

山东大学 计算机科学与技术 学院

机器学习 课程实验报告

学号：201618130133	姓名：代荣森	班级：2016. 4
实验题目：K-Means		
实验学时：4	实验日期：2018. 12. 13	
实验目的： 使用 K-means 通过减少其包含的颜色数来压缩图像。		
硬件环境： i5-6200U 8G RAM HD Graphics520		
软件环境： Visual Studio Code + Python		
实验步骤与内容： <ol style="list-style-type: none">1. 载入数据2. 设置迭代次数3. 随机取 16 种 RGB 值作为聚类4. 计算每个像素到这 16 个初始聚点的欧式距离，找到最小距离分为一类5. 计算每一类的平均值6. 将平均值作为下一次迭代的聚点7. 设置退出条件（当据点的值不再改变时）8. 重复 4, 5, 69. 读入原图10. 计算原图每个像素到 16 个迭代之后的据点的距离，找到最小距离分为一类11. 用聚点值替换该类像素 RGB 值12. 显示图片13. 保存图片		
实验结果		实验原图
		

结论分析与体会：

最近在学习 python，所以使用 python 来写了这个实验，由于一些规则用法不一样，所以有些地方需要特别注意

这次实验可能是机器学习实验中最简单的一次了，K-Means 很好理解，而且不怎么需要数学基础的支持。唯一有些不理解的是有些同学用 matlab 很快就跑完了，而我用 python 就要跑几分钟（平均迭代 40-60 次，每次迭代耗时 3-4 秒），可能是我的算法的问题，或者 python 真的很慢吧。

附录：程序源代码

K-Means.py

```
import matplotlib.image as img
import matplotlib.pyplot as plt
import numpy as np
import datetime

value = img.imread('D:\\GitHub\\Machine-Learning\\K-Means\\bird_small.tiff')
row = np.size(value,0)
col = np.size(value,1)
#u 为随机采样 16 种颜色 RGB 值作为聚类
u = np.zeros((16,3))
for i in range(16):
    for j in range(3):
        u[i,j] = np.random.randint(255)
print(u)
o_length = np.zeros(16)
label = np.zeros((row,col))
oldu = np.zeros((16,3))
starttime = datetime.datetime.now()
for iter in range(100):
    et3 = datetime.datetime.now()
    for i in range(row):
        for j in range(col):
            for k in range(16):#计算每个点到 16 个初始聚点的欧式距离,找到最小距离分为一
类
                o_length[k] = ((value[i,j,0]-u[k,0])**2+(value[i,j,1]-
u[k,1])**2+(value[i,j,2]-u[k,2])**2)**0.5
            label[i,j] = o_length.tolist().index(min(o_length))
            # print(label)
    for i in range(16):
        for j in range(3):
            oldu[i,j] = u[i,j]#保存原聚点

    for k in range(16):
        count = 0#计算每个颜色有多少个
        total = np.zeros(3)
        for i in range(row):
            for j in range(col):
```

```

        if label[i,j] == k:
            total = total + [value[i,j,0],value[i,j,1],value[i,j,2]]
            count += 1
    if count != 0:
        u[k,:] = total/count#计算每一类的平均值，将平均值作为下一次迭代的聚点

#退出条件，当 u 的差别不大时退出
len_u = 0
for i in range(16):
    for j in range(3):
        len_u = len_u + ((u[i,j] - oldu[i,j])**2)**0.5
if len_u < 1e-6:
    break
et4 = datetime.datetime.now()
print("第%s 次迭代,耗时: %s" %(iter,et4-et3))
et1 = datetime.datetime.now()
print("迭代总时间: %s" %(et1 - starttime))

lvalue = img.imread('D:\\GitHub\\Machine-Learning\\K-Means\\bird_large.tiff')
lvalue.flags.writeable = True#图像是只读模式,修改权限
lrow = np.size(lvalue,0)
lcol = np.size(lvalue,1)
lo_length = np.zeros(16)
for i in range(lrow):
    for j in range(lcol):
        for k in range(16):#计算原图每个像素到 16 个迭代之后的据点的距离，找到最小距离分
为一类
            lo_length[k] = ((lvalue[i,j,0]-u[k,0])**2+(lvalue[i,j,1]-
u[k,1])**2+(lvalue[i,j,2]-u[k,2])**2)**0.5
            lvalue[i,j,:] = u[lo_length.tolist().index(min(lo_length)),:]#用聚点值替换该
类像素 RGB 值
et2 = datetime.datetime.now()
print("替换原图耗时: %s" %(et2 - et1))
print("总耗时: %s" %(et2 - starttime))
plt.imshow(lvalue)
plt.show()
plt.imsave('bird_kmeans.tiff',lvalue)

```