



西安电子科技大学
XIDIAN UNIVERSITY

人工智能学院

智能数据挖掘课程作业报告

kmeans 算法练习

姓名：杨文韬

学号：18020100245

班级：1920012

2022 年 04 月 28 日

目录

1	问题描述	1
2	原理分析	1
2.1	waveform 数据集	1
2.2	k means 算法	1
2.3	图像分割	2
3	实验过程	2
3.1	waveform 数据集聚类	2
3.2	无噪声图像分割	2
4	附录	4

k means 算法练习

1 问题描述

编程实现 k means 算法针对 UCI 的 waveform 数据集中每类数据取 100 个；对一幅无噪图像进行分割；

2 原理分析

2.1 waveform 数据集

waveform 数据集是订货书的波形域，共有 5000 个样本，3 种标签，21 个属性，每个属性是 0 到 6 之间的连续值，没有缺失属性值，每个类别占 33%，近邻算法准确率在 78% 左右。

2.2 k means 算法

k 均值聚类的算法是一个迭代的过程，每次迭代包括两个步骤。首先选择 k 个类的中心，将样本逐个指派到与其最近的中心的类中，得到一个聚类结果；然后更新每个类的样本的均值，作为类的新的中心；重复以上步骤，直到收敛为止。 k means 均值聚法伪代码如算法 1 所示。

Algorithm 1 K-means Clustering

Input: A set X consisting of n samples.

Output: Clustering C^* of sample sets.

- 1: Select k sample points as initial clustering centroid randomly;
 - 2: $t \leftarrow 0$;
 - 3: **while** *uniterated convergence and not meet stop conditions* **do**
 - 4: Calculate the distance of each sample to the centroid of each clustering;
 - 5: Assign each sample to the clustering of its nearest centroid as clustering results $C^{(t)}$;
 - 6: Calculate the mean value of the samples in each clustering as new centroid;
 - 7: $t \leftarrow t + 1$;
 - 8: **end while**
 - 9: $C^* \leftarrow C^{(t)}$;
-

2.3 图像分割

在计算机视觉领域，图像分割（segmentation）指的是将数字图像细分为多个图像子区域（像素的集合）（也被称作超像素）的过程。图像分割的目的是简化或改变图像的表示形式，使得图像更容易理解和分析。图像分割通常用于定位图像中的物体和边界（线，曲线等）。更精确的，图像分割是对图像中的每个像素加标签的一个过程，这一过程使得具有相同标签的像素具有某种共同视觉特性。

彩色图像中的每一个像素是三维空间中的一个点，三维对应红、绿、蓝三原色的强度，基于 k means 聚类算法的图像分割以图像的像素为数据点，按照指定的簇数进行聚类，然后将每个像素点以其对应的聚类中心替代，重构该图像，不同的聚类簇数呈现不同的色彩特征。

3 实验过程

3.1 waveform 数据集聚类

导入数据集后发现 0 类 1657 个，1 类 1647 个，2 类 1696 个，每类数据取前 100 个拼接并打乱。然后实现 k means 步骤如下

Step 1. 随机生成 k 个中心。

Step 2. 依次求解当前（第 i 个）元素对 k 个中心的最小欧式距离，得到距当前元素欧式距离最小的中心。

Step 3. 如果当前（第 i 个）元素的与 k 个中心的最小欧式距离所对应的中心编号遇上一次记录不一致，则（第 i 个）元素所属簇发生改变，更新所属簇。

Step 4. 重新计算 k 个中心簇的各属性值的平均值，得到新的 k 个中心。

Step 5. 若 **Step 2~4** 的运行过程中，至少 1 个元素的所属簇发生改变，则不断重复运行 **Step 2~5** 过程，直至没有元素所属中心发生改变视为聚类成功，同时限定最大迭代次数为 100，聚类成功转 **Step 6**。

Step 6. 结束迭代，返回 k 个簇的中心值 centroids, 以及聚类划分 clusterAssment。

8 次迭代后算法停止，聚类中心和聚类划分结果见附录。

3.2 无噪声图像分割

使用 Python 库函数 `sklearn.cluster.KMeans` 分别取 $k = 4$ 和 $k = 8$ 进行图像分割得到的结果如图 1 所示。

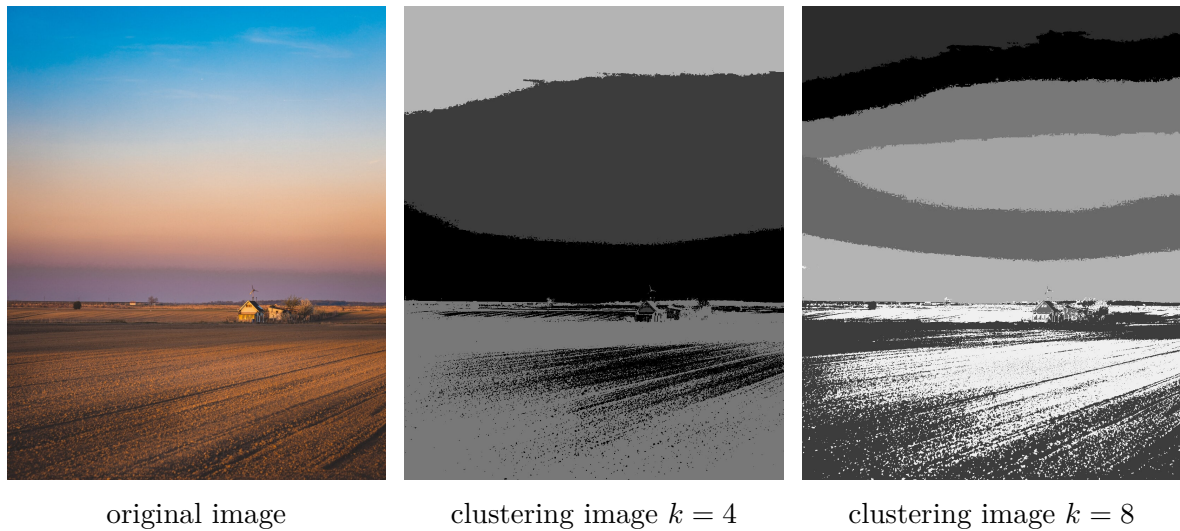


图 1: 图像分割结果

利用 cv2 库函数分别取 $k = 2, 4, 8, 16, 64$ 得到的彩色图像分割结果如图 2 所示。

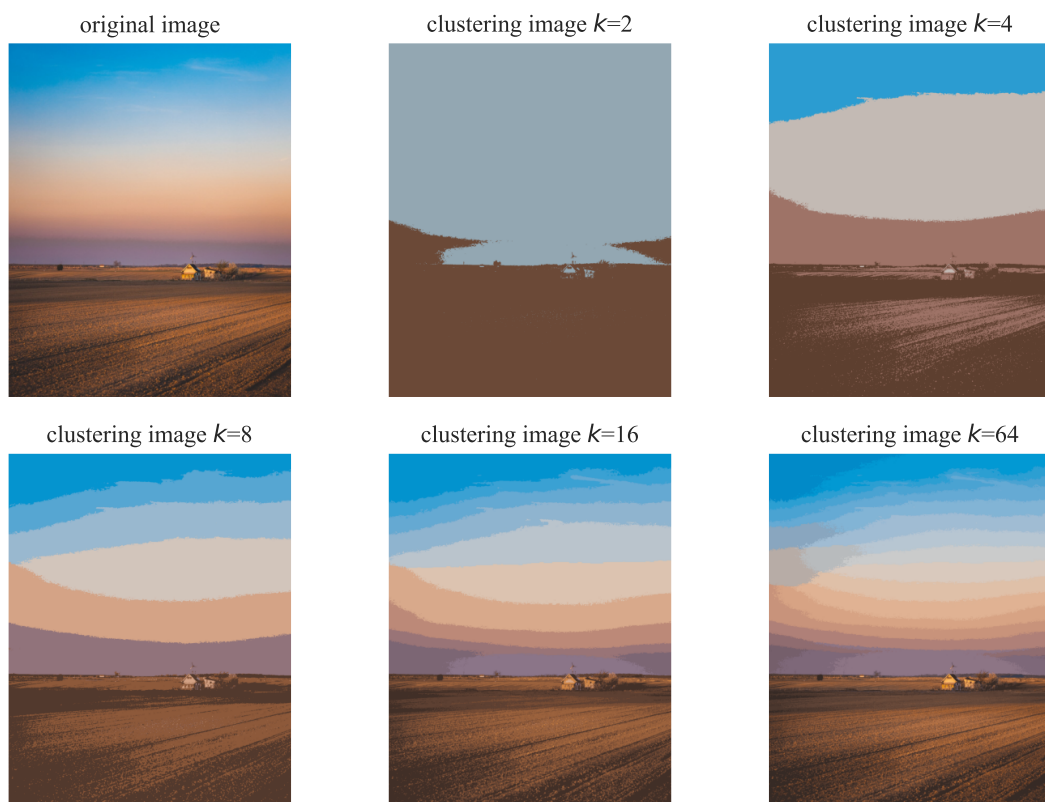


图 2: cv2 彩色图像分割结果

可以得出结论，在一定的迭代次数限制下，设置的聚类数越多，图像分割越明显，细节越丰富。并且程序的运行时间会随着聚类数 k 、最大迭代次数的增加而增加。

4 附录

```
In [1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import random
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline

In [2]: sns.set_style("darkgrid", {"grid.color": ".6", "grid.linestyle": ":"})
sns.set_theme(font='Times New Roman', font_scale=1.2)
plt.rc("figure", autolayout=True)
# plot settings
plt.rcParams["figure.figsize"] = (10, 7)
plt.rcParams['figure.dpi'] = 150
plt.rcParams['axes.grid'] = False
# Chinese support
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
```

4.1 Waveform 数据集聚类

```
In [3]: df = pd.read_csv('./waveform.data', header=None)
print(df[21].value_counts())
df.head()
```

```
2    1696
0    1657
1    1647
Name: 21, dtype: int64
```

```
Out[3]:
```

	0	1	2	3	4	5	6	7	8	9	...	12	\
0	-1.23	-1.56	-1.75	-0.28	0.60	2.22	0.85	0.21	-0.20	0.89	...	2.89	
1	-0.69	2.43	0.61	2.08	2.30	3.25	5.52	4.55	2.97	2.22	...	1.24	
2	-0.12	-0.94	1.29	2.59	2.42	3.55	4.94	3.25	1.90	2.07	...	2.50	
3	0.86	0.29	2.19	-0.02	1.13	2.51	2.37	5.45	5.45	4.84	...	2.58	

```
4  1.16  0.37  0.40 -0.59  2.66  1.00  2.69  4.06  5.34  3.53  ...  4.30
```

```

      13    14    15    16    17    18    19    20  21
0  7.75  4.59  3.15  5.12  3.32  1.20  0.24 -0.56  2
1  1.89  1.88 -1.34  0.83  1.41  1.78  0.60  2.42  1
2  0.12  1.41  2.78  0.64  0.62 -0.01 -0.79 -0.12  0
3  1.40  1.24  1.41  1.07 -1.43  2.84 -1.18  1.12  1
4  1.84  1.73  0.21 -0.18  0.13 -0.21 -0.80 -0.68  1

```

[5 rows x 22 columns]

```
In [4]: def cacldist(vec1, vec2):
        distance = np.sqrt(np.sum(np.square(vec2 - vec1)))
        return distance
```

```
In [5]: def initCentroids(dataSet, k):
        numSamples, dim = dataSet.shape
        centroids = np.zeros((k, dim))
        for i in range(k):
            index = int(random.uniform(0, numSamples))
            centroids[i, :] = dataSet[index, :]
        return centroids
```

```
In [6]: def kmeans(dataSet, k):
        numSamples = dataSet.shape[0]
        clusterAssment = np.mat(np.zeros((numSamples, 2)))
        flag = True
        centroids = initCentroids(dataSet, k)
        iternum = 0
        while flag and iternum <= 100:
            flag = False
            iternum = iternum + 1
            for i in range(numSamples):
                minDist = 1e5
                minIndex = 0
                for j in range(k):
                    distance = cacldist(centroids[j, :], dataSet[i, :])
                    if distance < minDist:
```

```

        minDist = distance
        minIndex = j
        if clusterAssment[i, 0] != minIndex:
            flag = True
            clusterAssment[i, :] = minIndex, minDist ** 2
''' # 由于前两维展示效果不好, 注释掉动态绘图部分
plt.ion()
plt.title('Result after ' + str(iternum) + ' iterations')
mark1 = ['Dr', 'Db', 'Dg', 'Dk', '~b', '+b', 'sb', 'db', '<b', 'pb']
mark2 = ['or', 'ob', 'og', 'ok', '~r', '+r', 'sr', 'dr', '<r', 'pr']
'''
for i in range(k):
    pointsInCluster = dataSet[np.nonzero(clusterAssment[:, 0].A == i)[0]]
    centroids[i, :] = np.mean(pointsInCluster, axis=0)
'''

    plt.plot(centroids[i, 0], centroids[i, 1], mark1[i], markersize=8)
    for j in range(pointsInCluster.shape[0]):
        point = pointsInCluster[j, :]
        plt.plot(pointsInCluster[j, 0], pointsInCluster[j, 1], mark2[i])
    plt.show()
    plt.pause(0.1)
plt.close()
plt.ioff()
'''

print("经过 %d 次迭代后算法停止" % iternum)
return centroids, clusterAssment

```

```

In [7]: df0 = df[df[21]==0].head(100) # 取 0 类数据前 100 个
        df1 = df[df[21]==1].head(100) # 取 1 类数据前 100 个
        df2 = df[df[21]==2].head(100) # 取 2 类数据前 100 个
        data = pd.concat([df0, df1, df2]) # 拼接
        data = data.sort_index() # 按原文件的索引从小到大排序
        data = data.reset_index(drop=True) # 重新建立索引

```

```

In [8]: data = np.mat(data)
        k = 3
        centroids, clusterAssment = kmeans(data, k)

```


经过 8 次迭代后算法停止

In [9]: centroids # 聚类中心

```
Out[9]: array([[ 0.04077586,  0.13784483,  0.29655172,  0.43275862,  0.39818966,
                 0.78517241,  1.05698276,  1.18767241,  1.4637931 ,  1.95103448,
                 2.73172414,  3.24560345,  3.54689655,  3.99258621,  4.59913793,
                 3.65137931,  2.64318966,  2.12465517,  1.39913793,  0.8212069 ,
                -0.06732759,  1.20689655],
               [-0.01473684,  0.42644737,  0.13407895,  0.48473684,  0.57263158,
                 1.33039474,  2.46039474,  3.12013158,  3.97434211,  4.45605263,
                 5.41710526,  4.5475      ,  3.17368421,  2.76960526,  2.00118421,
                 1.14184211,  0.15144737,  0.24618421,  0.16736842, -0.01868421,
                 0.23394737,  1.39473684],
               [ 0.01666667,  0.70657407,  1.4362037 ,  2.24666667,  3.06342593,
                 3.76074074,  4.8137037 ,  3.97064815,  3.48398148,  3.1287963 ,
                 2.52601852,  1.72138889,  1.23694444,  0.95111111,  0.96675926,
                 0.73787037,  0.60462963,  0.39648148,  0.085      ,  0.16037037,
                 0.05851852,  0.5          ]])
```

In [10]: # clusterAssment # 聚类划分, dim1 类, dim2 距离, 输出太长不显示

4.2 无噪图像分割

In [11]: import imageio

from sklearn.cluster import KMeans

```
def image_cluster(image_name, save_name, k_cluster=3):
```

```
    image = imageio.imread(image_name)
```

```
    # (w, h, 3) -> (w * h, 3)
```

```
    image2matrix = []
```

```
    for i in range(image.shape[0]):
```

```
        for j in range(image.shape[1]):
```

```
            r_v, g_v, b_v = image[i, j]
```

```
            image2matrix.append([r_v/255.0, g_v/255.0, b_v/255.0])
```

```
    data = np.mat(image2matrix)
```

```
cls = KMeans(n_clusters=k_cluster).fit_predict(data)
cls = cls.reshape(image.shape[0], image.shape[1])

container = np.zeros(shape=(image.shape[0], image.shape[1]))

for i in range(image.shape[0]):
    for j in range(image.shape[1]):
        container[i, j] = cls[i, j] * 60

container = container.astype(np.uint8)
imageio.imsave(save_name, container)

return True
```

```
In [12]: image_cluster("image-1080x1350.jpg", "cluster4.jpg", 4)
```

```
Out[12]: True
```

```
In [13]: image_cluster("image-1080x1350.jpg", "cluster8.jpg", 8)
```

```
Out[13]: True
```

```
In [14]: import cv2
```

```
img = cv2.imread('image-1080x1350.jpg')

data = img.reshape((-1,3))
data = np.float32(data)

criteria = (cv2.TERM_CRITERIA_EPS +
            cv2.TERM_CRITERIA_MAX_ITER, 10, 1.0)

flags = cv2.KMEANS_RANDOM_CENTERS

compactness, labels2, centers2 = cv2.kmeans(data, 2, None,
                                            criteria, 10, flags)
compactness, labels4, centers4 = cv2.kmeans(data, 4, None,
                                            criteria, 10, flags)
```

```
compactness, labels8, centers8 = cv2.kmeans(data, 8, None,
                                             criteria, 10, flags)

compactness, labels16, centers16 = cv2.kmeans(data, 16, None,
                                              criteria, 10, flags)

compactness, labels64, centers64 = cv2.kmeans(data, 64, None,
                                              criteria, 10, flags)


centers2 = np.uint8(centers2)
res = centers2[labels2.flatten()]
dst2 = res.reshape((img.shape))


centers4 = np.uint8(centers4)
res = centers4[labels4.flatten()]
dst4 = res.reshape((img.shape))


centers8 = np.uint8(centers8)
res = centers8[labels8.flatten()]
dst8 = res.reshape((img.shape))


centers16 = np.uint8(centers16)
res = centers16[labels16.flatten()]
dst16 = res.reshape((img.shape))


centers64 = np.uint8(centers64)
res = centers64[labels64.flatten()]
dst64 = res.reshape((img.shape))


img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
dst2 = cv2.cvtColor(dst2, cv2.COLOR_BGR2RGB)
dst4 = cv2.cvtColor(dst4, cv2.COLOR_BGR2RGB)
dst8 = cv2.cvtColor(dst8, cv2.COLOR_BGR2RGB)
dst16 = cv2.cvtColor(dst16, cv2.COLOR_BGR2RGB)
dst64 = cv2.cvtColor(dst64, cv2.COLOR_BGR2RGB)


titles = ['original image', 'clustering image $k$=2',
          'clustering image $k$=4', 'clustering image $k$=8',
          'clustering image $k$=16', 'clustering image $k$=64']
```

```
images = [img, dst2, dst4, dst8, dst16, dst64]
for i in range(6):
    plt.subplot(2, 3, i + 1), plt.imshow(images[i], 'gray'),
    plt.title(titles[i])
    plt.xticks([]), plt.yticks([])
plt.savefig('./document/figure/figsplit.pdf')
plt.show()
```

original image



clustering image $k=2$



clustering image $k=4$



clustering image $k=8$



clustering image $k=16$



clustering image $k=64$

