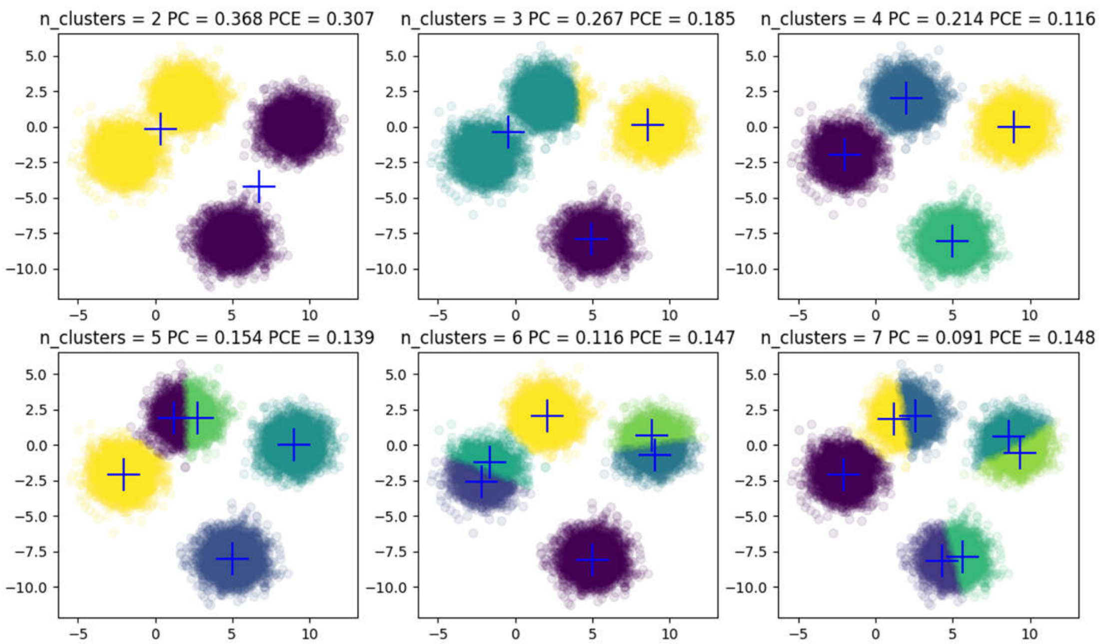
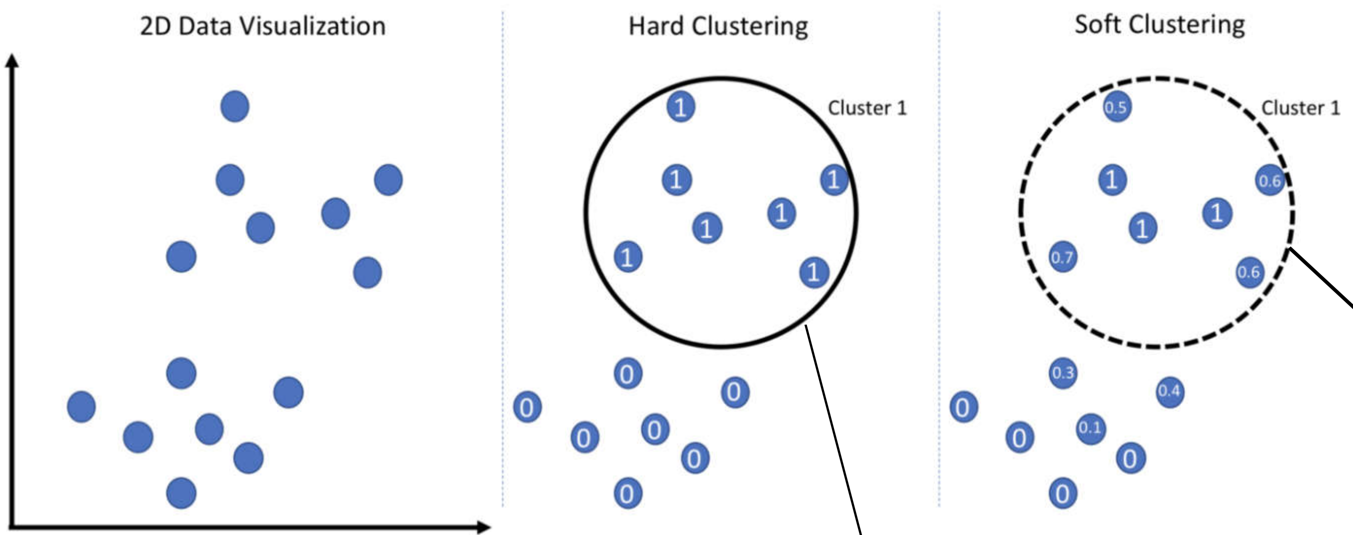
Python编程与人工智能实践



算法篇：Fuzzy C-Means(FCM)
模糊聚类

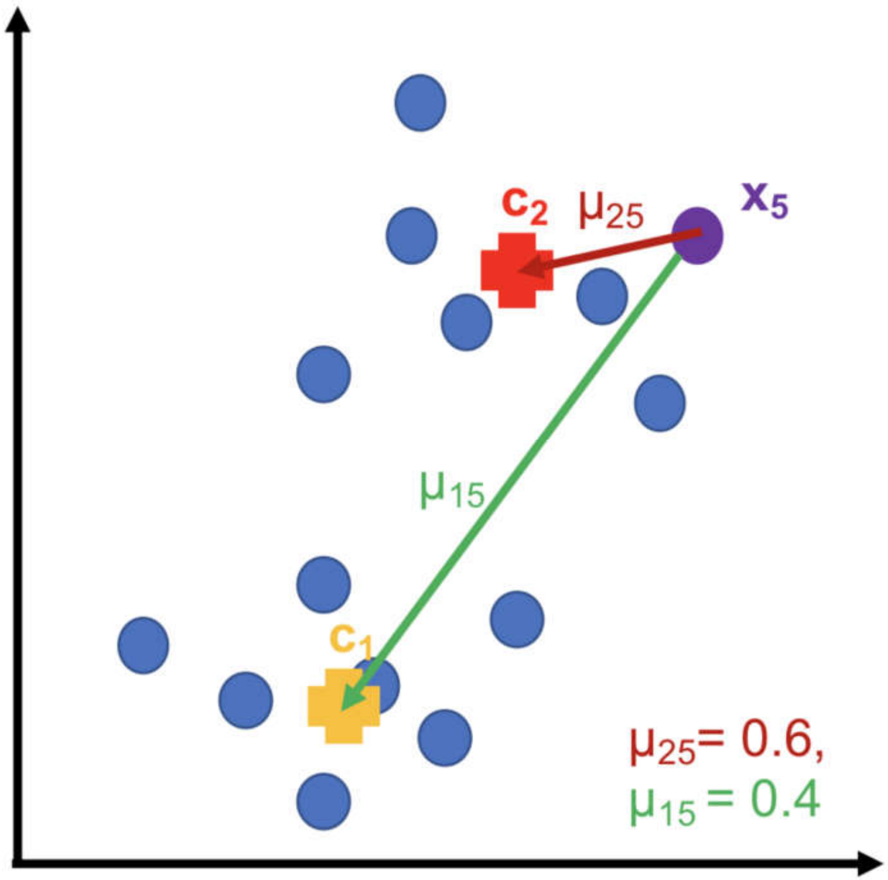
于泓
鲁东大学
信息与电气工程学院
2021.12.24



FCM 和 k-means的比较

K-means 利用硬聚类
来更新聚类
中心

利用软聚类
来更新聚类中心



每个样本和聚类中心之间有一个关系值 μ_{ij}

距离中心越近μ_{ij}越大，距离中心越远μ_{ij}越小

超参数

建模公式：$$\min(L)=\sum_{i=1}^{N}\sum_{j=1}^{C}\mu_{ij}^{m}\left\|x_i-c_j\right\|^2$$

μ_{ij}^{m} 相当于对距离进行加权
距离大：权重低
距离小：权重大

模型有2个要求的参数

当m过大时，加权的效果减少
距离趋近于一致

损失函数: **N个样本点**, C个聚类中心

$$L=\sum_{i=1}^{N}\sum_{j=1}^{C}\mu_{ij}^{m}\left\|x_i-c_j\right\|^2$$

超参数
越大越模糊

取最
小值

样本点与聚类
中心的关系

欧式距离

约束条件: **每个样本点与聚类中心的关系和为1**
N个样本点有N个约束关系

$$\sum_{j=1}^{C}\mu_{ij}=1 \qquad i=1,2,\ \ldots N$$

根据拉格朗日公式

$$L(\boldsymbol{\mu},\mathbf{c},\boldsymbol{\lambda})=\sum_{i=1}^{N}\sum_{j=1}^{C}\mu_{ij}^{m}\left\|x_i-c_j\right\|^2+\sum_{i=1}^{N}\left(\lambda_i\left(\sum_{j=1}^{C}\mu_{ij}-1\right)\right)$$

$$=\sum_{i=1}^{N}\sum_{j=1}^{C}\mu_{ij}^{m}\left\|x_i-c_j\right\|^2+\sum_{i=1}^{N}\sum_{j=1}^{C}\left(\lambda_i\mu_{ij}-\lambda_i\right)$$

$$\mathrm{L}(\boldsymbol{\mu},\mathbf{c},\boldsymbol{\lambda})=\sum_{i=1}^{N}\sum_{j=1}^{C}\mu_{ij}^{m}\left\|\mathrm{x}_i\text{-}\mathrm{c}_j\right\|^2+\sum_{i=1}^{N}\sum_{j=1}^{C}\left(\lambda_i\mu_{ij}\text{-}\lambda_i\right)$$

对μ_{ij}求导

$$\frac{\partial \mathrm{L}}{\mu_{ij}}=m\mu_{ij}^{m\text{-}1}\left\|\mathrm{x}_i\text{-}\mathrm{c}_j\right\|^2+\lambda_i=0$$

$$\mu_{ij}=-\left(\frac{\lambda_i}{m\left\|\mathrm{x}_i\text{-}\mathrm{c}_j\right\|^2}\right)^{\frac{1}{m-1}}=-\lambda_i^{\frac{1}{m-1}}\left(\frac{1}{m\left\|\mathrm{x}_i\text{-}\mathrm{c}_j\right\|}\right)^{\frac{2}{(m-1)}}$$

代入: $$\sum_{j=1}^{C}\mu_{ij}=1$$

$$-\lambda_i^{\frac{1}{m-1}}\sum_{j=1}^{C}\left(\frac{1}{m\left\|\mathrm{x}_i\text{-}\mathrm{c}_j\right\|}\right)^{\frac{2}{(m-1)}}=1$$

$$-\lambda_i^{\frac{1}{m-1}}=\frac{1}{\sum_{j=1}^{C}\left(\frac{1}{m\left\|\mathrm{x}_i\text{-}\mathrm{c}_j\right\|}\right)^{\frac{2}{(m-1)}}}$$

$$L(\boldsymbol{\mu},\mathbf{c},\boldsymbol{\lambda})=\sum_{i=1}^{N}\sum_{j=1}^{C}\mu_{ij}^{m}\left\|\mathrm{x}_i-\mathrm{c}_j\right\|^2+\sum_{i=1}^{N}\sum_{j=1}^{C}\left(\lambda_i\mu_{ij}-\lambda_i\right)$$

$$\frac{\partial L}{\mathrm{c}_j}=\sum_{i=1}^{N}\mu_{ij}^{m}(\mathrm{x}_i-\mathrm{c}_j)=0$$

$$\sum_{i=1}^{N}\mu_{ij}^{m}\mathrm{x}_i-\mathrm{c}_j\sum_{i=1}^{N}\mu_{ij}^{m}=0$$

$$\mathrm{c}_j=\frac{\sum_{i=1}^{N}\mu_{ij}^{m}\mathrm{x}_i}{\sum_{i=1}^{N}\mu_{ij}^{m}}$$

$$\mu_{ij}=\frac{\left(\frac{1}{m\left\|\mathrm{x}_i-\mathrm{c}_j\right\|}\right)^{\frac{2}{(m-1)}}}{\sum_{j=1}^{C}\left(\frac{1}{m\left\|\mathrm{x}_i-\mathrm{c}_j\right\|}\right)^{\frac{2}{(m-1)}}}$$

$$=\frac{\left(\frac{1}{\left\|\mathrm{x}_i-\mathrm{c}_j\right\|}\right)^{\frac{2}{(m-1)}}}{\sum_{j=1}^{C}\left(\frac{1}{\left\|\mathrm{x}_i-\mathrm{c}_j\right\|}\right)^{\frac{2}{(m-1)}}}=\frac{1}{\sum_{k=1}^{C}\left(\frac{\left\|\mathrm{x}_i-c_j\right\|}{\left\|\mathrm{x}_i-\mathrm{c}_k\right\|}\right)^{\frac{2}{(m-1)}}}$$

Fuzzy c means 训练方法：

（1）设计聚类数目以及超参数m

初始化关系矩阵 μ_{ij}

（2）更新聚类中心C

$$c_j = \frac{\sum_{i=1}^{N} \mu_{ij}^m x_i}{\sum_{i=1}^{N} \mu_{ij}^m}$$

（3） 更新 μ_{ij}

$$\mu_{ij} = \frac{1}{\sum_{k=1}^{C} \left(\frac{\left\| x_i - c_j \right\|}{\left\| x_i - c_k \right\|} \right)^{\frac{2}{(m-1)}}}$$

（4）重复 步骤（2）（3）直到收敛

两次计算得到关系矩阵U的差距较小

```python
def FCM_train(X,n_centers,m,max_iter = 100,theta=1e-5,seed = 0):

    rng  =  np.random.RandomState(seed)
    N,D = np.shape(X)

    # 随机初始化关系矩阵
    U = rng.uniform(size=(N, n_centers))
    # 保证每行和为1
    U = U/np.sum(U,axis=1,keepdims=True)

    # 开始迭代
    for i in range(max_iter):
        print(i)
        U_old = U.copy()
        centers = FCM_getCenters(U, X, m)
        U = FCM_getU(X,centers,m)

        # 两次关系矩阵距离过小，结束训练
        if np.linalg.norm(U - U_old) < theta:
            break

    return centers,U
```

$$\mu_{ij}=\frac{1}{\sum_{k=1}^{C}\left(\frac{\left\|x_i-c_j\right\|}{\left\|x_i-c_k\right\|}\right)^{\frac{2}{(m-1)}}}$$

```python
# 获取新的聚类中心
# U 关系矩阵 [N,C]
# X 输入数据 [N,D]
# 返回新的聚类中心 [C,D]

def FCM_getCenters(U,X,m):

    N,D = np.shape(X)
    N,C = np.shape(U)

    um = U ** m

    tile_X = np.tile(np.expand_dims(X,1),[1,C,1])
    tile_um = np.tile(np.expand_dims(um,-1),[1,1,D])
    temp = tile_X*tile_um

    new_C = np.sum(temp,axis=0)/np.expand_dims(np.sum(um,axis=0),axis=-1

    return new_C
```

$$c_j=\frac{\sum_{i=1}^{N}\mu_{ij}^{m}x_i}{\sum_{i=1}^{N}\mu_{ij}^{m}}$$

```python
def FCM_getU(X,Centers,m):
    N,D = np.shape(X)
    C,D = np.shape(Centers)

    temp = FCM_dist(X, Centers) ** float(2 / (m - 1))

    tile_temp =  np.tile(np.expand_dims(temp,1),[1,C,1])

    denominator_ = np.expand_dims(temp,-1)/tile_temp

    return 1 / np.sum(denominator_,axis=-1)
```

```python
# 计算N个样本，到C个中心的距离
#  X :  [N,D]
#  Centers :  [C,D]
# 返回 [N,C] 两两之间的距离
def FCM_dist(X,Centers):
    N,D = np.shape(X)
    C,D = np.shape(Centers)

    tile_x = np.tile(np.expand_dims(X,1),[1,C,1])
    tile_centers = np.tile(np.expand_dims(Centers,axis=0),[N,1,1])

    dist = np.sum((tile_x-tile_centers)**2,axis=-1)

    return np.sqrt(dist)
```

```python
def FCM_getClass(U):

    return np.argmax(U,axis=-1)

def FCM_partition_coefficient(U):

    return np.mean(U ** 2)

def FCM_partition_entropy_coefficient(U):

    return -np.mean(U * np.log2(U))
```

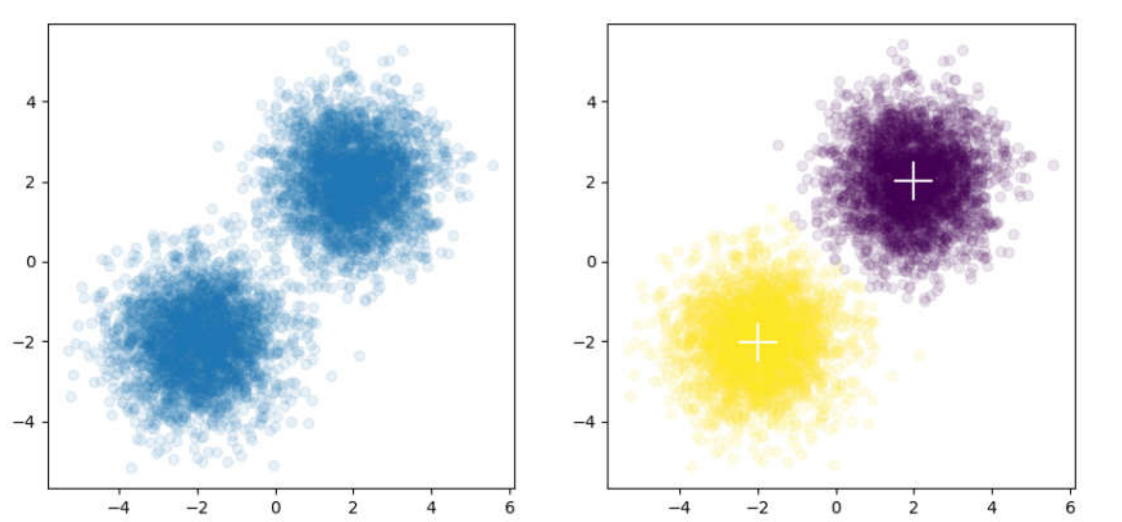
得到聚类标签

评估聚类效果

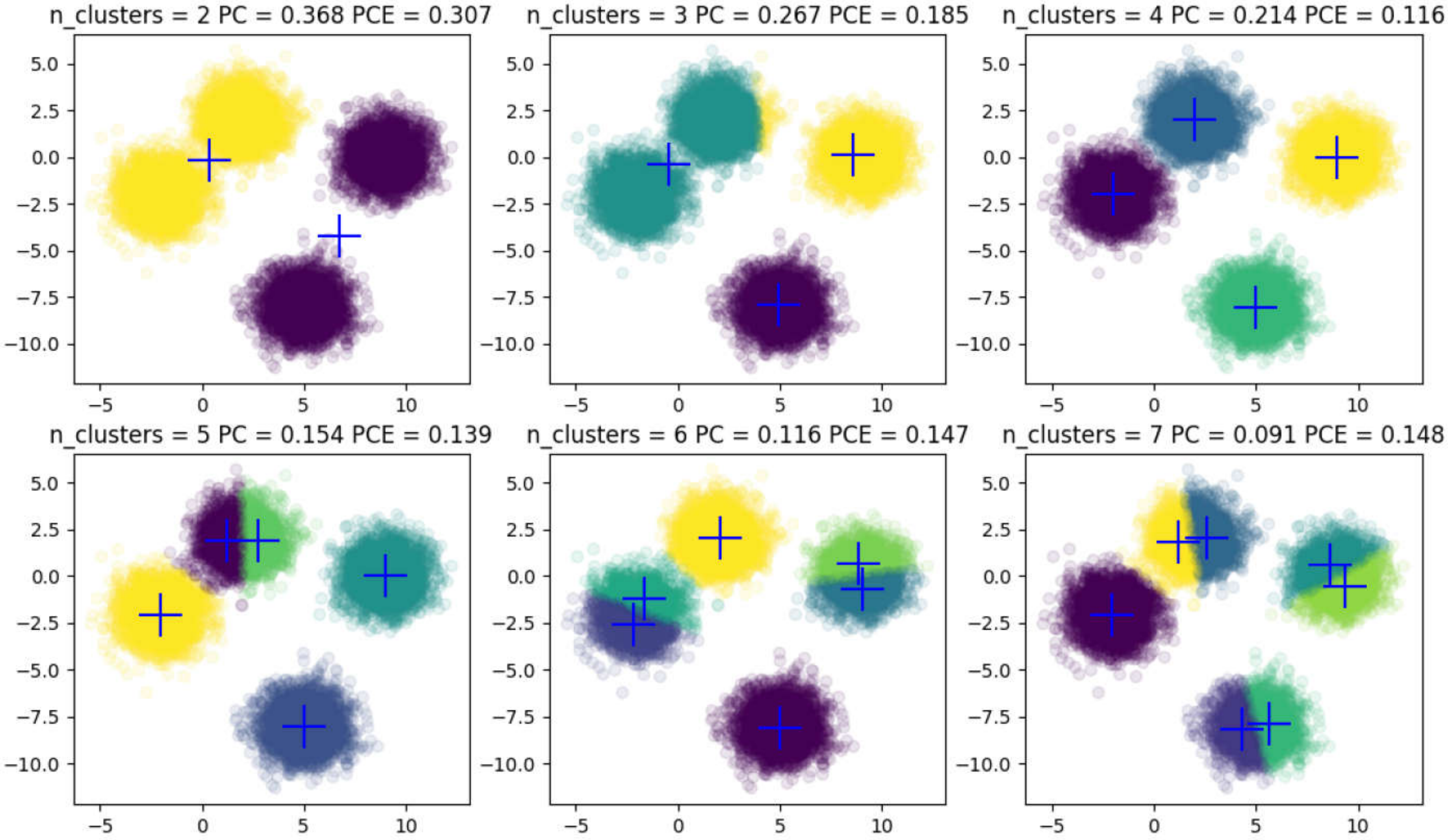
利用交叉熵评估聚类效果

测试：

```
# 简单测试
N = 3000

X = np.concatenate((
    np.random.normal((-2, -2), size=(N, 2)),
    np.random.normal((2, 2), size=(N, 2))
    ))


n_centers =2
m =2
centers,U = FCM_train(X,n_centers,m,max_iter = 100,theta=1e-5,seed = 0)

labels =  FCM_getClass(U)
print(labels)
f, axes = plt.subplots(1, 2, figsize=(11,5))
axes[0].scatter(X[:,0], X[:,1], alpha=.1)
axes[1].scatter(X[:,0], X[:,1], c=labels, alpha=.1)
axes[1].scatter(centers[:,0], centers[:,1], marker="+", s=500, c='w')

plt.show()
```



```python
# 测试聚类效果
n_samples = 3000

X = np.concatenate((
    np.random.normal((-2, -2), size=(n_samples, 2)),
    np.random.normal((2, 2), size=(n_samples, 2)),
    np.random.normal((9, 0), size=(n_samples, 2)),
    np.random.normal((5, -8), size=(n_samples, 2))
))


list_n_centers =[2, 3, 4, 5, 6, 7]
rows = 2
cols =3
f, axes = plt.subplots(rows, cols, figsize=(16,11))

for n_centers,axe in zip(list_n_centers,axes.ravel()):
    m = 2
    centers,U = FCM_train(X,n_centers,m,max_iter = 100,theta=1e-5,seed = 0)
    labels =  FCM_getClass(U)
    PC = FCM_partition_coefficient(U)
    PCE = FCM_partition_entropy_coefficient(U)

    axe.scatter(X[:,0], X[:,1], c=labels, alpha=.1)
    axe.scatter(centers[:,0], centers[:,1], marker="+", s=500, c='b')

    axe.set_title("n_clusters = %d PC = %.3f PCE = %.3f"%(n_centers,PC,PCE))

plt.show()
```


n_clusters = 2 PC = 0.368 PCE = 0.307
n_clusters = 3 PC = 0.267 PCE = 0.185
n_clusters = 4 PC = 0.214 PCE = 0.116
n_clusters = 5 PC = 0.154 PCE = 0.139
n_clusters = 6 PC = 0.116 PCE = 0.147
n_clusters = 7 PC = 0.091 PCE = 0.148

有最小的PCE