# Lab\_4 基于k-means的表情特征聚类

## 一、实习目的与要求

1、结合实际应用理解k-means过程；

2、理论结合实践，采用K-means方法对自发表情数据聚类；

3、通过进一步查阅文献，了解相关研究方向的最新研究进展。

## 二、实习题目

基于K-MEANS方法的表情特征聚类。

#### 【实验数据】

2479张人脸表情图像，要求聚类描述人脸的几种表情特征

#### 【过程描述】

1. 数据预处理，清洗，提取特征（至少选一种颜色直方图、sift、hog、lbp等）
2. 要求采用K-means自动聚类人脸表情特征
3. 要求可视化聚类特征
4. 要求采用聚类评估方法评估聚类结果

#### 【源代码】

（附上源代码）

1：数据处理，过滤掉错误的图像，与xlsx中没有对应标签的图像（去除没有对应标签的是为了评估时有依据）

**def** LoadData():  
 dataSet=[];dataLabel=[];  
 info=xlrd.open\_workbook(**"URLsWithEmotionCat.xlsx"**);  
 info\_sheet1=info.sheet\_by\_index(0);  
 rows,columns=info\_sheet1.nrows,info\_sheet1.ncols;  
   
 **for** i **in** range(1,rows):  
 *#根据xlsx中第一列的数据获取图像名称,为字符串中最后一个/后面的字符串* url=info\_sheet1.cell\_value(i,0);  
 temp=re.split(**'/'**,url);  
 imageName=temp[len(temp)-1];  
  
 *#读取图像* **try**:  
 *#获取初始图像* init\_image=mpimg.imread(**"urlf/"**+imageName);  
 *#缩放减少计算量* image=transform.resize(init\_image,(200,200));  
 *#灰度化* gravity = np.array([0.299, 0.587, 0.114])  
 data=np.array(image);  
 data=np.dot(data,gravity.T)  
 *#将其转为一维数组  
 #data=data.flatten()* **for** j **in** range(1,columns):  
 **if**(int(info\_sheet1.cell\_value(i,j))!=-1):  
 dataLabel.append(j-1);  
 **break**;  
  
 dataSet.append(data);  
  
 *#判断图像是否缩放成功* **except**:  
 **continue**;  
 print(i,imageName);  
 print(**"This picture isn't exist"**);  
 print(len(dataSet));  
  
 np.save(**"docData.npy"**,dataSet);  
 np.save(**"docLabel.npy"**,dataLabel);  
  
 dataSet=np.load(**"docData.npy"**);  
 print(np.array(dataSet).shape)  
 dataLabel=np.load(**"docLabel.npy"**);  
 print(np.array(dataLabel).shape)  
 **return** dataSet,dataLabel;

2.提取特征：提取图像的LBP特征（和原图像对应共分为16类）

**def** LBP\_doc(docMat):  
 hist\_doc=np.zeros((len(docMat),256));  
 *#np.save("LBPData.npy",hist\_doc);* **for** i **in** range(len(docMat)):  
 *#print(np.array(trainMat[i]).shape)* lbp=skft.local\_binary\_pattern(docMat[i],points,radius,**"default"**);  
 *#print(np.array(lbp).shape);* max\_bin=int(lbp.max()+1);  
 hist\_doc[i],\_=np.histogram(lbp,normed=**True**,bins=max\_bin,range=(0,max\_bin));  
 **return** hist\_doc;

前两步与实习三对图像的处理过程并无差异。

3.KMeans聚类：

*#求v1,v2的欧式距离***def** eculid\_distance(v1,v2):  
 distance=np.sqrt(np.sum(np.power(v1-v2,2)));  
 **return** distance;

*从数据集中选取k个质点***def** randCent(dataset,k):  
 n=np.shape(dataset)[1]; *#确定数据集的维数* centers=np.mat(np.zeros((k,n)));  
 list=np.random.randint(0,len(dataset)-1,k);  
 *#print(list);* centers=dataset[list];

**return** centers;

降维：T-SNE降维方法：

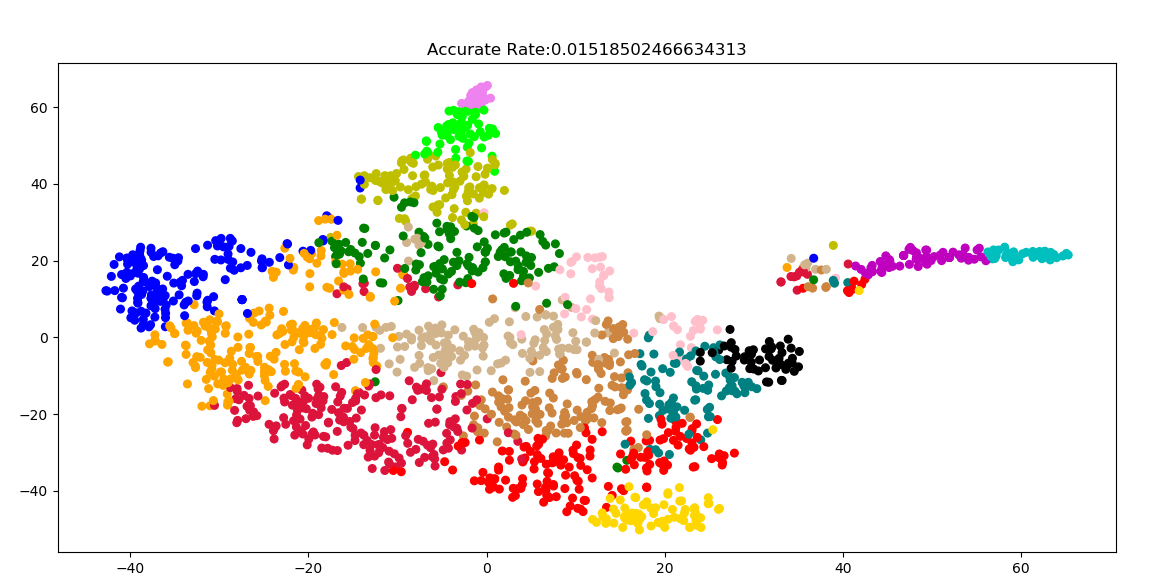
*#T-SNE降维方法***def** SneDepress(X\_mat):  
 tsne=TSNE(n\_components=2,init=**"pca"**,random\_state=0);  
 X\_tsne=tsne.fit\_transform(X\_mat);  
  
 **return** X\_tsne;

**1.普通K均值聚类算法：**

过程：

1. 随机选择k个种子点作为初始中心点/类簇中心；
2. 将每个文档指派到与其最相似的中心点所在的类簇；
3. 根据当前类簇文档重新计算类簇中心点；
4. 如果不收敛或不满足终止条件, 继续执行步骤2
5. **def** K\_means(dataSet,K):  
    n=np.shape(dataSet)[0];  
    clussResult=np.mat(np.zeros((n,2))); *#分类结果统计 mat是为了每一项都是1\*2的矩阵* centers=randCent(dataSet,K); *#随机产生k个初始质点* change=**True**;  
    **while** change:  
    change=**False**;  
    **for** i **in** range(n):  
    lowestdis=float(**"inf"**);minIndex=-1;  
    **for** j **in** range(K):  
    distance=eculid\_distance(centers[j],dataSet[i]);  
    **if** distance<lowestdis:  
    lowestdis,minIndex=distance,j;  
    *#clussResult[i]=j,distance;* **if** clussResult[i,0]!=minIndex:  
    change = **True**;  
    clussResult[i]=minIndex,lowestdis\*\*2; *#lowestdis\*\*2突出距离远的* **for** k **in** range(K):  
    points=dataSet[np.nonzero(clussResult[:,0]==k)[0]];  
    centers[k]=np.mean(points,axis=0); *#axis=0按列来取平均值  
     
    #返回每个簇的中心点和每个点所属的类别* **return** centers,clussResult;

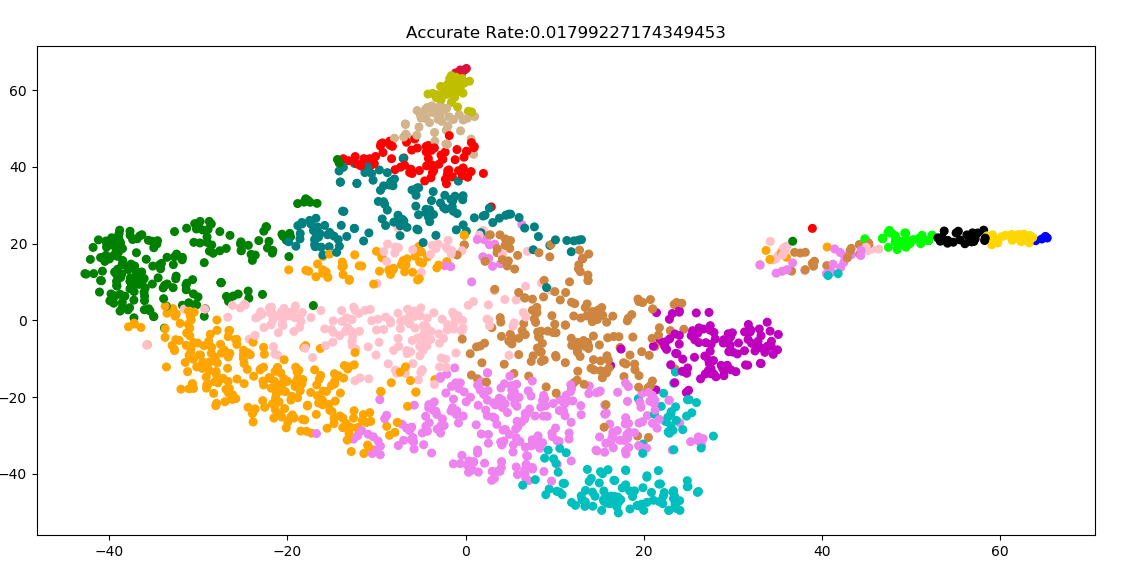
**结果：**



2.二分—K均值聚类算法：

过程：

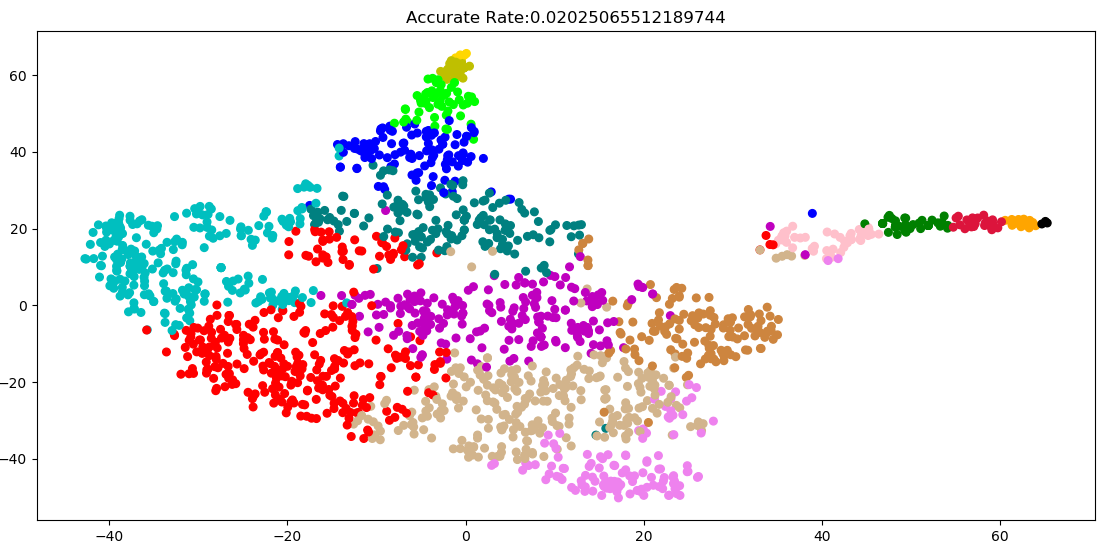
1. 将所有点看成一个簇
2. 当簇数小于K时，对每一个簇进行2分类（用方法一中的Kmeans），并计算总误差，找出使整个数据集的总误差最小的结果，对该簇进行2分类，使簇数加一。
3. 直到使簇数等于K
4. *#2分K-均值聚类***def** binary\_KMeans(dataSet,K):  
    *#初始为一个簇* n = np.shape(dataSet)[0];  
    clussResult = np.mat(np.zeros((n, 2))); *# 分类结果统计 mat是为了每一项都是1\*2的矩阵* center0=np.mean(dataSet,axis=0).tolist();  
    centers=[center0];  
    **for** i **in** range(n):  
    clussResult[i,1]=(eculid\_distance(center0,dataSet[i])\*\*2);  
     
    **while** len(centers)<K:  
    lowestSSE=float(**"inf"**);  
    print(len(centers));  
    length=len(centers)  
    *#对每一个簇进行2分类，求出SSE* **for** i **in** range(len(centers)):  
    points=dataSet[np.nonzero(clussResult[:,0]==i)[0]];  
    **if** len(points)<=1:**continue**;  
    binary\_centers,binary\_clussResukt=K\_means(points,2);  
    binarySSE=np.sum(binary\_clussResukt[:,1]);  
    Non\_binarySSE=np.sum(clussResult[np.nonzero(clussResult[:,0]!=i)[0],1]);  
     
    *#获取使整个数据集SSE最小的簇* **if** (binarySSE+Non\_binarySSE)<lowestSSE:  
    bestIndex=i;  
    lowestSSE=binarySSE+Non\_binarySSE;  
    best\_bicenters=binary\_centers;  
    best\_clussResult=binary\_clussResukt;  
     
    *#2分类后分成best\_clussResult[0]对应为0和1;  
    #先计算==1，再计算==0，防止0，1转化为1，2时，0->1,则转化的1类和原来的1无法区分* best\_clussResult[np.nonzero(best\_clussResult[:,0]==1)[0],0]=len(centers);  
    *#print(best\_clussResult[np.nonzero(best\_clussResult[:,0]==0)[0]]);  
    #print(len(best\_clussResult[np.nonzero(best\_clussResult[:,0]==bestIndex)[0]]))* best\_clussResult[np.nonzero(best\_clussResult[:,0]==0)[0],0]=bestIndex;  
    *#print(len(best\_clussResult[np.nonzero(best\_clussResult[:, 0] == len(centers))[0]]))* centers[bestIndex]=best\_bicenters[0];  
    centers.append(best\_bicenters[1]);  
    clussResult[np.nonzero(clussResult[:,0]==bestIndex)[0],:]=best\_clussResult;  
    print(**"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"**)  
    **for** i **in** range(len(centers)):  
    print(len(clussResult[np.nonzero(clussResult[:,0]==i)[0]]))  
     
    print(**"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"**);  
    **return** centers,clussResult;



3.SKlearn函数：

**def** Kmeas\_SKLearn(dataset,K):  
 estimator = KMeans(n\_clusters=K,random\_state=28)  
 estimator.fit(dataset);

label=estimator.labels\_;  
 centers=estimator.cluster\_centers\_;  
 **return** label,centers;

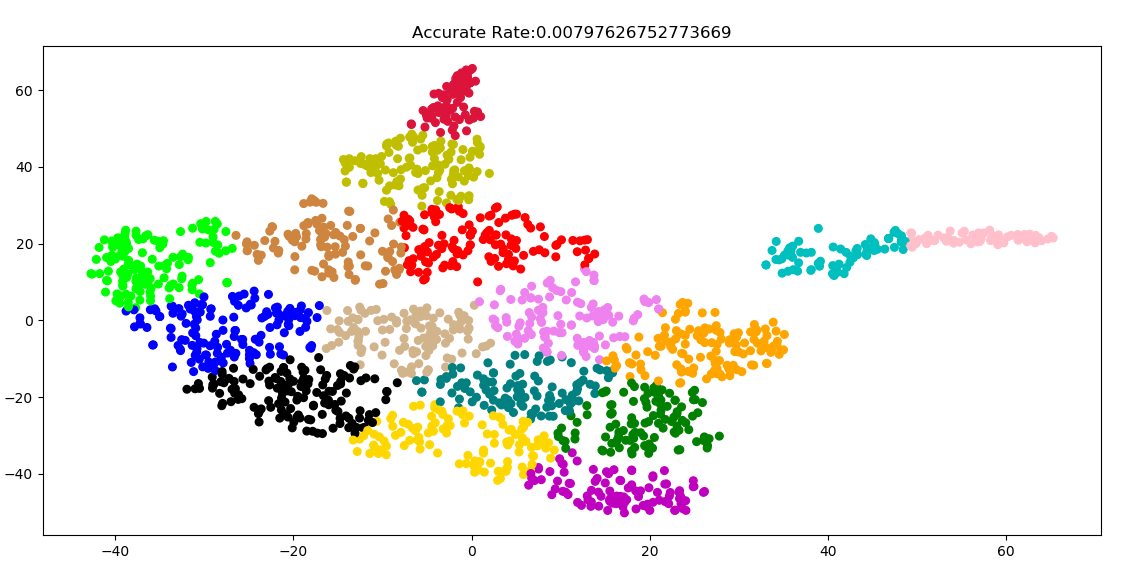


测试评估：采用

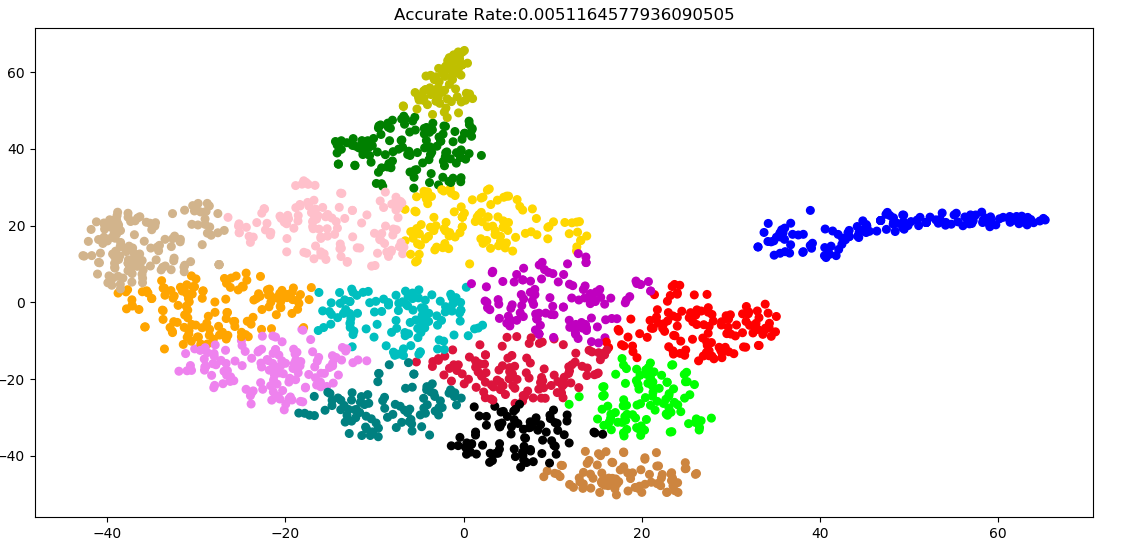
以上均是采用先聚类再降维的方式（降维目的是将图像提取特征后的数据显示出来），

下面拿SKlearn库函数先降维再聚类后的结果：

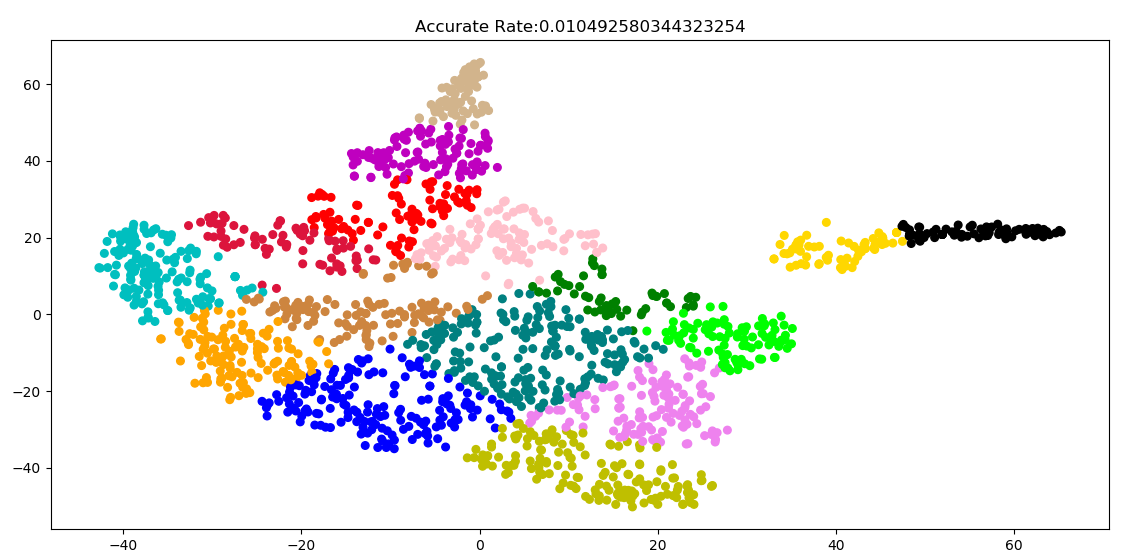
分类效果明显，但准确率下降：



KMeans:



二分K-均值聚类：



#### 【改进设想】

（或查阅文献提出改进设想；或考虑与第三方机器学习库分类结果的比较；或与采用其他分类算法实现的结果比较等）。

1.可以尝试通过层次凝聚式聚类（HAC的方法进行聚类）自底向上，看是否可以提高准确率）

## 三、本次实习小结

通过本次实习，对于非监督学习有了一定的了解，同时对于K-均值聚类也有所尝试，普通的K-均值聚类虽然简单，容易实现，但是准确性差，时间复杂度高。而2分K-均值基于普通K-均值聚类，采用自顶而下的方法，实现较为繁琐，但时间复杂度底，准确率较高，最后是SKLearn库，时间复杂度低，准确率高。而在可视化时先聚类再降维和先降维再聚类效果也极为不同，前者准确性高而分类效果不明显，后者准确性低而分类效果明显。在前面的结果中已有所证明。总的来说，相比于监督学习的方法，非监督学习算法准确性相对较低。

=======================================================

我的联系方式：

电话：13349830890

邮箱：liuyy@cug.edu.cn