Assignment05

April 11, 2019

20132915 Nam, Geunwoo

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]

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

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 $g(x)$.

Reference: https://www.imageeprocessing.com/2017/12/k-means-clustering-on-rgb-image.html

https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_iris.html

https://scikit-learn.org/stable/auto_examples/cluster/plot_color_quantization.html

https://scikit-learn.org/stable/modules/clustering.html

```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        from PIL import Image
        from IPython.display import display
        from sklearn.cluster import KMeans
        %matplotlib inline
```

In [2]:
```
#
# normalize the values of the input data to be [0, 1]
#
def normalize(data):

    data_normalized = (data - min(data)) / (max(data) - min(data))

    return(data_normalized)
```

In [3]:
```
#
# example of distance function between two vectors x and y
#
def distance(x, y):

    d = (x - y) ** 2
    s = np.sum(d)
    # r = np.sqrt(s)

    return(s)
```

In [4]:
```
im = Image.open('../../exercise/python/data/tiger.jpeg')
im_width, im_height = im.size
```

In [5]:
```
# Show the image inline
display(im)
```

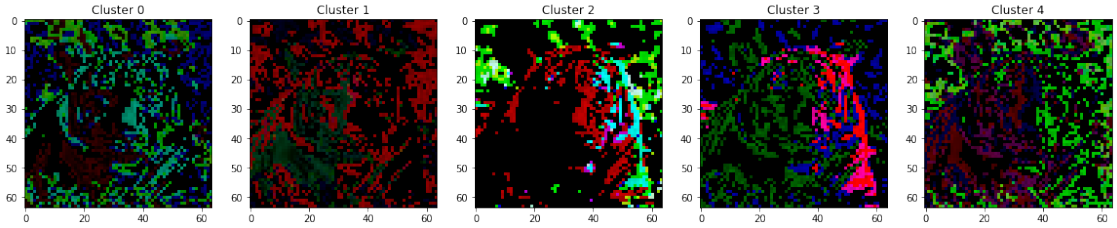
In [6]:
```
np_im = np.asarray(im)
im.close()

# Decompose by RGB colors and normalize
np_im_R = normalize(np_im[:,:,0].reshape(-1,1))
np_im_G = normalize(np_im[:,:,1].reshape(-1,1))
np_im_B = normalize(np_im[:,:,2].reshape(-1,1))
```

In [7]:
```
# K-Means clustering
# K = 5, ..., 10

cost =[]
for i in range(5, 11):
    kmeans_R = KMeans(n_clusters=i, random_state=0, verbose=0).fit(np_im_R)
```

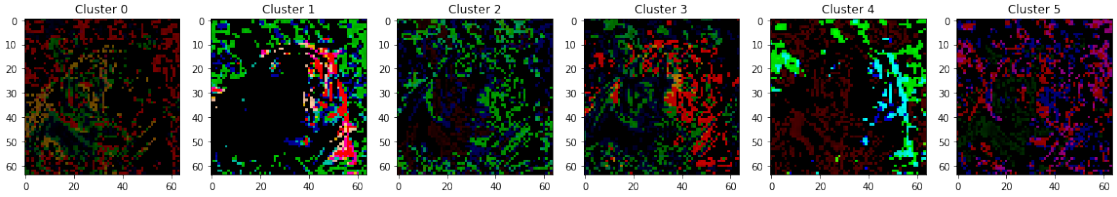
```
            kmeans_G = KMeans(n_clusters=i, random_state=0, verbose=0).fit(np_im_G)
            kmeans_B = KMeans(n_clusters=i, random_state=0, verbose=0).fit(np_im_B)

            cost.append(kmeans_R.inertia_+kmeans_G.inertia_+kmeans_B.inertia_)
            fig_result = plt.figure(figsize=(20,10))
            for j in range(0, i):
                kmeans_im_R_labels_tmp = (kmeans_R.labels_ == j).\
                reshape((im_width*im_height,1))
                kmeans_im_G_labels_tmp = (kmeans_G.labels_ == j).\
                reshape((im_width*im_height,1))
                kmeans_im_B_labels_tmp = (kmeans_B.labels_ == j).\
                reshape((im_width*im_height,1))

                kmeans_im_R = (np.multiply(kmeans_im_R_labels_tmp, np_im_R)).\
                reshape((im_width,im_height))
                kmeans_im_G = (np.multiply(kmeans_im_G_labels_tmp, np_im_G)).\
                reshape((im_width,im_height))
                kmeans_im_B = (np.multiply(kmeans_im_B_labels_tmp, np_im_B)).\
                reshape((im_width,im_height))

                kmeans_im = np.zeros((im_width,im_height,3))
                kmeans_im[:,:,0] = np.copy(kmeans_im_R)
                kmeans_im[:,:,1] = np.copy(kmeans_im_G)
                kmeans_im[:,:,2] = np.copy(kmeans_im_B)

                # Plot the result image.
                fig_result.suptitle("K = %d" % i, fontsize=25)
                #plt.imshow(kmeans_im)
                #plt.show()
                # Plot the result in one row.
                plt_sub = plt.subplot(1, i, j+1)
                plt_sub.title.set_text("Cluster %d" % j)
                plt_sub.imshow(kmeans_im)

        # Plot the cost
        fig_cost = plt.figure()
        fig_cost.suptitle("Cost function by K")
        plt.plot(range(5,11), cost)
        fig_cost.show()
```

```
/home/nam/anaconda3/lib/python3.7/site-packages/matplotlib/figure.py:445: UserWarning: Matplotl
  % get_backend())
```

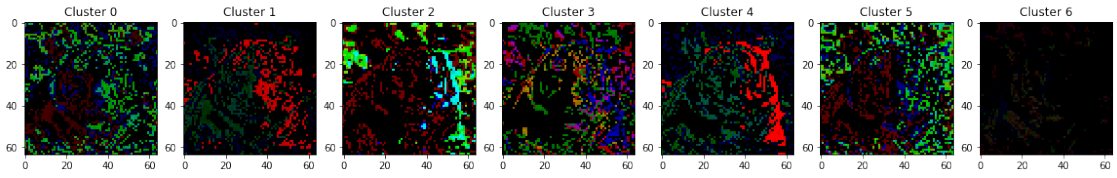
K = 5


Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4

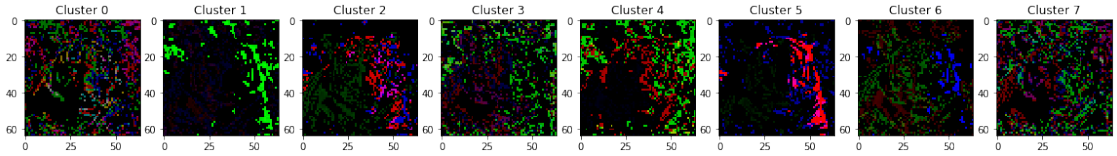
K = 6


Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5

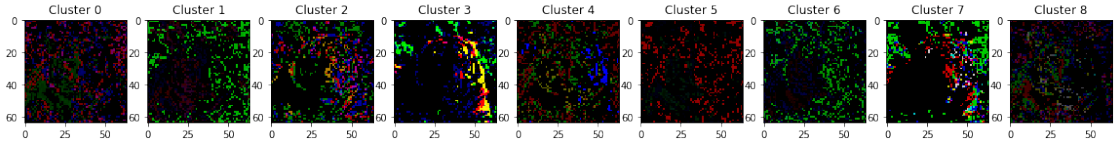
K = 7


Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6

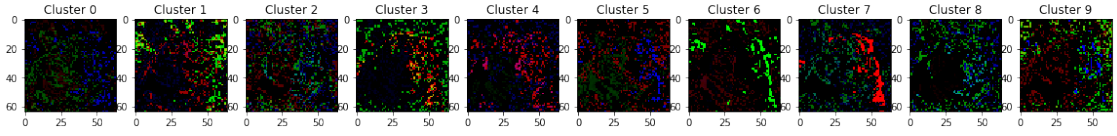
K = 8

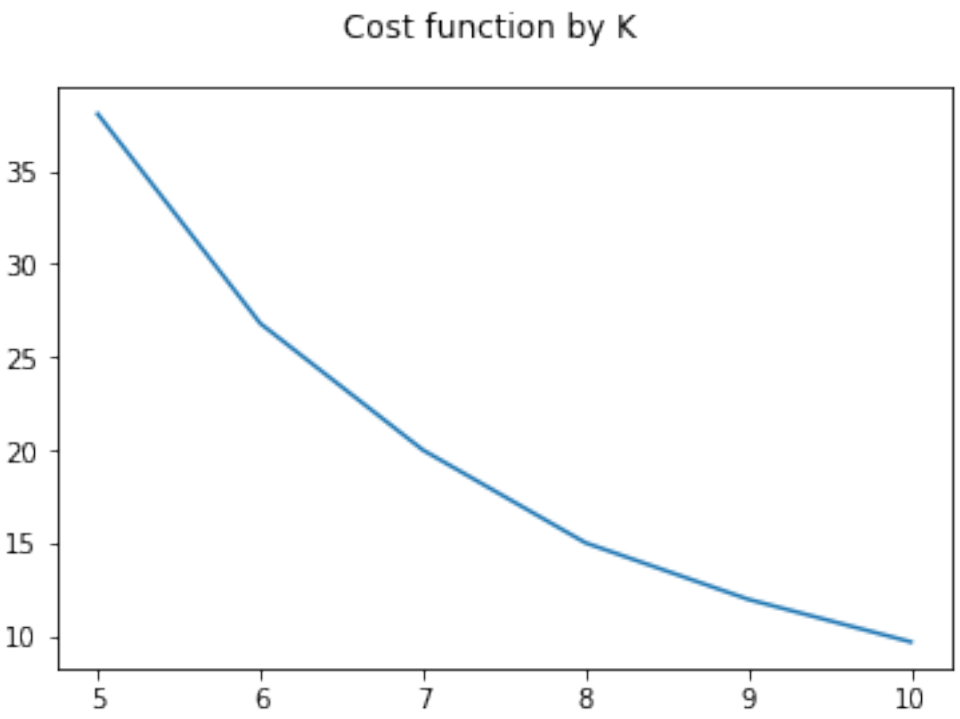

Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6
Cluster 7

K = 9


Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6
Cluster 7
Cluster 8

K = 10


Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6
Cluster 7
Cluster 8
Cluster 9



In [ ]: