



实 验 报 告

年 级 专 业: 20 级智能科学与技术

学 生 姓 名: 孙成

学 号: 20203101694

课 程 名 称: 机器学习

实 验 名 称: 神经网络

任 课 老 师: 胡祝华

实验报告成绩	
任课教师签名	

海南大学 · 信息与通信工程学院

School of Information and Communication Engineering, Hainan University

实验室	学院 118	计算机号	自带电脑
实验软硬件环境	1. 实验所用到的硬件环境 MacBook Pro (16-inch, 2019) 2.6GHz 六核 Intel Core i7 16 GB2667 MHz DDR4 2. 实验所用到的软件环境 PyCharm 2021.3.2 (Professional Edition)		
实验目的或要求	实验目的： 1. 理解聚类的基本概念，分类和聚类的区别，有监督聚类和无监督聚类的区别。 2. 理解并掌握 K-means 聚类算法的基本原理 3. 学会用 python 实现 K-means 聚类算法 实验要求： 1. 按照实验步骤独立完成实验。 2. 整理并上交实验报告和实验源程序。 ..\网络应用开发实验报告模板.doc 3. 考核：以学生的实验报告情况和做实验时的表现为考核依据。		
实验内容	1) 导入标准库 2) 定义一个函数计算欧式距离 3) 定义函数：初始化质心 4) 定义核心函数实现 kmeans 聚类 5) 定义函数：选择不同的 K 值进行交叉验证 6) 定义函数：作图可视化 kmeans 聚类效果 7) 对模型进行测试 8) 实验结果 9) 直接使用 sklearn 库里面封装好的 k-means 算法进行实验。		
实验结果	<pre>import numpy as np import pandas as pd import matplotlib.pyplot as plt #定义函数： 计算欧式距离 def euclDistance (point1,point2) : #计算两点 point1、 point2 之间的欧式距离 distance = np. sqrt (sum (pow (point2-point1,2))) return distance</pre>		

```

def initCentroids(dataSet, k):
    # dataSet 为数据集
    # k 是指用户设定的 k 个簇
    numSamples, dim = dataSet.shape # numSample: 数据集数量; dim:
    特征维度
    centroids = np.zeros((k, dim)) # 存放质心坐标, 初始化 k 行、 dim
    列零矩阵
    for i in range(k):
        # index = int(np.random.uniform(0, numSamples)) # 给出一个
        服从均匀分布的在 0~numSamples 之间的整数
        index = np.random.randint(0, numSamples) # 给出一个随机分布
        在 0~numSamples 之间的整数
        centroids[i, :] = dataSet[index, :] # 第 index 行作为质心
    return centroids

def kmeans(dataSet, k):
    # dataSet 为数据集
    # k 是指用户设定的 k 个簇
    numSamples = dataSet.shape[0]
    clusterAssment = np.zeros((numSamples, 2)) # clusterAssment 第
    1 列存放所属的簇, 第 2 列存放与质心的距离
    clusterChanged = True # clusterChanged=False 时迭代更新终止
    ## step 1: 初始化质心 centroids
    centroids = initCentroids(dataSet, k) # 循环体: 是否更新质心
    while clusterChanged:
        clusterChanged = False # 关闭更新
        # 对每个样本点
        for i in range(numSamples):
            minDist = 100000.0 # 最小距离
            minIndex = 0 # 最小距离对应的簇
            # step2: 找到距离每个样本点最近的质心
            # 对每个质心
            for j in range(k):
                distance =
            euclDistance(centroids[j, :], dataSet[i, :]) # 计算每个样本点到质
            心的欧式距离
            if distance < minDist: # 如果距离小小当前最小距离 minDist

```

```

        minDist = distance #最小距离更新
        minIndex = j # 样本所属的簇也会更新
        ## step3: 更新样本所属的簇
        if clusterAssment[i, 0] != minIndex: # 如当前样本不属于
该簇
            clusterChanged = True #聚类操作需要继续
            clusterAssment[i, :] = minIndex, minDist
        # step 4: 更新质心
        # 对每个质心
        for j in range(k):
            pointsInCluster = dataSet[np.nonzero(clusterAssment[:,
0] == j)[0]]
            # pointsInCluster 存储的是当前所有属于簇 j 的 dataSet 样本
点
            centroids[j, :] = np.mean(pointsInCluster, axis=0) # 更
新簇 j 的质心
        print("cluster complete!")
        return centroids, clusterAssment

def selectK(dataSet, k_list):
    # dataSet 为数据集
    # k_list 不同k 值列表
    distanceK=[] #存储不同k 值下每个样本点到质心的平均欧式距离
    for i, k in enumerate(k_list):
        centroids, clusterAssment = kmeans(dataSet, k) #调用 kmeans
函数
        distance = np.mean(clusterAssment[:, 1], axis=0) #
clusterAssment 所有 minDist 的平
        distanceK.append(distance)
    #best k =klist[np.argmin (distanceK)] #能够让距离最小的k 值
    return distanceK

def showCluster(dataSet, k, centroids, clusterAssment):
    # dataSet 为数据集
    #k 是指用户设定的k 个簇
    # centroids 存放质心坐标
    #clusterAssment 第1 列存放所属的簇, 第2 列存放与质心的距离
    numSamples, dim = dataSet.shape # numSample: 数据集数量; dim:

```

特征维度

```
if dim!= 2:
    print("The dimension of data is not 2!")
    return 1

mark= ['or', 'ob', 'og', 'ok', '^r', '+r', 'sr', 'dr', '<r',
'pr']
if k > len(mark):
    print("K is too large!")
    return 1
#画所有的样本
plt.figure()
for i in range(numSamples):
    markIndex = int(clusterAssment[i, 0])
    plt.plot(dataSet[i, 0], dataSet[i, 1], mark[markIndex])
    mark = ['Dr', 'Db', 'Dg', 'Dk', '^b', '+b', 'sb', 'db', '<b',
'pb']
#画所有的质心
for i in range(k):
    plt.plot(centroids[i, 0], centroids[i, 1], mark[i],
ms=12.0)
plt.title('K-means (K={})'.format(k))
plt.xlabel('x')
plt.ylabel('y')
plt.show()

#导入数据集
dataSet =
pd.read_csv('/Users/suncheng/PycharmProjects/machineLearning
/Lab3/dataSet.csv')
dataSet = dataSet.values #(80,2)
#选择不同的k 值对比
k_list = [2, 3, 4, 5, 6]
disK = selectK(dataSet, k_list) #画图
plt.figure()
plt.plot(k_list, disK, 'ro-')
plt.title('Cross-validation on k')
plt.xlabel('K')
```

```
plt.ylabel('Mean Euclidean distance')
plt.show()
# 使用 K-means 算法进行聚类
k = 4
centroids, clusterAssment = kmeans(dataSet, k) #作图可视化 kmeans
聚类效果
showCluster(dataSet, k, centroids, clusterAssment)
结果如下:
```

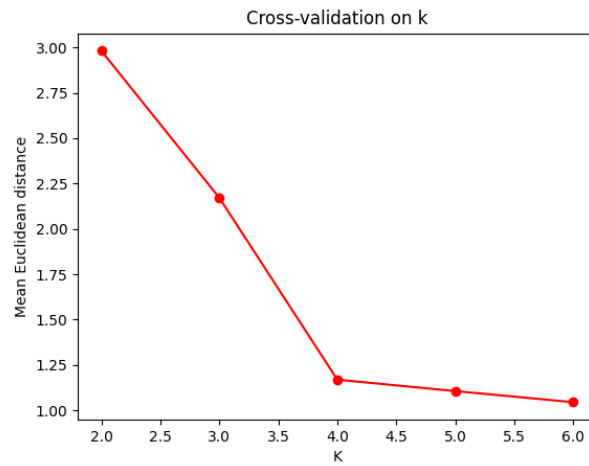


图 1 结果图

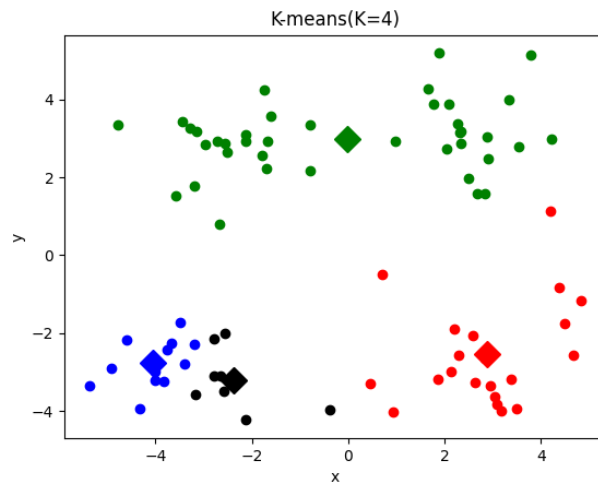


图 2 结果图

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans

plt.rcParams['font.sans-serif'] = ['Heiti TC']
# X 为样本特征, Y 为样本簇类别, 共 1000 个样本, 每个样本 4 个特征, 共 4
# 个簇, 簇中心 在[-1,-1], [0,0], [1,1], [2,2], 簇方差分别为[0.4, 0.2,
# 0.2]
X, y = make_blobs(n_samples=1000, n_features=2, centers=[[-1,
-1], [0, 0], [1, 1], [2, 2]],
                  cluster_std=[0.4, 0.2, 0.2, 0.2],
                  random_state=9)
plt.scatter(X[:, 0], X[:, 1], marker='o')
plt.title('数据集展示')
plt.show()
print(X)

# k=2

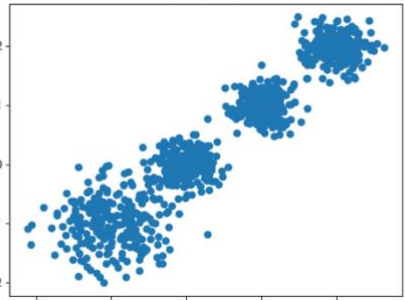
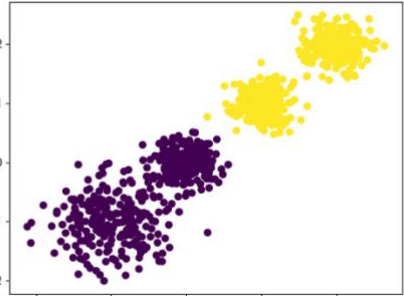
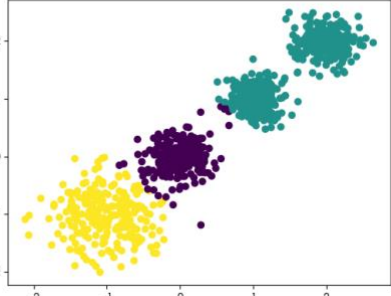
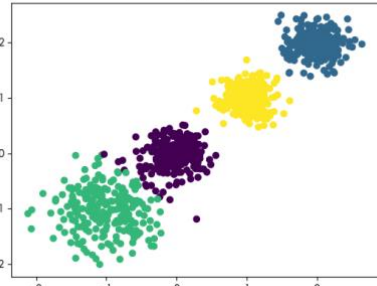
y_pred = KMeans(n_clusters=2, random_state=9).fit_predict(X)
plt.scatter(X[:, 0], X[:, 1], c=y_pred)
plt.title('k=2')
plt.show()

# k=3

y_pred = KMeans(n_clusters=3, random_state=9).fit_predict(X)
plt.scatter(X[:, 0], X[:, 1], c=y_pred)
plt.title('k=3')
plt.show()

# k=4

y_pred = KMeans(n_clusters=4, random_state=9).fit_predict(X)
plt.scatter(X[:, 0], X[:, 1], c=y_pred)
plt.title('k=4')
plt.show()
```

	<div data-bbox="315 250 735 588"> <p>数据集展示</p>  </div> <div data-bbox="791 235 1211 568"> <p>k=2</p>  </div> <div data-bbox="329 676 735 1009"> <p>k=3</p>  </div> <div data-bbox="819 686 1211 1009"> <p>k=4</p>  </div>
<p>心得 体会</p>	<p>本次试验我受益匪浅，其中最大的收获是对于 class 类的学习，明白在同一个 class 类中定义的函数调用其他函数，需要在函数名前加 ‘self.’ 一开始实验好几次都不太行，后面就快放弃时，找到了网上的在同一个 class 类中定义的函数调用其他函数的演示代码。除此之外我还学会了使用 model，以及调用其他自定义的 model，并且能够改变其文件位置，知道了__init__里面是放什么东西。本次试验是验证性试验，但是还是有难度的，希望早日消化其中的代码。</p>