# 机器学习导论

## 编程作业：聚类

姓名：陶威

班级：电信1604班

学号：U201613331

指导老师：王邦

### 任务1：对地理数据应用二分k-均值算法聚类

**实验思路：**

先利用二分k-均值的思想，根据总误差的减小程度，逐步并多次地将样本簇一分为二，直到达到指定的簇数目。再使用matplotlib画出每个簇的中心点并标识在地图上。

**实验流程：**

1. 载入数据，按行切割，提取出地点坐标，转化为矩阵

*for* line *in* open('data/places.txt').readlines():  
 line\_arr = line.split('\t')  
 dat\_list.append([float(line\_arr[4]), float(line\_arr[3])])  
dat\_mat = mat(dat\_list)

1. 对提取到的所有坐标点采用二分k-均值算法进行聚类，即先将所有点作为一个簇，然后将该簇一分为二，之后选择其中一个簇继续进行划分，直到达到指定的簇数目

*# dataset为数据集，k为分簇数目，dist\_eclud为距离函数  
def* bi\_k\_means(dataset, k, dist\_meas=dist\_eclud):  
 m = shape(dataset)[0]  
 cluster\_assment = mat(zeros((m, 2)))  
 *# 将所有样本的均值作为簇中心* centroid0 = mean(dataset, axis=0).tolist()[0]  
 *# 创建簇中心列表* cent\_list = [centroid0]  
 *# 计算每个样本的误差  
 for* j *in* range(m):  
 cluster\_assment[j, 1] = dist\_meas(mat(centroid0), dataset[j, :]) \*\* 2  
 *while* len(cent\_list) < k:  
 lowest\_sse = inf  
 *# 拆分每个簇，并计算拆分后的SSE，选择拆分后SSE最小的簇，保存拆分  
 for* i *in* range(len(cent\_list)):  
 pts\_in\_curr\_cluster = dataset[nonzero(cluster\_assment[:, 0].A == i)[0], :]  
 centroid\_mat, split\_clust\_ass = k\_means(pts\_in\_curr\_cluster, 2, dist\_meas)  
 sse\_split = sum(split\_clust\_ass[:, 1])  
 sse\_not\_split = sum(cluster\_assment[nonzero(cluster\_assment[:, 0].A != i)[0], 1])  
 print("sse\_split, and notSplit: ", sse\_split, sse\_not\_split)  
 *if* (sse\_split + sse\_not\_split) < lowest\_sse:  
 best\_cent\_to\_split = i  
 best\_new\_cents = centroid\_mat  
 best\_clust\_ass = split\_clust\_ass.copy()  
 lowest\_sse = sse\_split + sse\_not\_split  
 *# 一个簇拆分为二后，其中一个簇新增加簇索引，另一个保存原簇索引号* best\_clust\_ass[nonzero(best\_clust\_ass[:, 0].A == 1)[0], 0] = len(cent\_list)  
 best\_clust\_ass[nonzero(best\_clust\_ass[:, 0].A == 0)[0], 0] = best\_cent\_to\_split  
 print('the best\_cent\_to\_split is: ', best\_cent\_to\_split)  
 print('the len of best\_clust\_ass is: ', len(best\_clust\_ass))  
 *# 重置簇中心* cent\_list[best\_cent\_to\_split] = best\_new\_cents[0, :].tolist()[0]  
 cent\_list.append(best\_new\_cents[1, :].tolist()[0])  
 *# 调整样本的簇索引号及误差* cluster\_assment[nonzero(cluster\_assment[:, 0].A == best\_cent\_to\_split)[0], :] = best\_clust\_ass  
 *return* mat(cent\_list), cluster\_assment

1. 对分类后的结果进行可视化，即用不同的颜色和图形来标识每个簇，同时标注出每个簇的中心点（中心点为样本的均值）

*# 定义画布，背景*fig = plt.figure()  
rect = [0.0, 0.0, 1.0, 1.0]  
*# 不同图形标识*scatter\_markers = ['s', 'o', '^', '8', 'p', 'd', 'v', 'h', '>', '<']  
axprops = dict(xticks=[], yticks=[])  
ax0 = fig.add\_axes(rect, label='ax0', \*\*axprops)  
*# 导入地图*img\_p = plt.imread('Portland.png')  
ax0.imshow(img\_p)  
ax1 = fig.add\_axes(rect, label='ax1', frameon=*False*)  
*# 采用不同图形标识不同簇  
for* i *in* range(num\_clust):  
 pts\_in\_curr\_cluster = dat\_mat[nonzero(clust\_assing[:, 0].A == i)[0], :]  
 marker\_style = scatter\_markers[i % len(scatter\_markers)]  
 ax1.scatter(  
 pts\_in\_curr\_cluster[:, 0].flatten().A[0],  
 pts\_in\_curr\_cluster[:, 1].flatten().A[0],  
 marker=marker\_style,  
 s=90  
 )  
*# 采用‘+’表示簇中心*ax1.scatter(my\_centroids[:, 0].flatten().A[0], my\_centroids[:, 1].flatten().A[0], marker='+', s=300)  
plt.show()

**实验结果：**

K=3：



K=5:



### 任务2：根据用户采集的WiFi信息对用户进行聚类

**数据处理：**

先载入数据，调用内置csv的库提取原始信息

*def* load\_data(filename):  
 *with* open(filename, 'r', encoding='iso-8859-1') *as* f:  
 *return* list(csv.reader(f))

再分割处理数据，具体思想为：

1. 求出训练集数据中所有BSSID的并集，并计算其特征数
2. 将训练集数据根据finLabel进行分组，针对每个分组构造一个向量
3. 根据上述并集和特征数，再来逐个处理每个分组，将处于总集合中的BSSID的特征设置为其RSSLabel，不在总集合中的BSSID的特征（即为缺失值）设置为-100（测试过0和-100，发现设置为-100时的DBI值更小）
4. 同时提取每一个分组的RoomLabel作为其标签

整体代码如下：

*def* split\_data(dataset):  
 *# 求所有训练数据中BSSID的并集* bssids = list(set(np.array(dataset)[:, 0]))  
 feature\_num = len(bssids)  
  
 *def* generate\_feature(data):  
 labels = data[0]  
 dataset = data[1:]  
 X = []  
 y = []  
 *# 按指纹分组* train\_data\_group = groupby(dataset, *lambda* x: x[-1])  
 *for* fin, group *in* train\_data\_group:  
 feature = [-100] \* feature\_num  
 room\_label = 0  
 group = list(group)  
 *for* line *in* group:  
 *# 再做一次判断是因为测试数据集中可能出现训练数据集中没有的BSSID  
 if* line[0] *in* bssids:  
 feature[bssids.index(line[0])] = int(line[1])  
 room\_label = int(line[2])  
 X.append(feature)  
 y.append(room\_label)  
 *return* X, y, labels

**实现思路：**

1. 先根据数据处理部分的思想处理数据得到训练集特征及其标签

X, y, labels = split\_data(load\_data('data/DataSetKMeans1.csv'))

1. 再使用MDS对BSSID进行降维

*def* reduce\_dimension(X, n):  
 *# 使用MDS进行降维  
 from* sklearn.manifold *import* MDS  
 embedding = MDS(n\_components=n)  
 *return* embedding.fit\_transform(X)

X = reduce\_dimension(X, 5)

1. 调用sklearn的KMeans和davies\_bouldin\_score来对数据进行分类和计算误差，格式化输出结果

*from* sklearn.cluster *import* KMeans  
*from* sklearn.metrics *import* davies\_bouldin\_score  
  
clusters\_num = range(2, 10)  
  
X, y, labels = split\_data(load\_data('data/DataSetKMeans1.csv'))  
X = reduce\_dimension(X, 5)  
print("dataset1")  
print("clusters\_num:", ', '.join(map(str, clusters\_num)))  
print\_str = "DBI:"  
*for* n *in* clusters\_num:  
 kmeans = KMeans(n\_clusters=n).fit(X)  
 print\_str += str(' %.2f' % davies\_bouldin\_score(X, kmeans.labels\_))  
print(print\_str, '\n')  
  
X, y, labels = split\_data(load\_data('data/DataSetKMeans2.csv'))  
X = reduce\_dimension(X, 5)  
print("dataset2")  
print("clusters\_num:", ', '.join(map(str, clusters\_num)))  
print\_str = "DBI:"  
*for* n *in* clusters\_num:  
 kmeans = KMeans(n\_clusters=n).fit(X)  
 print\_str += str(' %.2f' % davies\_bouldin\_score(X, kmeans.labels\_))  
print(print\_str, '\n')

**实验结果：**

不同数据集中簇数目和DBI指数的关系：

