K-means algorithm on color image

April 10, 2019

Professor: **Hong Hyung-Woo**

Student NB: **20166450**

Major: **Software Engineering**

Name: **Kim Young Min**

[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]

$\frac{1}{n}\sum_{x\in\Omega}\|f(x)-m_c\|^2$,

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 $g(x)$.

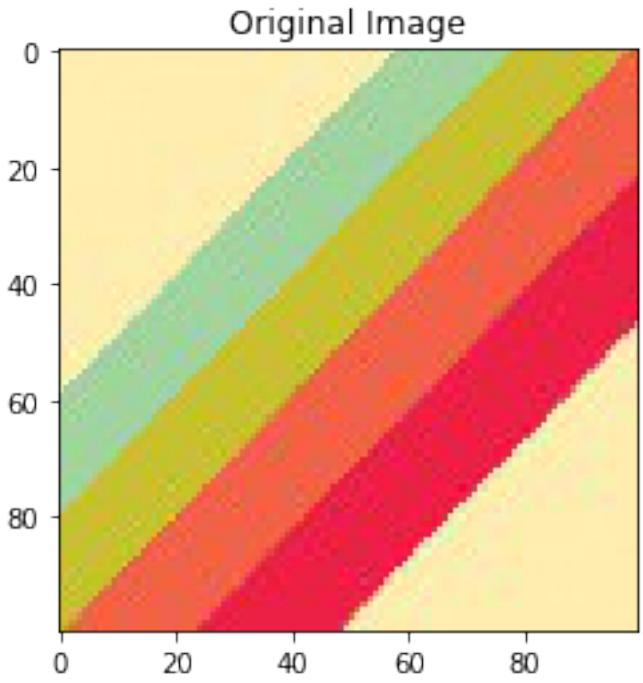
0.0.1 Start!

In [2]:
```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from random import *
import numpy as np
import cv2
```

1 Input color image

In [3]:
```
img = cv2.imread('colors.jpeg',1)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

r,g,b =cv2. split(img)
r= r.flatten()
g= g.flatten()
b= b.flatten()
print("R=",r)
print("G=",g)
print("B=",b)

plt.subplot(1,1,1)
plt.imshow(img)
plt.title("Original Image")
plt.show()

# For convenience about getting a clusterimage by K-means algorithms
img_10000=img.reshape(10000,3)
```

```
R= [255 255 255 ... 255 255 255]
G= [238 238 237 ... 237 237 237]
B= [176 176 175 ... 175 175 175]
```



In [4]:
```
def distance(x, y):

    d = (x - y) ** 2
    s = np.sum(d)
    return(s)



def kmeans_label(label,init_vec,img):

    z=[0]*len(init_vec)

    for j in range(len(img)):
        for i in range(len(init_vec)):
            z[i]=distance(img[j],init_vec[i])
        label[j]=np.argmin(z)

    return label


def energy_func(img,avg_image,avg_label,ener_sum):
    ener_sum=0
    for i in range(10000):
        d=(img[i]-avg_image[avg_label[i]])**2
        ener_sum+=np.sum(d)
```

```
    return(ener_sum)/10000



def Kmeans_algorithm(label,kmeans_vec,init_vec,img,ener_sum,list_ener):

    many=0
    kmeans_label(label,init_vec,img)
    avg_label=np.array([0]*(len(img)))
    avg_image=np.array([[0]*3]*(len(init_vec)))
    avg_label=np.copy(label)

    while(1):

        cnt=[0]*len(init_vec)
        label=np.copy(avg_label)
        avg_label=np.array([0]*len(img))
        avg_image=np.array([[0]*3]*(len(init_vec)))

        for k in range(len(label)):
            avg_image[label[k]]+=img[k]
            cnt[label[k]]+=1

        for l in range(len(init_vec)):
            if (cnt[l]!=0):
                avg_image[l]=avg_image[l]/cnt[l]

        kmeans_label(avg_label,avg_image,img)

        many+=1

        if(np.array_equal(label,avg_label)):
            break


        for i in range(len(label)):
            for j in range(len(init_vec)):
                if (avg_label[i]== j):
                    kmeans_vec[i]= avg_image[j]

        list_ener[many-1]= energy_func(img,avg_image,avg_label,ener_sum)

    print("Iteration Number:",many)
```

```
            return kmeans_vec
```

When k=2,

```
In [5]: label_2=np.array([0]*10000)
        init_vec_2=[[0]*3]*2
        kmeans_vec_2=np.array([[0]*3]*10000)
        ener_sum_2=0
        list_ener2=np.array([0]*20)

        for i in range(2):
                init_vec_2[i]= [np.random.randint(0,255),np.random.randint(0,255),np.random.ra



        Kmeans_algorithm(label_2,kmeans_vec_2,init_vec_2, img_10000,ener_sum_2,list_ener2)
```

```
Iteration Number: 4
```

```
Out[5]: array([[224, 228, 169],
               [224, 228, 169],
               [224, 228, 169],
               ...,
               [224, 228, 169],
               [224, 228, 169],
               [224, 228, 169]])
```

When k=4,

```
In [6]: #k=4
        label_4=[0]*10000
        init_vec_4=np.array([[0]*3]*4)
        kmeans_vec_4=np.array([[0]*3]*10000)
        ener_sum_4=0
        list_ener4=np.array([0]*100)

        for i in range(4):
                init_vec_4[i]= [np.random.randint(0,255),np.random.randint(0,255),np.random.ra



        Kmeans_algorithm(label_4,kmeans_vec_4,init_vec_4, img_10000,ener_sum_4,list_ener4)
```

```
Iteration Number: 6
```

```
Out[6]: array([[233, 220, 128],
               [233, 220, 128],
               [233, 220, 128],
               ...,
               [233, 220, 128],
               [233, 220, 128],
               [233, 220, 128]])
```

When k=6,

```
In [19]: #k=6
         label_6=[0]*10000
         init_vec_6=np.array([[0]*3]*6)
         kmeans_vec_6=np.array([[0]*3]*10000)
         ener_sum_6=0
         list_ener6=np.array([0]*100)

         for i in range(6):
                 init_vec_6[i]= [np.random.randint(0,255),np.random.randint(0,255),np.random.ra



         Kmeans_algorithm(label_6,kmeans_vec_6,init_vec_6, img_10000,ener_sum_6,list_ener6)
```

```
Iteration Number: 11
```

```
Out[19]: array([[252, 237, 175],
                [252, 237, 175],
                [252, 237, 175],
                ...,
                [252, 237, 175],
                [252, 237, 175],
                [252, 237, 175]])
```

When k=20,

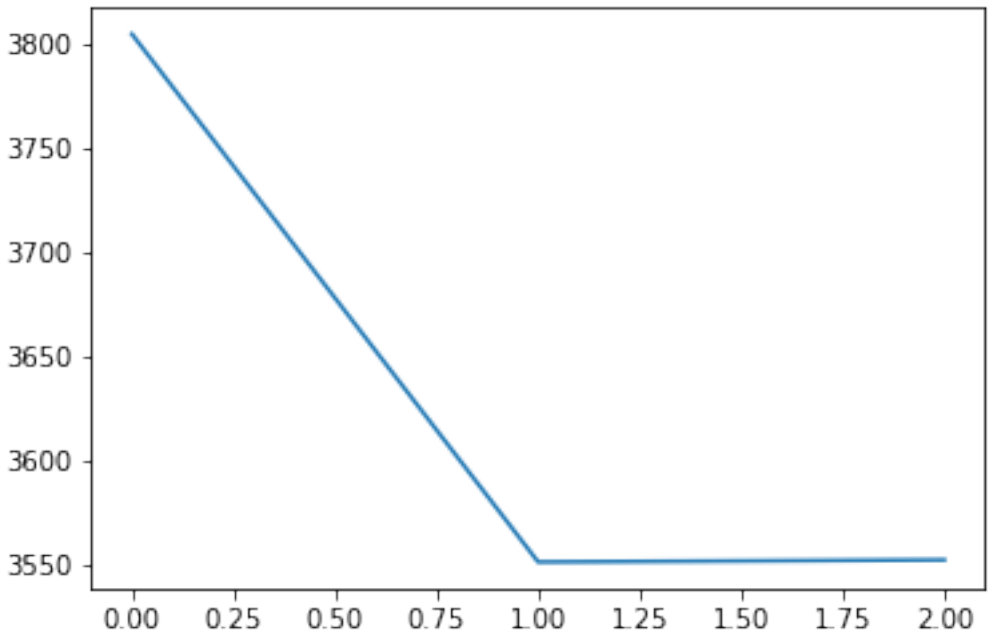
```
In [8]: #k=20
        label_20=[0]*10000
        init_vec_20=np.array([[0]*3]*20)
        kmeans_vec_20=np.array([[0]*3]*10000)
        ener_sum_20=0
        list_ener20=np.array([0]*30)


        for i in range(20):
                init_vec_20[i]= [np.random.randint(0,255),np.random.randint(0,255),np.random.ra
```

```
        Kmeans_algorithm(label_20,kmeans_vec_20,init_vec_20, img_10000,ener_sum_20,list_ener20)

Iteration Number: 11


Out[8]: array([[254, 236, 174],
               [254, 236, 174],
               [254, 236, 174],
               ...,
               [254, 236, 174],
               [254, 236, 174],
               [254, 236, 174]])
```

2 Energy curve for each

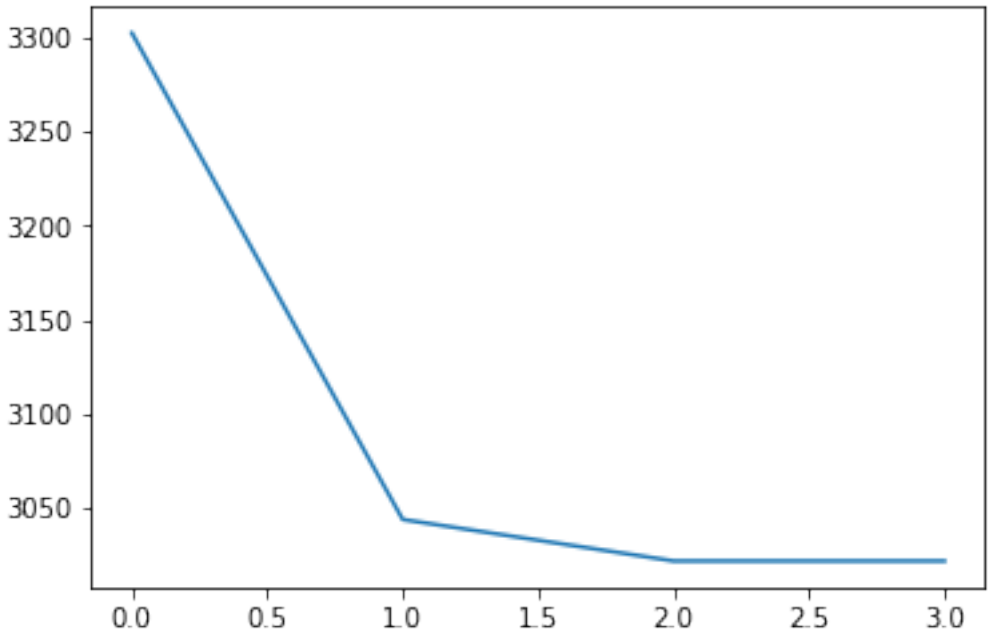
When k=2,

```
In [29]: y1=np.copy(list_ener2[:3])
         plt.plot(y1)

Out[29]: [<matplotlib.lines.Line2D at 0x2d73a455b00>]
```



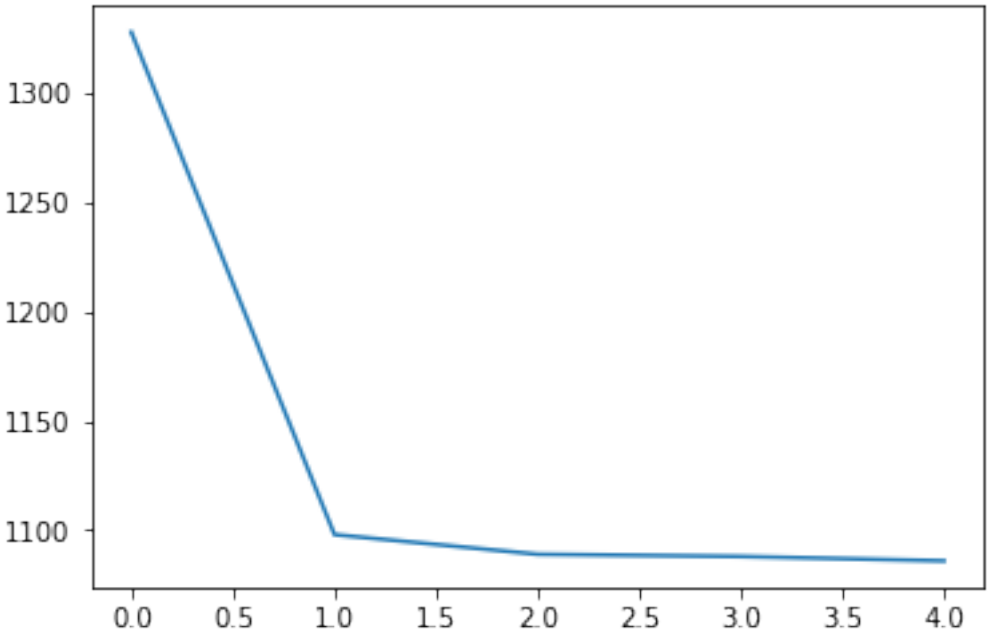
When k=4,

```
In [28]: y2=np.copy(list_ener4[:4])
         plt.plot(y2)

Out[28]: [<matplotlib.lines.Line2D at 0x2d73a3f9828>]
```



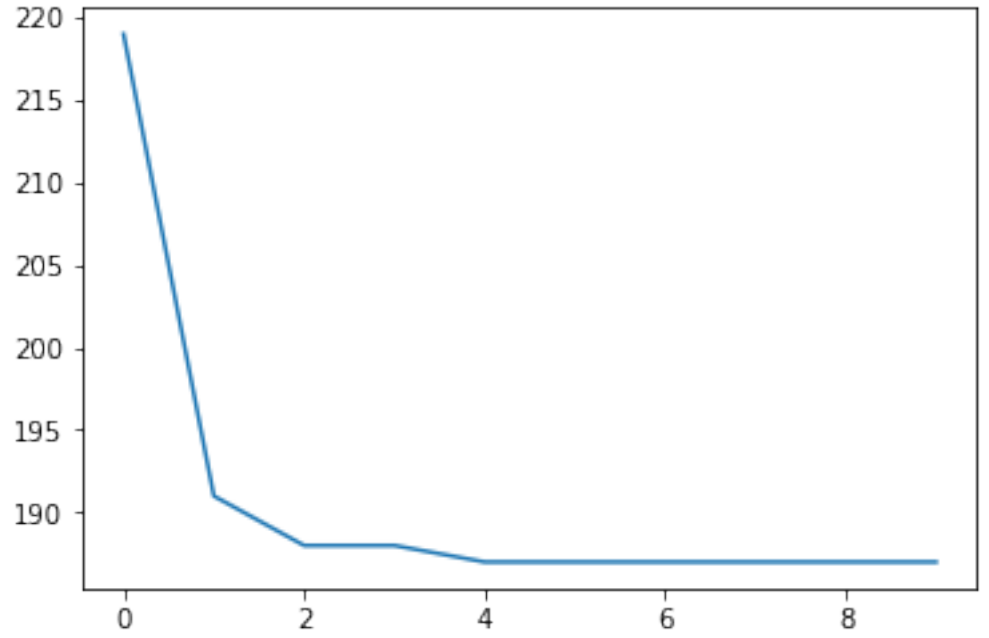
When k=6,

```
In [27]: y3=np.copy(list_ener6[:5])
         plt.plot(y3)
         plt.show()
```



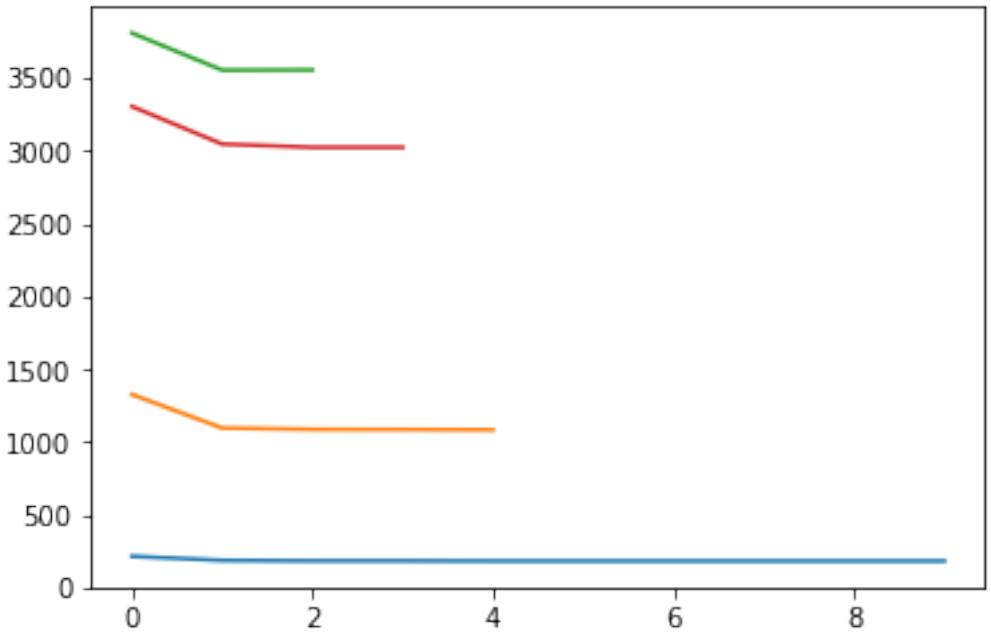
When k=20,

```
In [24]: y4=np.copy(list_ener20[:10])
         plt.plot(y4)

Out[24]: [<matplotlib.lines.Line2D at 0x2d739028f98>]
```



Plot the total of graph about each k.

```
In [30]: plt.plot(y4)
         plt.plot(y3)
         plt.plot(y1)
         plt.plot(y2)
         plt.show()
```



3 Output image for each

```
In [31]: output1 = img
         output2 = img
         output3 = img
         kmeans_vec_2=np.array(kmeans_vec_2)
         kmeans_vec_2=kmeans_vec_2.reshape(100,100,3)

         kmeans_vec_4=np.array(kmeans_vec_4)
         kmeans_vec_4=kmeans_vec_4.reshape(100,100,3)

         kmeans_vec_6=np.array(kmeans_vec_6)
         kmeans_vec_6=kmeans_vec_6.reshape(100,100,3)
```

```
kmeans_vec_20=np.array(kmeans_vec_20)
kmeans_vec_20=kmeans_vec_20.reshape(100,100,3)

output=[img,kmeans_vec_2, kmeans_vec_4, kmeans_vec_6, kmeans_vec_20 ]

titles = ['Original','k=2', 'k=4', 'k=6', 'k=20']

for i in range(5):
    plt.subplot(2, 3, i+1)
    plt.imshow(output[i])
    plt.title(titles[i])
    plt.xticks([])
    plt.yticks([])
plt.show()
```