K-Means Clustering: Image Segmentation

Name - Amith Korada

Course - AI & ML
(Batch - 4)

Duration - 12 Months

Problem Statement - Implement image segmentation using K-Means

Prerequisites -

What things you need to install the software and how to install them:

Python 3.6 This setup requires that your machine has the latest version of python. The following URL https://www.python.org/downloads/ can be referred to as download python.

The second and easier option is to download anaconda and use its anaconda prompt to run the commands. To install anaconda check this URL https://www.anaconda.com/download/ You will also need to download and install the below 3 packages after you install either python or anaconda from the steps above Sklearn (scikit-learn) numpy scipy if you have chosen to install python 3.6 then run the below commands in command prompt/terminal to install these packages pip install -U sci-kit-learn pip install NumPy pip install scipy if you have chosen to install anaconda then run the below commands in anaconda prompt to install these packages conda install -c sci-kit-learn conda install -c anaconda numpy conda install -c anaconda scipy.

1. Importing necessary libraries-

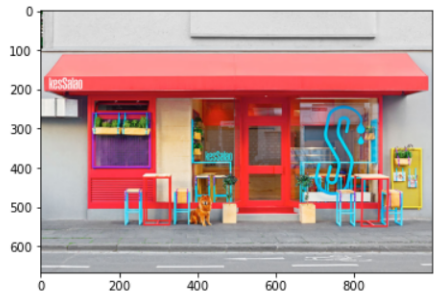
```
import matplotlib.image as mpimg
import matplotlib.pyplot as plt
import numpy as np
```

2. Loading the image and converting it to a NumPy array -

```
def ReadImage(img_file):
    img_arr = mpimg.imread(img_file)
    plt.imshow(img_arr)
    return(img_arr)
```

```
image_arr = ReadImage("./image.jpg")
print(image_arr.shape)
label_arr = np.zeros((image_arr.shape[0],image_arr.shape[1]))
print(label_arr.shape)
```

```
(667, 1000, 3)
(667, 1000)
```



3. Assigning random labels to all the pixels-

```
K = 7

for i in range(label_arr.shape[0]):
    for j in range(label_arr.shape[1]):
        label_arr[i,j] = np.random.choice(K)
print(label_arr)
np.unique(label_arr)
```

```
[[0. 3. 4. ... 2. 2. 2.]
 [3. 6. 5. ... 4. 6. 0.]
 [6. 0. 5. ... 5. 0. 1.]
 ...
 [0. 4. 1. ... 1. 3. 5.]
 [2. 1. 0. ... 6. 5. 0.]
 [4. 6. 1. ... 6. 3. 0.]]

array([0., 1., 2., 3., 4., 5., 6.])
```

4. Function to calculate pth order distance -

```
def dist_p(vec1,vec2,p):
    L = len(vec1)
    s1 = 0
    for l in range(L):
        diff = np.abs(vec2[l]-vec1[l])
        s1 = s1 + diff**p
    distance = s1**(1/p)
    return(distance)
```

5. Calculating initial means of all the clusters-

```
def init_mean(K,img_arr,label_arr):
    mean_ls = []
    pixel_ls = [[] for k in range(K)]

    for i in range(label_arr.shape[0]):
        for j in range(label_arr.shape[1]):
            for k in range(K):
                if label_arr[i,j] == k:
                    pixel_ls[k].append(np.ravel(img_arr[i,j,:]))

    for k in range(K):
        pixel_mat = np.matrix(pixel_ls[k])
        mean_k = np.mean(pixel_mat,axis=0)
        mean_ls.append(np.ravel(mean_k))
    return(mean_ls)
```

6. Function to update labels of all the pixels after an iteration-

```
def label_update(prev_mean,img_arr,label_arr,p):
    for i in range(img_arr.shape[0]):
        for j in range(img_arr.shape[1]):
            dist_ls = []
            for k in range(len(prev_mean)):
                dist = dist_p(img_arr[i,j,:],prev_mean[k],p)
                dist_ls.append(dist)
            dist_arr = np.array(dist_ls)
            new_label = np.argmin(dist_arr)
            label_arr[i,j] = new_label
    return(label_arr)
```

7. Generating new means of all clusters after updating the labels of all pixels-

```
def mean_from_label(K,prev_mean,img_arr,label_arr):
    pixel_ls = [[] for k in range(K)]

    for i in range(label_arr.shape[0]):
        for j in range(label_arr.shape[1]):
            for k in range(K):
                if label_arr[i,j] == k:
                    pixel_ls[k].append(np.ravel(img_arr[i,j,:]))

    for k in range(K):
        if len(pixel_ls[k]) != 0:
            pixel_mat = np.matrix(pixel_ls[k])
            mean_k = np.mean(pixel_mat,axis=0)
            prev_mean[k] = np.ravel(mean_k)
    new_mean = prev_mean
    return(new_mean)
```

8. Kmeans Clustering-

```
def KMeans(img_arr,label_arr,K,p,maxIter):
    mean_old = init_mean(K,img_arr,label_arr)
    for t in range(maxIter):
        new_label_arr = label_update(mean_old,img_arr,label_arr,p)
        mean_new = mean_from_label(K,mean_old,img_arr,new_label_arr)
        print("The mean obtained at {}th iteration is {}\n".format(t,mean_new))
        label_arr = new_label_arr
        mean_old = mean_new
    return(mean_new,label_arr)
```

```
mean_final,label_final = KMeans(image_arr,label_arr,K,2,5)
```

```
The mean obtained at 0th iteration is [array([120.4372934 , 131.58953054, 139.5153628 ]), array([172.45069836,  68.31156514,  63.01043926]), array([173.99305556,  64.46006944, 210.91840278]), array([201.55403011, 193.21486111, 191.80421283]), array([184.56842724, 145.06796025, 143.34918303]), array([121.02868526, 152.06215139, 175.03625498]), array([187.85714286, 162.57142857, 128.64285714])]

The mean obtained at 1th iteration is [array([115.4069155 , 114.36333693, 119.53352256]), array([177.41136774,  56.58513901,  51.86298261]), array([159.93470375,  62.85299706, 192.91760235]), array([208.73504124, 205.86700949, 203.45228045]), array([207.97701782, 153.35512815, 155.5975055 ]), array([ 87.74128873, 180.60389629, 199.99687615]), array([219.66940618, 176.50943396, 111.5343211 ])]

The mean obtained at 2th iteration is [array([111.5422608 , 105.35753255, 105.98647448]), array([190.94964923,  53.57431245,  49.43927581]), array([156.39906103,  64.10475352, 186.48048709]), array([212.00891412, 211.58126407, 209.32548872]), array([200.46840565, 159.76411662, 161.42860626]), array([ 38.64314783, 201.21657269, 230.64491297]), array([225.40072698, 180.51323197, 101.03058646])]

The mean obtained at 3th iteration is [array([103.38519933,  92.6199585 ,  90.22401691]), array([204.7218085 ,  53.88166145,  51.31799853]), array([155.41781006,  65.44955609, 183.2433414 ]), array([215.61303496, 215.26146173, 212.56043032]), array([194.02722121, 162.67614696, 163.88613002]), array([ 29.21535519, 204.20863388, 235.17666667]), array([227.81676083, 180.25696798,
99.36224105])]

The mean obtained at 4th iteration is [array([100.978133  ,  83.86426812,  77.99539286]), array([213.0898492 ,  53.63757762,  53.54342764]), array([153.71821306,  67.81725577, 179.69207167]), array([216.99199856, 216.51382068, 213.61516501]), array([189.6269846 , 162.98527136, 164.20274172]), array([ 28.25864828, 204.08827586, 235.22135172]), array([230.79340913, 175.48000477, 106.34274129])]
```

9. Image segmentation-

```
def segmentImage(image_arr,label_arr,mean_ls):
    seg_image = np.zeros((image_arr.shape[0],image_arr.shape[1],image_arr.shape[2]))
    for i in range(seg_image.shape[0]):
        for j in range(seg_image.shape[1]):
            k = label_arr[i,j]
            seg_image[i,j,:] = mean_ls[int(k)]
    seg_image = seg_image.astype(np.uint8)
    plt.imshow(seg_image)
```

```
segmentImage(image_arr,label_final,mean_final)
```