image-clustering

August 31, 2023

1 K_means:

K-means Clustering:

K-means is a popular unsupervised machine learning algorithm used for clustering and grouping data points into distinct clusters. The algorithm aims to partition a dataset into K clusters, where each data point belongs to the cluster with the nearest mean (centroid). It iteratively refines the cluster assignments and centroids until convergence.

K-means Clustering for Image Segmentation:

K-means clustering can also be applied to segment images into distinct regions based on color similarity. In this context, each pixel's color values are treated as feature vectors, and the K-means algorithm groups pixels with similar colors into clusters. This approach can effectively segment an image into regions that share similar color characteristics.

The steps for applying K-means clustering to image segmentation are as follows: 1. ***Load the image*** 2. ***Convert the image into an array of pixel values.*** 3. ***Choose a number 'k' of cluster centers, in this case, 'k' equals 6.*** 4. ***Apply the K-means algorithm to the pixel data using the 'k' cluster centers.*** 5. ***Repeat steps 3-4 until the cluster centers no longer significantly change or a maximum number of iterations is reached.*** 6. ***Assign each pixel to its closest group based on the distance between the pixel and the cluster centers.*** 7. ***Visualize the results by coloring each group with a different color.***

Importing the basic librarires

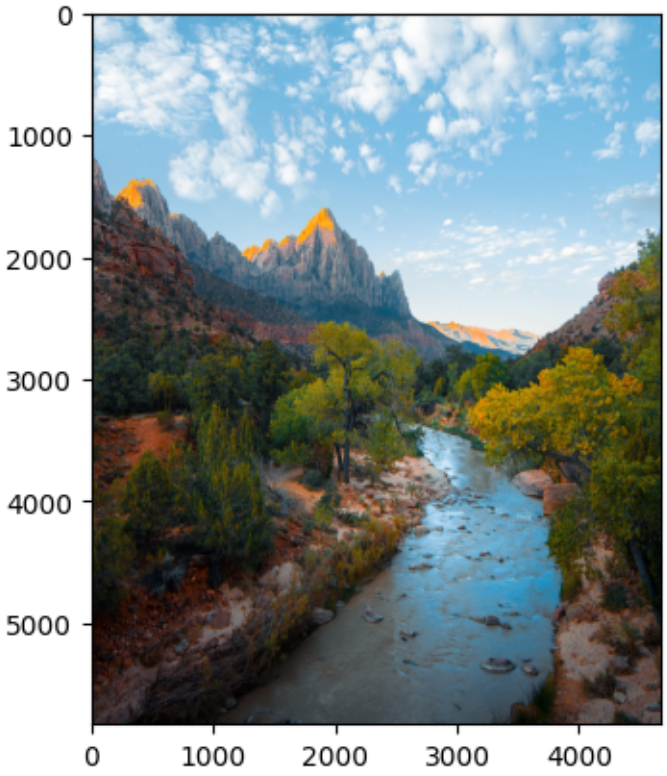
[3]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from PIL import Image
```

Load the image

[4]:
```
image = np.array(Image.open("michael.jpg"))
```

Display the image

[5]:
```
plt.imshow(image)
plt.show()
```



Convert the image into a pixel array

[6]:
```
pixel_valus = image.reshape(-1,3)
```

Apply K-means with 6 cluster centers

[7]:
```
kmeans = KMeans(n_clusters=6 ,random_state=0)
kmeans.fit(pixel_valus)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
```

[7]:
```
KMeans(n_clusters=6, random_state=0)
```

Assign each pixel to its nearest group

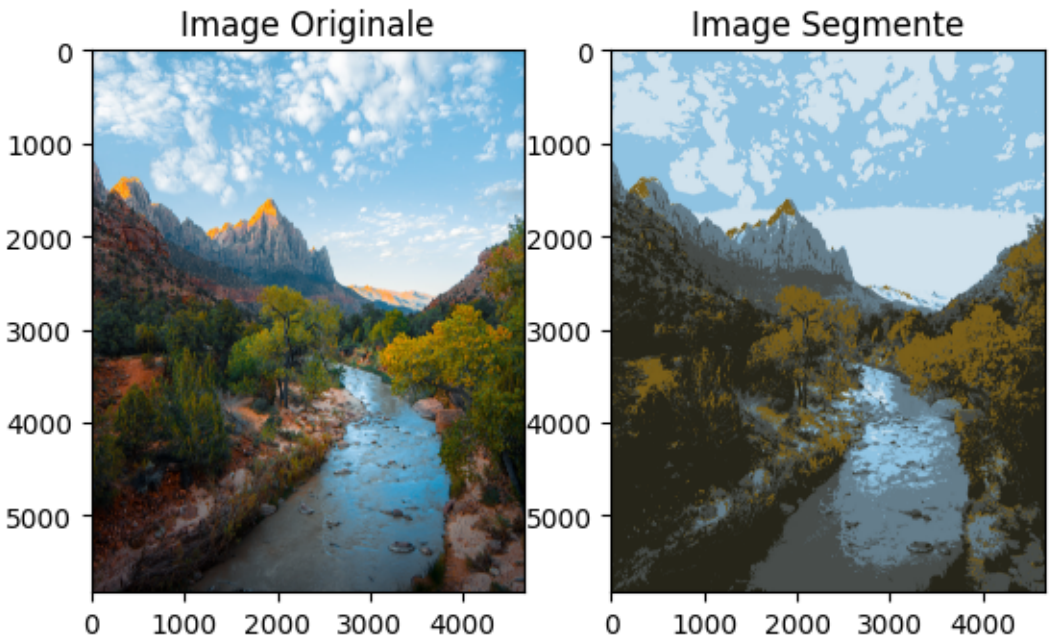
[8]:
```
segmented_image = kmeans.cluster_centers_[kmeans.labels_]
```

Restore the image to its original form

[9]:
```
segmented_image = segmented_image.reshape(image.shape)
```

Display the results

[16]:
```
fig,ax = plt.subplots(1 ,2 )
ax[0].imshow(image.astype('uint8'))
ax[0].set_title('Image Original')
ax[1].imshow(segmented_image.astype('uint8'))
ax[1].set_title('Image Segmented')
plt.show()
```



2 CHA (Classification Hiérarchique Ascendante):

Hierarchical Agglomerative Clustering (HAC)

is an unsupervised machine learning algorithm used for grouping data points into hierarchical structures. It builds a tree-like structure of clusters, where each node represents a cluster of data points. HAC starts with each data point as its own cluster and iteratively merges clusters together based on similarity, creating a hierarchy of clusters.

Importing the basic librarires

[17]:
```
import numpy as np
import cv2
from scipy.spatial.distance import pdist,squareform
from scipy.cluster.hierarchy import linkage,fcluster
import matplotlib.pyplot as plt
```

Load the image and convert it to grayscale

[29]:
```
image = cv2.imread("nature.jpg")
gray = cv2.cvtColor(image , cv2.COLOR_BGR2GRAY)
```

Display the image

[30]:
```
plt.imshow(image)
plt.axis('off')
plt.show()
```

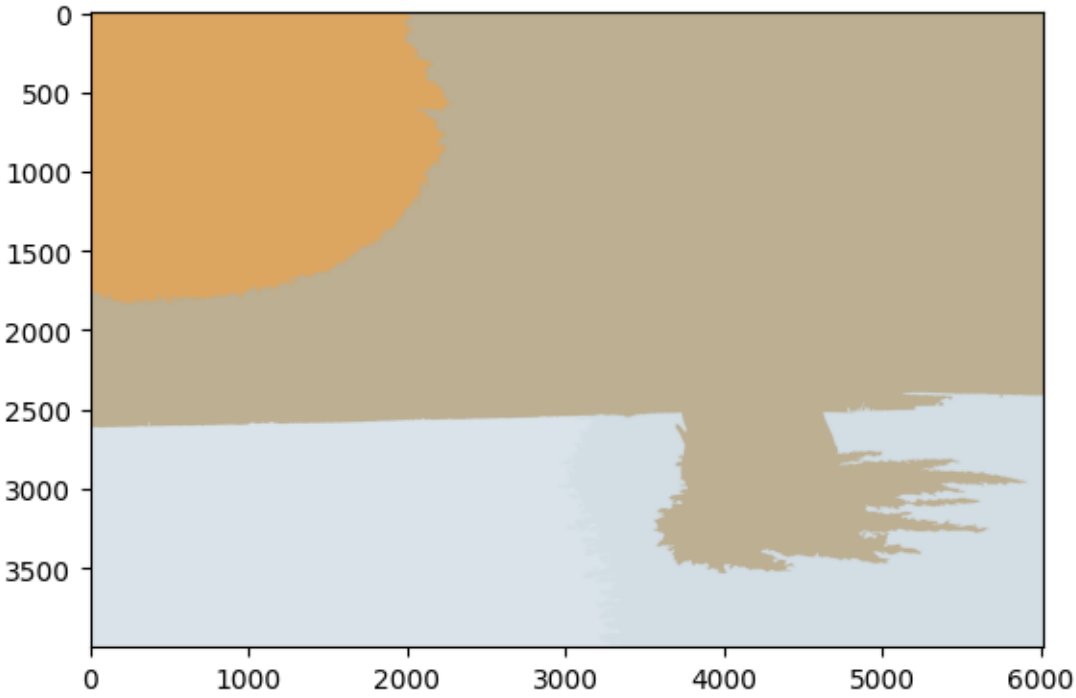
Apply the CHA algorithm to find 6 regions

[31]:
```
from skimage import io,segmentation ,color
from skimage import segmentation

labels = segmentation.slic(image , n_segments=6 ,compactness=10 ,sigma=1)
```

Display the results

[32]:
```
out =color.label2rgb(labels , image ,kind='avg')
plt.imshow(out)
plt.show()
```


0
500
1000
1500
2000
2500
3000
3500
0
1000
2000
3000
4000
5000
6000