人工智能实验Lab2-无监督学习问题

廖洲洲 PB17081504

实验目的

实现PCA和Kmeans算法，深入了解无监督学习问题

实验内容

数据预处理

读取数据，将数据缩放到合理范围，标准化。选择将特征值转化为0到1区间内的值

使用公式：$newValue = \frac{oldValue-min}{max-min}$

PCA算法

主要思想

在PCA中，数据从原来的坐标系转换到了新的坐标系，新坐标系的选择是由数据本身决定的。第一个新坐标轴选择的是原始数据中方差最大的方向，第二个新坐标轴的选择和第一个坐标轴正交且具有最大方差的方向。该过程一直重复，重复次数为原始数据中特征的数目。我们会发现，大部分方差都包含在最前面的几个新坐标系中。因此，我么可以忽略余下的坐标轴，即对数据进行了降维处理。

PCA算法（主成分分析）能降低数据的复杂度，识别最重要的多个特征，但可能损失有用信息。

伪码

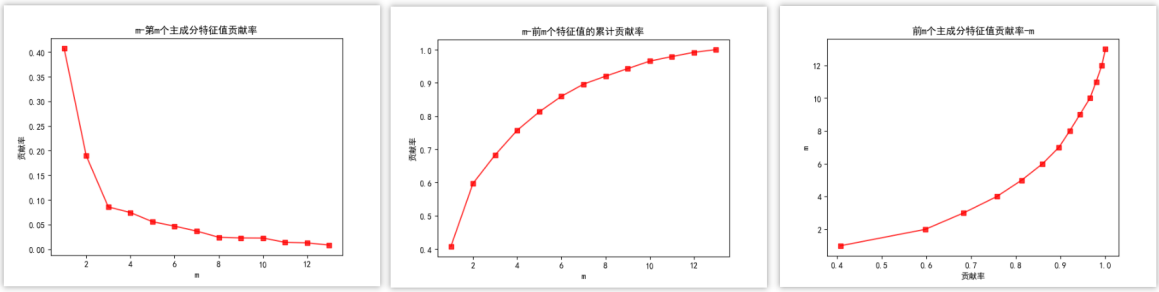
```
对所有样本进行去中心化，去除平均值
计算协方差矩阵
计算协方差矩阵的特征值和特征向量
将特征值从大到小排序
根据threshold计算N值
保留最上面的N个特征向量
将数据转换到上述n个特征向量构建的新空间中
```

主要代码

```
def pca(data,threshold=-1,firstm=2):
    """
    使用pca算法实现对数据的降维，当输入了threshold时，根据threshold值计算选择前m个特征向量
    当未输入threshold值,而输入了firstm值时，直接根据firstm值选择前firstm个特征向量
    当threshold和firstm值都未输入时，选择前2个特征向量
    :param data: 输入的高维数据
    :param threshold: 表示特征值的累计贡献度，根据该值选择前m个特征向量
    :param firstm: 直接选择前m个特征向量
    :return lowDimensionDataMat: 降维的数据
    """
    # 对所有样本进行去中心化，减去平均值
    meanValues= mean(data,axis=0)
    newData=data-meanValues
```

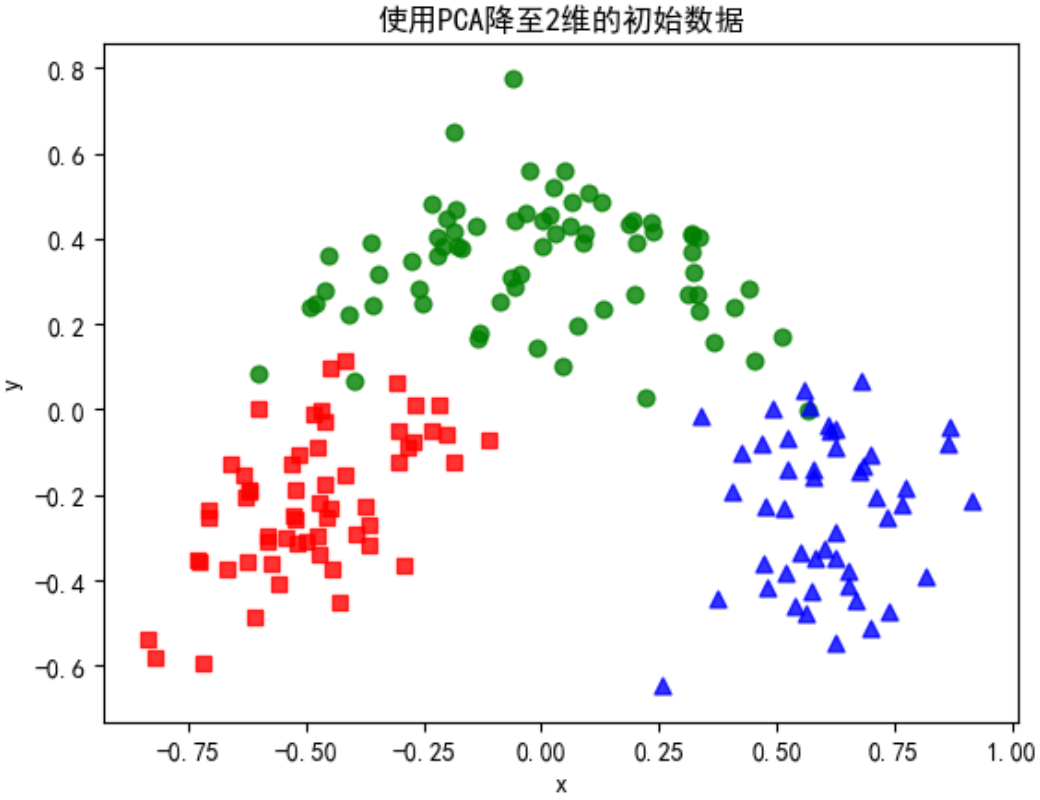
```
    # 计算协方差矩阵
    covMat = cov(newData,rowvar=0)
    # 计算协方差矩阵的特征值和特征向量
    eigValues,eigVectors = linalg.eig(mat(covMat))
    # 将特征值从大到小排序，得到索引值
    eigValuesIndex = argsort(eigValues)[-1::-1]
    # 得到排序后的特征值，特征向量
    eigValuesSort = eigValues[eigValuesIndex]
    eigVectorsSort = eigVectors[:,eigValuesIndex]
    if threshold!=-1:
        sumEigValues  = sum(eigValues)
        firstmSum = [0]
        for i in range(len(eigValuesSort)):
            firstmSum.append(firstmSum[i]+eigValuesSort[i])
        firstmSum=[x/sumEigValues for x in firstmSum]
        for i in range(len(firstmSum)):
            if(threshold <= firstmSum[i] ):
                firstm = i
                break
    firstmEigVectors = eigVectorsSort[:,:firstm]
    lowDimensionDataMat = newData * firstmEigVectors
    return lowDimensionDataMat
```

实验数据前m个主成分在总特征值中的贡献度及相应阈值对应的维度



- **左图为各个主成分特征值占总特征值和的百分比，中图为前m个主成分特征值和占总特征值和的百分比，右图为相应的threshold所对应的维度，**
- **由图中可以看出大部分特征值都包含在前几个主成分中，第一个主成分特征值贡献度达到了40%，而前5个主成分特征值贡献度就已经达到了80%。因此，舍弃后面的主成分不会损失太多信息**

对实验数据进行降维及可视化测试



- **将原13维数据降到2维后，发现数据聚集非常明显，说明pca算法在实现数据降维的同时保留了许多重要特征**

Kmeans算法

聚类是一种无监督的学习，它将相似的对象归到同一簇中，将不相似的对象归到不同簇。k-均值可以发现k个不同的簇，且每个簇的中心采用簇中所含值的均值计算而成。K-均值算法的工作流程是这样的。首先，随机确定k个初始点作为质心。然后将数据集中的每个点分配到一个簇中，具体来讲，为每个点找距其最近的质心，并将其分配给该质心所对应的簇。这一步完成之后，每个簇的质心更新为该簇所有点的平均值。

兰德系数

$$RI = \frac{a+d}{a+b+c+d}$$

假设用 C表示真实的分组情况，K表示聚类结果，那么：

a 为在 C 中为同一类且在K 中也为同一类别的数据点对数

b 为在 C中为同一类但在 K 中却隶属于不同类别的数据点对数

c 为在C 中不在同一类但在 K 中为同一类别的数据点对数

d 为在 C中不在同一类且在 K 中也不属于同一类别的数据点对

兰德系数值越大表明聚类结果匹配度更高

轮廓系数

$$S(i) = \frac{b(i)-a(i)}{max\{a(i),b(i)\}}$$

a(i) = average(i向量到所有它属于的簇中其它点的距离)

b(i) =i向量到与它相邻最近的一簇内的所有点的平均距离

将所有点的轮廓系数求平均，就是该聚类结果总的轮廓系数

轮廓系数值越大表明这个结点更匹配其属聚类而不与相邻的聚类匹配，轮廓系数接近-1，说明样本i更应该分类到另外的类，越接近0，说明样本i在两个簇的边界上

伪码

```
创建k个点作为起始质心（经常是随机选择）
当任意一个点的簇分配结果发生改变时
    对数据集中的每个数据点
        对每个质心
            计算质心与数据点之间的距离
        将数据点分配到距其最近的簇
    对每一个簇，计算簇中所有点的均值并将均值作为质心
```

(计算质心-分配-重新计算)反复迭代，直到所有数据点的簇分配结果不再改变位置。

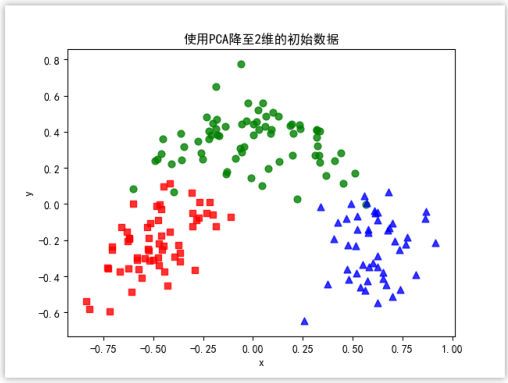
主要代码

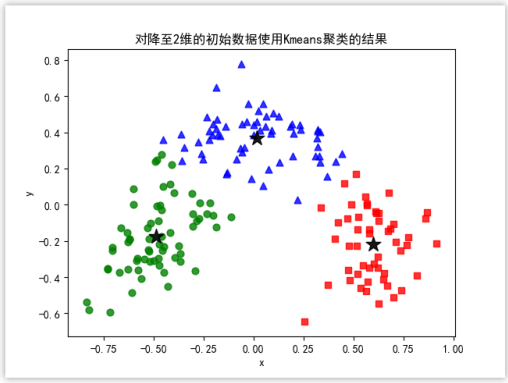
```
def Kmeans(k,data):
    """
    Kmeans算法主函数，输入数据和分类簇的个数，输出每一簇的质心和每一点的簇分配结果、与质心的距离
    :param k:分类簇的个数
    :param data:输入数据
    :return centers:各个簇的中心
    :return clusterAssign:每个数据点的分配结果和距离质心的距离
    """
    m,n = np.shape(data)
    # 记录每个数据点所分配的簇和距离该簇的质心的距离
    clusterAssign = np.mat(np.zeros((m,2)))
    # 初始随机k个质心
    centers = randCent(data,k)
    # 记录是否有点的簇分配发生改变
    isAssignChanged = True
    # 当任意一个点的簇分配发生改变时
    while isAssignChanged:
        isAssignChanged = False
        # 对数据集中的每个点
        for i in range(m):
            # 计算所有质心与该点距离，将其分配到最近的簇
            minDistance = np.Inf
            minIndex = -1
            for j in range(k):
                distance = euc_dis(data[i,:],centers[j,:])
                if distance < minDistance:
                    minDistance = distance
                    minIndex = j
            # 数据点被重新分配
            if clusterAssign[i,0] != minIndex:
                isAssignChanged = True
            clusterAssign[i,:] = minIndex,minDistance
        # 对每一个簇，计算簇中所有点的均值，并将其均值作为质心
        for j in range(k):
            points = data[np.nonzero(clusterAssign[:,0].A == j)[0]]
            centers[j,:] = np.mean(points,axis=0)
    return centers,clusterAssign
```

实验结果和分析

将原数据降到二维，其真实类别和使用Kmeans算法聚类的结果比较

左图为使用pca降到二维的原数据，右图是使用Kmeans算法对降维的数据聚类的结果，其中五角星表示簇的质心



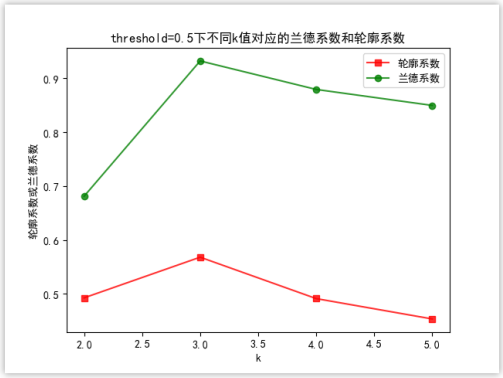


- 对比左右两图，在数据边界上会出现误分配的情况，但数据大体分布相同，说明Kmeans算法性能不错

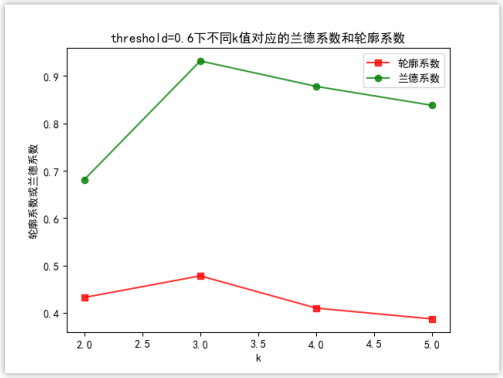
使用不同的threshold对数据进行降维，同时对降维数据使用不同数量类别聚类，比较轮廓系数和兰德系数

k=2,3,4,5 折线图

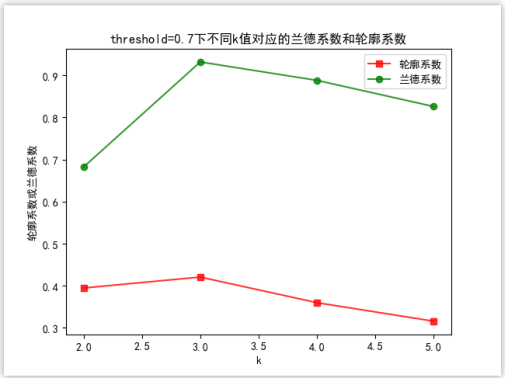
- threshold=0.5(此时原数据降为2维)



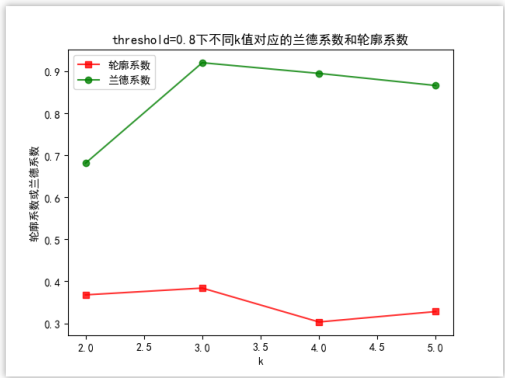
- threshold=0.6(此时原数据降为3维)



- threshold=0.7(此时原数据降为4维)

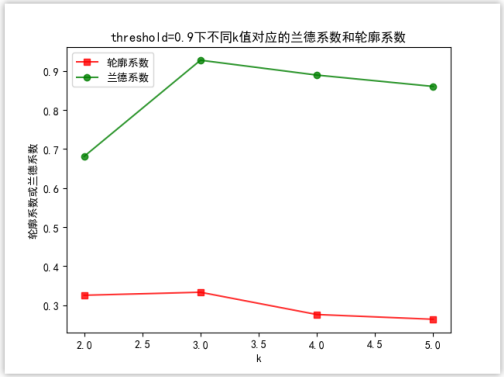


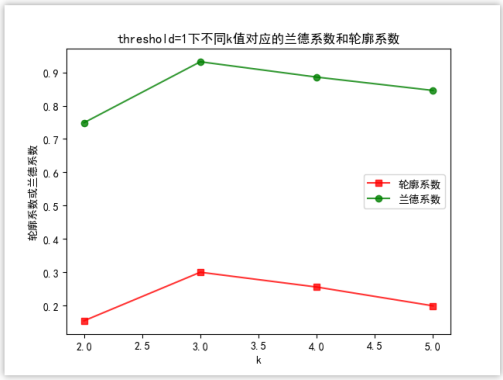
- threshold=0.8(此时原数据降为5维)



- threshold=0.9(此时原数据降为8维)

使用原始数据进行聚类





k=2,3,4,5 表格

```
threshold=0.5,dimension=2
          k=2    k=3    k=4    k=5
轮廓系数  0.493  0.568  0.491  0.454
兰德系数  0.681  0.932  0.879  0.849
```

```
threshold=0.6,dimension=3
          k=2    k=3    k=4    k=5
轮廓系数  0.432  0.478  0.410  0.387
兰德系数  0.681  0.932  0.878  0.838
```

```
threshold=0.7,dimension=4
          k=2    k=3    k=4    k=5
轮廓系数  0.395  0.421  0.360  0.316
兰德系数  0.682  0.932  0.888  0.826
```

```
threshold=0.8,dimension=5
          k=2    k=3    k=4    k=5
轮廓系数  0.368  0.384  0.303  0.328
兰德系数  0.681  0.920  0.895  0.866
```

```
threshold=0.9,dimension=8
          k=2    k=3    k=4    k=5
轮廓系数  0.326  0.333  0.276  0.264
兰德系数  0.681  0.927  0.889  0.860
```

```
使用原始数据进行聚类
          k=2    k=3    k=4    k=5
轮廓系数  0.154  0.300  0.255  0.199
兰德系数  0.749  0.932  0.886  0.846
```

- 当轮廓系数最大时，有threshold=0.5（此时dimension=2），k=3(分为3类)，从文件\output\k_3_dimension_2.csv查看聚类结果和原数据分类比较可以发现分类基本正确。与未经pca算法处理的数据聚类的兰德系数比较，发现两者相同，说明尽管原数据降为2维损失了一定的信息，但大部分的重要信息依旧保留，所以分类结果仍是比较好的。

- 可以发现，使用不同的降维数据，只要分类的k值不变，兰德系数变化均不大，由此可以说明pca算法将大部分信息依旧保留在了降维后的数据中。
- 对于使用相同的降维数据，一般情况下k=2，即分为2类是兰德系数较小，k=3时兰德系数最大，k=4、5时兰德系数在前两者之间，说明最好将数据分为3类，这也与原数据为3类相符
- 使用不同的降维数据，当k不变时，随着threshold的增大，即pca处理后数据维度的增加，轮廓系数越来越小，说明数据维度的增加会使得数据与数据边界附近的点的分类判断越来越困难
- 使用相同的降维数据，一般情况下k=3时轮廓系数最大，k=2、4、5时小于k=3时的值，也反因此最好将数据分为3类

实验总结

1. 实现了PCA和Kmeans算法，加强了对无监督学习的理解。
2. PCA算法可以降低数据的复杂度，识别最重要的多个特征。降低了数据的维度后，数据会更容易处理，相关特性也可能在数据中更加明确地显示出来。
3. Kmeans算法能发现给定数据集的k个簇，每一簇通过其质心描述，由此可以给数据进行聚类。