人工智能基础编程作业 2

非监督学习——学生表现预测

数据集

Wine Data Set

属性说明

数据集是自意大利同一地区但来自不同品种的葡萄酒的化学分析，是一经典的分类数据集，数据集共 13 个维度，第一个维度为葡萄酒的实际品种，其他维度均为葡萄酒化学分析特征，学习目标是对酒进行分类

学习算法

- PCA, Principal components analysis 主成分分析，用于数据降维
- KMeans k-均值算法，用于聚类

文件结构

```
unsupervise
|  README.md
|
├─input
|  └─wine
|          wine.data
|          wine.names
|
├─output
|      SilhouetteCoefficient.png
|      wine_clustered.csv
|
└─src
        main.py
```

依赖

程序依赖于`numpy`，并使用`matplotlib`包进行绘图

运行

本部分任务是单一的，在`src`目录下运行`main.py`文件即可

结果

程序运行后，将会把默认参数下的聚类结果保存至`../output/wine_clustered.csv`文件，并输出兰德系数

算法、实现与效果

数据读取与预处理

在`main.py`中使用`loadData(file)`函数从文件`file`中读取数据并进行标准化，Z-Score 标准化将属性平均值设置为 0，标准差设置为 1。返回数据与类别标签。

```
def loadData(file):

    print('start reading ' + file)
    Attributes = []
    Identifiers = []
    total = [0 for i in range(13)]  # 各分量总和

    with open(file, 'r') as fileStream:
        for line in fileStream.readlines():
            datum = line.strip().split(',')
            Attributes.append([float(x) for x in datum[1:]])
            Identifiers.append(int(datum[0]))
        Data = np.array(Attributes)
        average = np.mean(Data, axis=0)  # 属性平均值
        standardDeviation = np.std(Data, axis=0)  # 属性方差
        Data = (Data - average) / standardDeviation  # Z-Score 标准化
    return Data, Identifiers
```

主成分分析

在`main.py`中使用函数`PCA(data, threshold, dimension=None)`对数据`data`进行降维，降维的幅度由特征值的累计贡献率`threshold`指定或由`dimension`显式指定降维后的维度。通过计算协方差矩阵，目标是使得数据向某个低维度空间上的投影方差最大，投影函数由较大的几个特征向量组成。

```
def PCA(data, threshold, dimension=None):

    eigenValues, eigenVectors = np.linalg.eig(
        np.cov(data, rowvar=0))  # 由协方差矩阵计算特征值与特征向量
    eigenValuesIndices = np.argsort(eigenValues)[::-1]  # 由大到小排序后的下标
    total = np.sum(eigenValues)  # 总和
    total_m = 0
    for m in range(len(eigenValues)):
        if total_m / total >= threshold:
            break
        total_m += eigenValues[eigenValuesIndices[m]]
    # 选取前 m 个特征值对应的特征向量，作为新的特征空间的一组基
    if dimension != None:
        m = dimension  # 显式指定维数
    eigenVectors_m = eigenVectors[:, eigenValuesIndices[0:m]]
    lowerDimensionalData = np.dot(data, eigenVectors_m)  # 原始数据乘以基实现降
维

    return lowerDimensionalData
```

k-means 算法

在`main.py`中使用函数`KMeans(k, data)`基于参数`k`对数据`data`进行聚类。算法初始化时随机选取k个质心，每次迭代将所有数据点划分至最近的质心对于簇，并更新质心坐标，直到质心收敛即停止迭代。

函数同时计算了聚类的轮廓系数(Silhouette Coefficient)，随聚类结果一起返回。

```
def KMeans(k, data):

    centroids = np.empty([k, data.shape[1]], dtype=float)  # 簇质心
    for i in range(k):
        index = np.random.randint(data.shape[0])  # 随机下标
        centroids[i] = data[index]  # Forgy 方法：选取随机观测作为初始质心

    isClusteringChanged = True  # 聚类是否改变
    cluster = np.zeros(data.shape[0], dtype=int)   # 类别

    while isClusteringChanged:
        isClusteringChanged = False
        for i in range(data.shape[0]):  # 更新质心
            minDistance = float("inf")
            minCentroid = 0
            for j in range(k):
                if (distance:=distanceBetween(data[i], centroids[j])) < minDistance: # 计算观测到质心距离
                    minDistance = distance
                    minCentroid = j
            if cluster[i] != minCentroid:
                cluster[i] = minCentroid
                isClusteringChanged = True

        for j in range(k):
            centroids[j] = np.mean(
                data[np.nonzero(cluster == j)], axis=0)  # 根据类别更新质心

    a = [0] * data.shape[0]
    b = [0] * data.shape[0]
    Silhouette = [0] * data.shape[0]
    for i in range(data.shape[0]):
        a[i] = np.mean([distanceBetween(data[i], data[j]) for j in range(
            data.shape[0]) if i != j and cluster[i] == cluster[j]])  # i 到其簇中其他点距离的均值

        minNeighborDistance = float("inf")
        minNeighborCentroid = 0
        for j in range(k):
            if (distance:=distanceBetween(data[i], centroids[j])) < minNeighborDistance and j != cluster[i]: # 计算观测到质心距离
                minNeighborDistance = distance
                minNeighborCentroid = j
        b[i] = np.mean([distanceBetween(data[i], data[j]) for j in range(
            data.shape[0]) if cluster[j] == minNeighborCentroid])  # i 到相邻簇中所有点距离的均值
        Silhouette[i] = (b[i] - a[i]) / max(a[i], b[i])

    data_clustered = np.insert(
        data, 0, values=cluster + 1, axis=1)  # 以首列的正整数表示类别

    return data_clustered, np.mean(Silhouette)
```

结果评价

在`main.py`中通过函数`clusterTest(trueLabel, clusterLabel)`对聚类得到的标签`clusterLabel`进行评价，评价指标为兰德系数

```python
def clusterTest(trueLabel, clusterLabel):

    a = b = c = d = 0
    for i in range(len(trueLabel)):
        for j in range(i + 1, len(trueLabel)):  # 遍历数据点对
            if trueLabel[i] == trueLabel[j]:  # 在 trueLabel 中属同一类
                if clusterLabel[i] == clusterLabel[j]:  # 在 clusterLabel 中也属同一类
                    a += 1
                else:  # 在 clusterLabel 中不属同一类
                    b += 1
            else:  # 在 trueLabel 中不属同一类
                if clusterLabel[i] == clusterLabel[j]:  # 在 clusterLabel 中属同一类
                    c += 1
                else:  # 在 clusterLabel 中也不属同一类
                    d += 1
    print('a = {:5}  b = {:5}'.format(a, b))
    print('c = {:5}  d = {:5}'.format(c, d))

    return (a + d) / (a + b + c + d)  # 兰德系数
```

结果分析

首先讨论 PCA 的降维效果，由于数据集原始维度为 13，在经过测试后，可得不同 threshold 的降维后维度

threshold	0.00	0.37	0.56	0.67	0.74	0.81	0.86	0.90	0.93	0.95	0.97	0.98
demention	1	2	3	4	5	6	7	8	9	10	11	12

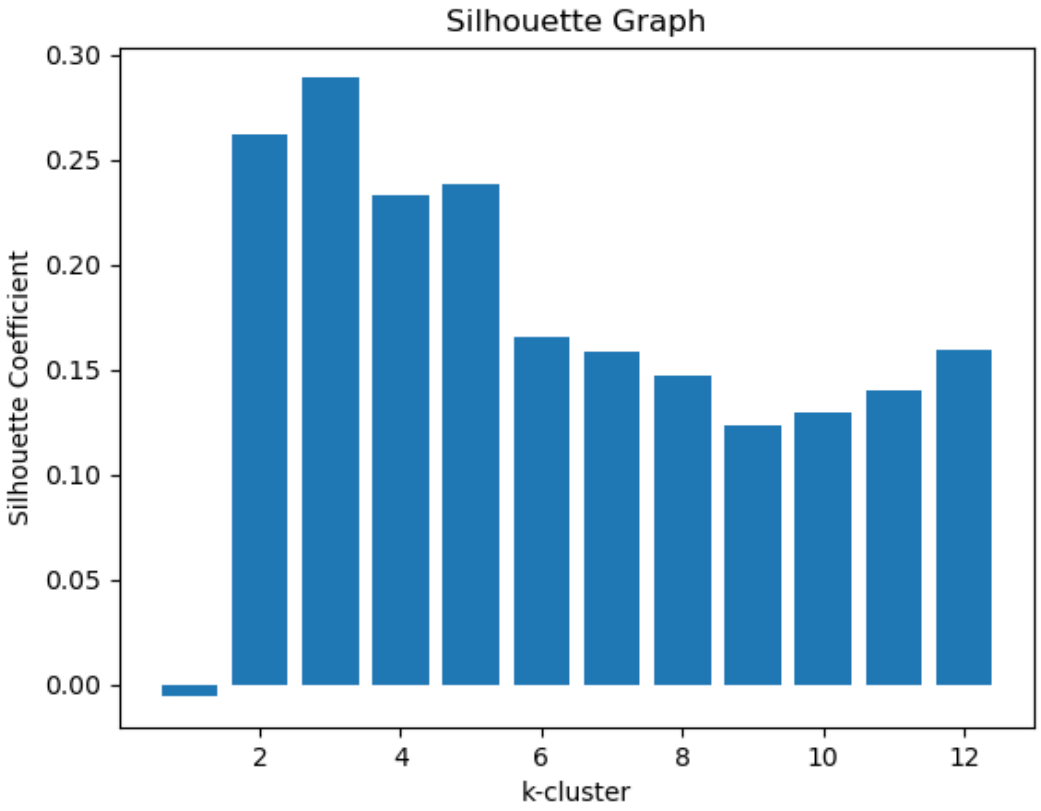
随后先在原始数据上对聚类效果进行评价，得到不同k值下的轮廓系数S和兰德系数RI

k	S	RI
2	0.1548	0.7730
3	0.2851	0.9542
4	0.2515	0.8873
5	0.2916	0.9426
6	0.2138	0.8416
7	0.2025	0.8320
8	0.1482	0.7956
9	0.1253	0.7761
10	0.1667	0.7900

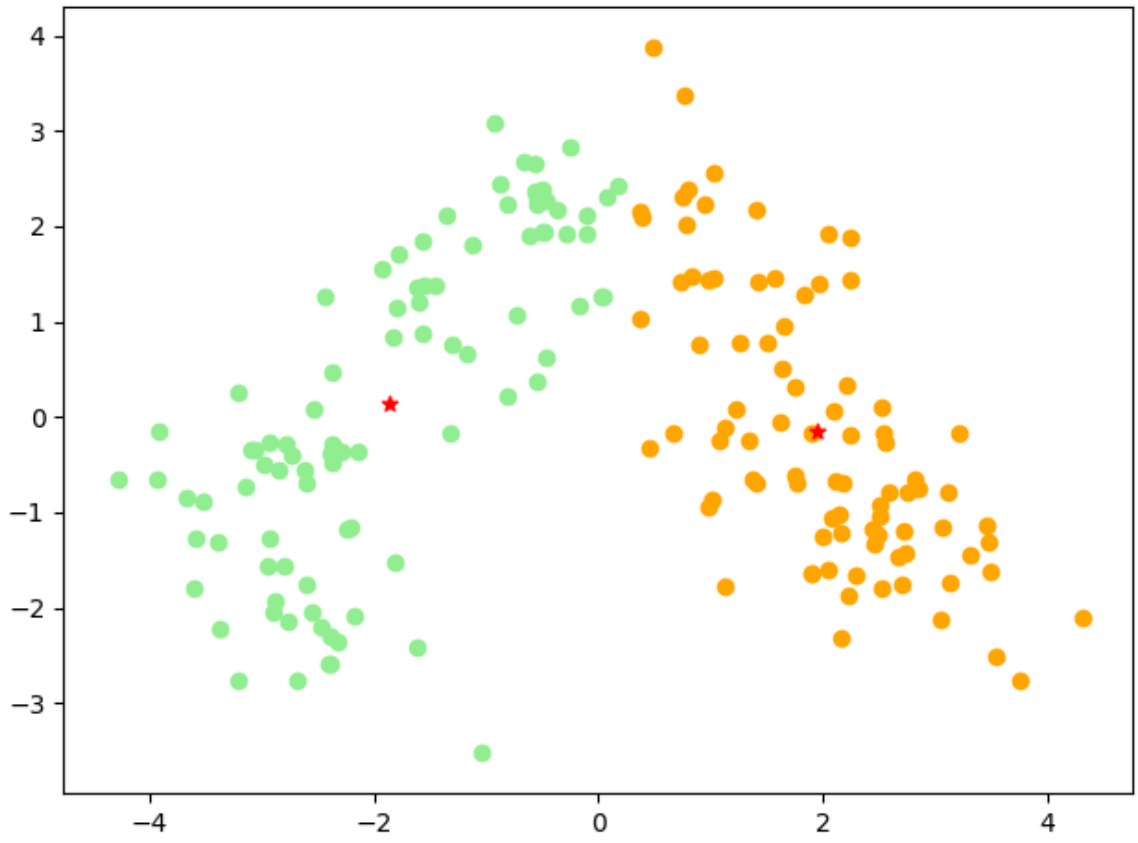
注意到在$3 \leqslant k \leqslant 5$时，指标的水平较高，随后我们加上主成分分析进行降维，得到轮廓系数与兰德系数

S|RI	1	2	3	4	5	6	7	8	9	10	11	12
1	0.54|0.51	0.35|0.50	0.29|0.60	0.33|0.70	0.27|0.70	0.30|0.71	0.27|0.72	0.26|0.71	0.28|0.76	0.30|0.75	0.25|0.73	0.25|0.75
3	0.54|0.54	0.35|0.62	0.29|0.59	0.33|0.76	0.28|0.74	0.29|0.79	0.28|0.84	0.27|0.83	0.29|0.88	0.28|0.90	0.29|0.90	0.28|0.95
4	0.54|0.57	0.35|0.65	0.29|0.62	0.28|0.77	0.27|0.77	0.28|0.85	0.25|0.84	0.25|0.84	0.25|0.89	0.29|0.93	0.28|0.93	0.27|0.97
5	0.54|0.58	0.35|0.64	0.28|0.65	0.29|0.77	0.27|0.79	0.27|0.81	0.24|0.83	0.25|0.82	0.30|0.85	0.29|0.90	0.24|0.90	0.28|0.96
6	0.57|0.61	0.35|0.67	0.27|0.67	0.26|0.78	0.26|0.77	0.24|0.81	0.23|82	0.25|0.82	0.25|0.85	0.26|0.87	0.27|0.89	0.28|0.93

可以发现，虽然随着k值的选取与维数的不同，数据呈现一定的变化趋势，主要为$k = 3$时聚类效果较好，但降维对于各项指标的提升并不明显，综上选取的默认参数为$k = 3$与$\text{threshold} = 0.99$，下面给出的是在$\text{threshold} = 0.99$的轮廓系数柱状图，图中数据与上表并不一致，这是由于随机选取的初始质心对结果影响较大



为了便于可视化，我们显式设定降维后的维度为 2，得到如下的结果


4
3
2
1
0
−1
−2
−3
−4
−2
0
2
4