TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

***

BÁO CÁO ĐỒ ÁN THỰC HÀNH TOÁN ỨNG DỤNG VÀ THỐNG KÊ

Họ Và Tên: Hà Minh Toàn

MSSV: 18120599

Mục Lục:

I. Ý tưởng thực hiện:

Em đã được học và cài đặt thuật toán Kmeans ở môn học khác, em dùng lại kết quả đó và nâng cấp lên bằng numpy để tăng tốc độ tính toán

II. Mô tả các hàm:

```
class k_means:
    def __init__(self, k=2, thresold = 0.001, max_iter = 300, has_converged=False):

        '''
        Class constructor

        Parameters
        ----------
        - k: number of clusters.
        - thresold (percentage): stop algorithm when difference between prev cluster
                                 and new cluster is less than thresold
        - max_iter: number of times centroids will move
        - has_converged: to check if the algorithm stop or not
        '''

        self.k = k
        self.thresold = thresold
        self.max_iter = max_iter
        self.has_converged= has_converged
```

init: khởi tạo các biến cho model.

- K: số cluster.
- Thresold: ngưỡng khác nhau giữa các cluster mà thuật toán sẽ dừng.
- max_iter: số lần lặp lại tối đa của thuật toán.
- has_converged: Các điểm đã tụ lại cluster hay chưa (True, False).

```
def initCentroids(self, X):
    '''
    Parameters
    ----------
    X: input data.
    '''
    #Starting clusters will be random members from X set
    indexes = np.random.randint(0, len(X)-1,self.k)
    self.centroids=X[indexes].astype('float')
```

initCentroids: khởi tạo các tâm cụm ban đầu là random của các điểm dữ liệu đầu vào.

```
def updateCentroids(self, cur_centroids):
    '''
    Class constructor

    Parameters
    ----------
    cur_centroids: list of new centroids

    '''
    self.has_converged=True
    "Create Distance Array"
    centroid_Distance =np.sqrt(np.sum((
        (self.centroids.astype(float) - cur_centroids)**2),axis=1))

    "If 1 of the centroid have distance more than thresold => move centroid"
    if (centroid_Distance > self.thresold).sum() !=0:
        self.has_converged = False
        self.centroids = cur_centroids
```

updateCentroids: tính toán khoảng cách của các centroids sau mỗi vòng lặp => ra mảng khoảng cách, nếu không có khoảng cách nào lớn hơn **thresold** thì dừng, ngược lại thì cập nhật tâm cụm mới

```
def fit(self, X):
    '''
    FIT function, used to find clusters

    Parameters
    ----------
    X: input data.
    '''
    #Init list cluster centroids
    self.initCentroids(X)
    centroid_id=np.arange(self.k).reshape(self.k,1)

    #Main loop
    for _ in range(self.max_iter):
        "Create Centroid, input 3d Matrix"
        centroid_compute =  self.centroids[np.newaxis,:,:]
        input_compute = X[:,np.newaxis,:].astype('float')

        "Create a 3d distance Matrix of input and Centroid"
        distance_Matrix = np.sqrt(np.sum((centroid_compute -
                                          input_compute)**2,axis=2))
        "Get min distance to centroid of each input "
        belong_flag_Matrix = np.argmin(distance_Matrix,axis =1)

        "Create 2d Mask"
        mask = (centroid_id == belong_flag_Matrix)

        "Create a 3d Matrix with point belong to centroid by surface  "
        mean_compute_df = (mask.T[:,:,np.newaxis] * input_compute ).astype('float')

        "replace 0 with np.nan to use np.nanmean"
        mean_compute_df[mean_compute_df==0.0]= np.nan
        "calculate mean of a surface by np.nanmean"
        cur_centroids = np.nanmean(mean_compute_df,axis=0)


        "Update centroid"
        self.updateCentroids(cur_centroids)

        if self.has_converged:
            break
    "Change data type of centroid"
    self.centroids = np.array(self.centroids).round().astype('uint8')
    return self.centroids
def clusterApply(self,img_point):
```

fit: tạo ra ma trận khoảng cách giữa các điểm và cluster mà nó thuộc về, tính trung bình giá trị của cluster và update cluster mới bằng **updateCentroids.**

```
    def clusterApply(self,img_point):
        "Create a distance array of point and centroid, and return closet centroid "
        distance_Matrix  = np.sqrt(np.sum(((self.centroids.astype(float) - img_point)
                                           **2),axis=1))
        return self.centroids[np.argmin(distance_Matrix)]

    def compressAndPerform(self,img_arr,shape):
        "Apply cluserApply and return new image"
        newImg = np.apply_along_axis(self.clusterApply, 1, img_arr)
        return PIL.Image.fromarray(newImg.reshape(shape))
```

clusterApply: trả về tâm cụm tương ứng với điểm đầu vào.

compresAndPerform: chạy hàm **clusterApply** với tất cả các điểm ảnh.

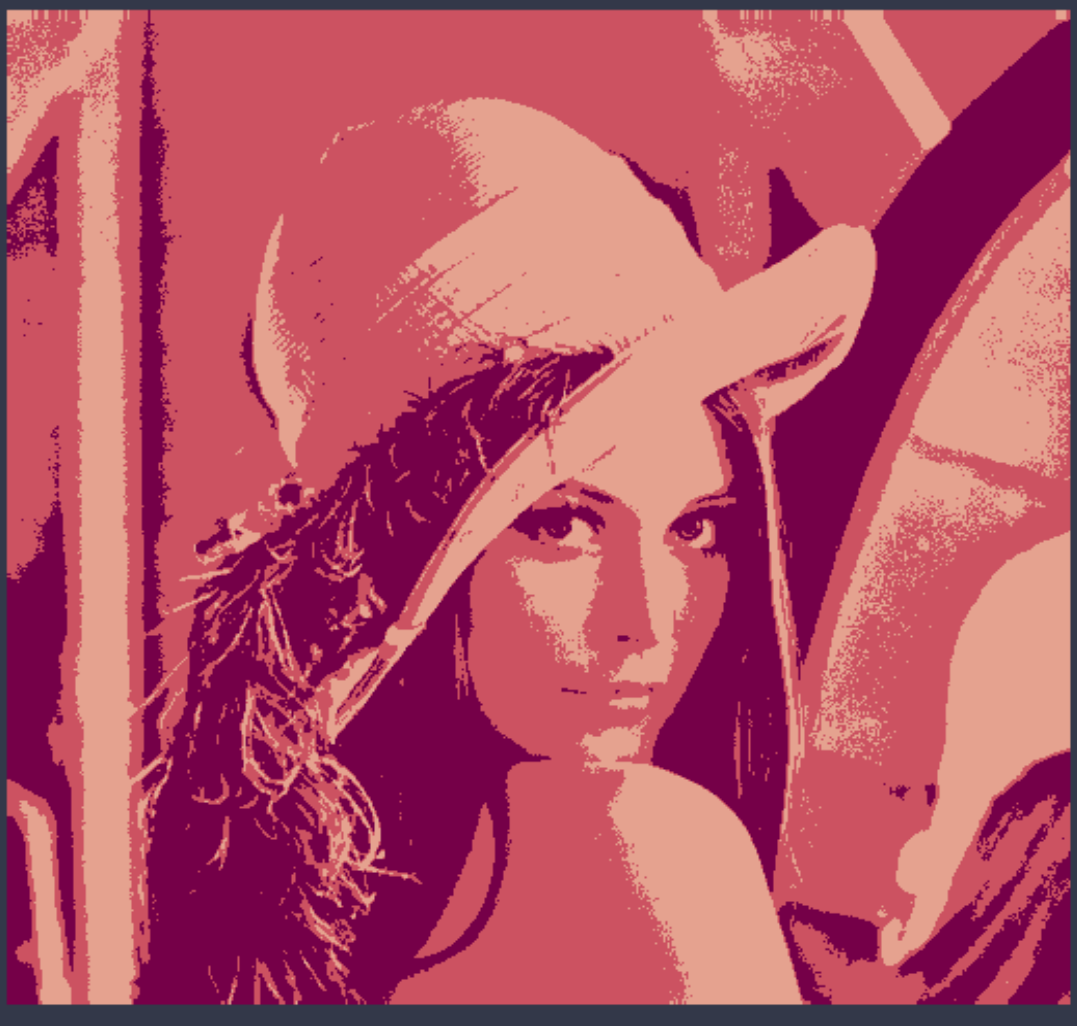
```
    def fit_and_compress(self,numpy_img):
        fit_input = numpy_img.reshape(numpy_img.shape[0]*numpy_img.shape[1],3)
        self.fit(fit_input)
        return self.compressAndPerform(fit_input,numpy_img.shape)
```

fit_and_compress: Gộp **fit** và **compressAndPerfrom** với đầu vào là mảng ảnh gốc.

III. Hình ảnh kết quả:

- với k =3

```
k_means(k=3,thresold = 0.01).fit_and_compress(numpy_arr)
```



- với k =5

```
2 k_means(k=5,thresold = 0.01).fit_and_compress(numpy_arr)
```

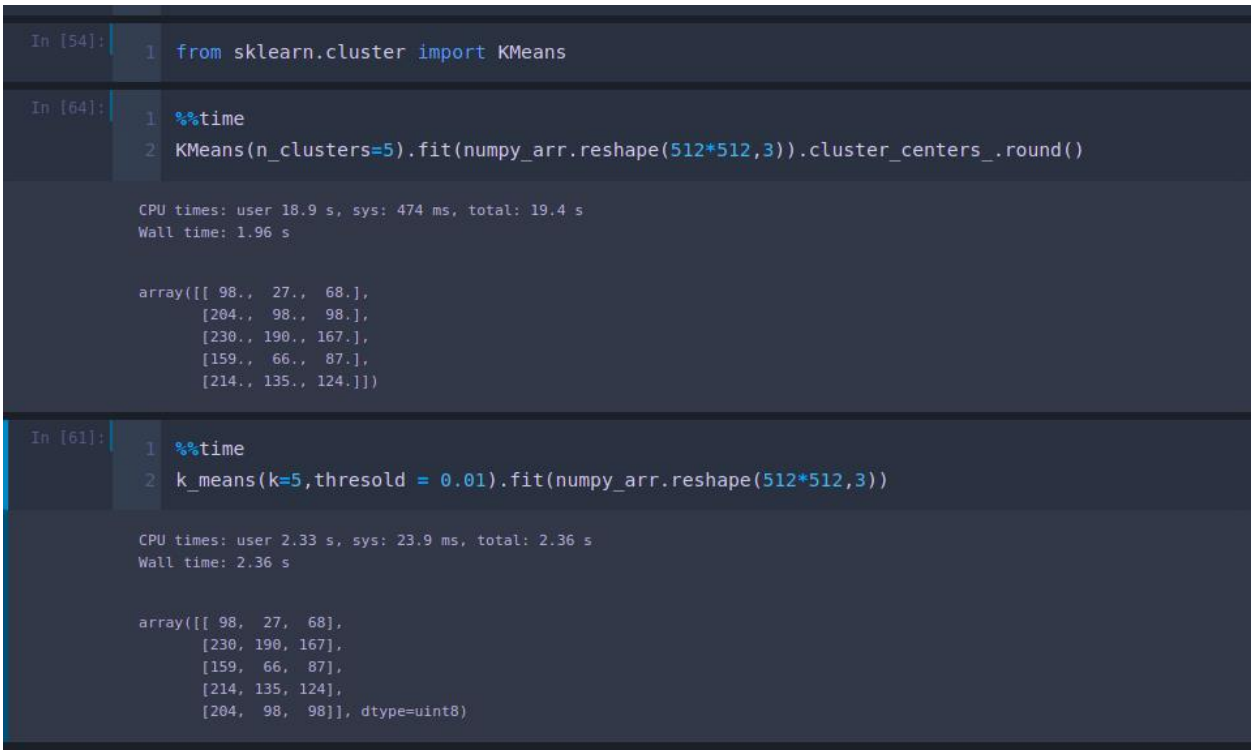
```
CPU times: user 7.57 s, sys: 228 ms, total: 7.79 s
Wall time: 7.83 s
```

- với k =7

```
[53]:
1 k_means(k=7,thresold = 0.01).fit_and_compress(numpy_arr)
```

IV. Nhận xét kết quả:

- với k=5 hình ảnh có vẻ giống mới hình mẫu giáo viên đã gửi.



```
In [54]:
from sklearn.cluster import KMeans

In [64]:
%%time
KMeans(n_clusters=5).fit(numpy_arr.reshape(512*512,3)).cluster_centers_.round()

CPU times: user 18.9 s, sys: 474 ms, total: 19.4 s
Wall time: 1.96 s


array([[ 98.,  27.,  68.],
       [204.,  98.,  98.],
       [230., 190., 167.],
       [159.,  66.,  87.],
       [214., 135., 124.]])

In [61]:
%%time
k_means(k=5,thresold = 0.01).fit(numpy_arr.reshape(512*512,3))

CPU times: user 2.33 s, sys: 23.9 ms, total: 2.36 s
Wall time: 2.36 s


array([[ 98,  27,  68],
       [230, 190, 167],
       [159,  66,  87],
       [214, 135, 124],
       [204,  98,  98]], dtype=uint8)
```

- thuật toán ra tâm cụm tương đương với KMeans của Sklearn nhưng chậm hơn 0.4s