

基于K-Means算法的图像分割

王铭嵩

2023 年 10 月 16 日

1 K-Means算法概述

1.1 原理

K-Means算法是一种无监督分类算法，该算法的任务是将无标签数据集聚类成 k 个簇，利用贪心策略求得近似解。具体步骤为：

1. 在样本集中随机选取 k 个样本作为初始各簇中心点 μ_i 。
2. 计算所有样本点与各个簇中心之间的距离，并将其划入最近的簇中。
3. 根据簇中已有的样本点，重新计算簇中心 $\mu_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$
4. 重复2, 3.

1.2 优化

K-Means++优化了各簇中心点的选取：

1. 随机选取一个样本点作为第一个簇中心 μ_1 ,
2. 计算剩余样本点与所有簇中心的最短距离， $D(x^{(i)}) = \min[j | dist(x^{(i)}, C_j)]$ ，某样本点被选为下一个簇中心的概率为 $\frac{D(x^{(i)})^2}{\sum D(x^{(j)})^2}$,
3. 重复2，直到选出 k 个簇中心。

2 实验过程

2.1 图片输入

输入图片：



已将原始图片的宽高等比缩放之原来的25%。

2.2 代码实现

首先利用cv2库读入图片，并利用numpy将图片转化为二维数组，其中每一行代表一个像素，包含RGB三个列的值。

```
# read the image
image = cv2.imread('crane.jpg')

# break the image into 2D array
pixels = image.reshape(-1, 3)
```

然后创建KMeans对象，并利用fit方法和predict方法得到聚类后的结果。

```
kmeans = KMeans(n_clusters=cluster_cnt)
kmeans.fit(pixels)
labels = kmeans.predict(pixels)
```

最后将聚类后的结果转化为图片，并保存和展示。

```

# reconstructed segmented image
segmented_image = labels.reshape(image.shape[0] ,
                                 image.shape[1])

segments = []

for i in range(cluster_cnt):
    segments.append(np.copy(image))
    segments[i][segmented_image != i] = 0

# display and save the segmented images
for i in range(cluster_cnt):
    cv2.imshow('Segment-' + str(i), segments[i])
    cv2.imwrite('Segment-' + str(i) + '.png', segments[i])
    cv2.waitKey(0)
cv2.destroyAllWindows()

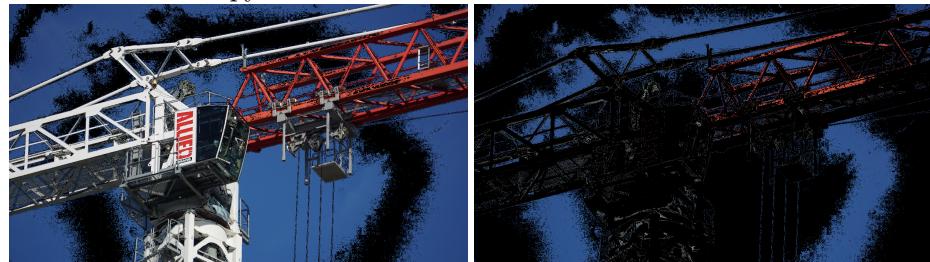
```

2.3 结果展示

聚类结果：

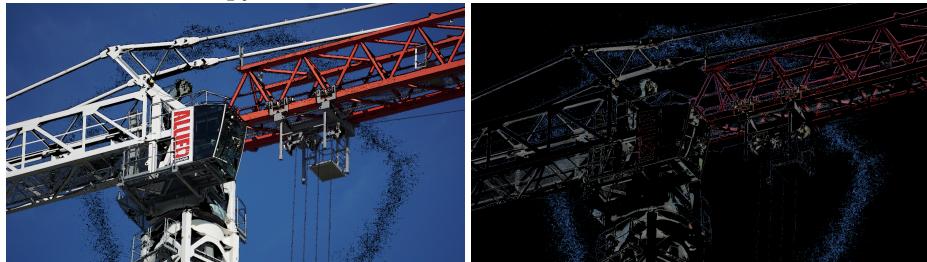
2.3.1 clusters=2, iterations=300

使用KMeans.py实现。



2.3.2 clusters=2, iterations=3000

使用KMeans.py实现。



2.3.3 clusters=3, iterations=default

使用sklearn库实现。



2.3.4 问题

sklearn库中的KMeans默认采用Lloyd算法，距离计算采用欧氏距离；而KMeans.py中的KMeans调用向量取模函数，在距离算法上应该与其相同，但结果却与sklearn库大相径庭。可能是在选点和迭代条件上有约束？