# Kemans 算法

## 【算法原理】

Kemans算法是一种无监督算法，用于将相似的样本归为一个类中。对于给定包含N个样本的数据集{$x_1, x_2, \ldots, x_N$}, K-means算法的目标是将样本分配到k个类中。 K-means算法需要假定有K个中心点作为每个类的代表，目标函数如下：

$$\min_{\mu_k, c_{ik}} \sum_{i=1}^{N} \sum_{k=1}^{K} c_{ik} \|\mathbf{x}_i - \mu_k\|_2^2$$

$$\text{s.t. } c_{ik} \in \{0, 1\}, \sum_{k=1}^{K} c_{ik} = 1,$$

其中，$\mu_k$是k个类的中心， $c_{ik}$代表第i个样本是否属于第k个类。

k-means的算法流程是寻找合适中心点并将样本分配到k个中心点的过程。K-means算法需要初始化k个中心点并将样本分配到离它最近的中心点，然后在新的聚类中重新计算各个聚类的中心点，这样不断更新k个中心点直到收敛。

## 【伪代码】

```
选择K个点作为初始质心
repeat
    将每个点指派到最近的质心，形成k个簇
    重新计算每个簇的质心
until 簇不发生变化或达到最大的迭代次数
```

## 【代码实现】

1. 计算每个中心点到所有样本的距离

```python
def cal_dis(data, clu, k):
    """
    计算质点与数据点的距离
    :param data: 样本点
    :param clu:  质点集合
    :param k: 类别个数
    :return: 质心与样本点距离矩阵
    """
    dis = []
    for i in range(len(data)):
        dis.append([])
        for j in range(k):
            dis[i].append(m.sqrt((data[i, 0] - clu[j, 0])**2 + (data[i, 1]-clu[j, 1])**2))
    return np.asarray(dis)
```

2. 将所有样本分配到最近的中心点

```
def divide(data, dis):
    """
    对数据点分组
    :param data: 样本集合
    :param dis: 质心与所有样本的距离
    :param k: 类别个数
    :return: 分割后样本
    """
    clusterRes = [0] * len(data)
    for i in range(len(data)):
        seq = np.argsort(dis[i])
        clusterRes[i] = seq[0]

    return np.asarray(clusterRes)
```

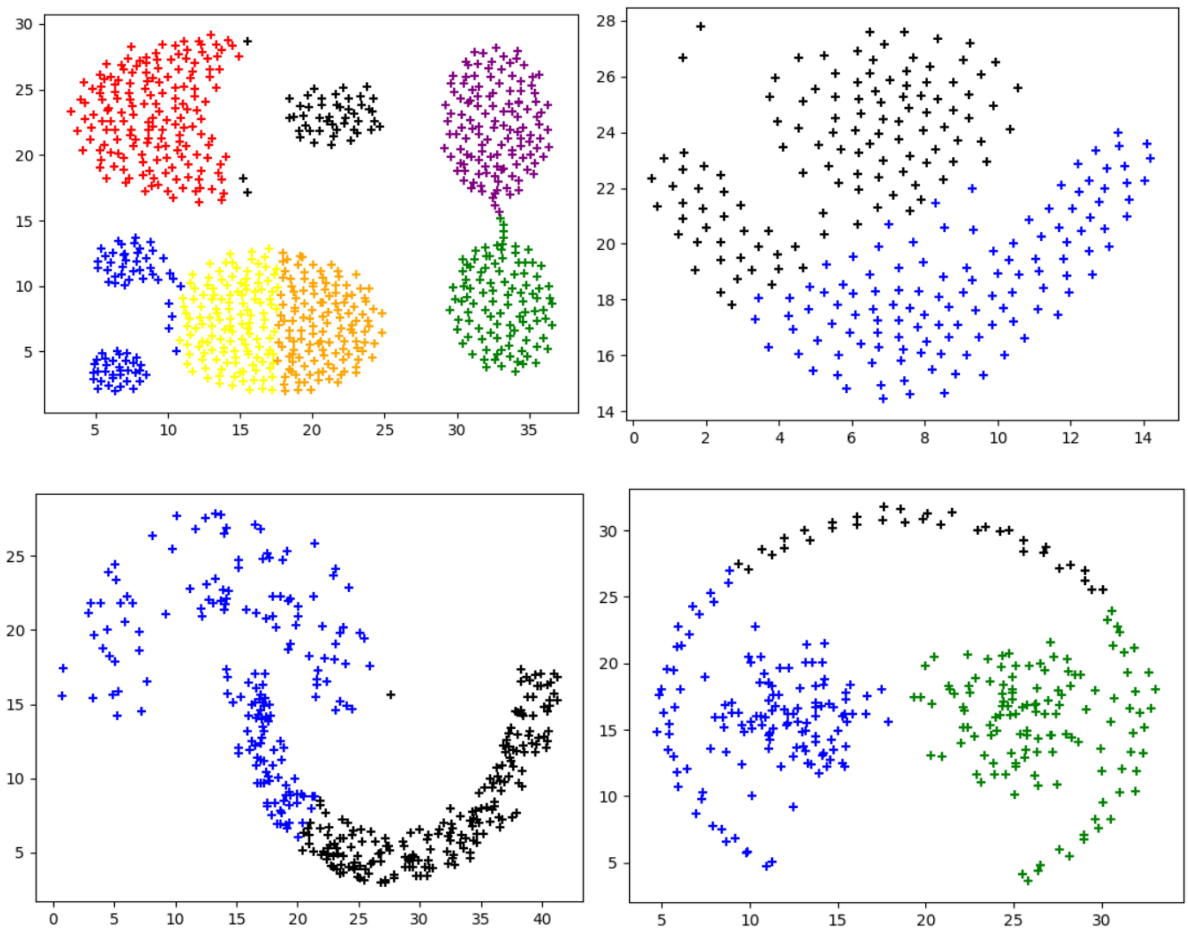
3. 重新计算中心点

```
def center(data, clusterRes, k):
    """
    计算质心
    :param group: 分组后样本
    :param k: 类别个数
    :return: 计算得到的质心
    """
    clunew = []
    for i in range(k):
        # 计算每个组的新质心
        idx = np.where(clusterRes == i)
        sum = data[idx].sum(axis=0)
        avg_sum = sum/len(data[idx])
        clunew.append(avg_sum)
    clunew = np.asarray(clunew)
    return clunew[:, 0: 2]
```

4. 因为初始化的中心点不一定是期望中心点，需要不断更新中心点直到收敛

```
def classfy(data, clu, k):
    """
    迭代收敛更新质心
    :param data: 样本集合
    :param clu: 质心集合
    :param k: 类别个数
    :return: 误差， 新质心
    """
    clulist = cal_dis(data, clu, k)
    clusterRes = divide(data, clulist)
    clunew = center(data, clusterRes, k)
    err = clunew - clu
    return err, clunew, k, clusterRes
```

【实验效果】



通过上述结果展示，我们可以看到K-means算法有很多明显的缺陷：

1. K-means算法在多种不同情况下的聚类表现得并不太好，我们可以看到K-means得到的簇更偏向于球形，这意味着**K-means算法不能处理非球形簇的聚类问题**，而现实中数据的分布情况是十分复杂的，所以K-means算法不太适用于现实大多数情况。
2. **K-means算法需要预先确定聚类的个数**。
3. **K-means算法对初始选取的聚类中心点敏感**。我们可以看到在Aggression数据聚类中，K-means算法会把不同簇聚合在一起来满足球形状簇的聚类，而这种情况下得到的中心点在两个簇中间。这说明此时K-means陷入了一个局部最优解，而陷入局部最优的一个原因是初始化中心点的位置不太好。