电子科技大学信息与软件工程学院

**实 验 报 告**

学 号 2019091604016

姓 名 夏野

（实验） 课程名称 人工智能

实验时间 11月21日

实验教师 周帆

**电 子 科 技 大 学**

**实 验 报 告**

1. **实验名称：**

无监督聚类算法：dbscan与k-means

1. **实验学时：**

4学时

1. **实验目的：**

了解无监督学习中重要的聚类算法，学会使用sklearn包搭建聚类算法解决分类任务。基本了解dbscan和k-means的算法区别，以及编程思想的区别。最后，通过算法原理分析体会这两个算法的优缺点。

1. **实验原理：**
2. 无监督学习：是一种机器学习的训练方式，它本质上是一个统计手段，在没有标签的数据里可以发现潜在的一些结构的一种训练方式。它主要具备3个特点：
   1. 无监督学习没有明确的目的
   2. 无监督学习不需要给数据打标签
   3. 无监督学习无法量化效果
3. 聚类：聚类是把相似的数据分成不同的组（相似或距离或其他），我们不关心组的类型。我们只需要将数据分组，并确保组之间尽可能地不相关。因此聚类不需要合适的数据和学习。这是一种无监督的学习。
4. k-means：初始化k个不同的中心点{u1, u2, ..., uk}，重复一下两个步骤直到收敛：每个训练样本分配到最近的中心点ui所代表的聚类i；每一个中心点ui更新为聚类i中所有训练样本的均值。
5. DBSCAN：由密度可达关系导出的最大密度相连的样本集合，即为我们最终聚类的一个类别，或者说一个簇。这个DBSCAN的簇里面可以有一个或者多个核心对象。如果只有一个核心对象，则簇里其他的非核心对象样本都在这个核心对象的ϵ-邻域里；如果有多个核心对象，则簇里的任意一个核心对象的ϵ-邻域中一定有一个其他的核心对象，否则这两个核心对象无法密度可达。这些核心对象的ϵ-邻域里所有的样本的集合组成的一个DBSCAN聚类簇。
6. **实验内容：**
7. 加载数据集，Data\_for\_Cluster.npz ；
   1. X 为特征，labels\_true为标签。
8. 搭建模型，k-means与dbscan。
9. 训练模型，调参，得出分类结果。
10. 结果分析及可视化。
    1. 绘制散点图。(根据分类结果进行着色) ；
    2. 从算法原理的角度分析两个算法优缺点，及适应的数据集特征。
11. 评估标准，轮廓系数法（Silhouette Cofficient），用来评估聚类算法的效果。
12. **实验器材（设备、元器件）：**

1. 硬件：mac笔记本电脑

2. 软件：ios系统，pycharm

1. **实验步骤：**
2. dbscan聚类算法：
   1. 加载数据集：将文件保存在本地，格式为npz，我使用numpy包中的load方法对数据进行加载并读取。其中有两列数据，一个X表示特征，一个labels\_true表示标签。打印数据形状后，得到X的大小是(1000, 2)，标签的大小是(1000,)，可以看到X有2个特征；
   2. 搭建模型：使用sklearn.cluster包下的DBSCAN类进行模型搭建；
   3. 模型训练：在这个任务里，我首先对eps进行了参数的判断，设置范围为0.1-0.5，步长为0.001，每一次我都进行模型的训练，并且通过labels\_获得预测数据，然后通过metrics中的轮廓系数作为评价指标，描绘出该系数随eps变化的曲线，得到大概eps在0.34时轮廓系数最大；其次我又对min\_samples进行参数调整，将范围设置为2-41，同时固定eps值为0.34，获得轮廓系数随min\_samples变化的曲线，得到在20的位置轮廓系数最大。因此，通过调参之后，最终确定eps=0.34，min\_samples=20。
   4. 结果可视化：使用matplotlib进行结果可视化，将数据的两个特征分别通过散点图的形式打印出来;
   5. 模型评估：通过metrics中的轮廓系数作为评价指标，获得其平均轮廓系数。
3. k-means算法：
   1. 加载数据集：将文件保存在本地，格式为npz，我使用numpy包中的load方法对数据进行加载并读取。其中有两列数据，一个X表示特征，一个labels\_true表示标签。打印数据形状后，得到X的大小是(1000, 2)，标签的大小是(1000,)，可以看到X有2个特征；
   2. 搭建模型：通过数据可视化，发现数据一共是3个种类，因此我设置n\_clusters=3，使用sklearn.cluster中的KMeans搭建模型，将数据X放入fit中，获得具备分类能力的模型；
   3. 模型训练：将X送入模型中，获得其预测值，然后通过metrics中的轮廓系数获得其平均轮廓系数，作为评价指标；
   4. 结果可视化：首先模型中的训练得来的个数据中心centers，然后通过numpy中where方法找到相同类别的索引集合，这样就获得了n\_clusters个集合，然后将它和centers一起在图中以散点图的形式显示出来即可。
4. **实验结果与分析（含重要数据结果分析或核心代码流程分析）**

1. 实验一 - dbscan聚类算法：

a. 重要数据结果分析：

（1）结果可视化：

Chart, scatter chart

Description automatically generated

结果分析：

成功将数据可视化为散点图。

（2）运行结果：

Text

Description automatically generated

结果分析：

成功估计出平均轮廓系数，完成实验。

b. 核心代码及流程分析

(1)装载数据

path = 'Data\_for\_Cluster.npz'  
data = np.load(path)  
X = data['X']  
y = data['labels\_true']

代码分析：

将数据及文件相对路径`Data\_for\_cluster.nzp`赋值给path变量。接着调用load()函数将数据集装载进data变量中。进行数据划分，分别给X和y变量。

（2）模型训练代码

dbscan = DBSCAN(eps=0.34, min\_samples=20).fit(X)  
y\_pre = dbscan.labels\_

代码分析:

调用DBSCAN()函数，并将参数`eps=0.34`, `min\_sample=20`传入给方法。参数的具体确定是经过多次实验而最终确定的。

（3）数据可视化代码

plt.scatter(X[:, 0], X[:, 1], c=y\_pre)  
plt.show()

代码分析：

分别调用scatter()方法以及show()方法来存入数据以及展示数据。

（4）模型评估代码

score = metrics.silhouette\_score(X, y\_pre)  
print("平均轮廓系数: ", score)

代码分析：

通过metrics中的轮廓系数作为评价指标，调用metrics中的silhouette\_scoure()函数获得其平均轮廓系数。并打印相关值。

3. 实验二 - k-means算法：

a. 重要数据结果分析

（1）结果可视化：

Chart, scatter chart

Description automatically generated

结果分析：

成功作出不同cluster的中心点的图。

（2）代码运行结果

Text

Description automatically generated

结果分析:

成功运行代码并且成功计算出了平均轮廓系数以及centers的三个三个坐标。

b. 核心代码及流程分析

（1）装载数据集代码

path = 'Data\_for\_Cluster.npz'  
data = np.load(path)  
X = data['X']  
y = data['labels\_true']

代码分析：

将文件保存在本地，格式为npz，我使用numpy包中的load方法对数据进行加载并读取。其中有两列数据，一个X表示特征，一个labels\_true表示标签。打印数据形状后，得到X的大小是(1000, 2)，标签的大小是(1000,)，可以看到X有2个特征。

（2）数据可视化

plt.scatter(X[:, 0], X[:, 1], c=y)  
plt.show()

代码分析：

具体可视化方法和前面几个实验类似，这里不具体描述。

（3）搭建数据集

n\_clusters = 3  
cls = KMeans(n\_clusters).fit(X)

代码分析：

根据上面代码数据可视化的结果，我们可以发现数据一共是3个种类，因此我们将变量 n\_clusters的值设置为3，使用sklearn.cluster中的KMeans搭建模型，将数据X放入fit中，使cls获得具备分类能力的模型；

（4）模型训练

y\_pre = cls.predict(X)  
n\_samples, n\_features = X.shape   
inertias = cls.inertia\_   
silhouette\_s = metrics.silhouette\_score(X, y\_pre, metric='euclidean')   
print("平均轮廓系数: ", silhouette\_s)  
centers = cls.cluster\_centers\_   
print("centers: ", centers)

代码分析：

首先我们调用cls中的predirct()函数，将X送入模型中，获得其预测值y\_pre，然后，然后，我们通过通过metrics中的silhouette\_scoure()函数获得其平均轮廓系数silhouette\_s，作为评价指标；并且将cls中cluster\_centers的坐标赋值给centers变量。

（5）结果数据可视化代码

colors = ['b', 'g', 'r']  
plt.figure()   
for i in range(n\_clusters):   
 index\_sets = np.where(y\_pre == i)   
 cluster = X[index\_sets]   
 plt.scatter(cluster[:, 0], cluster[:, 1], c=colors[i], marker='.')   
 plt.plot(centers[i][0], centers[i][1], '\*', markerfacecolor=colors[i], markeredgecolor='k', markersize=6)  
plt.show()

代码解析：

首先，调用plt中的figure()方法建立画布。接着循环遍历3次，分别将我们的中心坐标点标记在图中。

1. **总结及心得体会：**

通过实验三，我了解了无监督学习中重要的聚类算法，并且学会了使用sklearn包搭建聚类算法解决分类任务的方法。同时我还对于dbscan和k-means有了基本的了解，了解了他们算法的区别以及编程思想上的一些区别：

1. k-means需要指定聚类簇数k，并且且初始聚类中心对聚类影响很大。k-means把任何点都归到了某一个类，对异常点比较敏感。DBSCAN能剔除噪声，需要指定邻域距离阈值eps和样本个数阈值MinPts，可以自动确定簇个数。
2. K均值和DBSCAN都是将每个对象指派到单个簇的划分聚类算法，但是K均值一般聚类所有对象，而DBSCAN丢弃被它识别为噪声的对象。
3. K均值算法的时间复杂度是O(m)，而DBSCAN的时间复杂度是O(m^2)
4. K均值可以发现不是明显分离的簇，即便簇有重叠也可以发现，但是DBSCAN会合并有重叠的簇。