机器学习学习报告

周珊琳

上海电力大学
上海, 2019-07-11, 中国

第一节　算法概述

算法名称：K-means

算法原理：k均值聚类算法（k-means clustering algorithm）是一种迭代求解的聚类分析算法，其步骤是随机选取K个对象作为初始的聚类中心，然后计算每个对象与各个种子聚类中心之间的距离，把每个对象分配给距离它最近的聚类中心。聚类中心以及分配给它们的对象就代表一个聚类。每分配一个样本，聚类的聚类中心会根据聚类中现有的对象被重新计算。这个过程将不断重复直到满足某个终止条件。终止条件可以是没有（或最小数目）对象被重新分配给不同的聚类，没有（或最小数目）聚类中心再发生变化，误差平方和局部最小。

欧式距离：

$$d = \sqrt{\sum_{k=1}^{n}(x_{1k}-x_{2k})^{2}} \tag{1.1}$$ [1]

误差平方和：

$$SSE = \sum_{i=1}^{k}\sum_{x\epsilon C_i} d(C_i,x)^{2} \tag{1.2}$$ [2]

[1] 其中有a(x11,x12,···,x1n)与 b(x21,x22,···,x2n)为两个n维度向量

[2] d()表示两个对象之间的距离，通常为欧式距离
对于相同的k值，更小的SSE说明簇中的对象越集中；对于不同的k值，越大的k值应该对应越小的SSE

第二节 算法设计

2.1 算法流程

Algorithm 1 k-means 算法

```
输入：数据集D,划分簇的个数k;
输出：k个簇的集合;
从数据集D中任意选择k个对象作为初始簇中心
repeat
    for 数据集D中每个对象P do
        计算对象P到k个簇中心的距离
        将P指派到与其最近（距离最短）的簇
    end for
    计算每个簇中对象的均值，作为新的簇的中心
    until k个簇的簇中心不再发生变化
```

2.2 核心代码

源代码 1:

```python

# -*- coding: utf-8 -*-
"""
Created on Sun Jul 7 16:38:52 2019

@author: zsl
"""
from numpy import *
from sklearn.datasets import load_iris,load_wine
from sklearn import preprocessing
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

def calDistance(a,b):
    return sqrt(sum(power(a-b,2)))

def randCent(dataSet, k):
    n = shape(dataSet)[1]
    centroids =mat(zeros((k,n)))
    for j in range(n):
        minJ =min(dataSet[:,j])
        rangeJ =float(max(dataSet[:,j]) -minJ)
        centroids[:,j] =mat(minJ +rangeJ *random.rand(k,1))
    return centroids

def kMeans(dataSet, k, distMeas=calDistance, createCent=randCent):
    m = shape(dataSet)[0]
    clusterAssment =mat(zeros((m,2)))
    SSE=[]
    centroids =createCent(dataSet, k)
    clusterChanged =True
    while clusterChanged:
        clusterChanged =False
        for i in range(m):
            minDist =inf
            minIndex =-1
            for j in range(k):
                distJI =distMeas(centroids[j,:],dataSet[i,:])
                if distJI <minDist:
                    minDist =distJI
                    minIndex =j
            if clusterAssment[i,0] !=minIndex:
                clusterChanged =True
            clusterAssment[i,:] =minIndex,minDist**2
        print(centroids)
        SSE.append(sum(clusterAssment[:,1]))
        for cent in range(k):
            ptsInClust =dataSet[nonzero(clusterAssment[:,0].A==cent)[0]]
            centroids[cent,:] =mean(ptsInClust, axis=0)
    return centroids, clusterAssment,SSE

def process_data(data):
    min_max_scaler =preprocessing.MinMaxScaler()
    return min_max_scaler.fit_transform(dataset)

def plot_scatter(feature_name,dataset,target,mycentroids,myclusterAssment):
    x=[];y=[]
    n=shape(mycentroids)[0]
    for cent in range(n):
        x.append(dataset[nonzero(myclusterAssment[:,0].A==cent)[0]])

    for cent in range(n):
        y.append(dataset[nonzero(target==cent)[0]])

    fig =plt.figure(figsize=(10, 5), facecolor=’w’)
    ax =fig.add_subplot(121)
    ax.scatter(y[0][:,0], y[0][:,1], c=’r’, s=30, marker=’o’, edgecolors=’k’)
    ax.scatter(y[1][:,0], y[1][:,1], c=’g’, s=30, marker=’^’, edgecolors=’k’)
    ax.scatter(y[2][:,0], y[2][:,1], c=’#6060FF’, s=30, marker=’s’, edgecolors=’k’)

    ax.set_xlabel(feature_name[0],fontsize=15)
    ax.set_ylabel(feature_name[1],fontsize=15)
    ax.set_title(u’origin’, fontsize=15)
    ax =fig.add_subplot(122)
    ax.scatter(x[0][:,0], x[0][:,1], c=’r’, s=30, marker=’o’, edgecolors=’k’)
    ax.scatter(x[1][:,0], x[1][:,1], c=’g’, s=30, marker=’^’, edgecolors=’k’)
    ax.scatter(x[2][:,0], x[2][:,1], c=’#6060FF’, s=30, marker=’s’, edgecolors=’k’)
    ax.set_xlabel(feature_name[0],fontsize=15)
    ax.set_ylabel(feature_name[1],fontsize=15)
    ax.set_title(u’K-means’, fontsize=15)
    plt.tight_layout()
    plt.show()

#iris
data=load_iris()
dataset =data.data
target =data.target
dataset=process_data(dataset)
datMat=mat(dataset)
mycentroids,myclusterAssment,sse1=kMeans(datMat,3)
plt.plot(sse1)
#7.13
feature_name =[’Calyx length’,’Calyx width’]
plot_scatter(feature_name,dataset,target,mycentroids,myclusterAssment)


##wine
#data=load_wine()
#dataset = data.data
#target = data.target
#dataset=process_data(dataset)
#df = pd.DataFrame(dataset)
#dfcorr=df.corr()
##plot heatmap
#plt.subplots(figsize=(13, 13))
#sns.heatmap(dfcorr, annot=True, vmax=1, square=True,
#            cmap="Blues")
#plt.savefig(’d:/heatmap.png’)
#plt.show()
##01245
#df = df[[0,1,2,4,5]]
#datMat=mat(df.values)
#mycentroids,myclusterAssment,sse2=kMeans(datMat,3)
##16.90
#plt.plot(sse2)
#feature_name =[’alcohol’,’malic acid’]
#plot_scatter(feature_name,dataset,target,mycentroids,myclusterAssment)

##car
def loadDataSet(fileName):
    dataMat =[]; labelMat =[]
    fr = open(fileName)
    for line in fr.readlines():
        lineArr =line.strip().split(’,’)
        # print(lineArr)
        if lineArr[0] ==’vhigh’:
            lineArr[0] =1
        if lineArr[0] ==’high’:
            lineArr[0] =2
        if lineArr[0] ==’med’:
            lineArr[0] =3
        if lineArr[0] ==’low’:
            lineArr[0] =4
        if lineArr[1] ==’vhigh’:
            lineArr[1] =1
        if lineArr[1] ==’high’:
            lineArr[1] =2
        if lineArr[1] ==’med’:
            lineArr[1] =3
        if lineArr[1] ==’low’:
            lineArr[1] =4
        if lineArr[2]==’2’:
            lineArr[2]=1
        if lineArr[2]==’3’:
            lineArr[2]=2
        if lineArr[2]==’4’:
            lineArr[2]=3
        if lineArr[2]==’5more’:
            lineArr[2]=4
        if lineArr[3]==’2’:
            lineArr[3]=1
        if lineArr[3]==’4’:
            lineArr[3]=2
        if lineArr[3]==’more’:
            lineArr[3]=3
        if lineArr[4]==’small’:
            lineArr[4]=1
        if lineArr[4]==’med’:
            lineArr[4]=2
        if lineArr[4]==’big’:
            lineArr[4]=3
        if lineArr[5]==’low’:
            lineArr[5]=1
        if lineArr[5]==’med’:
            lineArr[5]=2
        if lineArr[5]==’high’:
            lineArr[5]=3
        dataMat.append([float(lineArr[0]),float(lineArr[1]), float(lineArr[2]),
                        float(lineArr[3]),float(lineArr[4]),
                        float(lineArr[5])])


        if lineArr[6] ==’unacc’:
            lineArr[6] =0
        elif lineArr[6] ==’acc’:
            lineArr[6] =1
        elif lineArr[6] ==’good’:
            lineArr[6] =2
        else:
            lineArr[6] =3

        labelMat.append(float(lineArr[6]))

    return dataMat,labelMat

#dataset,target = loadDataSet(’car.data’)
#target=int32(target)
#dataset = array(dataset)
```

```
#datMat=mat(dataset)
#mycentroids,myclusterAssment,sse3=kMeans(datMat,4)
##5937.575268319763
#plt.plot(sse3)
#x=[];y=[]
#n=shape(mycentroids)[0]
#for cent in range(n):
#    x.append(dataset[nonzero(myclusterAssment[:,0].A==cent)[0]])
#
#for cent in range(n):
#    y.append(dataset[nonzero(target==cent)[0]])
#
#fig = plt.figure(figsize=(10, 5), facecolor=’w’)
#ax = fig.add_subplot(121)
#ax.scatter(y[0][:,0], y[0][:,1], c=’r’, s=30, marker=’o’, edgecolors=’k’)
#ax.scatter(y[1][:,0], y[1][:,1], c=’g’, s=30, marker=’^’, edgecolors=’k’)
#ax.scatter(y[2][:,0], y[2][:,1], c=’#6060FF’, s=30, marker=’s’, edgecolors=’k’)
#ax.scatter(y[3][:,0], y[3][:,1], c=’gold’, s=30, marker=’s’, edgecolors=’k’)
#feature_name=[’buying’,’maint’]
#ax.set_xlabel(feature_name[0],fontsize=15)
#ax.set_ylabel(feature_name[1],fontsize=15)
#ax.set_title(u’origin’, fontsize=15)
#ax = fig.add_subplot(122)
#ax.scatter(x[0][:,0], x[0][:,1], c=’r’, s=30, marker=’o’, edgecolors=’k’)
#ax.scatter(x[1][:,0], x[1][:,1], c=’g’, s=30, marker=’^’, edgecolors=’k’)
#ax.scatter(x[2][:,0], x[2][:,1], c=’#6060FF’, s=30, marker=’s’, edgecolors=’k’)
#ax.scatter(x[3][:,0], x[3][:,1], c=’gold’, s=30, marker=’s’, edgecolors=’k’)
#ax.set_xlabel(feature_name[0],fontsize=15)
#ax.set_ylabel(feature_name[1],fontsize=15)
#ax.set_title(u’K-means’, fontsize=15)
#plt.tight_layout()
#plt.show()


ln1, =plt.plot(sse1, color =’red’, linewidth =2.0, linestyle =’--’)
ln2, =plt.plot(sse2, color =’blue’, linewidth =2.0, linestyle =’--’)
ln3, =plt.plot(sse3, color =’pink’, linewidth =2.0, linestyle =’--’)
plt.legend(handles=[ln1,ln2,ln3], labels=[’iris’, ’wine’,’car’],
    loc=’uper right’)
```

第三节　选用数据

iris行数：150 列数：5
列属性及取值:
1)萼片长度cm，数值型
2)萼片宽度cm，数值型
3)花瓣长度cm，数值型
4)花瓣宽度cm数值型
类别：
Iris Setosa
Iris Versicolour
Iris Virginica
car，行数：1728，列数：6
列属性及取值：
1)buying: vhigh, high, med, low.
2)maint: vhigh, high, med, low.
3)doors: 2, 3, 4, 5more.
4)persons: 2, 4, more.
5)lugboot: small, med, big.
6)safety: low, med, high.
类别：
unacc, acc, good, vgood
wine，行数：178，列数：13
属性：
1) Alcohol
2) Malic acid
3) Ash
4) Alcalinity of ash
5) Magnesium
6) Total phenols
7) Flavanoids
8) Nonflavanoid phenols
9) Proanthocyanins
10) Color intensity
11) Hue
12) OD280/OD315 of diluted wines
13) Proline
类别：
Alcohol 1，2，3

第四节　实验结果展示

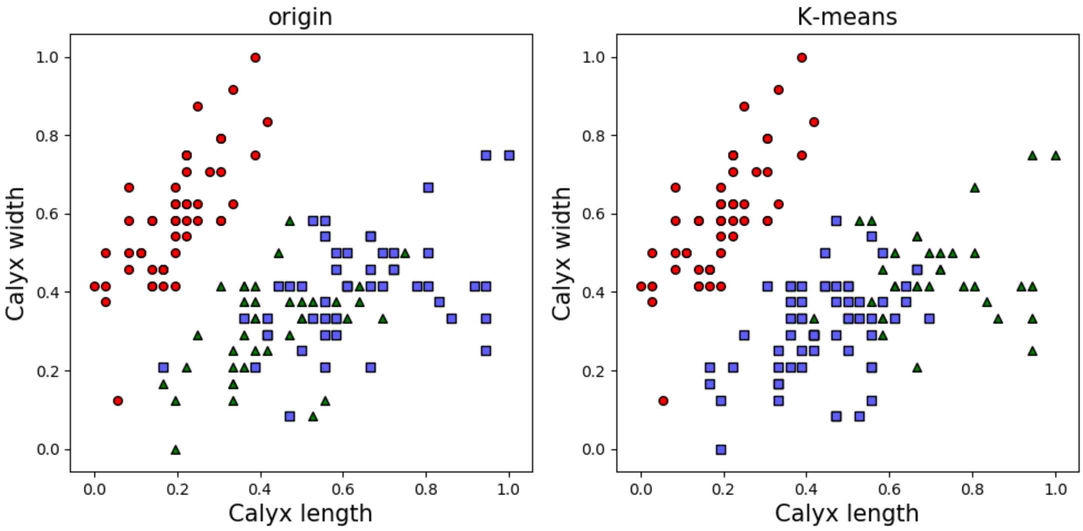

图 1: iris聚类对比图

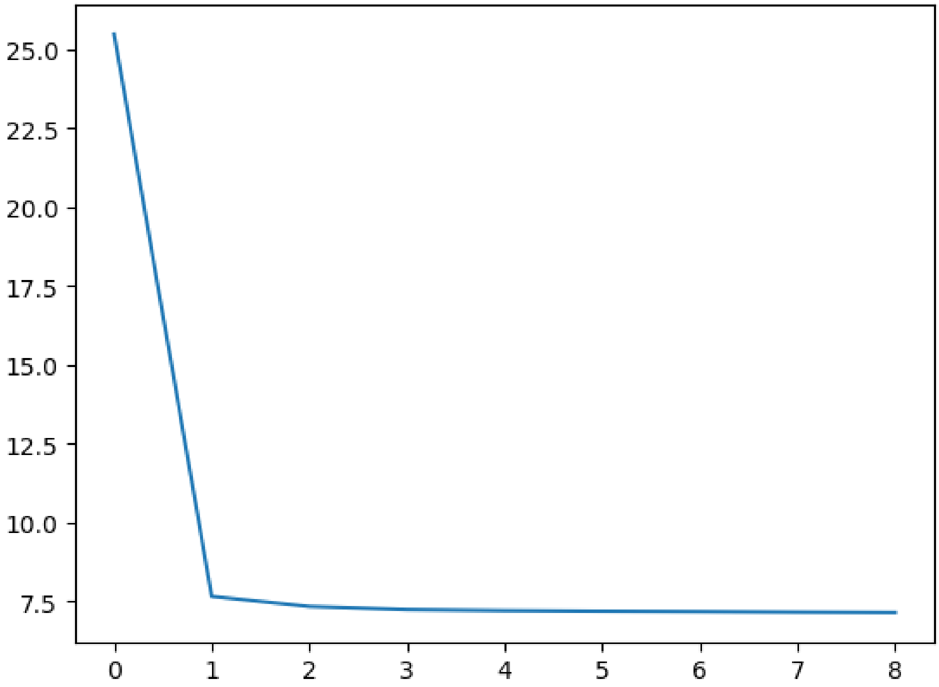

图 2: iris的SSE变化曲线

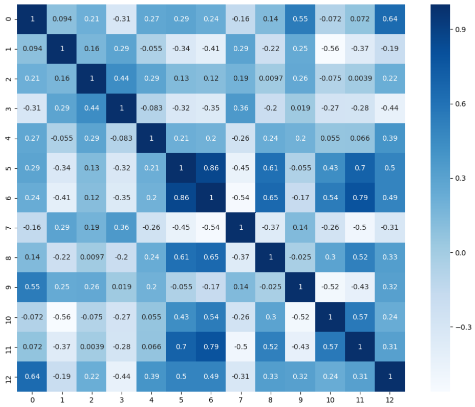

图 3: wine变量相关性热力图

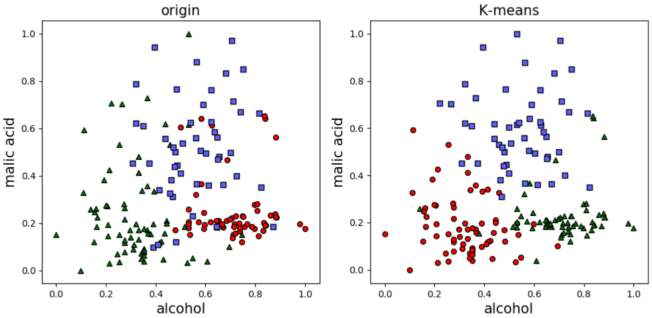

图 4: wine聚类对比图

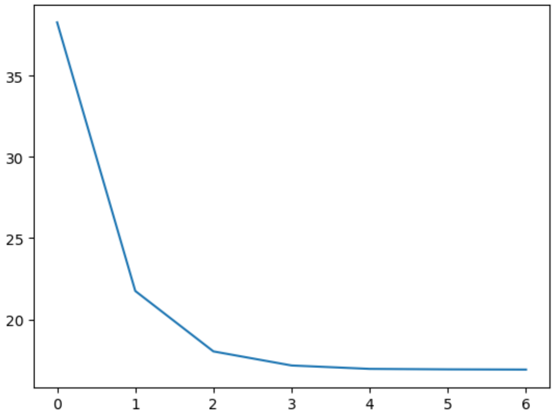

图 5: wine的SSE变化曲线

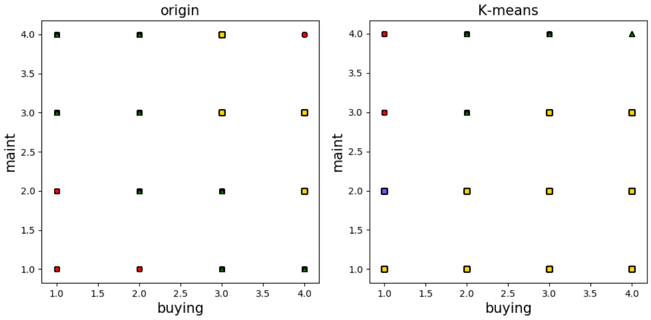

图 6: car聚类对比图

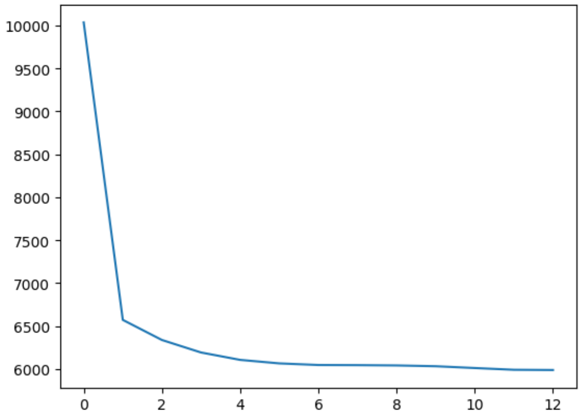

图 7: car的SSE变化曲线

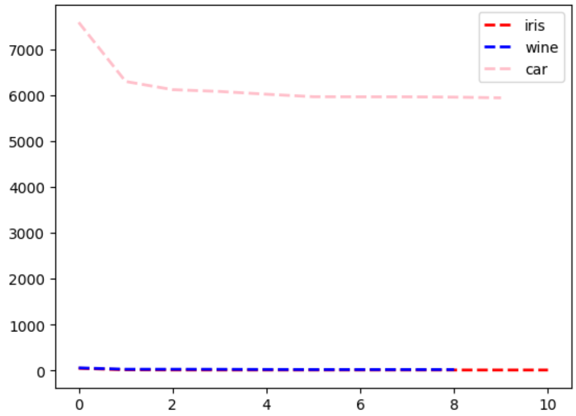

图 8: SSE变化曲线对比图

第五节　实验分析和比较

k-means需要数值型数据，对于car这个数据，不适合用这个算法。这次，在使用wine之前，对wine进行了相关性分析，并对iris和wine数据进行了最大最小归一化处理，这样处理之后的数据，再使用聚类算法，效果明显更好。