

KMeans-Answer

August 7, 2023

1 Bài toán:

Phân cụm dữ liệu bằng giải thuật K-means (unsupervised learning)

Mục tiêu - Xây dựng mô hình Kmeans sử dụng thư viện sklearn. - Áp dụng hai mô hình để giải bài toán thực tế (nén ảnh). - Hiểu được cách tinh chỉnh tham số và đánh giá chất lượng mô hình

Dữ liệu - Bài toán mô phỏng Kmeans: Sử dụng hàm sinh dữ liệu tự động của sklearn (sinh ra các điểm ngẫu nhiên theo phân phối Gauss). Mỗi dữ liệu là một điểm trên mặt phẳng Oxy - Bài toán nén ảnh: Ảnh bird_small.png (Hình ảnh về một chú chim)

Mô hình hóa bài toán: bài toán phân cụm.

- Đầu vào: N vector D chiều, tương ứng N điểm dữ liệu với D thuộc tính trong không gian. Số cụm khởi tạo.
- Đầu ra: Các tâm cụm và chỉ số cụm của mỗi điểm dữ liệu đã cho ban đầu.

2 Các bước làm

2.1 Các thư viện sử dụng

```
[ ]: import numpy as np
import matplotlib.pyplot as plt

from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs
```

2.2 Bài toán 1: Mô phỏng Kmeans với dữ liệu tự sinh (không gian Oxy)

2.2.1 Chuẩn bị dữ liệu

- Sinh dữ liệu ngẫu nhiên $n_samples = 100$ tương đương 100 điểm
 - `random_state`: biến cố định hàm random - để các điểm sinh ngẫu nhiên
- Mỗi điểm dữ liệu có 2 chiều

```
[ ]: n_samples = 100
random_state = 170
center_points = [[1, 1], [-1, -1], [1, -1]] # sinh ngẫu nhiên các điểm xung
↳ quanh vị trí tâm cố định
# center_points = 3 # tâm cụm được chọn ngẫu nhiên
```

```
X, y = make_blobs(n_samples=n_samples, random_state=random_state,
↳centers=center_points, cluster_std=0.6)
print("Số chiều dữ liệu: ", X.shape, y.shape)
print("5 điểm dữ liệu đầu tiên: \n", X[:5])
```

Số chiều dữ liệu: (100, 2) (100,)

5 điểm dữ liệu đầu tiên:

```
[[ 1.26241305  0.94872541]
 [-0.39743873 -1.18567406]
 [ 1.35081331  0.48041993]
 [ 1.21219555  0.98929291]
 [-0.75344338 -1.09784774]]
```

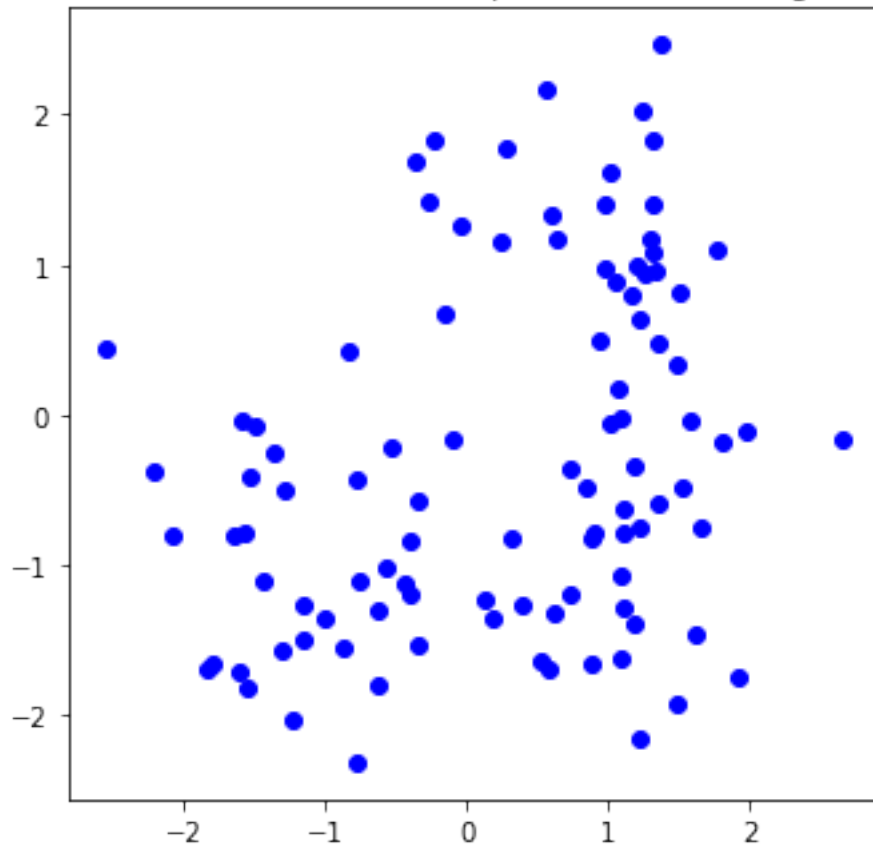
```
[ ]: set(y)
```

```
[ ]: {0, 1, 2}
```

Vẽ các điểm ảnh sử dụng matplotlib

```
[ ]: plt.figure(figsize=(12, 12))
plt.subplot(221)
plt.scatter(X[:, 0], X[:, 1], c='blue') # c là tham số chọn màu sắc, có thể
↳truyền vào string hoặc số id 1,2,3 ...
plt.title("Các điểm dữ liệu trước khi phân cụm. Số lượng: {}".format(n_samples))
plt.show()
```

Các điểm dữ liệu trước khi phân cụm. Số lượng: 100



2.2.2 Dựng giải thuật K-means và huấn luyện

- Sử dụng thư viện sklearn để xây dựng giải thuật K-means, xem chi tiết tại [tài liệu hướng dẫn](#)

```
[ ]: k_cluster = 3
k_mean_model = KMeans(n_clusters=k_cluster, random_state=random_state)
k_mean_model.fit(X)

centers = np.array(k_mean_model.cluster_centers_) # cluster_centers_: là thuộc tính lưu trữ các
# tâm cụm sau khi training
print("Tâm cụm sau khi training ({} tâm): \n".format(k_cluster),
      centers)
```

```
Tâm cụm sau khi training (3 tâm):
[[ 0.88823619  1.19442485]
 [-1.13949326 -0.97100768]
 [ 1.11177838 -0.94555162]]
```

2.2.3 Đánh giá mô hình

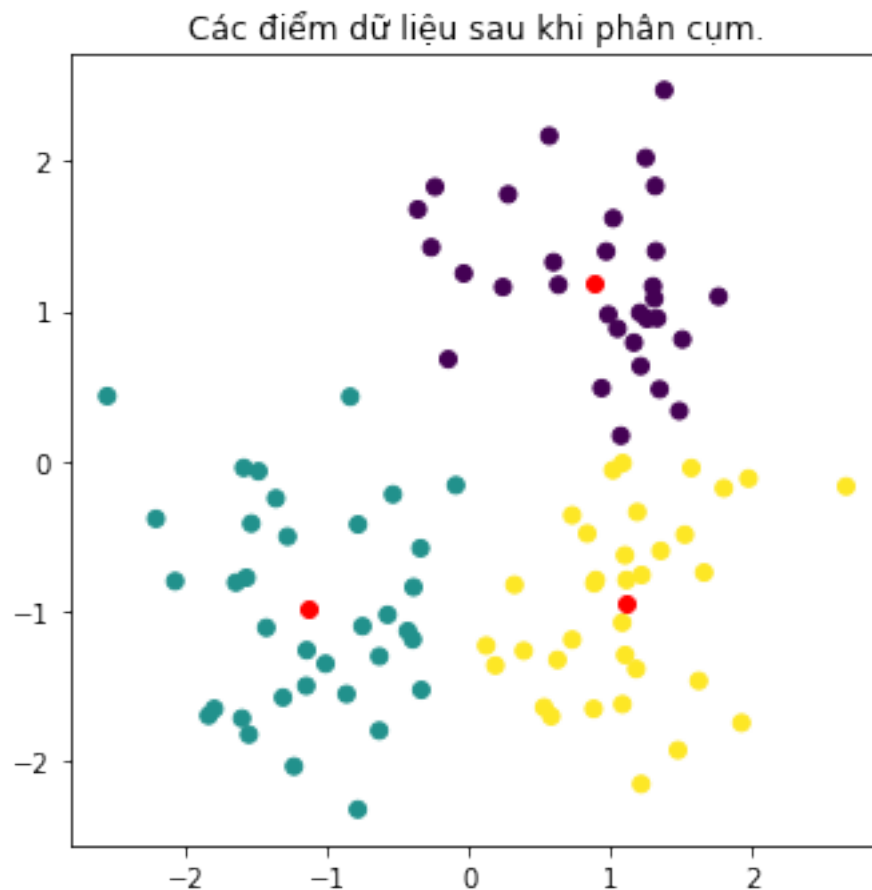
- Kiểm tra các điểm dữ liệu thuộc vào cụm nào
- Vẽ biểu đồ hiển thị, trong đó các điểm thuộc các cụm khác nhau sẽ có các màu khác nhau

```
[ ]: y_pred = k_mean_model.predict(X)
print("Kết quả dự đoán cho 5 mẫu dữ liệu đầu tiên trong tập data: \n")
print(y_pred[:5])

plt.figure(figsize=(12, 12))
plt.subplot(222)
plt.scatter(X[:, 0], X[:, 1], c=y_pred)
plt.scatter(centers[:, 0], centers[:, 1], c='red')
plt.title("Các điểm dữ liệu sau khi phân cụm.")
plt.show()
```

Kết quả dự đoán cho 5 mẫu dữ liệu đầu tiên trong tập data:

```
[0 1 0 0 1]
```



2.2.4 Bài tập 1

Yêu cầu: Thử nghiệm trường hợp dữ liệu sinh ra chỉ có 2 cụm nhưng huấn luyện K-means với các tham số $k = 3, 4, 5$ cụm

- Tự viết code sinh dữ liệu tương tự bên trên
- Xây dựng mô hình 3, 4, 5 cụm

Gợi ý: thay đổi tham số số cụm khi dựng giải thuật K-means

Kết quả phải ra được hình ảnh thể hiện đúng số tâm cụm và phân bố cụm.

```
[ ]: n_samples = 100
random_state = 170
center_points = 2                                # tâm cụm được chọn random

X, y = make_blobs(n_samples=n_samples, random_state=random_state,
                  ↪centers=center_points, cluster_std=0.6)
print("Số chiều dữ liệu: ", X.shape, y.shape)
print("5 điểm dữ liệu đầu tiên: \n", X[:5])

plt.figure(figsize=(12, 12))
plt.subplot(221)
plt.scatter(X[:, 0], X[:, 1], c='blue') # c là tham số chọn màu sắc, có thể
    ↪truyền vào string hoặc số id 1,2,3 ...
plt.title("Các điểm dữ liệu trước khi phân cụm. Số lượng: {}".format(n_samples))
```

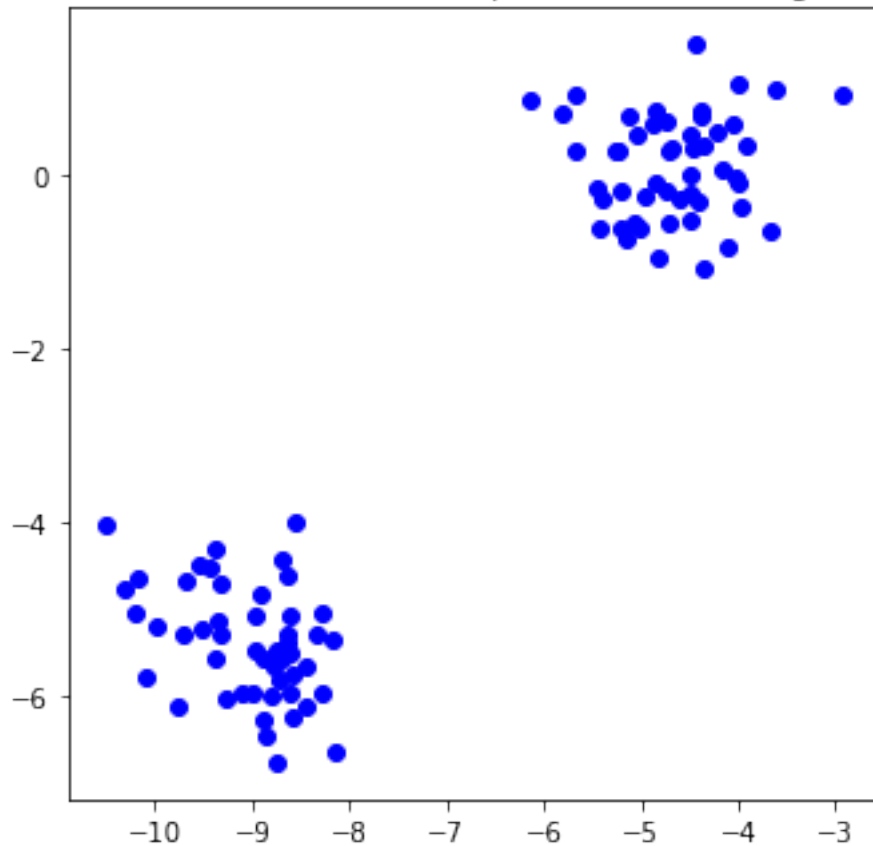
Số chiều dữ liệu: (100, 2) (100,)

5 điểm dữ liệu đầu tiên:

```
[[-8.87090181 -6.2923396 ]
 [-5.68131011  0.92989934]
 [-4.48237839 -0.2011274 ]
 [-9.70586418 -5.3021706 ]
 [-8.77808596 -5.67185751]]
```

```
[ ]: Text(0.5, 1.0, 'Các điểm dữ liệu trước khi phân cụm. Số lượng: 100')
```

Các điểm dữ liệu trước khi phân cụm. Số lượng: 100

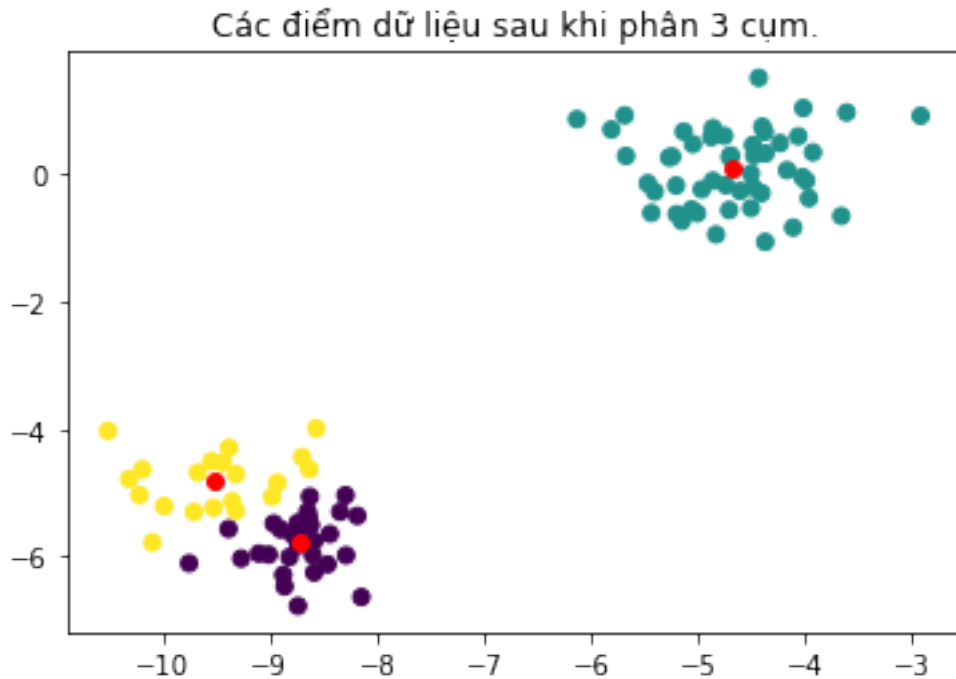


```
[ ]: # K-means với số cụm bằng 3
k_mean_model = KMeans(n_clusters=3, random_state=random_state)
k_mean_model.fit(X)

centers = np.array(k_mean_model.cluster_centers_) # cluster_centers_: là thuộc tính lưu trữ các tâm cụm sau khi training
y_pred = k_mean_model.predict(X)

plt.subplot(111)
plt.scatter(X[:, 0], X[:, 1], c=y_pred)
plt.scatter(centers[:, 0], centers[:, 1], c='red')
plt.title("Các điểm dữ liệu sau khi phân 3 cụm.")
```

```
[ ]: Text(0.5, 1.0, 'Các điểm dữ liệu sau khi phân 3 cụm.')
```



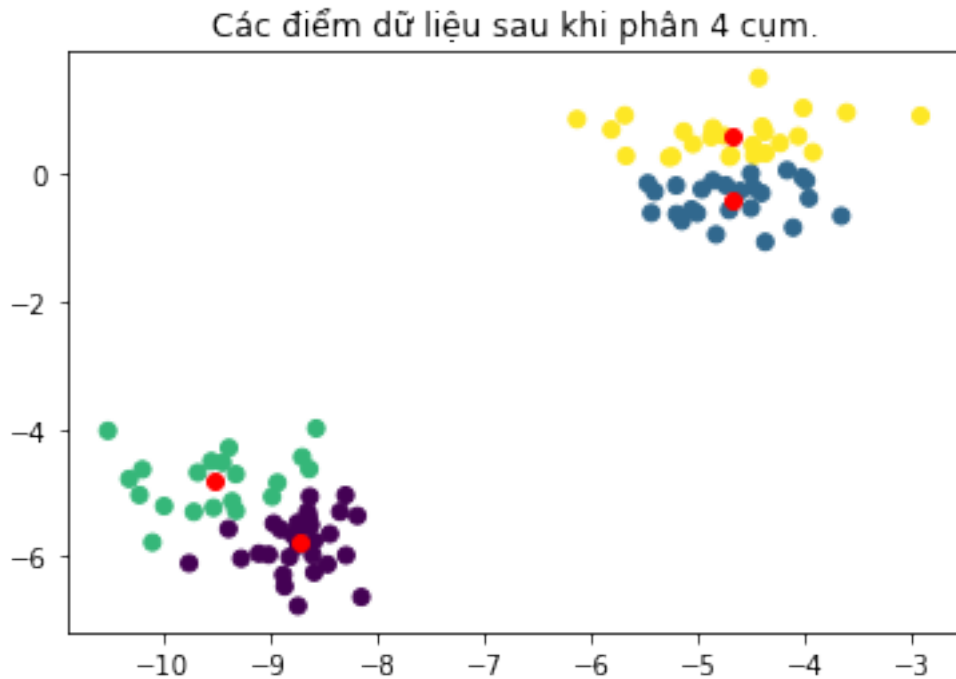
```
[ ]: # K-means với số cụm bằng 4
k_mean_model = KMeans(n_clusters=4, random_state=random_state)
k_mean_model.fit(X)

centers = np.array(k_mean_model.cluster_centers_) # cluster_centers_: là thuộc tính lưu trữ các
# tâm cụm sau khi training

y_pred = k_mean_model.predict(X)

plt.subplot(111)
plt.scatter(X[:, 0], X[:, 1], c=y_pred)
plt.scatter(centers[:, 0], centers[:, 1], c='red')
plt.title("Các điểm dữ liệu sau khi phân 4 cụm.")
```

```
[ ]: Text(0.5, 1.0, 'Các điểm dữ liệu sau khi phân 4 cụm.')
```



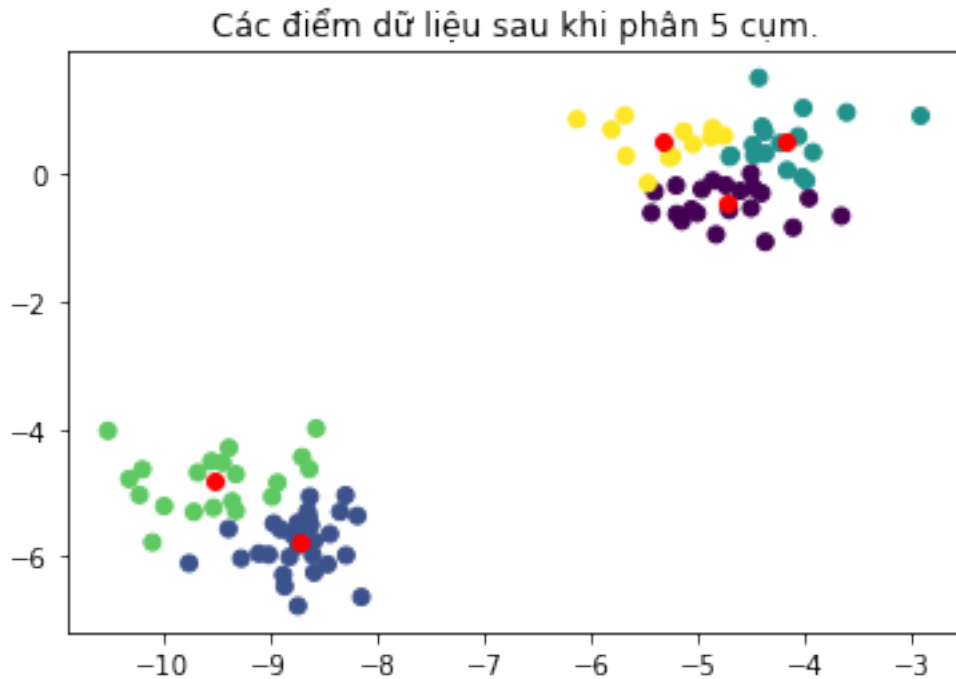
```
[ ]: # K-means với số cụm bằng 5
k_mean_model = KMeans(n_clusters=5, random_state=random_state)
k_mean_model.fit(X)

centers = np.array(k_mean_model.cluster_centers_) # cluster_centers_: là thuộc tính lưu trữ các
                                                    # tâm cụm sau khi training

y_pred = k_mean_model.predict(X)

plt.subplot(111)
plt.scatter(X[:, 0], X[:, 1], c=y_pred)
plt.scatter(centers[:, 0], centers[:, 1], c='red')
plt.title("Các điểm dữ liệu sau khi phân 5 cụm.")

plt.show()
```

2.3 Bài toán 2: Ứng dụng Kmeans cho nén ảnh

- Đặt vấn đề:
 - Muốn xây dựng 1 hệ thống nén dữ liệu hình ảnh
 - Có thể tùy chỉnh được độ sắc nét, giảm kích thước bộ nhớ, nhưng không làm sai lệch quá nhiều dưới mắt nhìn.
- Giải pháp
 - Sử dụng giải thuật K-means, tự động phân cụm các điểm ảnh, giới hạn số lượng màu để giảm kích thước ảnh
 - Mỗi điểm ảnh sẽ được quy về 1 cụm nào đó, mang giá trị màu bằng màu của tâm cụm.

2.3.1 Thư viện sử dụng - hỗ trợ hình ảnh

```
[ ]: from skimage import io
from sklearn.cluster import KMeans
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as image
from IPython.core.display import Image, display #Hiển thị ảnh
```

2.3.2 Đọc dữ liệu hình ảnh

- Mỗi điểm ảnh là 1 mẫu quan sát
- Phân cụm tập dữ liệu (tập các điểm ảnh) về k nhãn

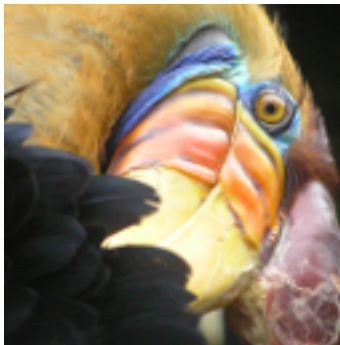
```
[ ]: path_img = 'bird_small.png'
Image(path_img, width=250, unconfined=True)
```

```
[ ]:
```



```
[ ]: path_img = 'bird_small.png'

display(Image(path_img, width=250, unconfined=True))
```



```
[ ]: img = io.imread(path_img)
img
```

```
[ ]: path_img = 'bird_small.png'

display(Image(path_img, width=250, unconfined=True))

img = io.imread(path_img)
print("Dữ liệu ảnh trước khi reshape:", img.shape)

img_shape = img.shape # 128x128x3
data_img = (img / 255.0).reshape(-1,img.shape[2]) # chuyển ma trận 128x128x3 về
↳ mảng 2 chiều, giữ lại chiều .shape[2]

print("Số chiều của dữ liệu hình ảnh: ", data_img.shape)
```

```
print("Tổng số điểm ảnh là: ", data_img.shape[0])
print("Mỗi điểm ảnh có số chiều = ", data_img.shape[1])
```



Dữ liệu ảnh trước khi reshape: (128, 128, 3)
 Số chiều của dữ liệu hình ảnh: (16384, 3)
 Tổng số điểm ảnh là: 16384
 Mỗi điểm ảnh có số chiều = 3

```
[ ]: img.shape
```

```
[ ]: (128, 128, 3)
```

```
[ ]: data_img.shape
```

```
[ ]: (16384, 3)
```

```
[ ]: a = np.array([[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]], [[13, 14, 15],  

    ↪ [16, 17, 18]])  

    a.shape
```

```
[ ]: (3, 2, 3)
```

```
[ ]: a.reshape(-1, 3)
```

```
[ ]: array([[ 1,  2,  3],  

    [ 4,  5,  6],  

    [ 7,  8,  9],  

    [10, 11, 12],  

    [13, 14, 15],  

    [16, 17, 18]])
```

2.3.3 Xây dựng mô hình kmean để nén ảnh

- Số lượng cụm chính là số lượng màu ta giữ lại
- Số lượng cụm càng nhỏ thì kích thước ảnh cho ra càng nhỏ

```
[ ]: n_color = 10
      k_mean_model = KMeans(n_clusters=n_color)
```

Huấn luyện mô hình

```
[ ]: k_mean_model.fit(data_img)
```

```
[ ]: KMeans(n_clusters=10)
```

```
[ ]: # Hiển thị một số thông tin đã học của mô hình
      print("Số chiều của tâm cụm: ", k_mean_model.cluster_centers_.shape)
      print(k_mean_model.cluster_centers_)
      print(k_mean_model.labels_[0:20])
```

```
Số chiều của tâm cụm: (10, 3)
[[0.86231668 0.69520561 0.42362336]
 [0.09711114 0.1042277 0.09435887]
 [0.47524466 0.43347112 0.44338103]
 [0.97036443 0.93594302 0.79803499]
 [0.48720222 0.36193737 0.22141937]
 [0.24372436 0.22561487 0.21985492]
 [0.74394647 0.53433707 0.25326797]
 [0.67681521 0.59349376 0.51435532]
 [0.90544752 0.7933047 0.64511694]
 [0.5540016 0.6744898 0.81887755]]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
```

2.3.4 Hiển thị ảnh mới sau khi nén

```
[ ]: # k_mean_model.labels_: chứa nhãn của tất cả các điểm ảnh
      # k_mean_model.cluster_centers_: chứa các tâm cụm.
      # new_arr = arr1[index]
      img128=k_mean_model.cluster_centers_[k_mean_model.labels_]

      print(img128.shape)

      # chuẩn hoá lại kích thước ảnh theo chiều dài, rộng ban đầu
      img128=np.reshape(img128, img_shape)
      print(img128.shape)
      image.imsave('img128.png', img128)
```

```
(16384, 3)
(128, 128, 3)
```

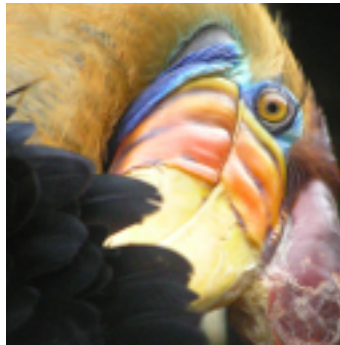
```
[ ]: # hiển thị kích thước hình ảnh trước và sau khi nén
      import os
      print('Size of compressed image: ' + str(os.path.getsize('img128.png')) + ' KB')
```

```
print('Size of original image: ' + str(os.path.getsize('bird_small.png')) + '␣  
↪KB')
```

Size of compressed image: 7245 KB

Size of original image: 33031 KB

```
[ ]: from IPython.core.display import Image, display  
  
#Save image  
display(Image('img128.png', width=250, unconfined=True))  
display(Image(path_img, width=250, unconfined=True))
```



2.3.5 Bài tập 2

Yêu cầu: Nén ảnh trên thành ảnh có số màu < 5 và kiểm tra

Gợi ý: thay đổi tham số “số cụm” khi xây dựng K-means

```
[ ]: n_color = 3  
k_mean_model = KMeans(n_clusters=n_color)  
k_mean_model.fit(data_img)
```

```
[ ]: KMeans(n_clusters=3)
```

```
[ ]: # Hiển thị một số thông tin đã học của mô hình
print("Số chiều của tâm cụm: ", k_mean_model.cluster_centers_.shape)
print(k_mean_model.cluster_centers_)
print(k_mean_model.labels_[0:20])
```

```
Số chiều của tâm cụm: (3, 3)
[[0.64101374 0.51229053 0.35837297]
 [0.15439697 0.14866772 0.1363279 ]
 [0.89002949 0.80491521 0.64769657]]
[2 2 2 2 2 2 2 2 2 0 2 2 2 2 2 2 0 2 2 2]
```

```
[ ]: # k_mean_model.labels_: chứa nhãn của tất cả các điểm ảnh
# k_mean_model.cluster_centers_: chứa các tâm cụm.
#new_arr = arr1[index]
img128=k_mean_model.cluster_centers_[k_mean_model.labels_]

print(img128.shape)

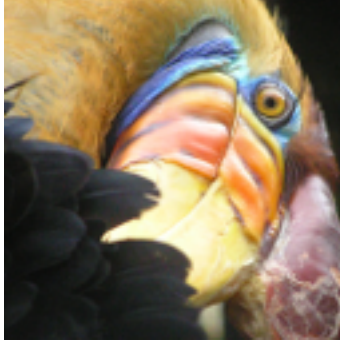
# chuẩn hoá lại kích thước ảnh theo chiều dài, rộng ban đầu
img128=np.reshape(img128, img_shape)
print(img128.shape)
image.imsave('img128.png', img128)
```

```
(16384, 3)
(128, 128, 3)
```

```
[ ]: from IPython.core.display import Image, display

#Save image
display(Image('img128.png', width=250, unconfined=True))
display(Image(path_img, width=250, unconfined=True))
```





2.4 Bài tập bổ sung

Làm sao để biết số cụm bằng bao nhiêu là phù hợp với dữ liệu ?

Sử dụng tiêu chuẩn inertia để đánh giá: inertia được tính bằng tổng bình phương khoảng cách của các mẫu đến trung tâm cụm gần nhất của chúng

Một mô hình tốt là mô hình có chỉ số inertia nhỏ và số lượng cụm bé. Vấn đề: số cụm càng lớn thì inertia càng nhỏ. Chọn số cụm hợp lý bằng cách: chọn điểm mà độ giảm inertia bắt đầu chậm lại.

2.4.1 Với dữ liệu các điểm ban đầu

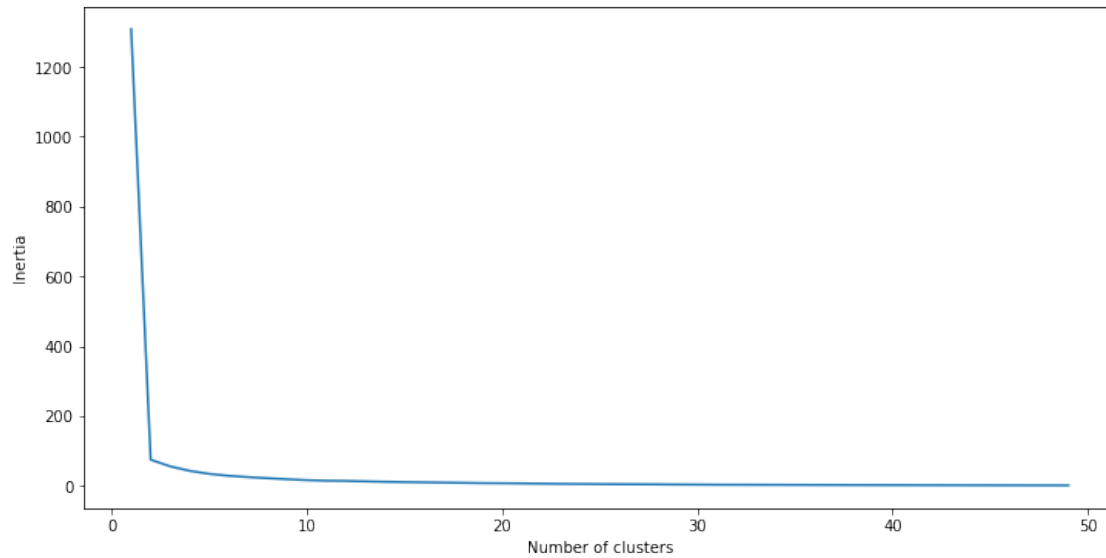
```
[ ]: k_mean_model.inertia_
```

```
[ ]: 70.3265626999078
```

```
[ ]: import pandas as pd
SSE = []
for cluster in range(1,50):
    kmeans = KMeans(n_clusters = cluster, random_state=random_state)
    kmeans.fit(X)
    SSE.append(kmeans.inertia_)

# converting the results into a dataframe and plotting them
frame = pd.DataFrame({'Cluster':range(1,50), 'SSE':SSE})
plt.figure(figsize=(12,6))
plt.plot(frame['Cluster'], frame['SSE'])
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')
```

```
[ ]: Text(0, 0.5, 'Inertia')
```

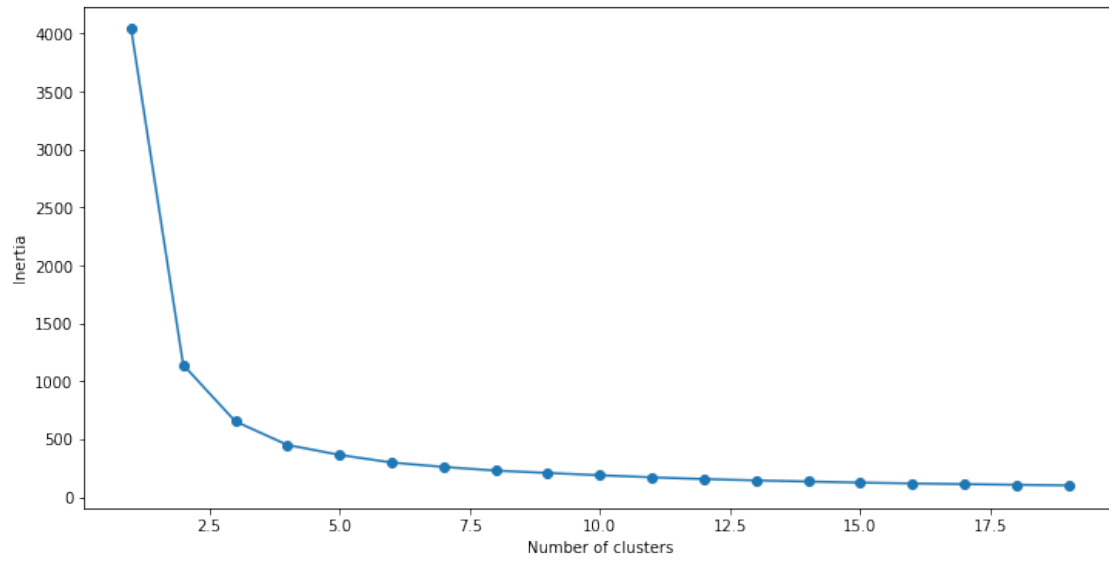


2.4.2 Với dữ liệu ảnh

```
[ ]: SSE = []
    for cluster in range(1,20):
        kmeans = KMeans(n_clusters = cluster, random_state=random_state)
        kmeans.fit(data_img)
        SSE.append(kmeans.inertia_)

    # converting the results into a dataframe and plotting them
    frame = pd.DataFrame({'Cluster':range(1,20), 'SSE':SSE})
    plt.figure(figsize=(12,6))
    plt.plot(frame['Cluster'], frame['SSE'], marker='o')
    plt.xlabel('Number of clusters')
    plt.ylabel('Inertia')
```

```
[ ]: Text(0, 0.5, 'Inertia')
```

[]: