# 人工智能基础课程实验报告 2

**姓名：**章东泉　　**学号：**PB17121706

## 问题二　无监督学习

### Part1 KMEANS

算法的主要思路是，先通过 PCA 对标准化的数据进行降维，再使用 KMEANS 算法对降维后的数据聚类。

标准化分为两个步骤：1. 将每个样本的每个属性减去该属性在所有样本中的均值；2. 将每个样本的每个属性除以该属性在所有样本中的标准差。

第一步的目的是将数据集的中心移到坐标原点，方便后续协方差矩阵的计算，第二步的目的是抹平各个属性之间的数值差异，防止由于数值差异过大而掩盖数据特征。

PCA 算法首先计算数据集的协方差矩阵和其特征值、特征向量，将特征向量按照特征值排序，取出前 m 个特征向量作为矩阵 P，矩阵乘积 PX 即为将 X 的属性降到 m 维后的矩阵。

部分代码如下：

代码 1-1　PCA 算法降维

```
COV = np.cov(dataSet)
eigenvalue, eigenvector = np.linalg.eig(COV)
eigen_list = []
for i in range(len(eigenvalue)):
    eigen_data = {'value': eigenvalue[i], 'vector': eigenvector[i]}
    eigen_list.append(eigen_data)
sorted(eigen_list, key= lambda x:x['value'])
k = 0
```

```
for m in range(len(eigen_list)):
    if(check_threshold(eigen_list, m, threshold) == True):
        k = m
        break
matrix_P = []
for i in range(k):
    matrix_P.append(eigen_list[i]['vector'])
reduced_matrix = np.matmul(matrix_P, dataSet)
```

然后使用 KMEANS 算法，先随机生成 k 个样本质心，将数据集分配到距离最近的质心，再计算每个簇新质心的位置，循环直到质心位置不发生变化。部分代码实例如下：

代码 1-2　KMEANS 算法示例

```
# 计算新的质心
newCenter = massCenter[:]
for i in range(k):
    coorSum = []
    labelNum = 0
    for j in range(len(LabelSet[0])):
        if(LabelSet[0][j] == i):
            if(len(coorSum) == 0):
                coorSum = newSet[j][:]
            else:
                coorSum = coorSum + newSet[j]
            labelNum = labelNum + 1
    for p in range(len(coorSum)):
        coorSum[p] = coorSum[p]/labelNum
    if(len(coorSum) != 0):
        newCenter[i] = coorSum[:]
```

先固定 threshold 为 0.9，对应的降维后的维度是 7。测试用数据集的实际分类是三类，即数据的第一维度。用我们的算法将数据集(抛弃了第一维度)聚成三类，生成的标签如下：

图 1-1　聚类(3)标签

```
philip@ubuntu:~/AI/unsupervise/src$ python3 main.py
[0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0
, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 1, 2, 2, 1, 1, 2, 1, 1, 1, 2, 0, 1, 1, 1, 0, 1, 1, 1, 2, 0, 1,
 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
, 1, 1, 1, 0, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2,
 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]
```

可以看到基本上是与实际分类一致的。聚为 3 类的兰德系数为 0.8669。

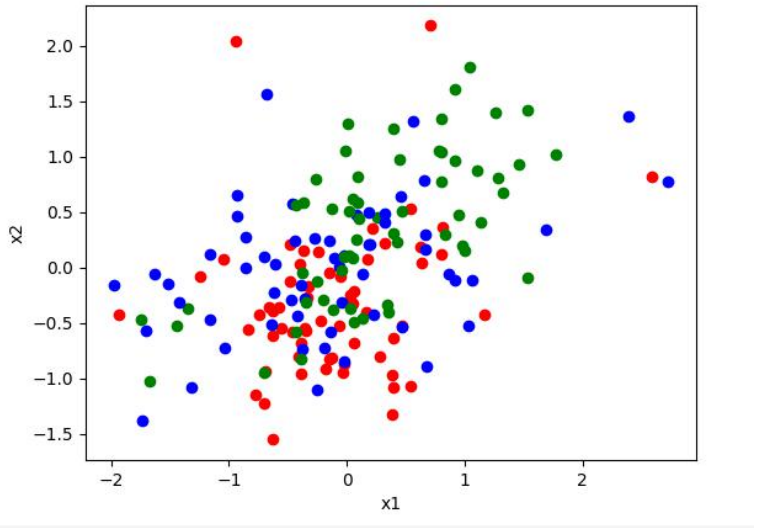
我们尝试了不同的聚类程度，得到的轮廓系数和兰德系数如下表所示：

表 1-1 聚类结果分析

| K | Sil(轮廓系数) | RI(兰德系数) |
|---|---|---|
| 3 | 0.3226 | 0.8669 |
| 5 | 0.3896 | 0.7875 |
| 7 | 0.3629 | 0.7775 |
| 2 | 0.2948 | 0.7111 |

综上考虑，聚类为 3 类比较合适。将原数据降至 2 维绘图如下：

图 1-2 聚类结果可视化(threshold=0.9)



由于降维程度较高，数据点在 2 维上的分布不是很明显。同时可以看到聚类后的轮廓系数也不是很好，说明降维后数据集的划分并不明确。

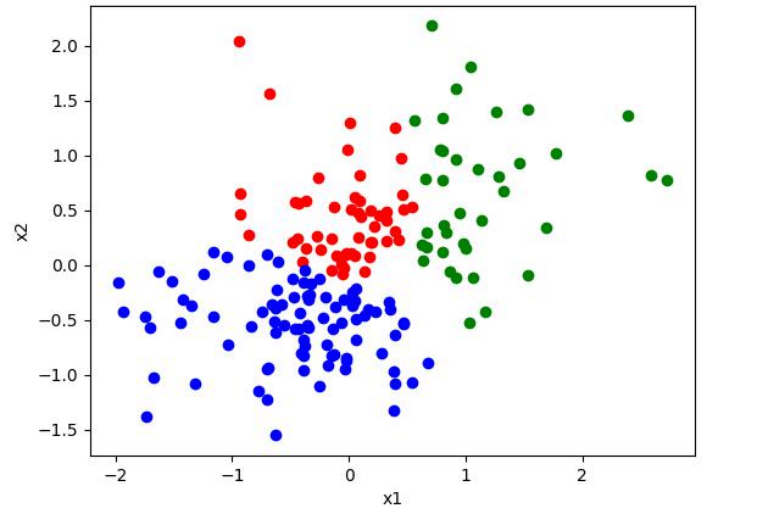
再更改 threshold 进行多次 3 类的聚类：

表 1-2 降维维数对聚类结果分析

| threshold | dimension | Sil | RI |
|---|---|---|---|
| 0.4 | 1 | 0.7136 | 0.5687 |
| 0.6 | 2 | 0.5547 | 0.5776 |
| 0.9 | 7 | 0.3226 | 0.8669 |
| 1.0 | 12(原数据维度) | 0.3223 | 0.9339 |

随着聚类数据维数的增加，轮廓系数越来越小兰德系数越来越大。在降维 2 维聚类时，我们对聚类后的标签进行了二维可视化，效果如下：

图 1-3 聚类结果可视化(threshold=0.6)




可以看出这次的划分比较明显了，三个簇分隔明确。

轮廓系数越高，代表数据集之间的划分越明确，兰德系数越高，

代表聚类后的类别与真实类别越接近。

利用不同的 threshold 降维后的结果说明，原数据的三个簇之间划分并不是很明确，而随着维数的不断下降，三个簇之间的划分虽然明确了但是开始与实际划分有较大差异。这是数据集一个比较有意思的特点。