765185986,
 0.7255218057951645,
 0.7051323713926129,
 0.6476321331764263,
 0.6423730633835298,
 0.618596093021743,
 0.5998045561061369,
 0.5758419637461036]
```

In [51]:
```
fig, ax = plt.subplots()
ax.plot(range(2,10), fpcs)
ax.set_xlabel('Número de clusters')
ax.set_ylabel('Coeficiente de partição');
```



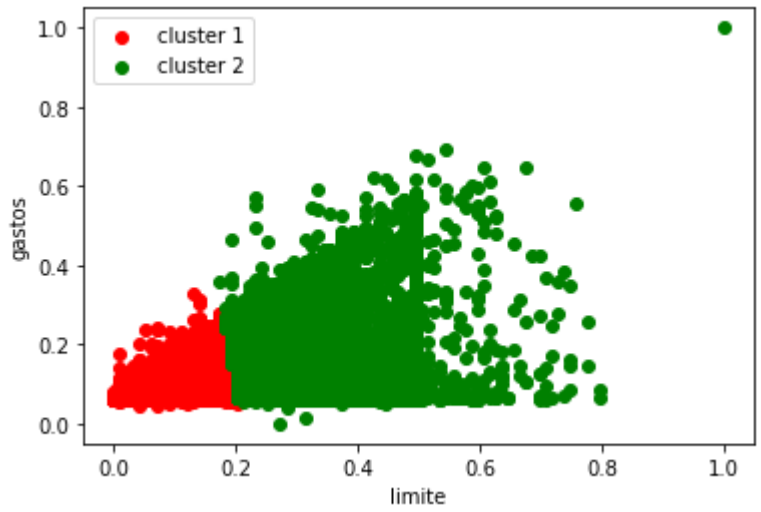
Novo agrupamento com FPC - análise do agrupamento

In [60]:
```
agrupamento = skfuzzy.cmeans(data=x.T, c=2, m= 2, error=0.005, maxiter= 100, init=None
```

In [61]:
```
previsoes_porcentagem = agrupamento[1]
previsoes = previsoes_porcentagem.argmax(axis=0)
previsoes, np.unique(previsoes, return_counts=True)
```

Out[61]:
```
(array([0, 0, 0, ..., 0, 0, 0], dtype=int64),
 (array([0, 1], dtype=int64), array([20248,  9752], dtype=int64)))
```

In [62]:
```
plt.scatter(x[previsoes==0,0], x[previsoes == 0, 1], c = 'red', label = 'cluster 1')
plt.scatter(x[previsoes==1,0], x[previsoes == 1, 1], c = 'green', label = 'cluster 2')
plt.xlabel('limite')
plt.ylabel('gastos')
plt.legend();
```



In [63]:
```
centroides = agrupamento[0]
centroides
```

Out[63]:
```
array([[0.0777615 , 0.09533229],
       [0.31603448, 0.1221696 ]])
```

In [65]:
```
centroides = scaler.inverse_transform(centroides)
centroides = pd.DataFrame(data=centroides, columns=['Limite', 'Gastos'])
centroides
```

Out[65]:

	Limite	Gastos
0	0.077761	0.095332
1	0.316034	0.122170

In [66]:
```
previsoes, previsoes.shape
```

Out[66]:
```
(array([0, 0, 0, ..., 0, 0, 0], dtype=int64), (30000,))
```

In [67]:
```
base_grupos = pd.concat([base, pd.DataFrame({'grupo': previsoes})], axis=1)
base_grupos
```

Out[67]:

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	...	BIL
0	1	20000	2	2	1	24	2	2	-1	-1	...	
1	2	120000	2	2	2	26	-1	2	0	0	...	
2	3	90000	2	2	2	34	0	0	0	0	...	
3	4	50000	2	2	1	37	0	0	0	0	...	
4	5	50000	1	2	1	57	-1	0	-1	0	...	
...	...	...	...	...	...	...	...	...	...	...	...	
29995	29996	220000	1	3	1	39	0	0	0	0	...	
29996	29997	150000	1	3	2	43	-1	-1	-1	-1	...	
29997	29998	30000	1	2	2	37	4	3	2	-1	...	
29998	29999	80000	1	3	1	41	1	-1	0	0	...	
29999	30000	50000	1	2	1	46	0	0	0	0	...	

30000 rows × 27 columns