Assignment05_20133096_HyunjaeLee

April 10, 2019

20133096 LEEHYUNJAE
Assignment 05
[K-means algorithm on color image]
Let $f(x)$ be a color image and x be the index of image in the domain. The values of image $f(x)$ consist of [red, green, blue] intensity.

Apply K-means algorithm to image $f(x)$ based on its color value with given number of clusters K and visualize the progress of optimization and results of the algorithm for each selected number of clusters K.

1. Select any color image that consists of distinctive regions with different colors.
2. Apply K-means algorithm to the given image with at least 4 different choice of K.
3. For each K, plot the energy curve and the result image.

[Visualisation]

1. Input color image
2. Energy curve for each K
3. Output image for each K

[Energy]
$1_{\overline{n\sum\_\{x\in\Omega\}|f(x)-m\_c|^2}}$,
where Ω denotes the image domain and the number of pixels $|\Omega|$ is n, and m_c denotes the centroid for cluster c that is the cluster label of $f(x)$.

[Output Image]
g(x) = m_c where label(x) = c
Each pixel of the output image $g(x)$ should be its centroid m_c where c is the cluster label of $g(x)$.

Set up
Input an Image by io.imread having 0-255 value for each pixel

```
In [2]: import matplotlib.pyplot as plt
        import numpy as np
        import random
        import math
        from scipy import signal
        from skimage import io, color
        from skimage import exposure
        import sys
```

```
file_image         = 'image.jpg'

im_color           = io.imread(file_image) # 0-255
```

In [148]:
```
# Print input image
plt.title('input image')
plt.imshow(im_color)
plt.axis('off')
plt.show()
```

Define functions

In [4]:
```
# Width = 168 pixel, Height = 300 pixel
Function_X = np.array(im_color) # 300 x 168 x 3
Function_X_Size = list(Function_X.shape) # [ 300, 168, 3]
centroid_list = np.zeros((5,3))
average = np.zeros((5,3))
count_list = np.zeros((5,1))
Energy = []

# some variables
threshold = 0.5

def initialize(labelsize, k):
    #print(labelsize)
```

```
    return np.random.randint(k, size = labelsize)

def Average(lst):
    return sum(lst) / len(lst)

def centroid(Image,Label_Array, K, width, height):
    #print(Image.shape) 300,168,3
    for row in range(height):
        for col in range(width):
            for k in range(K):
                if Label_Array[row][col] == k:
                    centroid_list[k] += Image[row][col]
                    count_list[k] += 1
    #print(count_list)
    return centroid_list/count_list

def labeling(Image,avg, Label_Array, K, width, height):

    energy = 0

    for row in range(height):
        for col in range(width):
            temp = []
            avg_temp = np.zeros(K)
            for k in range(K):
                #temp.append(math.sqrt((Image[row][col] - centroid_list[k])**2))
                input = (Image[row][col] - avg[k])**2
                temp.append(input)

            for rk in range(K):
                avg_temp[rk] = Average(temp[rk])

            Label_Array[row][col] = np.argmin(avg_temp)

    # Calculate Energy
    for row in range(height):
        for col in range(width):
            for k in range(K):
                if(Label_Array[row][col] == k):
                    energy += sum((Image[row][col] - avg[k])**2)

    energy /= (row*col*3)

    return Label_Array,energy

# Print average image
def create_average_image(average,Label_Array,K):
    im_avg = np.zeros(((300,168,3)))
```

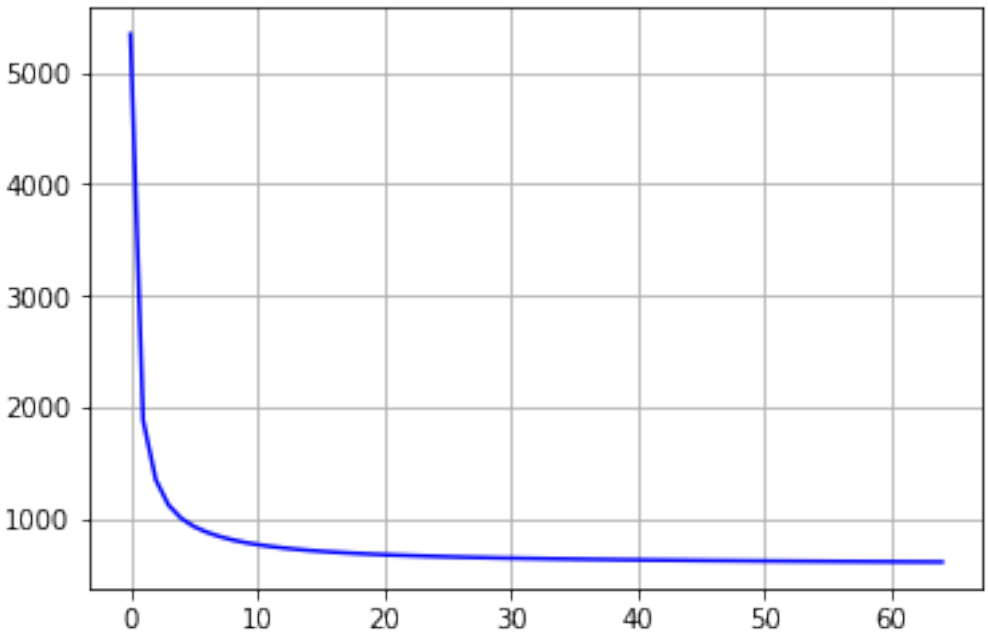
```
            for row in range(300):
                for col in range(168):
                    for k in range(K):
                        if(Label_Array[row][col] == k):
                            im_avg[row][col] = average[k]

            return im_avg
```

K = 5

```
In [228]: #Initialize a label matrix (height X width) and K (k-means)
          iteration = 0
          Energy = []
          Label_Array = initialize((Function_X_Size[0],Function_X_Size[1]), 5)
          Energy.append(99999)

          while True:
              average = centroid(Function_X, Label_Array,5, Function_X_Size[1],Function_X_Size[0
              Label_Array , energy= labeling(Function_X,average, Label_Array, 5, Function_X_Size
              Energy.append(energy)

              #print(Energy)
              iteration += 1

              if Energy[-2] - Energy[-1] < threshold:
                  print('done')
                  break
```

```
done
```

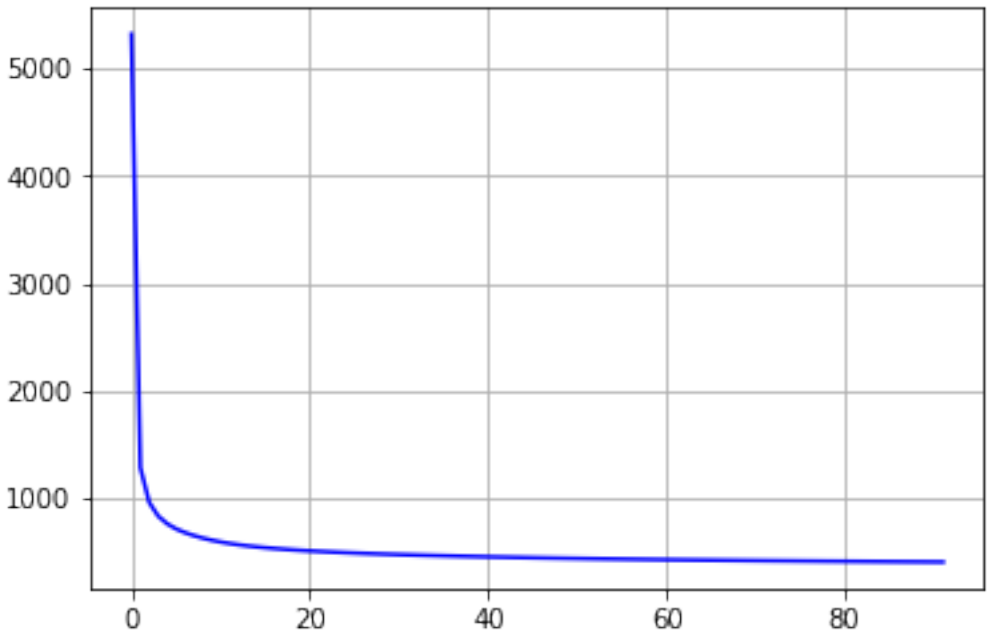
```
In [235]: im_avg = create_average_image(average,Label_Array,5)
          plt.figure(2)
          plt.title('K = 5 average image')
          plt.imshow(im_avg.astype(np.uint8))
          plt.axis('off')
          plt.show()

          # energy graph
          plt.figure(3)
          plt.plot(Energy[1:],"b")
          #plt.title("Energy graph when K = ",5)
          plt.grid(True)
          plt.show()
```



K = 10

```
In [236]: # Initialization
          centroid_list = np.zeros((10,3))
```

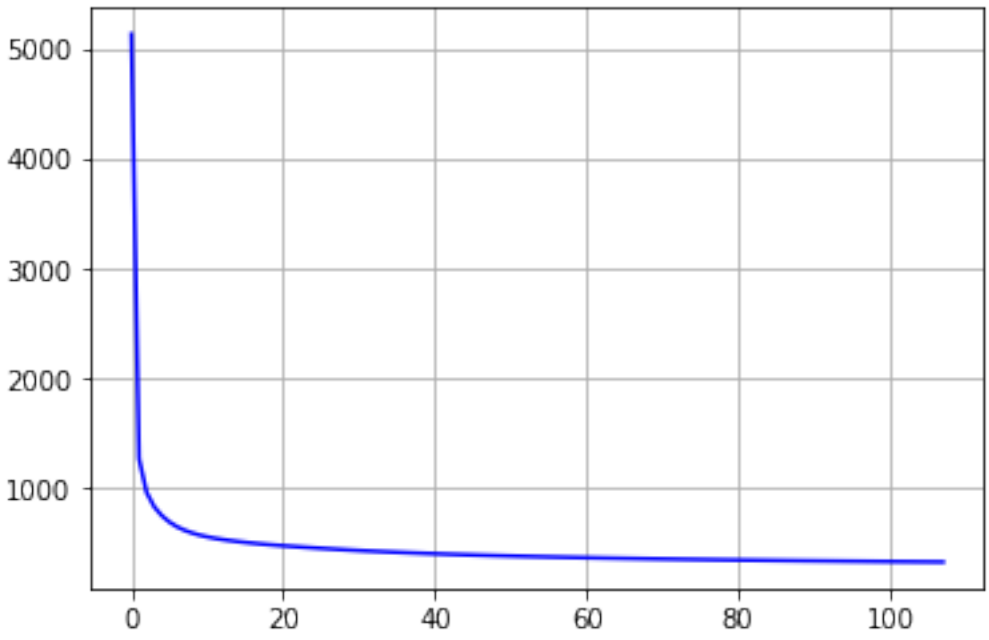
```
        average = np.zeros((10,3))
        count_list = np.zeros((10,1))
        Energy = []
        iteration = 0
        Label_Array = initialize((Function_X_Size[0],Function_X_Size[1]), 10)
        Energy.append(99999)

        while True:
            average = centroid(Function_X, Label_Array,10, Function_X_Size[1],Function_X_Size[
            Label_Array , energy= labeling(Function_X,average, Label_Array, 10, Function_X_Siz
            Energy.append(energy)

            #print(Energy)
            iteration += 1

            if Energy[-2] - Energy[-1] < threshold:
                print('done')
                break

done


In [237]: im_avg = create_average_image(average,Label_Array,10)
          plt.figure(4)
          plt.title('K = 10 average image')
          plt.imshow(im_avg.astype(np.uint8))
          plt.axis('off')
          plt.show()

          # energy graph
          plt.figure(5)
          plt.plot(Energy[1:],"b")
          plt.grid(True)
          plt.show()
```



K = 15

```
In [5]: # Initialization
        centroid_list = np.zeros((15,3))
```

```
        average = np.zeros((15,3))
        count_list = np.zeros((15,1))
        Energy = []
        iteration = 0
        Label_Array = initialize((Function_X_Size[0],Function_X_Size[1]), 15)
        Energy.append(99999)

        while True:
            average = centroid(Function_X, Label_Array,15, Function_X_Size[1],Function_X_Size[0]
            Label_Array , energy= labeling(Function_X,average, Label_Array, 15, Function_X_Size[
            Energy.append(energy)

            #print(Energy)
            iteration += 1

            if Energy[-2] - Energy[-1] < threshold:
                print('done')
                break

done


In [6]: im_avg = create_average_image(average,Label_Array,15)
        plt.figure(5)
        plt.title('K = 15 average image')
        plt.imshow(im_avg.astype(np.uint8))
        plt.axis('off')
        plt.show()

        # energy graph
        plt.figure(6)
        plt.plot(Energy[1:],"b")
        plt.grid(True)
        plt.show()
```

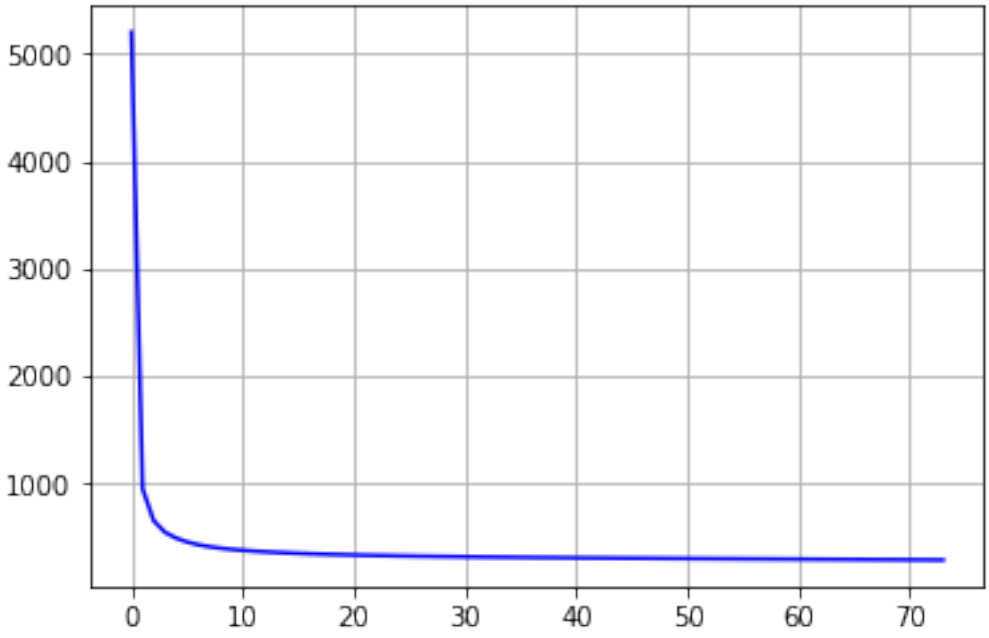


K = 30

```
In [8]: # Initialization
        centroid_list = np.zeros((30,3))
```

```
        average = np.zeros((30,3))
        count_list = np.zeros((30,1))
        Energy = []
        iteration = 0
        Label_Array = initialize((Function_X_Size[0],Function_X_Size[1]), 30)
        Energy.append(99999)

        while True:
            average = centroid(Function_X, Label_Array,30, Function_X_Size[1],Function_X_Size[0]
            Label_Array , energy= labeling(Function_X,average, Label_Array, 30, Function_X_Size[
            Energy.append(energy)

            #print(Energy)
            iteration += 1

            if Energy[-2] - Energy[-1] < threshold:
                print('done')
                break

done


In [9]: im_avg = create_average_image(average,Label_Array,30)
        plt.figure(5)
        plt.title('K = 30 average image')
        plt.imshow(im_avg.astype(np.uint8))
        plt.axis('off')
        plt.show()

        # energy graph
        plt.figure(6)
        plt.plot(Energy[1:],"b")
        plt.grid(True)
        plt.show()
```


K = 30 average image


5000
4000
3000
2000
1000
0
10
20
30
40
50
60
70