CASE STUDY 4

**Bài toán: “Gom cụm hình ảnh lá cây” dùng Folio dataset**



***Môn học - Lớp:***

**MÁY HỌC - CS114.J21.KHTN**

***Giảng viên:***

**HUỲNH THỊ THANH THƯƠNG**

***Sinh viên:***

**17520689 - NGUYỄN THÙY LINH**

**17520015 - NGUYỄN HỮU PHONG**

Nội dung

[**I - GIỚI THIỆU CHUNG** 1](#_Toc12443992)

[**II - XỬ LÝ DỮ LIỆU** 2](#_Toc12443993)

[**1 - Dữ liệu gốc ban đầu** 2](#_Toc12443994)

[**2 - Tiền xử lý dữ liệu:** 2](#_Toc12443995)

[**3 – Feature Extraction (Trích xuất đặc trưng):** 5](#_Toc12443996)

[**III – CLUSTERING** 6](#_Toc12443997)

[**1 – K mean** 6](#_Toc12443998)

[**2 – Hierarchical agglomerative clustering - HAC** 7](#_Toc12443999)

[**3 – Density based:** 7](#_Toc12444000)

[**IV – ĐÁNH GIÁ CHẤT LƯỢNG GOM CỤM** 8](#_Toc12444001)

[**V – NHẬN XÉT** 9](#_Toc12444002)

# **I - GIỚI THIỆU CHUNG**

**Folio dataset:**

Bao gồm 32 loài thực vật (species), mỗi loài thu thập 20 hình lá của chúng (photos of leaves).Tổng cộng có 637 hình. Hình chụp từ các cây trong khuôn viên của đại học University of Mauritius và khu vực lân cận. Lá được đặt trên nền trắng, được chụp trong điều kiện lý tưởng.

Tên của 32 loài thực vật (List of plant species):

|  |  |  |  |
| --- | --- | --- | --- |
| Beaumier du perou | Eggplant | Fruitcitere | Guava |
| Hibiscus | Betel | Rose | Chrysanthemum |
| Ficus | Duranta gold | Ashanti blood | Bitter Orange |
| Coeur Demoiselle | Jackfruit | Mulberry Leaf | Pimento |
| Pomme Jacquot | Star Apple | Barbados Cherry | Sweet Olive |
| Croton | Thevetia | Vieux Garcon | Chocolate tree |
| Carricature plant | Coffee | Ketembilla | Chinese guava |
| Lychee | Geranium | Sweet potato | Papaya |

Download từ trang web: http://archive.ics.uci.edu/ml/machine-learning-databases/00338/

**Bài toán đặt ra:**

Sử dụng các phương pháp gom cụm: ***K-means***, ***Hierachical agglomerative*** và ***Density based*** để gom dữ liệu hình ảnh về các loại lá cây và đánh giá chất lượng gom cụm.

# **II - XỬ LÝ DỮ LIỆU**

# **1 - Dữ liệu gốc ban đầu**

Bao gồm 12 folder, mỗi folder gồm 20 ảnh, riêng Mulberry Leaf có 19 ảnh, Eggplant có 18 ảnh. Mặc dù được chụp trong điều kiện lý tưởng là nền trắng và không có các vật nhiễu, nhưng thời điểm chụp khác nhau, dẫn đến độ sáng, tương phản, … của mỗi bức ảnh không thể giống nhau, do đó, nền nói chung vẫn không đồng nhất. Mặt khác, khoảng cách chụp không giống nhau nên vùng lá trong bức ảnh cái to cái nhỏ. Kích thước của mỗi ảnh cũng không giống nhau.

Vì những lí do trên, trước khi phân cụm, ta phải tiền xử lí dữ liệu trước.

# **2 - Tiền xử lý dữ liệu:**

**BƯỚC 1: Đồng nhất nền (màu đen)**

Dữ liệu vào: 1 ảnh lá cây RGB với nền không đồng nhất

Dữ liệu ra: 1 ảnh lá cây có nền màu đen, vùng lá có màu như ảnh gốc

**BƯỚC 2: Đồng nhất kích thước**

Dữ liệu vào: Ảnh lá cây đã được đồng nhất nền màu đen nhưng kích thước chưa đồng nhất

Dữ liệu ra: Ảnh lá cây đã được đồng nhất về màu nền và kích thước



**Sources code:**

|  |
| --- |
| import cv2  import glob  import os  import numpy as np |
| path = 'Folio\\*\\*.jpg'  folder = 'Mixed Folio'  if not os.path.exists(folder):  os.makedirs(folder)  folder\_path = folder + '\\*\\*.jpg'  dataset = []  feature = [] |
| def convert3D(bin\_image):  \_3D = []  for row in bin\_image:  new = [[x] \* 3 for x in row]  \_3D.append(new)  return np.array(\_3D) |
| def extractROI(image):  # resize  max\_dimension = max(image.shape)  scale = 700 / max\_dimension  image = cv2.resize(image, None, fx=scale, fy=scale)  # smooth image  image\_blur = cv2.GaussianBlur(image, (7, 7), 0)  # grayscale  gray = cv2.cvtColor(image\_blur, cv2.COLOR\_BGR2GRAY)  # binary  if gray[gray.shape[0] // 2][gray.shape[1] // 2] > 95:  \_, bin\_img = cv2.threshold(gray, 190, 1, cv2.THRESH\_BINARY\_INV)  else:  \_, bin\_img = cv2.threshold(gray, 130, 1, cv2.THRESH\_BINARY\_INV)  # dilation  kernel = np.ones((4, 1), np.uint8)  img\_dilation = cv2.dilate(bin\_img, kernel, iterations=1)  img\_dilation = convert3D(img\_dilation)  # roi  roi = image\*img\_dilation  return roi |
| def preprocess(path):  max\_width = 0  # resize + findmax width  for image\_path in glob.glob(path):  head, name = os.path.split(image\_path)  \_, subfolder = os.path.split(head)  dir = os.path.join(folder, subfolder)  if not os.path.exists(dir):  os.makedirs(dir)  image = cv2.imread(image\_path)  image = extractROI(image.copy())  cv2.imwrite(os.path.join(dir, name), image)  # find max width  if image.shape[1] > max\_width:  max\_width = image.shape[1]  # padding  for file in glob.glob(folder\_path):  img = cv2.imread(file)  margin = (max\_width - img.shape[1])//2  pad = np.zeros([700, max\_width, 3])  pad[:, margin:margin+img.shape[1]] = img  cv2.imwrite(file, pad) |
| def load\_data(folder\_path):  for file in glob.glob(folder\_path):  img = cv2.imread(file)  dataset.append(img) |

# **3 – Feature Extraction (Trích xuất đặc trưng):**

**Color feature:**

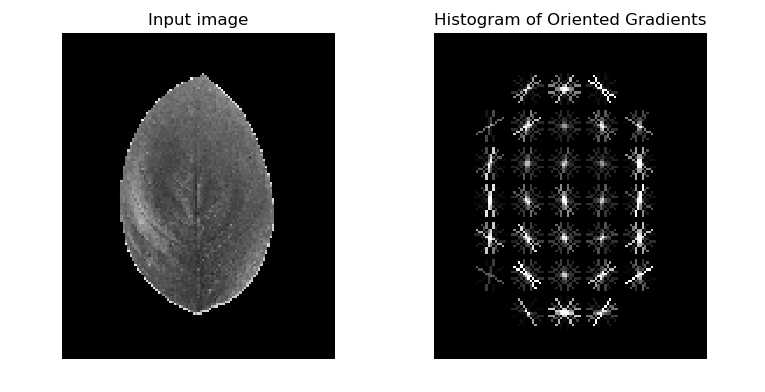
B1/ Chuyển ảnh RGB sang ảnh HSV

B2/ Từ kênh H, ta tính kì vọng theo công thức:

**Shape feature:**

B1/ Chuyển ảnh RGB sang ảnh Gray

B2/ Sử dụng hàm skimage.feature.hog để trích xuất shape feature:



**Source code:**

|  |
| --- |
| from skimage.feature import hog |
| def featureExtr():  for img in dataset:  # color - hsv  hsv = cv2.cvtColor(img, cv2.COLOR\_RGB2HSV)  c = sum(sum(hsv[:, :, 0]))/(hsv.shape[0]\*hsv.shape[1])  # shape - hog  gray = cv2.cvtColor(img, cv2.COLOR\_RGB2GRAY)  h, \_ = hog(gray, orientations=8, pixels\_per\_cell=(16, 16),  cells\_per\_block=(1, 1), visualize=True, multichannel=False)  np.insert(h, 0, c)  feature.append(h) |

# **III – CLUSTERING**

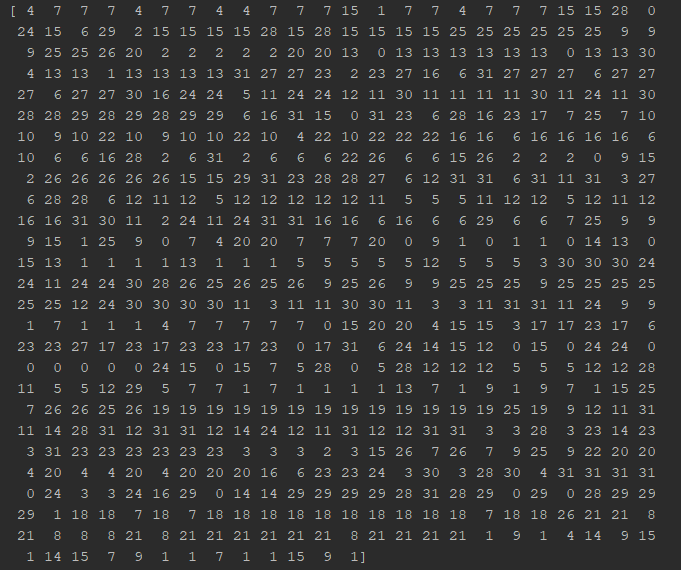
# **1 – K mean**

**Sources code:**

|  |
| --- |
| from sklearn.cluster import KMeans |
| kmeans = KMeans(n\_clusters=32, random\_state=0).fit(feature) |

**Kết quả:**

|  |
| --- |
| print(kmeans.labels\_) |



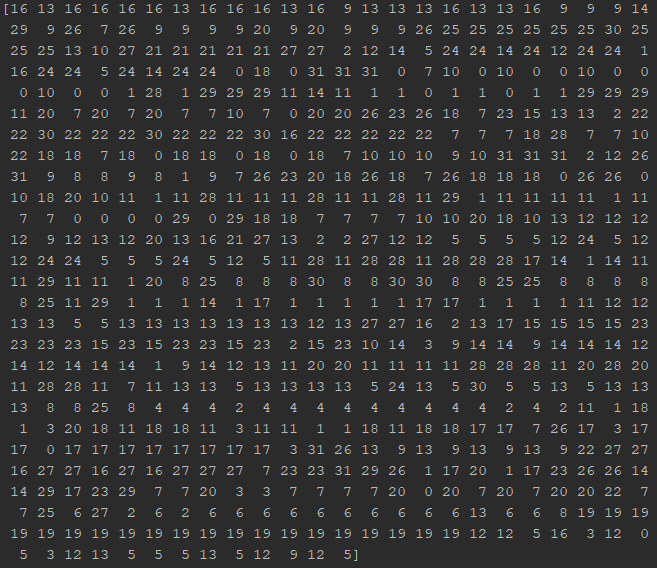
# **2 – Hierarchical agglomerative clustering - HAC**

**Sources code:**

|  |
| --- |
| from sklearn.cluster import AgglomerativeClustering as AC |
| hac = AC(n\_clusters=32).fit(feature) |

**Kết quả:**

|  |
| --- |
| print(hac.labels\_) |



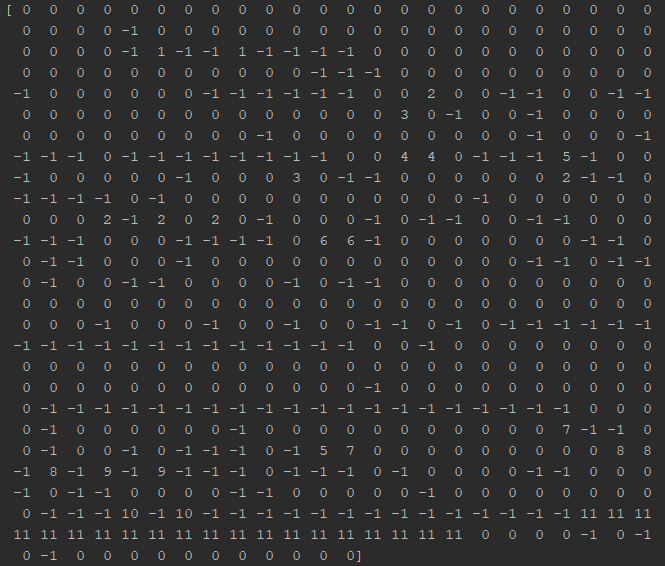
# **3 – Density based:**

**Sources code:**

|  |
| --- |
| from sklearn.cluster import DBSCAN |
| dbscan = DBSCAN(eps=12, min\_samples=2).fit(feature) |

**Kết quả:**

|  |
| --- |
| print(dbscan.labels\_) |

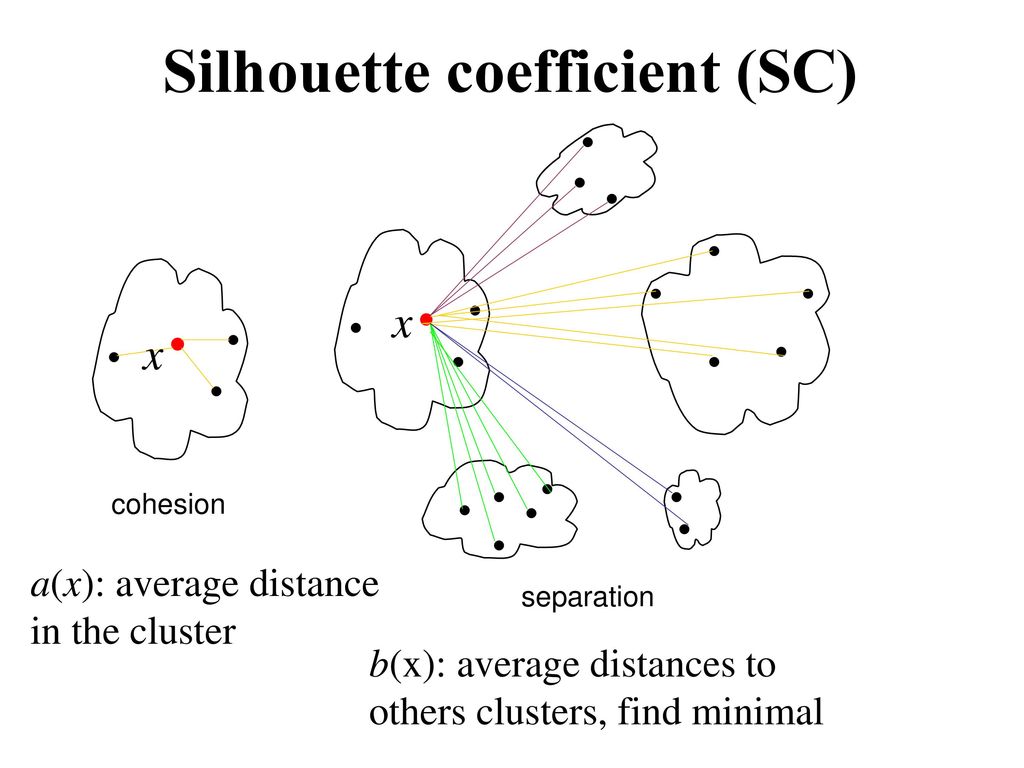


# **IV – ĐÁNH GIÁ CHẤT LƯỢNG GOM CỤM**

Trong bài báo cáo này, chúng em sử dụng phương pháp Silhouette Coefficient để đánh giá chất lượng gom cụm của cả 3 phương pháp K-means, HAC, DBSCAN.

Silhouette Coefficient là một phương pháp đánh giá chất lượng gom cụm nội tại. Với một bộ dữ liệu D, gồm n điểm dữ liệu, chia D thành k cụm: C1, C2, … Ck. Với mỗi điểm dữ liệu x, ta tính 2 thông số sau:

* a(x): khoảng cách trung bình giữa x và các điểm dữ liệu khác trong cùng một cụm.
* b(x): khoảng cách trung bình giữa x và các điểm dữ liệu khác trong cụm gần nhất với cụm chứa x.



Silhouette Coeficient s(x) cho mỗi điểm dữ liệu đó là:

Với:

* s(x) thuộc đoạn [-1, 1]
* s(x) càng lớn thì chất lượng gom cụm càng tốt và ngược lại

Cuối cùng, lấy giá trị trung bình của các s(x) thì ra được Silhouette Coefficient cần tìm.

Sử dụng hàm ***sklearn.metrics.silhouette\_score()*** ta tính được các kết quả như sau:

|  |
| --- |
| from sklearn import metrics |
| # output: 0.03986165752472994  print('silhouette\_score = ', metrics.silhouette\_score(feature, kmeans.labels\_, metric='euclidean'))  #output: 0.04852955681361866  print('silhouette\_score = ', metrics.silhouette\_score(feature, hac.labels\_, metric='euclidean'))  #output: -0.058876030540880364  print('silhouette\_score = ', metrics.silhouette\_score(feature, dbscan.labels\_, metric='euclidean')) |

# **V – NHẬN XÉT**

Thứ nhất, trong phần tiền xử lí, việc tách region of interest khỏi bachground cũng gặp một số vấn đề. Với một threshold đặt ra, vùng interest có thể bị lẫn vào background nên sau khi tách vùng đó bị mất; hoặc background bị lẫn vào vùng interest:

Thứ 2, chưa xử lí được những ảnh cùng loại, nhưng kích thước, vị trí vùng ROI chênh lệch quá lớn:

Thứ 3, trong phần trích xuất đặc trưng cũng thế. Có một vài loại lá có hình dạng giống nhau, gây khó khăn cho việc phân cụm:

Thứ 4, Dataset quá lớn và máy tính chúng em không được mạnh cho lắm nên qua các bước tiền xử lí, trích xuất đặc trưng, phân cụm, thời gian thực thi rất lâu (khoảng 30 phút). Do đó không thể thử được nhiều eps và min\_samples (DBSCAN) cho phù hợp. Cùng với các lí do trên thì kết quả gom cụm của cả 3 thuật toán K-means, Hierarchical Agglomerative, DBSCAN không được tốt cho lắm (thậm chí DBSCAN khá tệ).

Trong thời gian tới, chúng em sẽ cố gắng cải thiện việc phâm cụm để đạt được kết quả tốt cũng như để kịp ứng dụng vào các công việc liên quan.