TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN - ĐHQG TP.HCM

University of Information Technology

BÁO CÁO:

Bài thực hành 1:

Clustering

- Giảng viên hướng dẫn:

Lê Đình Duy

Mai Tiến Dũng

- Sinh viên thực hiện:

Hoàng Hữu Tín – 14520956

- Mã Lớp:

CS332.I11.KHTN

Tp. Hồ Chí Minh, Tháng 10 - 2017

Mục Lục

Mục lục hình

Mục lục bảng

I Lý Thuyết Clustering:

Phân cụm (clustering) là kỹ thuật nhóm các đối tượng trong tập dữ liệu thành các nhóm (cluster), các đối tượng trong cùng một nhóm tương đồng (gần giống) nhau về một khía cạnh nào đó hơn so với nhóm khác. Đây là một trong các kỹ thuật chính trong data mining, và được sử dụng trong khá nhiều lĩnh vực như: machine learning, pattern recognition, image analysis, information retrieval, data compression…

Có thể xem Clustering chưa phải là một thuật toán hoàn chỉnh vì kết quả của clustering thường là không thể xác định rõ ràng đúng hay sai một cách tuyệt đối. Kỹ thuật này thuộc lớp các phương pháp **Unsupervised Learning** trong Machine Learning.

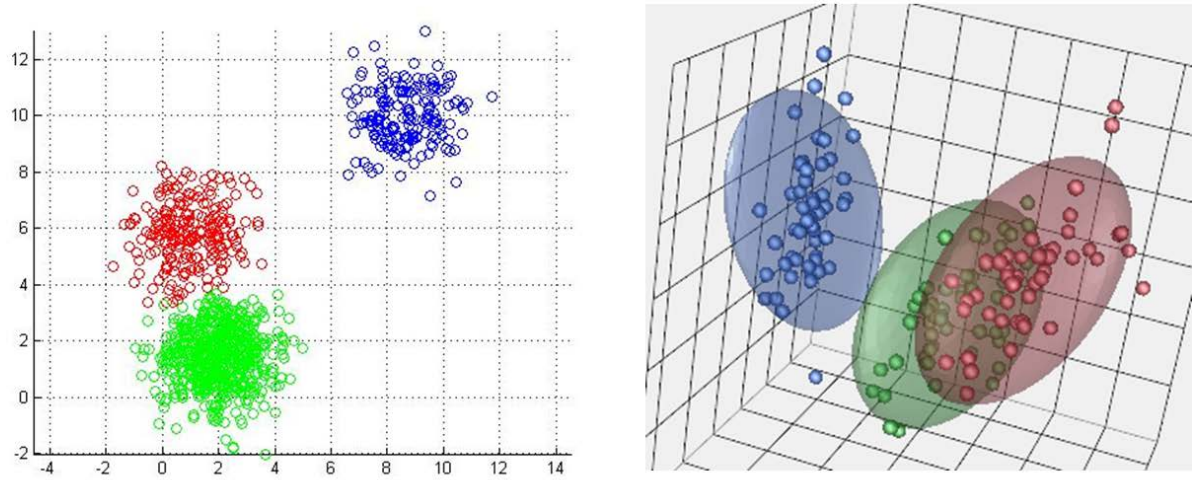
I.1 K-Means:

K-means là một trong các kỹ thuật được biết đến nhiều nhất trong bài toán Clustering, và là kỹ thuật đầu tiên sẽ được học nếu nhắc đến phân cụm. Ý tưởng của nó khá đơn giản, trực quan với thời gian chạy (trên các bộ dữ liệu nhỏ và trung bình) có thể chấp nhận được, kết quả thì cũng không quá tệ.

Thuật toán Standard K-means **[3]**:

---

1. *Clusters the data into k groups where k is predefined.*
2. *Select k points at random as cluster centers.*
3. *Assign objects to their closest cluster center according to the Euclidean distance function.*
4. *Calculate the centroid or mean of all objects in each cluster.*
5. *Repeat steps 2, 3 and 4 until the same points are assigned to each cluster in consecutive rounds.*



Hình I-1. K-Means trong không gian 2D và 3D

Phía trên là giải thuật của một thuật toán K-means tiêu chuẩn với điểm khởi tạo ngẫu nhiên và dùng Euclidean làm độ đo khoảng cách. Trên thực tế việc lựa chọn vị trí ngẫu nhiên k điểm ban đầu không phải là một cách hay → K-means++ ra đời.

K-means++ là một thuật toán chọn k điểm ban đầu cho K-means **[4]**:

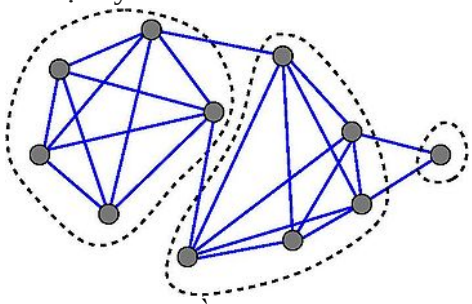
1. Choose one center uniformly at random from among the data points.
2. For each data point x, compute D(x), the distance between x and the nearest center that has already been chosen.
3. Choose one new data point at random as a new center, using a weighted probability distribution where a point x is chosen with probability proportional to D(x)^2.
4. Repeat Steps 2 and 3 until k centers have been chosen.
5. Now that the initial centers have been chosen, proceed using standard *k*-means clustering.

❖ Ưu điểm K-means:
- Đơn giản dễ thực hiện.
- Trực quan dễ hiểu, dễ học.
- Thời gian chạy chấp nhận được trên dữ liệu nhỏ và trung bình.

❖ Nhược điểm K-means:
- Độ phức tạp lớn: $O(n^{dk+1})$, với n số điểm cần clustering và d là số chiều của dữ liệu.
- Thời gian chạy lâu trên bộ dữ liệu lớn.
- Nhạy cảm với các độ đo khoảng cách.
- Số cluster k phải định nghĩa trước, việc lựa chọn số k sai có thể cho kết quả khá tồi.
- Kết quả của K-means không ổn định, bị phụ thuộc vào khởi chọn vị trí k điểm ban đầu.

I.2 Spectral Clustering:

Spectral là một trong các phương pháp Clustering, nó sử dụng eigenvectors của ma trận tương đồng (similarity matrix) được dẫn xuất từ data. Ý tưởng chung của Spectral Clustering là xem việc clustering như một vấn đề phân tách đồ thị (graph partitioning): xem data points như các nodes của một đồ thị, liên kết nối 2 node là độ tương đồng của chúng; các clusters đạt được bằng cách phân tách đồ thị này.



Hình I-2. Clustering bằng cách Graph Partitioning

Thuật toán Spectral Clustering **[5]:**

By Ng, Jordan and Weiss

- Given a data set $S = \{s_1,... s_n\}$ to be clustered

1. Calculate the affinity matrix $\boldsymbol{A_{ij} = \exp(-\|s_i - s_j\|^2/2\sigma^2)}$, if $i \neq j$ and $A_{ii} = 0$ where σ^2 is the scaling parameter
2. Define $\boldsymbol{D}$ to be the diagonal matrix whose (i,i)-element is the sum of A's i-th row, and construct the matrix $\boldsymbol{L = D^{-1/2} A D^{-1/2}}$
3. Find k largest eigenvectors of L and form the matrix $\boldsymbol{X = [x_1 x_2 \ldots x_k]}$
4. Form the matrix Y from X by normalizing each of X's rows to have unit length, $\boldsymbol{Y_{ij} = X_{ij}/(\Sigma_j X_{ij}^2)^{1/2}}$
5. Treating each row of Y as a point, cluster them into k clusters via K-means or any other algorithm
6. Assign the original point s_i to cluster j if and only if row i of the matrix Y was assigned to cluster j

❖ Ưu điểm:
- Dễ thực hiện.
- Kết quả clustering tương đối tốt.
- Tương đối nhanh với bộ dữ liệu vài nghìn points.
- Không quá chú trọng vào data: cluster trực tiếp trên đồ thị theo độ tương đồng của các points.
- Quan tâm đến sự liên kết giữa các points hơn là sự lân cận của nó (như K-means) → chạy tốt trên các bộ dữ liệu not-well geometrically separated.

❖ Nhược điểm:
- Nhạy cảm với việc lựa chọn tham số: độ đo tương đồng, eigenvalue decomposition strategy
- Chi phí tính toán lớn cho large datasets.

I.3 DBSCAN:

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) là phương pháp clustering dựa trên mật độ của dữ liệu. Khác với 2 phương pháp trước, DBSCAN không cần biết trước số lượng cluster.

Ý tưởng DBSCAN:

- Giả sử có dataset với n-dimensional points.
- Với mỗi point lấy một n-dimensional sphere xung quanh điểm đó với bán kính ***epsilon***, tính số lượng của data point trong sphere.

- Nếu số lượng points trong sphere lớn hơn giá trị ***min_points***, ta đánh dấu tâm của sphere và các points nằm trong đó cùng thuộc một cluster. Tiếp tục thực hiện đệ quy mở rộng sphere cho các points trong đó.
- Nếu số lượng points trong hình cầu ít hơn ***min_points***, ta bỏ qua nó và tiếp tục thử với hình cầu có tâm là point khác.

❖ Ưu điểm:
- Không cần biết trước số lượng cluster.
- Độ phức tạp trung bình: O(n logn)
- Có thể tìm các cluster có hình dạng tùy ý, kể cả cluster được bao bọc hoàn toàn bởi cluster khác.
- Mạnh về xử lý outliers, không quá nhạy cảm với noise.

❖ Nhược điểm:
- Độ phức tạp lớn
- Phụ thuộc lớn vào tham số: ε, min_points và độ đo khoảng cách.
- Đối với các data có scale chưa biết trước, việc chọn được threshold ε không phải dễ dàng.
- Chạy không tốt trên high-dimensional datasets.

I.4 Agglomerative Hierarchical Clustering:

Agglomerative là một phương pháp bottom-up clustering, xem các clusters gồm các sub-clusters, và các sub-clusters đó tiếp tục có sub-clusters của chúng. Agglomerative thực hiện merge từng các sub-clusters và tiếp tục như vậy để move up trên cây có thứ bậc (hierarchy)

Ý tưởng chính của Agglomerative:

- Assign mỗi point là một cluster.
- Đánh giá khoảng cách giữa các cặp cluster (pair-wise distances)
- Tìm cặp cluster có distance nhỏ nhất, merge 2 cluster này.
- Update distances từ cluster mới merge được với các cluster còn lại.
- Lặp lại cho đến khi đã merge tất cả các cluster thành 1.

Muốn phân tách data ban đầu với số lượng cluster k đã biết trước chỉ cần chọn ra bậc trên Hierarchical tree với số nhánh bằng k. Các points trong các nhánh con của một nhánh thuộc cùng một cluster.

❖ Ưu điểm:
- Tổ chức dataset thành các cluster có thứ bậc, thuận tiện cho
- Một khi xây dựng xong Hierarchical tree, việc thay đổi số lượng cluster có thể dễ dàng thực hiện mà không cần chạy lại thuật toán.

❖ Nhược điểm:
- Độ phức tạp lớn $O(n^2 \log n)$

- Phụ thuộc vào distance metrics.
- Trong quá trình chạy thuật toán, không thể relocation lại các sub-clusters đã incorrectly merge trước đó.

I.5 Principal component analysis:

Principal component analysis (PCA) là một phương pháp nhận biết các kiểu mẫu (identifying patterns) trong data, thể hiện data theo một cách làm nổi bật sự tương đồng cũng như khác biệt trong data. Một trong những lợi ích chính của PCA là một khi đã tìm ra các patterns, chúng ta có thể nén data ví dụ như giảm số chiều của nó mà không làm mất quá nhiều thông tin.

Các bước chính của PCA:

1. Giả sử chúng ta có một 2-D data với hai chiều x, y.
2. Subtract the mean:
   a. Trừ các giá trị x cho giá trị trung bình mean $\bar{x}$
   b. Tương tự, trừ các giá trị y cho giá trị trung bình mean $\bar{y}$

   Ví dụ ta có data như sau:

Data =

x	y
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3.0
2.3	2.7
2	1.6
1	1.1
1.5	1.6
1.1	0.9

DataAdjust =

x	y
.69	.49
-1.31	-1.21
.39	.99
.09	.29
1.29	1.09
.49	.79
.19	-.31
-.81	-.81
-.31	-.31
-.71	-1.01

$$\bar{x} = 1.81;\ \bar{y} = 1.91$$

3. Calculate the covariance matrix:

$$cov = \begin{pmatrix} .616555556 & .615444444 \\ .615444444 & .716555556 \end{pmatrix}$$

4. Calculate the eigenvectors and eigenvalues of the covariance matrix:

$$eigenvalues = \begin{pmatrix} .0490833989 \\ 1.28402771 \end{pmatrix}$$

$$eigenvectors = \begin{pmatrix} -.735178656 & -.677873399 \\ .677873399 & -.735178656 \end{pmatrix}$$

5. Choosing components and forming a feature vector
   a. Nhìn vào eigenvectors và eigenvalues ta có thể thấy giá trị eigenvalues khá khác biệt. Thực tế, eigenvector với giá trị eigenvalue lớn nhất chính là principle component của dataset.
   b. Như vậy từ các eigenvectors có được từ covariance matrix ta sắp xếp chúng theo thứ tự giảm dần của eigenvalues → Các eigenvectors càng ở trên đầu càng có ý nghĩa nhất đối với dataset. Nếu chúng ta muốn giảm số chiều dữ liệu chỉ còn p chiều thì chỉ cần lấy p eigenvectors có giá trị eigenvalues lớn nhất.
   c. Tạo một feature vector chỉ gồm các eigenvectors cần giữ lại:

$$FeatureVector = (eig_1\ eig_2\ eig_3\ ....\ eig_n)$$

   Như ví dụ ở trên ta chỉ chọn một eigenvector có value lớn nhất vậy FeatureVector sẽ là:

$$\begin{pmatrix} -.677873399 \\ -.735178656 \end{pmatrix}$$

6. Deriving the new data set:
   Sau khi đã chọn được components (eigenvectors) cần giữ, để nhận được data mới ta đơn giản tính:

$$FinalData = RowFeatureVector \times RowDataAdjust$$

   Với RowFeatureVector là ma trận chuyển vị của FeatureVector để nhận được mỗi eigenvectors trên một hàng, eigenvector với value lớn nhất ở trên đầu. Tương tự với RowDataAdjust là ma trận chuyển vị của Data đã tính ở bước 2.

II Lý Thuyết Image Features:

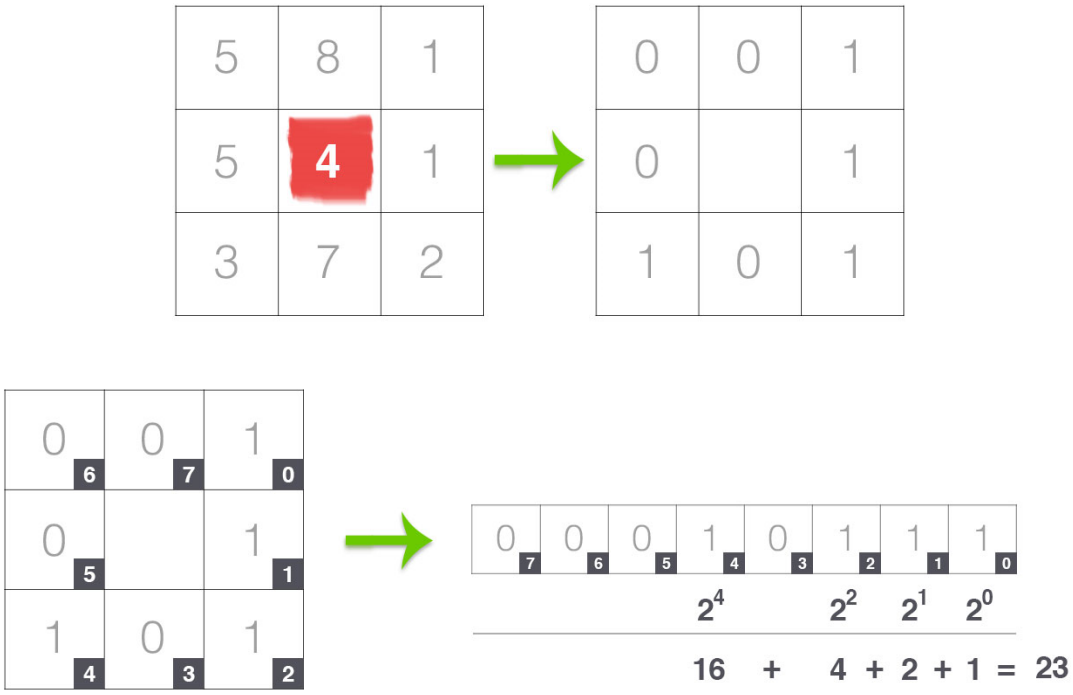
II.1 Pixel Intensity:

Các images thường được biểu diễn trên máy tính là một ma trận 2 chiều gồm các pixels. Mỗi pixel mang một giá trị (Pixel Intensity) thể hiện mức sáng của nó. Với kênh màu RGB mỗi pixels gồm 3 giá trị Red, Green, Blue với giá trị từ 0→255, để cho đơn giản ta thường chuyển một ảnh RGB thành một ảnh Grayscale với một channel duy nhất khi dùng đến pixel intensity.

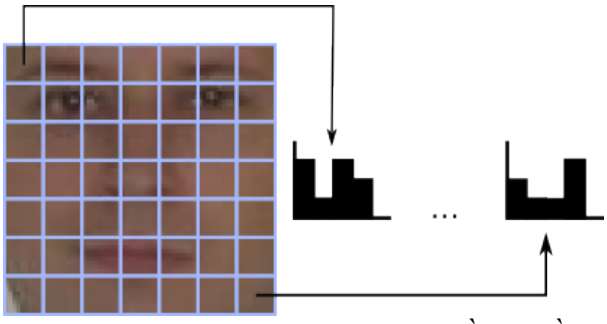
Ta có thể xem pixel intensity như một đặt trưng của ảnh: đơn giản ta sẽ chuyển một ảnh hai chiều thành một vectors một chiều với các giá trị là mức sáng của các pixels.

II.2 Local Binary Pattern:

Local Binary Pattern (LBP) là một texture descriptor, thực hiện bằng cách tính local representation of texture. Local representation được xây dựng bằng các so sánh mỗi pixel với các neighbors lân cận của nó.



Hình II-1. Cách tính LBP ***[9]***



Hình II-2. Tính LBP cho face recognition: Chia ảnh thành grid gồm nhiều cells, tính histogram của LBP cho mỗi cell, nối các histogram cho các cells này lại thành một feature vector cho ảnh.

Các bước để rút trích LBP cho một ảnh ***[9]***:

---

1. *Divide the examined window into cells (e.g. 16x16 pixels for each cell).*
2. *For each pixel in a cell, compare the pixel to each of its 8 neighbors (on its left-top, left-middle, left-bottom, right-top, etc.). Follow the pixels along a circle, i.e. clockwise or counter-clockwise.*
3. *Where the center pixel's value is greater than the neighbor's value, write "0". Otherwise, write "1". This gives an 8-digit binary number (which is usually converted to decimal for convenience).*
4. *Compute the histogram, over the cell, of the frequency of each "number" occurring (i.e., each combination of which pixels are smaller and which are greater than the center). This histogram can be seen as a 256-dimensional feature vector.*
5. *Optionally normalize the histogram.*
6. *Concatenate (normalized) histograms of all cells. This gives a feature vector for the entire window.*

- **Uniform LBP**: là một kỹ thuật cải tiến LBP để rút ngắn độ dài của feature vector. Ý tưởng này dựa trên việc quan sát thấy một vài binary patterns xuất hiện nhiều hơn trong texture images hơn các patterns khác. Một local binary pattern được xem là uniform nếu pattern bao gồm tối đa hai chuyển đổi 0-1 hay 1-0. Ví dụ: 000-1-0000 (2 chuyển đổi) là uniform, 00-11-0-1-00 (4 chuyển đổi) không phải uniform. Như vậy trong việc tính LBP Histogram, ta tạo mỗi bin cho mỗi giá trị uniform patterns và 1 bin duy nhất cho các giá trị non-uniform.

II.3 Facenet Embeddings:

Paper: **FaceNet: A Unified Embedding for Face Recognition and Clustering**

Authors: Florian Schroff, Dmitry Kalenichenko, James Philbin

Published in: Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on 17 June 2015

Cited by 831 (Google scholar)

PDF: https://arxiv.org/pdf/1503.03832.pdf

❖ Ý tưởng chính:

- Embedding được định nghĩa là một hàm $f(x)$ với giá trị nhận vào là ảnh x và cho ra feature trong không gian $\mathbb{R}^d$, để mà squared distance giữa tất cả ảnh của cùng một người thì nhỏ trong khi squared distance giữa 2 ảnh của 2 người khác nhau thì lớn.
- Facenet định nghĩa một Triplet Loss với motivation là: try to enforce a margin between each pair of faces from one person to all other faces
- Facenet sử dụng một deep convolutional network để training minimize Triplet loss, với layer đầu tiên là image và layer cuối cùng là embeddings (128-Dimension).

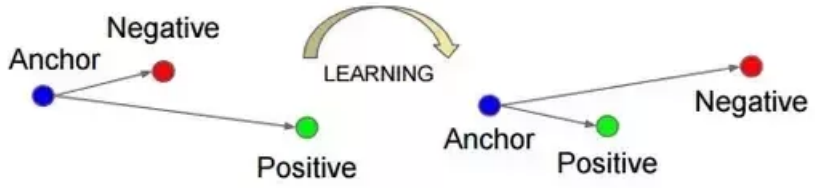


Figure 3. The **Triplet Loss** minimizes the distance between an *anchor* and a *positive*, both of which have the same identity, and maximizes the distance between the *anchor* and a *negative* of a different identity.

Triplet Loss

The triplet loss is trained on a series of triplets $\{x_a, x_p, x_n\}$ where x_a and x_p come from the same class and x_n comes from a different class. The goal of the triplet loss is to keep x_a closer to x_p than x_n. The triplet loss is formulated as:

$$L_{trp} = \sum_{a,p,n}^{N} \left[\|f(x_a) - f(x_p)\|_2^2 - \|f(x_a) - f(x_n)\|_2^2 + \alpha_{margin} \right]_+$$

Hình II-3. Triplet Loss Explanation

III Clustering Performance Evaluation:

Sau khi đã thực hiện Clustering xong, việc đánh giá các giải thuật này là khá cần thiết để có thể so sánh các phương pháp với nhau và chọn ra phương pháp tối ưu tùy theo bộ dữ liệu. Việc đánh giá một clustering algorithm thường không đơn giản như tính số lỗi sai hay precision và recall như ở một supervised classification algorithm.

III.1 Adjusted Rand index:

Adjusted Rand index (ARI) là độ đo chỉ sự tương đồng (similarity) giữa groundtruth và kết quả của clustering. ARI cho phép bỏ qua đổi trật tự của label (permutations): ví dụ mã label sau khi clustering thuộc cùng một cluster có thể khác với label cũng của cluster đó ở groundtruth.

Range [-1, 1]: Với 1 là điểm perfect; các kết quả bad sẽ có giá trị âm hay gần với 0.0

Cách tính Raw Rand Index (unadjusted):

$$\mathrm{RI} = \frac{a + b}{C_2^{n_{samples}}}$$

Với a: số lượng cặp elements trùng nhau trong groundtruth và kết quả clustering

b: số lượng cặp elements khác nhau trong groundtruth và kết quả clustering

Tính Adjusted Rand index:

$$\mathrm{ARI} = \frac{\mathrm{RI} - E[\mathrm{RI}]}{\max(\mathrm{RI}) - E[\mathrm{RI}]}$$

Với: E[RI]: expected RI of random labelings

III.2 Mutual Information based scores:

Mutual information là một độ đo đối xứng cho mức độ phụ thuộc (dependency) giữa clustering và groundtruth. Nó dựa trên khái niệm cluster *purity* – đánh giá chất lượng của một cluster C_i bởi p_i^j, số objects lớn nhất trong cluster C_i mà C_i có chung với M_j (groundtruth cho C_i) so sánh với tất cả groundtruth M

$$purity(C_i) = \frac{1}{|C_i|} max_j(p_i^j)$$

Mutual Information (MI) giữa kết quả cluster C và groundtruth M được tính dựa trên shared object membership:

$$MI(C, M) = \frac{1}{n} \sum_{i=1}^{k} \sum_{j=1}^{l} p_i^j \frac{log(\frac{p_i^j * n}{\sum_{a=1}^{k} p_a^j \sum_{b=1}^{l} p_l^b})}{log(k * l)}$$

$$= \frac{1}{n} \sum_{i=1}^{k} \sum_{j=1}^{l} t_{ij} \frac{log(\frac{t_{ij} * n}{t_{i.} * t_{.j}})}{log(k * l)}$$

Adjusted Mutual Information:

Tương tự **Rand Index** (ở trên), Mutual Information giữa 2 cluster có xu hướng tăng lên khi số lượng clusters cần đánh giá tăng lên (với cùng số lượng elements N)

⇨ Cần adjust Mutual Information theo expected mutual information

$$E[\mathrm{MI}(U,V)] = \sum_{i=1}^{|U|}\sum_{j=1}^{|V|}\sum_{n_{ij}=(a_i+b_j-N)^+}^{\min(a_i,b_j)} \frac{n_{ij}}{N}\log\left(\frac{N.n_{ij}}{a_i b_j}\right)\frac{a_i!b_j!(N-a_i)!(N-b_j)!}{N!n_{ij}!(a_i-n_{ij})!(b_j-n_{ij})!(N-a_i-b_j+n_{ij})!}$$

$$\mathrm{AMI} = \frac{\mathrm{MI} - E[\mathrm{MI}]}{\max(H(U),H(V)) - E[\mathrm{MI}]}$$

*Với H(U), H(V) tính theo **[11]***

Giá trị AMI trong khoảng [0,1]: với giá trị gần 0 nghĩa là clustering và groundtruth độc lập với nhau (independent), giá trị càng gần 1 nghĩa là chúng càng thống nhất (agreement)

III.3 Homogeneity, completeness and V-measure:

❖ **Homogeneity:**

Homogeneity được dùng để đánh giá tính đồng nhất: mỗi cluster chỉ bao gồm members của một groundtruth class.

$$h = 1 - \frac{H(C|K)}{H(C)}$$

Với H(C|K) là conditional entropy of the classes given the cluster assignments:

$$H(C|K) = -\sum_{c=1}^{|C|}\sum_{k=1}^{|K|}\frac{n_{c,k}}{n}\cdot\log\left(\frac{n_{c,k}}{n_k}\right)$$

Và H(C) là entropy of the classes:

$$H(C) = -\sum_{c=1}^{|C|}\frac{n_c}{n}\cdot\log\left(\frac{n_c}{n}\right)$$

Trong đó: n_c và n_k là số samples thuộc cluster c và cluster k

$n_{c,k}$ là số samples của class c (theo groundtruth) được assigned (bởi giải thuật) cho cluster k

❖ **Completeness:**

Completeness được dùng để đánh giá tính trọn vẹn: tất cả members của một groundtruth class được assigned vào cùng một cluster.

$$c = 1 - \frac{H(K|C)}{H(K)}$$

Như vậy có thể thấy:

homogeneity_score(a, b) == completeness_score(b, a)

❖ **V-measure:**

V-measure tương đương với mutual information (unadjusted) ở trên, nó có thể được dùng để đánh giá agreement giữa 2 assignments (clustering và groundtruth; hoặc giữa 2 kết quả của 2 giải thuật clusterings)

V-measure được tính bằng cách trung bình hài hòa giữa homogeneity và completeness:

$$v = 2 \cdot \frac{h \cdot c}{h + c}$$

Tất cả các độ đo ở trong phần III đều yêu cầu có groundtruth. Vì datasets trong bài thực hành này đề đã được label nên chúng ta sẽ sử dụng các phương pháp evaluation này. Với các dataset không có label sẵn, ta có thể đánh giá theo: Silhouette Coefficient, Calinski-Harabaz Index.

IV Nội dung thực hành:

Ngôn ngữ sử dụng: Python 3.6 (64-bit)

OS: Windows 10 x64

Gói thư viện chính: Anaconda 4.4 (gồm sklearn, skimage, matplotlib, numpy, pickle, …)

Các thư viện phụ:

+ dlib: https://pypi.python.org/pypi/dlib - Tìm Face landmarks

+ Facenet: https://github.com/davidsandberg/facenet - Tính Facenet embeddings

IV.1 Bài tập 1:

Thực hiện K-means clustering trên 2 phân bố Gaussian ngẫu nhiên.

Hàm tạo 2 random Gaussian:

```
sklearn.datasets.make_blobs(n_samples , centers , random_state)
```

Với `n_samples` : Số lượng points

`Centers` : Số clusters

`random_state` : giá trị để sinh ra trạng thái ngẫu nhiên của blobs

IV.2 Bài tập 2:

Thực hiện Kmeans, Spectral Clustering, DBSCAN, Agglomerative Clustering

Dataset UCI ML hand-written digits datasets:

http://archive.ics.uci.edu/ml/datasets/optical+recognition+of+handwritten+digits

Hàm load dataset của sklearn:

```
sklearn.datasets.load_digits(n_class=10, return_X_y=False)
```

Image size: 8x8

#Samples: 1797

#clusters: 10

Features: Pixel Intensity (16-Dimensional vectors)

Các hàm Sklearn Clustering và tham số sử dụng:

```
cluster.KMeans(init='k-means++', n_clusters=n_digits, n_init=10).fit(reduced_data)

cluster.SpectralClustering(affinity="nearest_neighbor",
                                        n_clusters=n_digits,
                                        eigen_solver='arpack').fit(reduced_data)

cluster.DBSCAN(eps=0.3, min_samples=10).fit(reduced_data)

cluster.AgglomerativeClustering(linkage="ward", n_clusters=n_digits).fit(reduced_data)
```

❖ Tiến trình thực hiện:

File “Bai_Tap_2.py”:

+ Load Dataset

+ Thực hiện PCA – để reduce dataset xuống còn 2 dimension.

+ Thực hiện các phương pháp Clustering.

+ Đánh giá các phương pháp Clustering theo các độ đo ở Phần III.

+ Visualize kết quả bằng matplotlib.

File "Bai_Tap_2_NonPCA.py":

+ Load Dataset

+ **Không** thực hiện PCA trước khi chạy clustering.

+ Thực hiện các phương pháp Clustering.

+ Đánh giá các phương pháp Clustering theo các độ đo ở Phần III.

+ Thực hiện PCA – 2D và dùng nó để visualize kết quả bằng matplotlib.

IV.3 Bài tập 3:

Thực hiện Kmeans, Spectral Clustering, DBSCAN, Agglomerative Clustering cho Dataset Labeled Faces in the Wild:

http://vis-www.cs.umass.edu/lfw/

Hàm load dataset của sklearn:

`sklearn.datasets.fetch_lfw_people(`*min_faces_per_person*`)`

Image size: 62x47 (default resize = 0.5)

#Samples: 13233

#clusters: 5749

Features: Local Binary Histogram (Xem Mục **II.2**)

Các hàm Sklearn Clustering và tham số sử dụng:

```
cluster.KMeans(init='k-means++', n_clusters=n_digits, n_init=10).fit(reduced_data)

cluster.SpectralClustering(affinity="nearest_neighbor",
                                        n_clusters=n_digits,
                                        eigen_solver='arpack').fit(reduced_data)

cluster.DBSCAN(eps=0.3, min_samples=10).fit(reduced_data)

cluster.AgglomerativeClustering(linkage="ward", n_clusters=n_digits).fit(reduced_data)
```

❖ Tiến trình thực hiện:

File “Bai_Tap_3_all.py”:

+ Load Dataset – lấy tất cả 13233 ảnh, 5749 clusters

+ Thực hiện **PCA** – để reduce dataset xuống còn 2 dimension.

+ Chỉ thực hiện các phương pháp K-means Clustering, các phương pháp còn lại có thời gian chạy quá lâu (> 3 tiếng – trên cấu hình một máy laptop bình thường) nên tạm thời bỏ qua cho đến khi kiếm được một máy cấu hình mạnh hơn.

+ Đánh giá Clustering theo các độ đo ở Phần III.

+ **Không** thực hiện Visualize kết quả vì số clusters quá lớn.

File “Bai_Tap_3_visualize.py”:

+ Load Dataset - *min_faces_per_person = 60,* chỉ lấy các ảnh của những người có số lượng ảnh >= 60:

#total_images: 1348

#clusters: 8

+ Thực hiện **PCA** – để reduce dataset xuống còn 2 dimension.

+ Thực hiện các phương pháp Clustering (cả 4 phương pháp).

+ Đánh giá các phương pháp Clustering theo các độ đo ở Phần III.

+ Visualize kết quả bằng matplotlib.

Các dữ liệu features đã extracted và models đã trained sau khi chạy được lưu lại thành file với đuôi .data và .model để thuận tiện dùng lại sau này, trong đó:

HIST_all.data : LBP Histogram features cho mỗi ảnh chạy từ file “Bai_tap_3_all.py”

HIST_visual_minface60.data : LBP Histogram features cho mỗi ảnh chạy từ file “Bai_tap_3_visualize.py”

Tương tự cho các file .model với cú pháp đặt tên như trên.

Chi tiết cấu trúc được lưu trữ trong file xem ở comment trong các file code.

IV.4 Bài tập 4:

Thực hiện Clustering cho Dataset Labeled Faces in the Wild:

http://vis-www.cs.umass.edu/lfw/

❖ Tiến trình thực hiện:

- Training Facenet model:
  Chi tiết tại:
  https://github.com/davidsandberg/facenet/wiki/Classifier-training-of-inception-resnet-v1
  - Prepare training dataset: tải raw LFW dataset về tại:
    http://vis-www.cs.umass.edu/lfw/lfw.tgz
  - Face Alignment:
    Thực hiện Alignment cho face: rotation tất cả faces về cùng một hướng, center faces theo thuật toán tại:
    https://kpzhang93.github.io/MTCNN_face_detection_alignment/
    Dùng file alignment có sẵn tại thư viện Facenet – src/align/ align_dataset_mtcnn.py, với command:

    python align_dataset_mtcnn.py "Dataset\lfw" "Dataset\lfw_mtcnnpy_160" --image_size 160 --margin 32 --random_order --gpu_memory_fraction 0.25

    Ta được thư mục lfw_mtcnnpy_160 với các faces đã aligned và centered, kích thước mỗi image 160x160

  - Start classifier training:
    Dùng file `src/train_softmax.py` của Facenet để bắt đầu training model dựa trên thư mục dataset lfw_mtcnnpy_160
    Sau khi train ta sẽ có một model neural network với layer input là một ảnh và layer cuối là embeddings của ảnh đó. Có thể tải về model đã được train sẵn tại đây.

- Sử dụng model đã train để rút trích ra các embedding features,

  File code tại: face_utlis/face_embedding.py

  Hàm chính get_features_from_images_path()

  Với tham số chính:

  images_path : đường dẫn đến thư mục ảnh cần lấy embeddings.

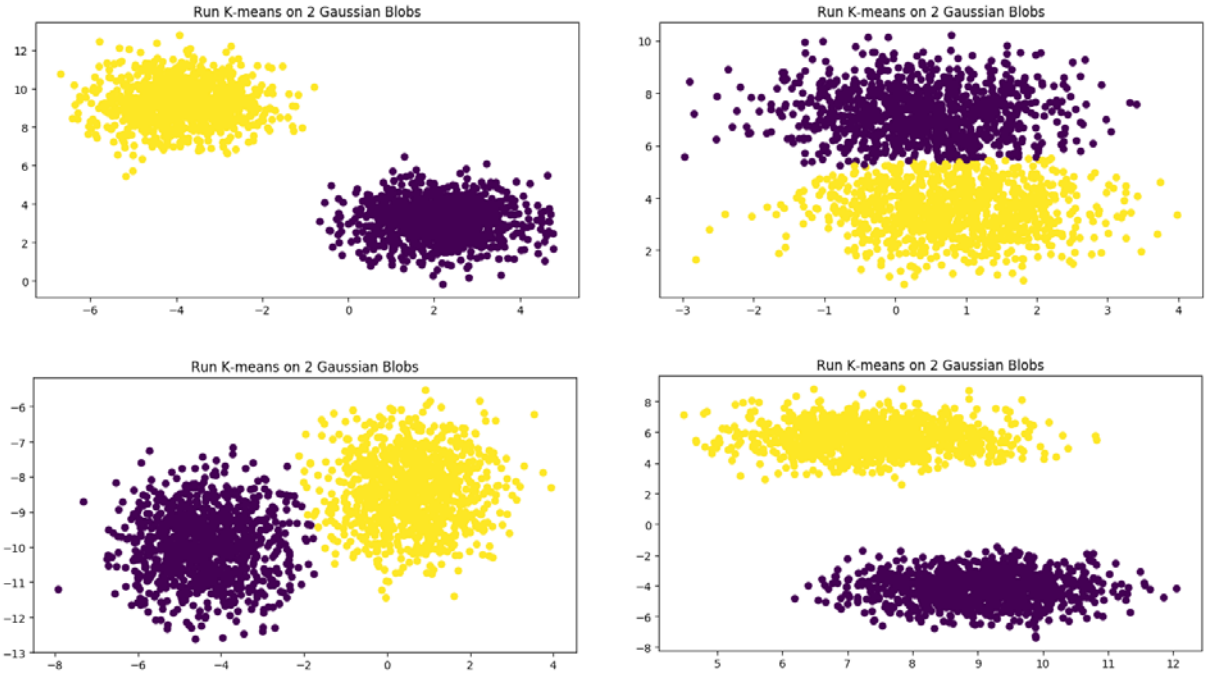
  model_dir : đường dẫn đến thư mục facenet model đã train

  Kết quả trả về:

  Mảng 2 chiều gồm các embedding features với mỗi vector trên một dòng

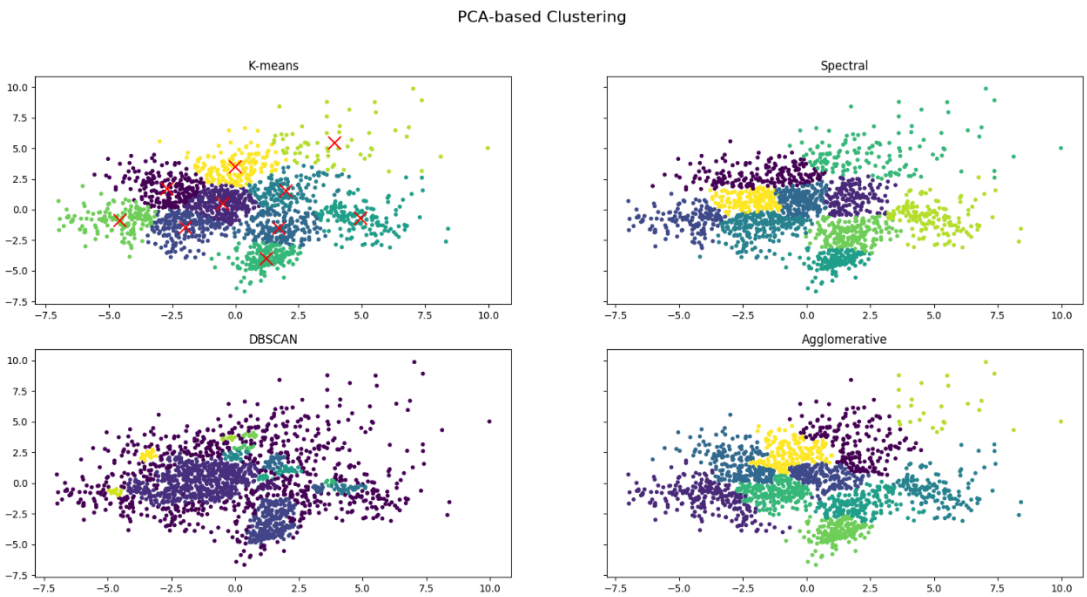
V Kết quả và đánh giá:

V.1 Bài tập 1:



Hình V-1. Một số kết quả chạy K-means trên 2 Random Gaussian Blobs

V.2 Bài tập 2:



Hình V-2. Kết quả chạy PCA-based clustering trên hand-written Digit dataset

Hình V-3. Kết quả chạy Non-PCA based clustering trên hand-written Digit dataset

Method	Time	ARI	AMI	Homogeneity	Completeness	V-measure
K-means	0.31s	0.325	0.454	0.459	0.470	0.465
Spectral	0.50s	0.314	0.452	0.458	0.461	0.459
DBSCAN	0.02s	0.110	0.236	0.251	0.368	0.298
Agglomerative	0.12s	0.310	0.438	0.444	0.458	0.451

Bảng V-1. Performance của PCA-based clustering (file Bai_tap_2.py)

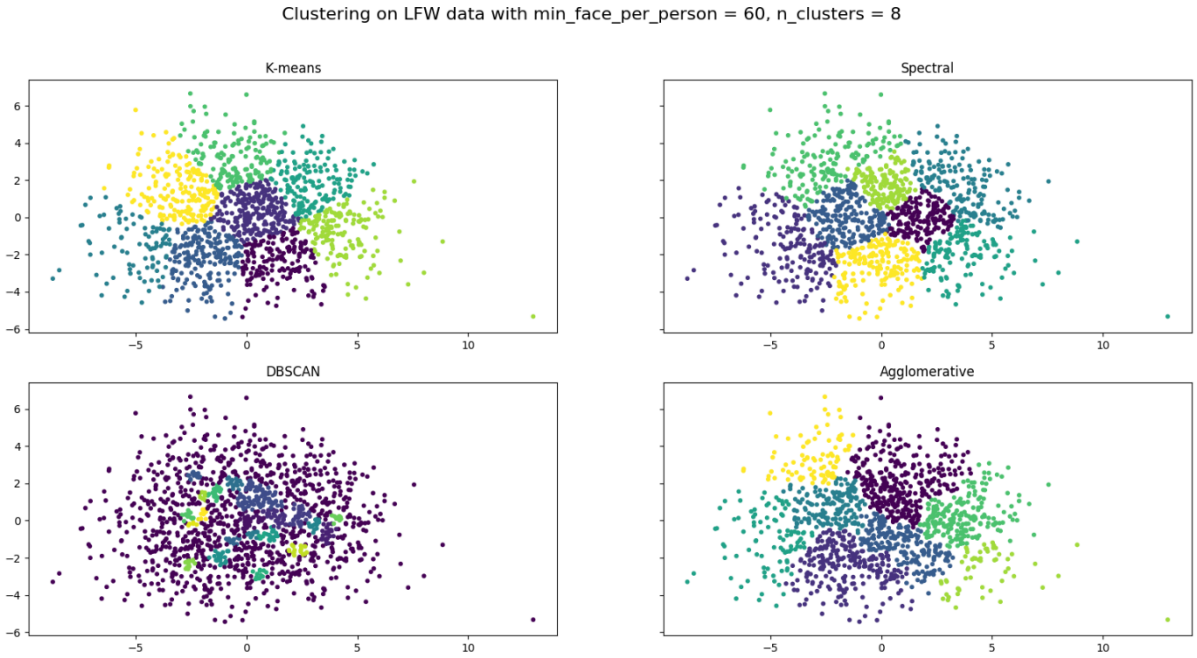
Method	Time	ARI	AMI	Homogeneity	Completeness	V-measure
K-means	0.53s	0.465	0.598	0.602	0.650	0.625
Spectral	0.84s	0.655	0.782	0.784	0.844	0.813
DBSCAN	0.68s	0.000	0.002	0.003	0.132	0.006
Agglomerative	0.17s	0.664	0.756	0.758	0.836	0.796

Bảng V-2. Performance của Non-PCA based clustering (file Bai_tap_2_NonPCA.py)

❖ ***Đánh giá:***

- Như vậy có thể thấy nếu dùng PCA thì thời gian chạy clustering được rút ngắn khá nhiều. Tuy nhiên kết quả performance nhìn chung tệ hơn so với Clustering trên data gốc (Non-PCA).
- DBSCAN: Các thông số epsilon, min_points dùng cho DBSCAN được test trên PCA-based data nên kết quả ở Non-PCA data gần như bằng 0. Chứng tỏ thuật toán DBSCAN phụ thuộc quá lớn vào các tham số truyền vào, rất khó để một người không phải chuyên gia có thể hiệu chỉnh các thông số này cho phù hợp với một dataset bất kỳ.

V.3 Bài tập 3:



Hình V-4. Visualization của Clustering trên LFW dataset, min_faces = 60

Method	Time	ARI	AMI	Homogeneity	Completeness	V-measure
K-means	0.28s	0.069	0.115	0.141	0.123	0.131
Spectral	0.49s	0.063	0.108	0.134	0.116	0.124
DBSCAN	0.01s	0.005	0.025	0.060	0.072	0.066
Agglomerative	0.09s	0.049	0.101	0.120	0.109	0.114

Bảng V-3. Performance của clustering trên LFW dataset, min_faces = 60 (file Bai_tap_3_visualize.py)

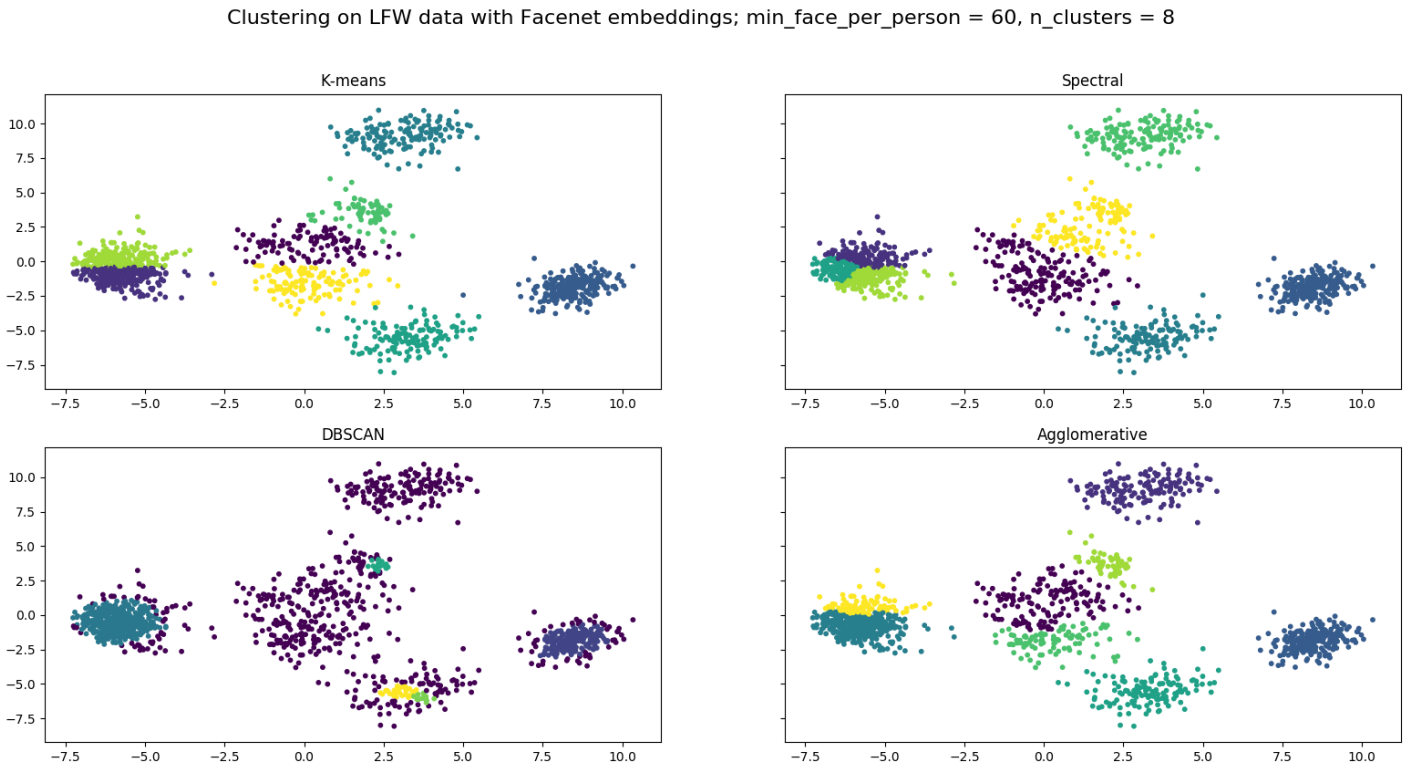
Method	Time	ARI	AMI	Homogeneity	Completeness	V-measure
K-means	347.75s	0.000	0.000	0.874	0.799	0.834

Bảng V-4. Performance của clustering trên LFW dataset (file Bai_tap_3_all.py)

❖ ***Đánh giá**:*

- Kết quả ARI, AMI chạy trên dữ liệu tất cả cluster bằng 0, có vẻ features LBP chưa thật sự tốt để có thể sử dụng khi chạy clustering trên dữ liệu face thực tế.
- Các phương pháp Clustering khác (Spectral, DBSCAN, Agglomerative) chạy quá lâu với dữ liệu lớn (hơn 13000 points) – mặc dù đã sử dụng PCA để giảm số chiều - gần như không thể thực hiện trên một máy tính có cấu hình trung bình.
- Một lần nữa kết quả performance của DBSCAN là tệ nhất do không xác định được epsilon, min_points tối ưu cho dataset này.

V.4 Bài tập 4:



Hình V-5. Visualization của Clustering trên LFW dataset với Facenet Embeddings, min_faces = 60

Method	Time	ARI	AMI	Homogeneity	Completeness	V-measure
K-means	0.18s	-0.001	0.002	0.013	0.012	0.012
Spectral	0.44s	-0.001	-0.000	0.010	0.009	0.009
DBSCAN	0.02s	-0.001	-0.001	0.007	0.011	0.009
Agglomerative	0.06s	-0.001	0.000	0.011	0.010	0.010

Bảng V-5. Performance của clustering trên LFW dataset with Facenet Embeddings (file Bai_tap_4_visualize.py)

Method	Time	ARI	AMI	Homogeneity	Completeness	V-measure
K-means	350.41s	0.000	0.000	0.871	0.798	0.833

Bảng V-6. Performance của clustering trên LFW dataset with Facenet Embeddings (file Bai_tap_4_all.py)

❖ ***Đánh giá***:

- Kết quả performance quá thấp so với Paper, nguyên nhân chưa rõ, 3 hướng đang tìm hiểu nguyên nhân:
  - Lỗi ở code: xem lại code rút trích embeddings từ model, và xem lại các thông số khi training facenet model có thể không phù hợp.
  - Feature không thật sự tốt khi sử dụng cho các phương pháp clustering ở trên: thử với một phương pháp clustering khác như rank-order
  - Thử trên một dataset khác là: Youtube Faces DB

VI Tài liệu tham khảo:

[1] Clustering Wiki:

https://en.wikipedia.org/wiki/Cluster_analysis

[2] K-means Wiki:

https://en.wikipedia.org/wiki/K-means_clustering

[3] K-means pseudo-code:

http://www.saedsayad.com/clustering_kmeans.htm

[4] K-means++ Wiki:

https://en.wikipedia.org/wiki/K-means%2B%2B

[5] Spectral Clustering – Andrew Y. Ng, Michael I. Jordan, Yair Weiss:

http://ai.stanford.edu/~ang/papers/nips01-spectral.pdf

[6] DBSCAN:
https://algorithmicthoughts.wordpress.com/2013/05/29/machine-learning-dbscan/
[7] Agglomerative Hierarchical Clustering:

http://www.improvedoutcomes.com/docs/WebSiteDocs/Clustering/Agglomerative_Hierarchical_Clustering_Overview.htm

https://en.wikipedia.org/wiki/Hierarchical_clustering

[8] PCA:
http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf
[9] LBP:
https://www.pyimagesearch.com/2015/12/07/local-binary-patterns-with-python-opencv/
https://en.wikipedia.org/wiki/Local_binary_patterns

[10] Facenet Embeddings:

https://arxiv.org/pdf/1503.03832.pdf

[11] Clustering Performance Evaluation:
http://scikit-learn.org/stable/modules/clustering.html#clustering-performance-evaluation
[12] Mutual Information Index:
http://www.ims.uni-stuttgart.de/institut/mitarbeiter/schulte/theses/phd/algorithm.pdf
[13] Dlib:
https://pypi.python.org/pypi/dlib
[14] Facenet Github Code:
https://github.com/davidsandberg/facenet
[15] Facial Clustering Pipeline:
https://github.com/kevinlu1211/FacialClusteringPipeline
[16] Triplet loss:
http://cs231n.stanford.edu/reports/2017/posters/108.pdf