TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KHOA HỌC MÁY TÍNH
MÔN HỌC THỊ GIÁC MÁY TÍNH VÀ MÁY HỌC

BÁO CÁO BÀI TẬP 1
CLUSTERING

SVTH:

1. **Đoàn Trí Đức** **14520178**

Tp. Hồ Chí Minh, tháng 20 năm 2017

Contents

1. CƠ SỞ LÝ THUYẾT

1.1 Clustering

- Thuật toán clustering thực hiện phân nhóm dữ liệu hay phân tách dữ liệu vào những nhóm có đặc điểm giống nhau. Clustering thuộc lớp bài toán unsupervised learning . Trong lớp bài toán này chúng ta không biết được nhãn (label) của dữ liệu đầu vào

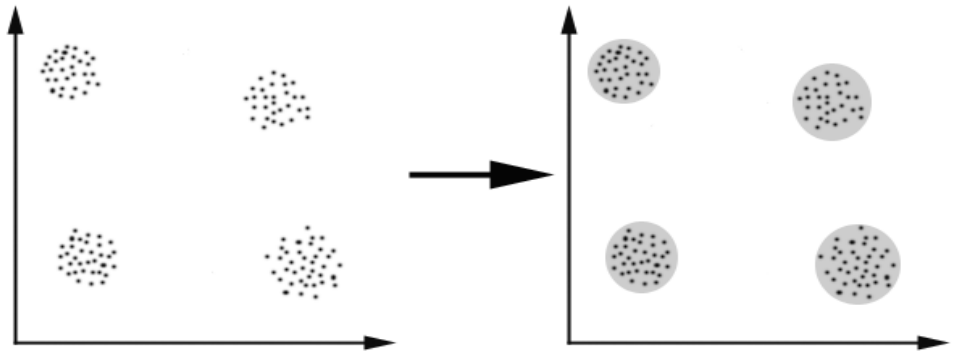


Figure 1 Ví dụ của Clustering

a. Kmean

- Tư tưởng cốt lõi của thuật toán Kmean là tìm cách phân nhóm các tối tượng đã cho vào K cụm cho trước.
- Phát biểu bài toán:

  + Input: Tập các đối tượng (data point), số cụm : K

  + Output: Các cụm và các điểm trong tập cho trước đã thuộc một cụm nhất định

- Thuật toán:
  1. Chọn ngẫu nhiên K trọng tâm cho K cụm.
  2. Tính khoảng cách giữa các đối tượng đến K tâm . Khoảng cách thường được tính bằng Minkowski, Euclidean, cosin, …
  3. Nhóm các đối tường vào nhóm gần nhất.
  4. Xác định lại tâm cho các nhóm.
  5. Thực hiện bước 2 cho đến khi không có sự thay đổi nào của các đối tượng hoặc nhỏ hơn một thredhold (hội tụ).

b. Thuật toán DBSCAN (Density Based Spatial Clustering of Application with Noise)

- Ý tưởng cơ bản của thuật toán là vùng lân cận mỗi đối tượng trong một cụm có số đối tượng lớn hơn ngưỡng tối thiểu, các cụm có hình dạng tùy ý, khả năng phát hiện nhiễu tốt. Hình dạng vùng lân cận phụ thuộc vào hàm khoảng cách giữa các đối tượng.

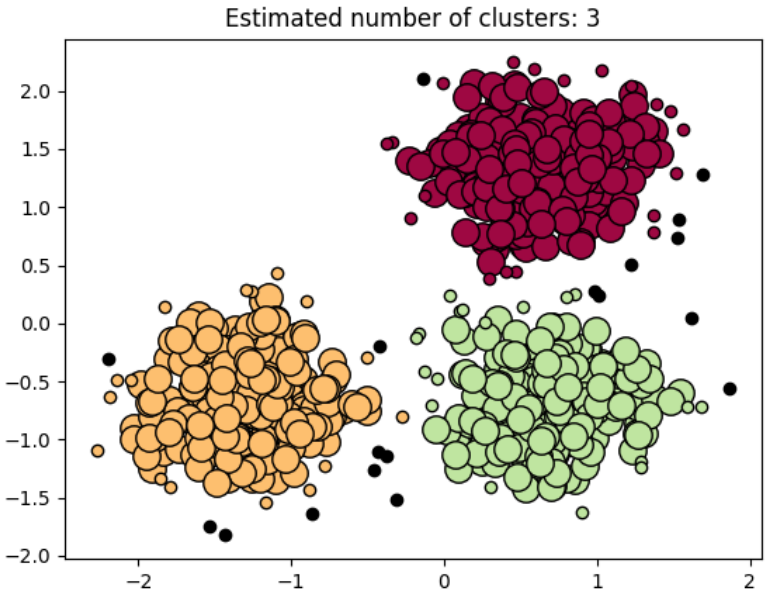


Figure 2 Ví dụ cho DBSCAN

- Các khái niệm trong DBSCAN [1]:
    1. **Định nghĩa 1**: Eps-neighborhood
        - Gọi D là tập cơ sở dữ liệu điểm. Eps- neighborhood của điểm p, ký hiệu là Gọi D là tập cơ sở dữ liệu điểm. Epsneighborhood của điểm p, ký hiệu là NEps(p), được xác định bởi NEps(p) = {q ∈D | dist(p,q) ≤ Eps}.

    2. **Định nghĩa 2**: Directly density- reachable
        - Một điểm p∈D là directly density - reachable từ một điểm q∈D khi và chỉ khi: p ∈ NEps(q) và |NEps(q)| ≥ MinPts (điều kiện điểm lõi).

3. **Định nghĩa 3**: (density–reachable)
    - Một điểm p là density–reachable từ q khi và chỉ khi có một dây chuyền các điểm p1, …, pn với p1 = q và pn = p mà pi+1 là directly density – reachable từ pi.
4. **Định nghĩa 4**: (density-connected)
    - Một điểm p là density-connected đến một điểm q khi và chỉ khi có một điểm o mà cả hai điểm p và q đều density-reachable từ o. Một nhóm là một tập các điểm density-connected lớn nhất. Nhiễu là điểm không thuộc nhóm nào.
5. **Định nghĩa 5**: Nhóm
    - Một nhóm C là một tập con không rỗng của D thỏa các điều kiệu sau: ∀ p, q: nếu p ∈ C và q là density-reachable từ p thì q ∈ C. ∀ p, q ∈ C: p là density-connected từ q.
6. **Định nghĩa 6**: Nhiễu
    - Gọi C1, …, Ck là các nhóm của tập dữ liệu D. Nhiễu là tập hợp tất cả các điểm không thuộc bất kỳ nhóm Ci nào với i=1, …, k.

- Giải thuật DBSCAN:

Đầu vào:

Tập dữ liệu gồm m phần tử: $X = \{x_i\}_{i=1,m}$ với $x_i \in R^n$

Bán kính vùng lân cận: eps

Số lượng điểm tối thiểu: minPts

Giải thuật:

```
//Khởi tạo
ClusterId = 1
for i = 1 to m do
    if (x_i chưa được duyệt qua)
        N = regionQuery(x_i, X, eps)
        if (|N| < minPts) //x_i không là điểm lõi
            changeCluster(x_i, 0) //Gán x_i vào nhiễu
        else
            changeCluster(N, ClusterId)
            N = N \ {x_i}
            while (N != rỗng)
                N' = regionQuery(p, X, eps) // p ∈ N
                if (|N'| >= minPts)
                    for j = 1 to sizeof(N') do
                        q = N'.get(j)
                        if (q chưa được duyệt hoặc là noise)
                            if (q chưa được duyệt)
                                N = N ∪ {q}
                            changeCluster(q, ClusterId)
                N = N \ {p}
            ClusterId++
```

c. Spectral Clustering

- Thuật toán Spectral Clustering xem xét data clustering như bài toán graph partitioning mà không cần quan tâm đến sự phân bố của data. [2]
- **Phát biểu bài toán**

  + Input: Dữ liệu đầu vào và số cluster (K)

  + Output: Các cụm và các điểm trong tập cho trước đã thuộc một cụm nhất định.

- **Thuật giải:** Giả sử các data là x1, ..,xn và cạnh liên kết là sij= s(xi,xj).
  1. Xây dựng đồ thi giữa các điểm data, và trọng số có thể được tính bằng các cách tùy theo bài toán.
  2. Tính Laplacian L .
  3. Tính k egenvector v1,..vk thỏa phương trình L*v=lamda*D*v.
  4. Gán lần lượt giá trị từ egenvector cho các điểm trong đồ thị.
  5. Tìm đường cắt đồ thị thành 2 đồ thị con.
- **Ví dụ**

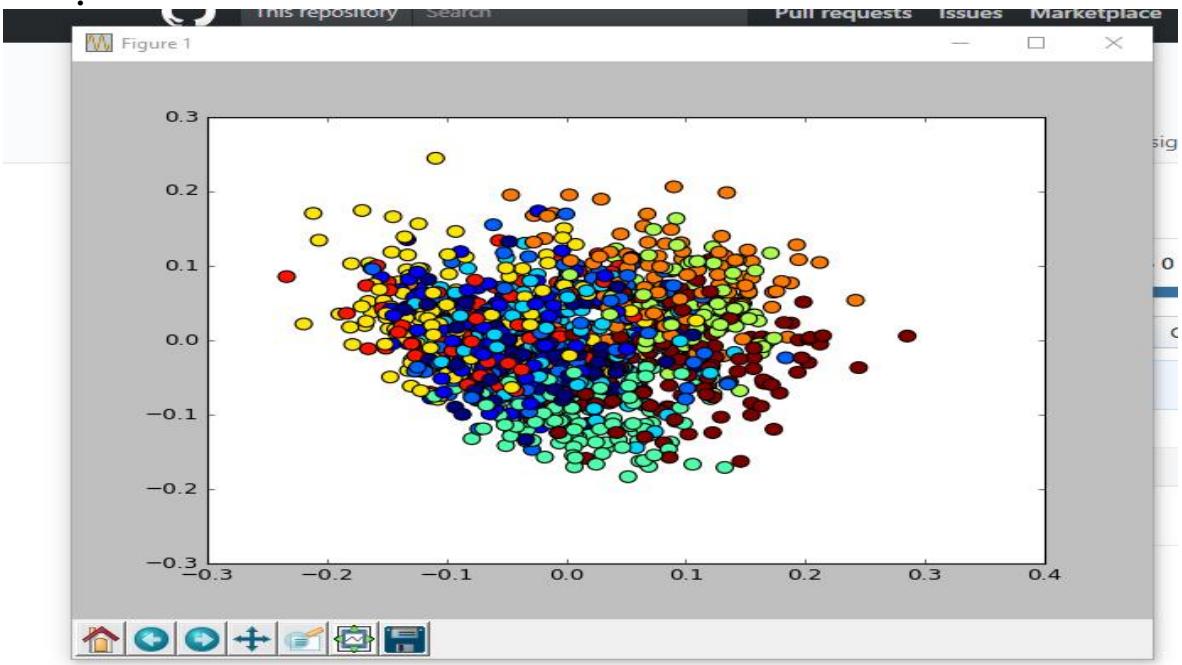
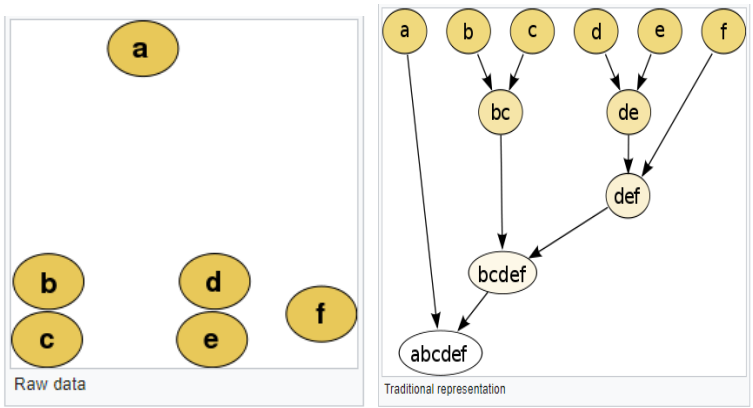


Figure 3 Spectral Clustering với HOG feature

d. Agglomerative Approach

- **Ý tưởng** : Ban đầu, chúng ta xem mỗi đối tượng là 1 nhóm (cluster) và nhóm 2 đối tượng gần nhất thành 1 cluster. Quá trình này lặp lại cho đến khi tất cả các đối tượng được nhóm vào 1 cluster cuối cùng.
- **Thuật toán**:
    1. Chuyển đổi các đặc trưng của đối tượng vào ma trận khoảng cách
    2. Xem các đối tượng là một cluster
    3. Lập lại bước 2 cho đến khi số cluster bằng 1
       3.1.Gộp 2 cluster gần nhất
       3.2. : Cập nhật mâ trận khoảng cách.
- **Minh họa thuật giải**



- **Ví dụ**

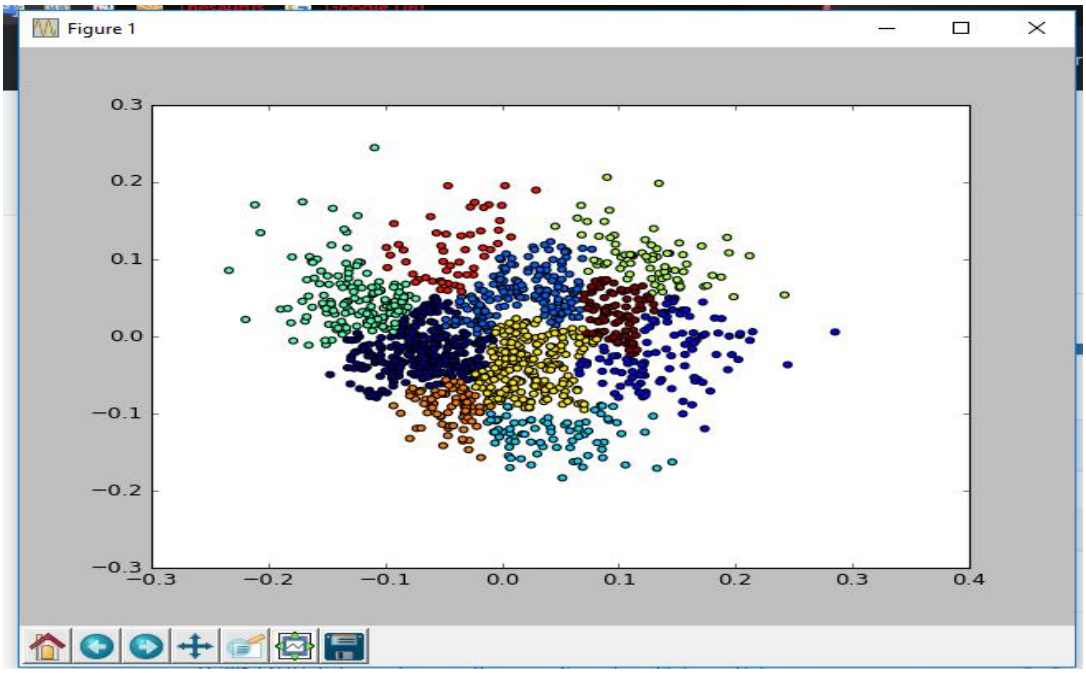


Figure 4 Agglomerative Clustering với HOG feature

2. PHƯƠNG PHÁP ĐÁNH GIÁ HIỆU SUẤT CỦA CÁC THUẬT TOÁN CLUSTERIG

- Đánh giá hiệu suất của các thuật toán clustering không giống như việc đánh giá độ lỗi hoặc độ chính xác giống như phương pháp học có giám sát (supervised classification algorithm).
- **Adjusted Rand index:** Adjusted Rand index tính toán độ tương tự giữa 2 clustering bằng cách xem xét tất cả cách cặp mẫu. Sau đó điểmm các cặp đã được chỉ định thuộc cluster được gán label đúng hoặc khác với label thật sự của điểm đó. Công thức tính:

```
ARI = (RI - Expected_RI) / (max(RI) - Expected_RI)
```

3. CÁC PHƯƠNG PHÁP EXTRACT FEATURE

a. Local Binary pattern

- Local binary patterm sử dụng 8 điểm ảnh xung quanh và sử dụng giá trị của điểm ảnh ở trung tâm làm ngưỡng. Giá trị LBP được xác định bằng cách nhân các giá trị ngưỡng với trọng số ứng với mỗi điểm ảnh sau đó cộng tổng lại.Hình dưới minh họa cách tính độ tương phản trực giao (C) là hiệu cấp độ xám trung bình của các điểm ảnh lớn hơn hoặc bằng ngưỡng với các điểm ảnh thấp hơn ngưỡng.
- Kể từ khi được đưa ra, theo định nghĩa là bất biến với những thay đổi đơn điệu trong ảnh đen trắng. Để cải tiến phương pháp, bổ sung thêm phương pháp tương phản trực giao địa phương. Hình dưới minh họa cách tính độ tương phản trực giao (C) là ký hiệu cấp độ xám trung bình của các điểm ảnh lớn hơn hoặc bằng ngưỡng với các điểm ảnh thấp hơn ngưỡng. Phân phối hai chiều của mã LBP và độ tương phản cục bộ được lấy làm đặc trưng gọi là LBP/C.

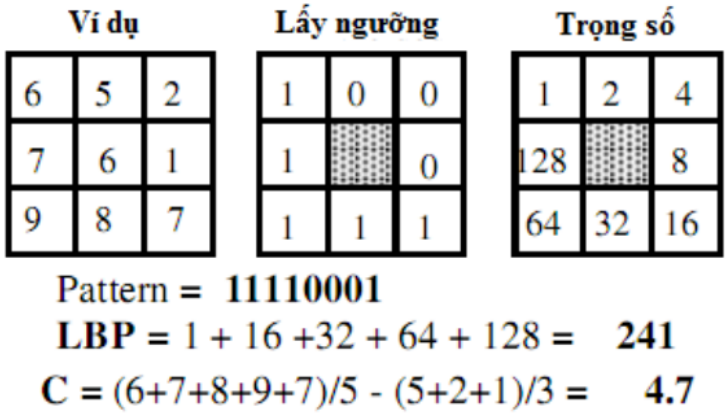


Figure 5 Ví dụ về LBP và độ tương phản cục bộ C.

b. Histogram of Oriented Gradients.

- **Phương pháp**: HOG dựa trên việc đếm số lần xuất hiện của các hướng đạo hàm (gradient orientation) trong các vùng cục bổ của ảnh. Bản chất của phương pháp HOG là khai thác đối tượng ở hình dáng và vẻ bề ngoài trong ảnh.
- **Thuật toán HOG**:
    1. Tiền sử lý hình ảnh: size , …
    2. Tính Grandient của ảnh: Thường được sử dụng thực hiện bằng hai phép nhân chập ảnh gốc với 2 nhân 1 chiều tương ứng với các toán tử lấy đạo hàm theo 2 hướng Õ và Oy.
    3. Gán hướng
    4. Tính histogram
    5. Chuẩn hóa

4. HIỆN THỰC CÁC THUẬT TOÁN VỚI DATASE

a. Ứng dụng 1: Thực hiện thuật toán Kmean với dữ liệu gồm 2 Gaussians

- **Các hàm chính**

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.spatial.distance import cdist
from sklearn import metrics
from sklearn.cluster import KMeans
from sklearn.datasets import load_digits
from sklearn.decomposition import PCA
from sklearn.preprocessing import scale
np.random.seed(11)
means = [[2, 2], [8, 3]]
cov = [[1, 0], [0, 1]]
N = 500
X0 = np.random.multivariate_normal(means[0], cov, N)
X1 = np.random.multivariate_normal(means[1], cov, N)
#X2 = np.random.multivariate_normal(means[2], cov, N)
X = np.concatenate((X0, X1), axis = 0)
K = 2
original_label = np.asarray([0]*N + [1]*N).T
def kmeans_display(X, label):
    K = np.amax(label) + 1
    X0 = X[label == 0, :]
    X1 = X[label == 1, :]
    #X2 = X[label == 2, :]
    plt.plot(X0[:, 0], X0[:, 1], 'b^', markersize = 4, alpha = .8)
    plt.plot(X1[:, 0], X1[:, 1], 'go', markersize = 4, alpha = .8)
    #plt.plot(X2[:, 0], X2[:, 1], 'rs', markersize = 4, alpha = .8)
    plt.axis('equal')
    plt.plot()
```

```
x_min, x_max = reduced_data[:, 0].min() - 1, reduced_data[:, 0].max() + 1
y_min, y_max = reduced_data[:, 1].min() - 1, reduced_data[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
# Obtain labels for each point in mesh. Use last trained model.
Z = kmeans.predict(np.c_[xx.ravel(), yy.ravel()])
# Put the result into a color plot
Z = Z.reshape(xx.shape)
plt.figure(1)
plt.clf()
plt.imshow(Z, interpolation='nearest',
           extent=(xx.min(), xx.max(), yy.min(), yy.max()),
           cmap=plt.cm.Paired,
           aspect='auto', origin='lower')

plt.plot(reduced_data[:, 0], reduced_data[:, 1], 'k.', markersize=2)
# Plot the centroids as a white X
centroids = kmeans.cluster_centers_
plt.scatter(centroids[:, 0], centroids[:, 1],
            marker='x', s=169, linewidths=3,
            color='w', zorder=10)
plt.title('K-means clustering on the digits dataset (PCA-reduced data)\n'
          'Centroids are marked with white cross')
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
plt.yticks(())
plt.show()
```

- **Kết quả**:

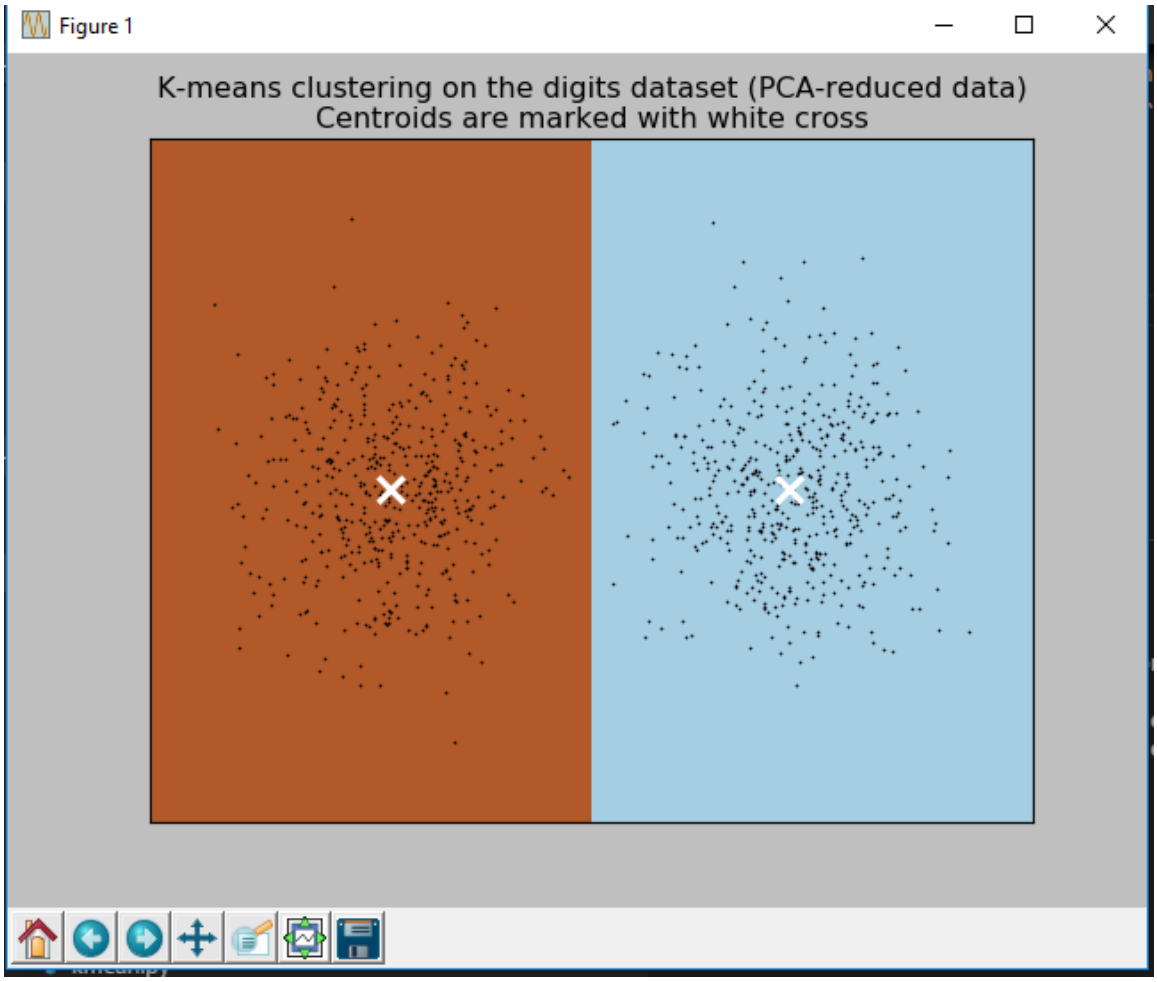


Figure 6 Kết quả đạt được với Kmean

b. Ứng dụng 2: Sử dụng tập dữ liệu hand-written digit

- **Phát biểu bài toán**:

+ Input: Tập dữ liệu là các số viết tay, gồm các số từ 0 đến 9. Feature được sử dụng là pixel intensity

+ Output: 10 cluster tương ứng với các kí tự số, đánh giá kết quả của các giải thuật.

- **Hướng giải quyết**: Sử dụng các thuật toán: Kmeans, Spectral Clustering, DBSCAN, Agglomerative Clustering.

❖ **Kmean**:

- Các hàm chính:

```
print(__doc__)
```

```

from time import time
import numpy as np
import matplotlib.pyplot as plt

from sklearn import metrics
from sklearn.cluster import KMeans
from sklearn.datasets import load_digits
from sklearn.decomposition import PCA
from sklearn.preprocessing import scale
np.random.seed(42)
digits = load_digits()
data = scale(digits.data)
n_samples, n_features = data.shape
n_digits = len(np.unique(digits.target))
labels = digits.target
sample_size = 300
print("n_digits: %d, \t n_samples %d, \t n_features %d"
      % (n_digits, n_samples, n_features))
print(82 * '_')
print('init\t\ttime\tinertia\thomo\tcompl\tv-meas\tARI\tAMI\tsilhouette')
def bench_k_means(estimator, name, data):
    t0 = time()
    estimator.fit(data)
    print('%-9s\t%.2fs\t%i\t%.3f\t%.3f\t%.3f\t%.3f\t%.3f\t%.3f'
          % (name, (time() - t0), estimator.inertia_,
             metrics.homogeneity_score(labels, estimator.labels_),
             metrics.completeness_score(labels, estimator.labels_),
             metrics.v_measure_score(labels, estimator.labels_),
             metrics.adjusted_rand_score(labels, estimator.labels_),
             metrics.adjusted_mutual_info_score(labels,  estimator.labels_),
             metrics.silhouette_score(data, estimator.labels_,
                                      metric='euclidean',
                                      sample_size=sample_size)))

bench_k_means(KMeans(init='k-means++', n_clusters=n_digits, n_init=10),
              name="k-means++", data=data)

bench_k_means(KMeans(init='random', n_clusters=n_digits, n_init=10),
              name="random", data=data)

# in this case the seeding of the centers is deterministic, hence we run the
# kmeans algorithm only once with n_init=1
pca = PCA(n_components=n_digits).fit(data)
bench_k_means(KMeans(init=pca.components_, n_clusters=n_digits, n_init=1),
              name="PCA-based",
              data=data)
print(82 * '_')
```

```
# ##############################################################################
```

```
# Visualize the results on PCA-reduced data

reduced_data = PCA(n_components=2).fit_transform(data)
kmeans = KMeans(init='k-means++', n_clusters=n_digits, n_init=10)
kmeans.fit(reduced_data)

# Step size of the mesh. Decrease to increase the quality of the VQ.
h = .02     # point in the mesh [x_min, x_max]x[y_min, y_max].

# Plot the decision boundary. For that, we will assign a color to each
x_min, x_max = reduced_data[:, 0].min() - 1, reduced_data[:, 0].max() + 1
y_min, y_max = reduced_data[:, 1].min() - 1, reduced_data[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))

# Obtain labels for each point in mesh. Use last trained model.
Z = kmeans.predict(np.c_[xx.ravel(), yy.ravel()])

# Put the result into a color plot
Z = Z.reshape(xx.shape)
plt.figure(1)
plt.clf()
plt.imshow(Z, interpolation='nearest',
           extent=(xx.min(), xx.max(), yy.min(), yy.max()),
           cmap=plt.cm.Paired,
           aspect='auto', origin='lower')

plt.plot(reduced_data[:, 0], reduced_data[:, 1], 'k.', markersize=2)
# Plot the centroids as a white X
centroids = kmeans.cluster_centers_
plt.scatter(centroids[:, 0], centroids[:, 1],
            marker='x', s=169, linewidths=3,
            color='w', zorder=10)
plt.title('K-means clustering on the digits dataset (PCA-reduced data)\n'
          'Centroids are marked with white cross')
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
plt.yticks(())
plt.show()
```

- **Kết quả**

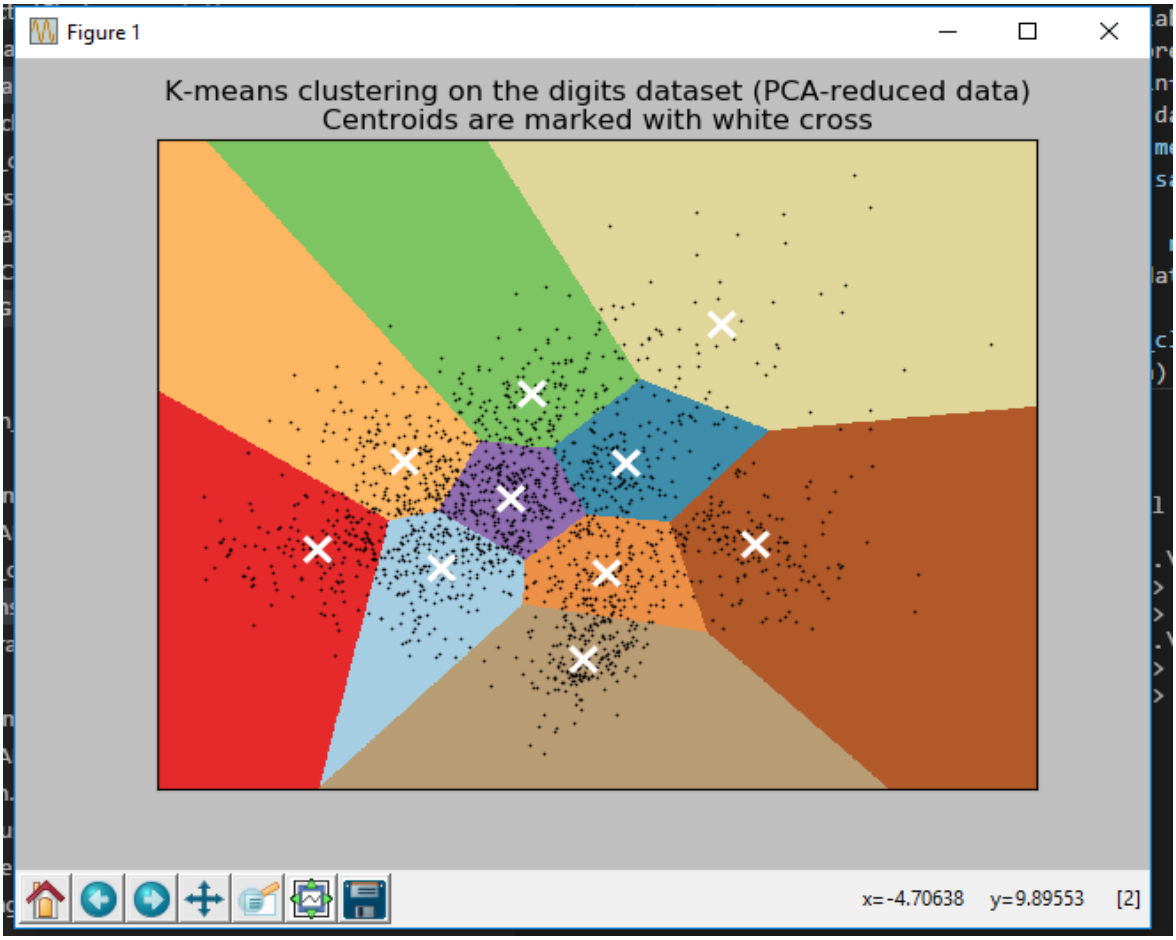


Figure 1
K-means clustering on the digits dataset (PCA-reduced data)
Centroids are marked with white cross
x=-4.70638 y=9.89553 [2]

❖ **Spectral Clustering**

- **Các hàm chính**

```
#MSSV: 14520178
```

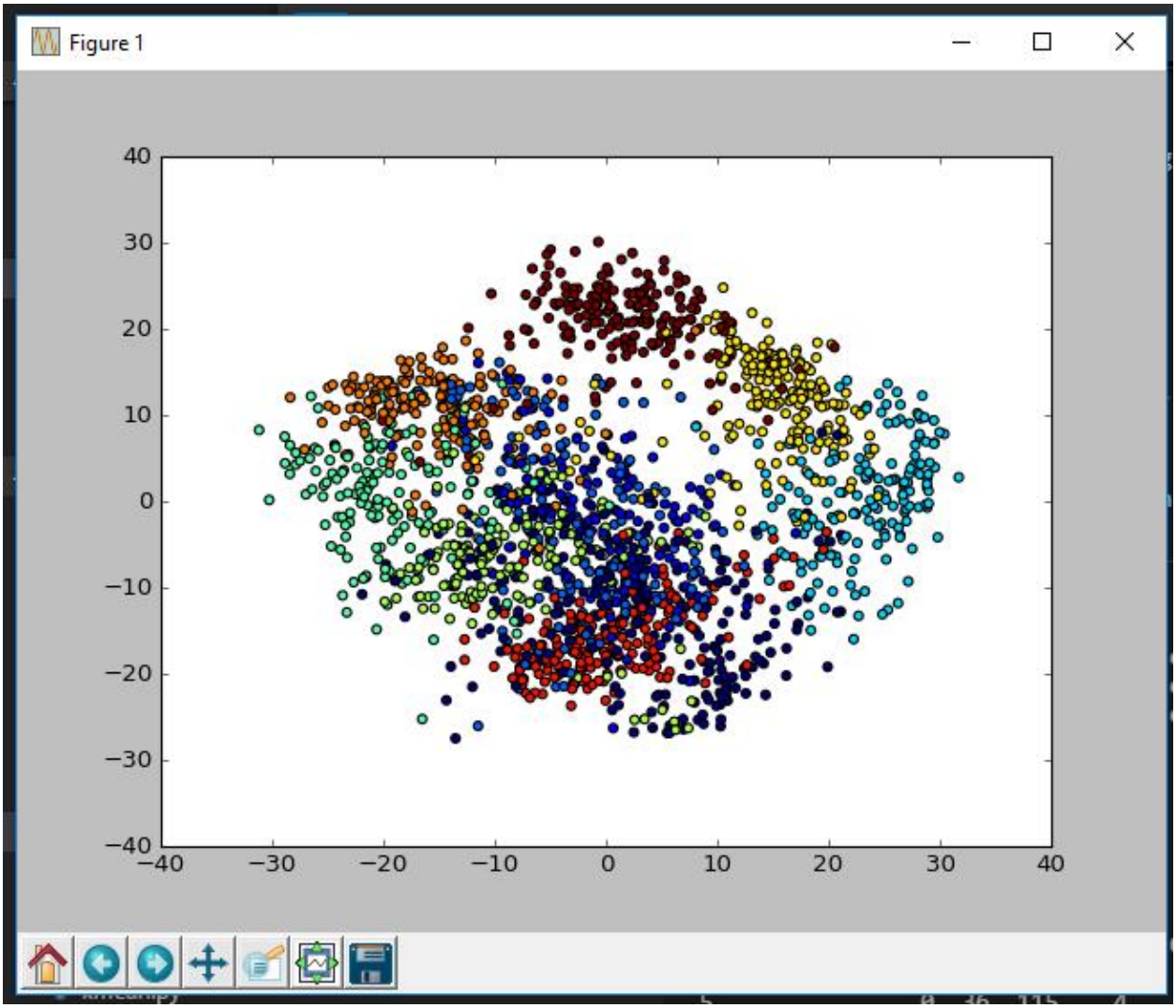
```
# special clustering

print(__doc__)

from time import time
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn import metrics
from sklearn.cluster import KMeans
from sklearn.datasets import load_digits
from sklearn.decomposition import PCA
from sklearn.preprocessing import scale
from sklearn.cluster import spectral_clustering

np.random.seed(42)

digits = load_digits()
```

```
X = metrics.pairwise.cosine_similarity(digits.data)
```

```
labels = spectral_clustering(X, n_clusters=10, eigen_solver='arpack')
```

```
print("Result")
df = pd.DataFrame({'Labels':labels,'Truth labels':digits.target})
ct = pd.crosstab(df['Labels'],df['Truth labels'])
print(ct)
print("----------------------------------------------------------------------")

# visualize
pca_converter = PCA(n_components = 2)
# convert digits data to 2D points
data_2d = pca_converter.fit_transform(digits.data)
plt.scatter(data_2d[:,0], data_2d[:,1], c=labels)
plt.show()
```

- **Kết quả:**



❖ **DBSCAN**

- **Các hàm chính**

```
print(__doc__)
```

```
import numpy as np

from sklearn.cluster import DBSCAN
from sklearn import metrics
from sklearn.datasets.samples_generator import make_blobs
from sklearn.preprocessing import StandardScaler

from sklearn.datasets import load_digits
from sklearn.decomposition import PCA
from sklearn.preprocessing import scale
np.random.seed(42)
digits = load_digits()
data = scale(digits.data)
X = PCA(n_components=2).fit_transform(data)
db = DBSCAN(eps=0.3, min_samples=10,algorithm='brute').fit(X)
core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
core_samples_mask[db.core_sample_indices_] = True
labels = db.labels_
# Number of clusters in labels, ignoring noise if present.
n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)
import matplotlib.pyplot as plt
# Black removed and is used for noise instead.
unique_labels = set(labels)
colors = [plt.cm.Spectral(each)
          for each in np.linspace(0, 1, len(unique_labels))]
for k, col in zip(unique_labels, colors):
    if k == -1:
        # Black used for noise.
        col = [0, 0, 0, 1]

    class_member_mask = (labels == k)

    xy = X[class_member_mask & core_samples_mask]
    plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=tuple(col),
             markeredgecolor='k', markersize=14)

    xy = X[class_member_mask & ~core_samples_mask]
    plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=tuple(col),
             markeredgecolor='k', markersize=6)

plt.title('Estimated number of clusters: %d' % n_clusters_)
plt.show()
```

- **Kết quả :**

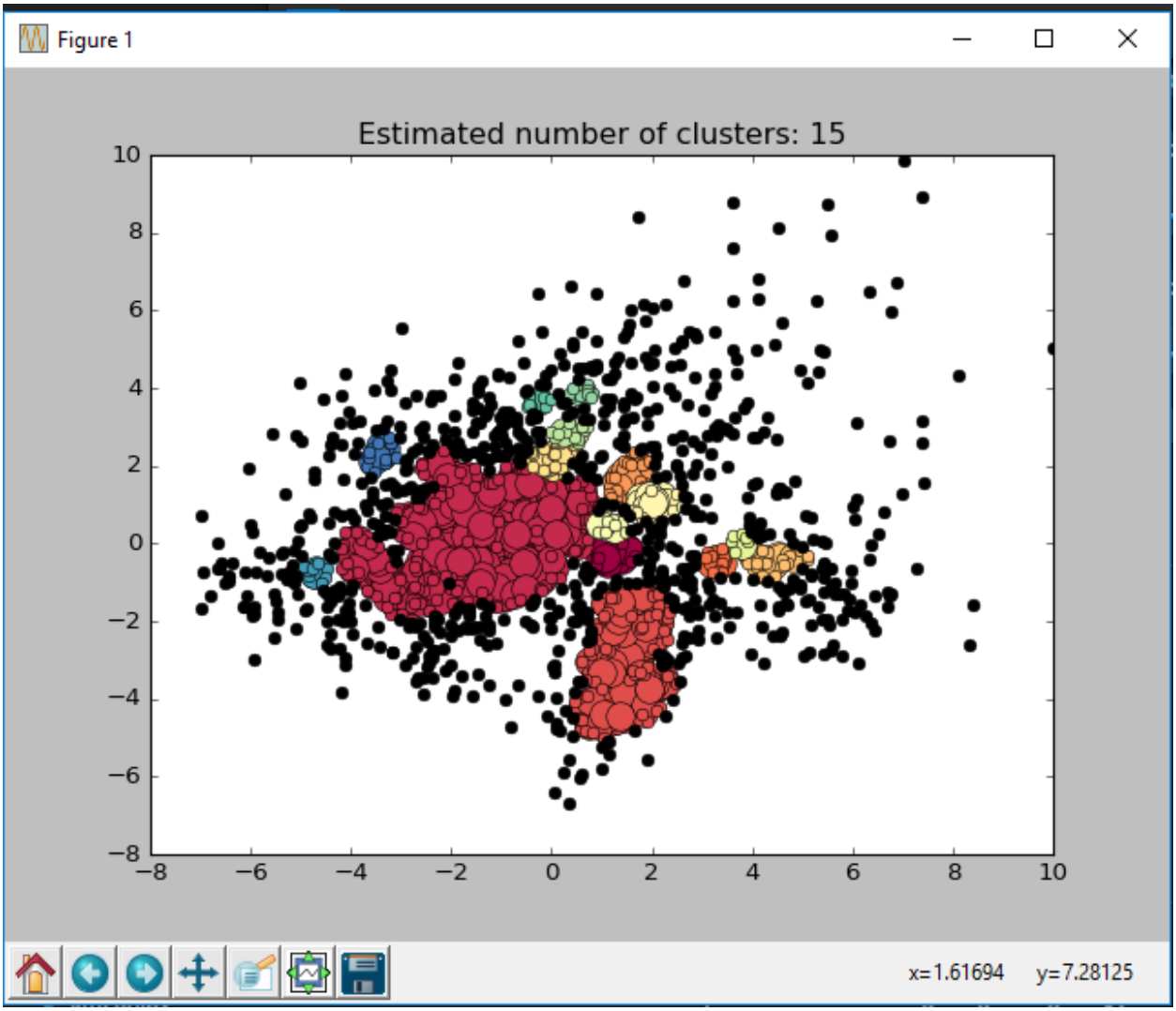


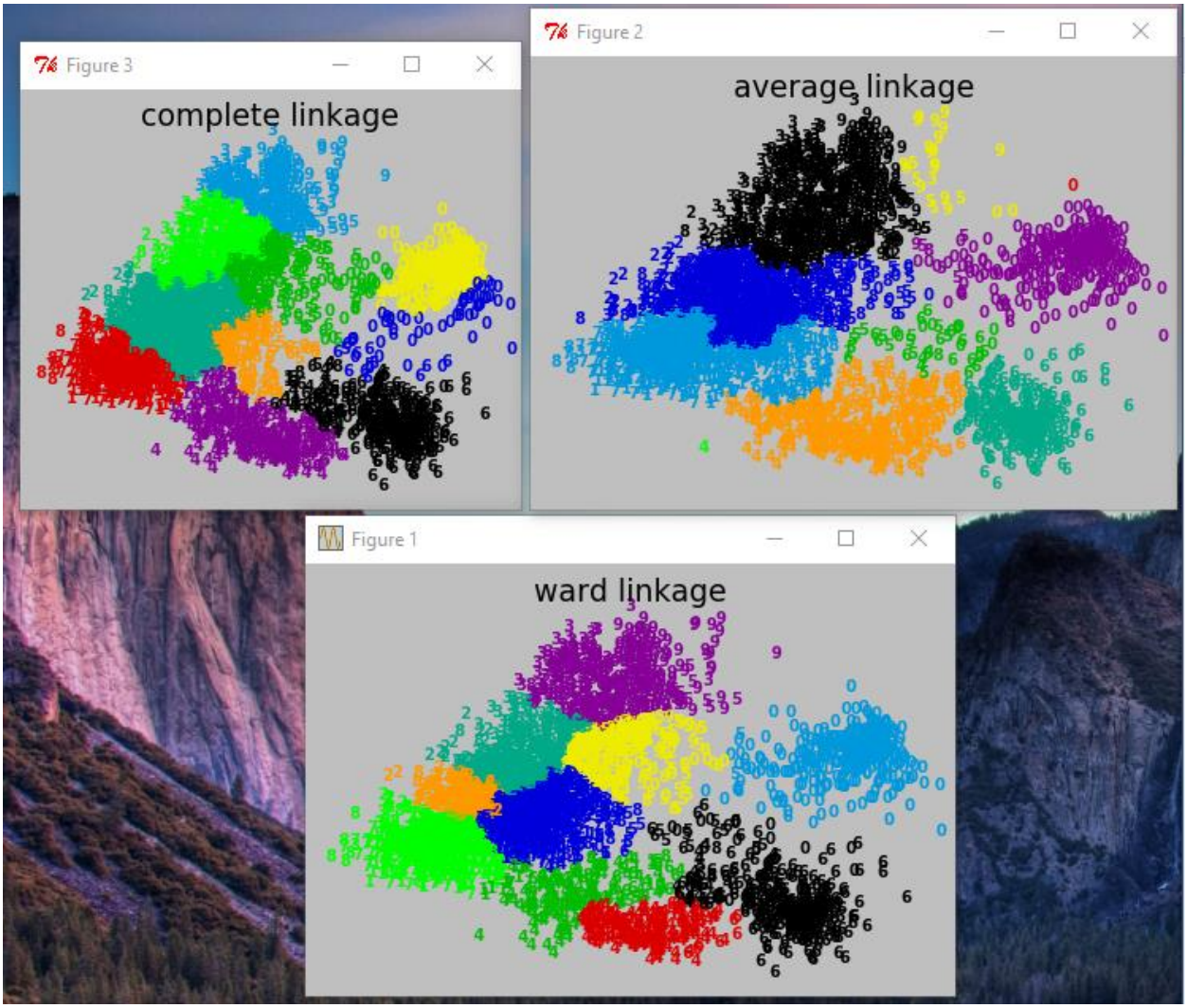
Figure 1
Estimated number of clusters: 15
10
8
6
4
2
0
−2
−4
−6
−8
−8
−6
−4
−2
0
2
4
6
8
10
x=1.61694
y=7.28125

❖ **Agglomerative Clustering.**

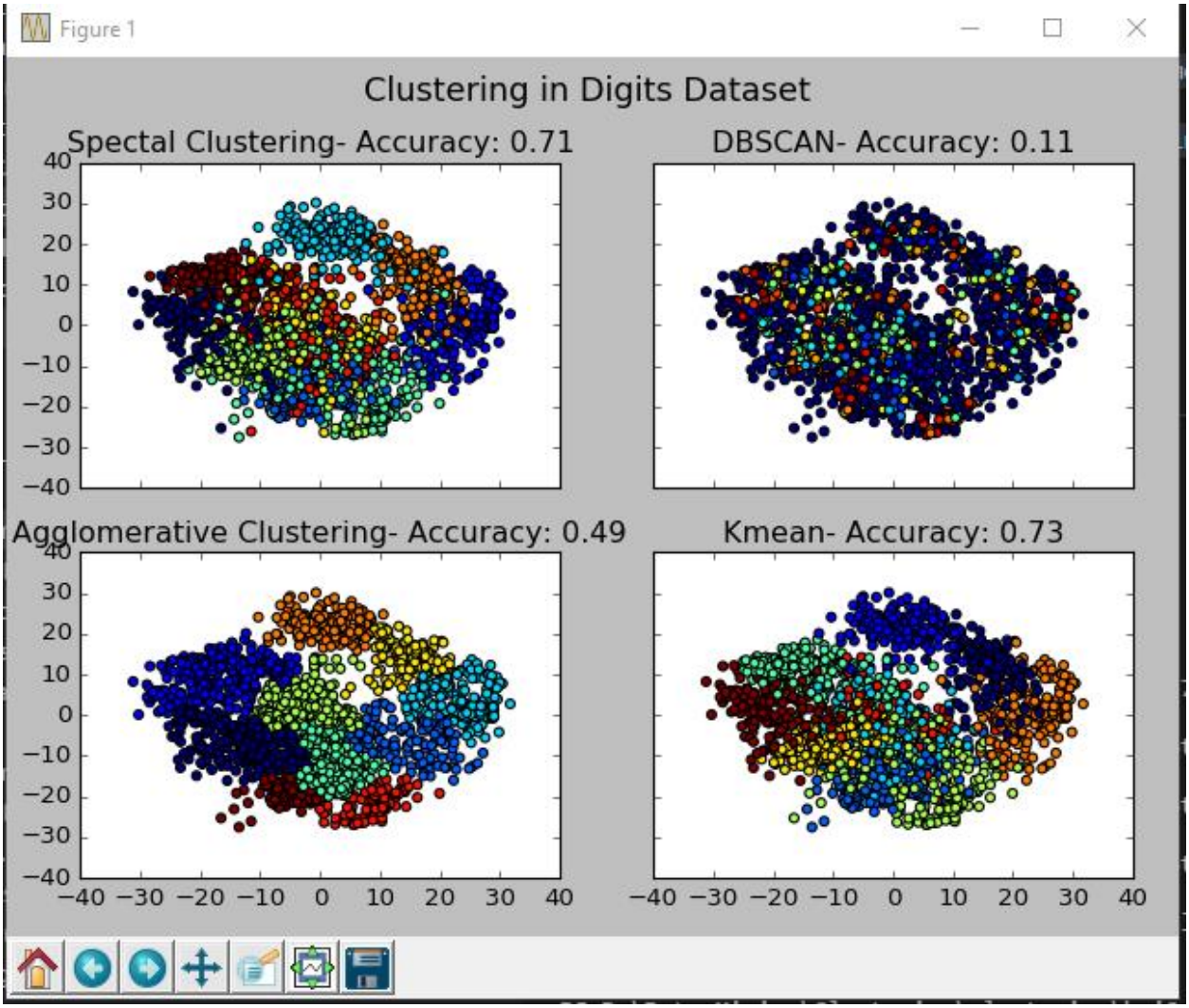
- **Các hàm chính**

```
print(__doc__)
from time import time

import numpy as np
from scipy import ndimage
from matplotlib import pyplot as plt
from sklearn import manifold, datasets
digits = datasets.load_digits(n_class=10)
X = digits.data
y = digits.target
n_samples, n_features = X.shape
np.random.seed(0)
def nudge_images(X, y):
    shift = lambda x: ndimage.shift(x.reshape((8, 8)),
                                    .3 * np.random.normal(size=2),
                                    mode='constant',
                                    ).ravel()
    X = np.concatenate([X, np.apply_along_axis(shift, 1, X)])
    Y = np.concatenate([y, y], axis=0)
    return X, Y
X, y = nudge_images(X, y)
def plot_clustering(X_red, X, labels, title=None):
    x_min, x_max = np.min(X_red, axis=0), np.max(X_red, axis=0)
    X_red = (X_red - x_min) / (x_max - x_min)
    plt.figure(figsize=(6, 4))
    for i in range(X_red.shape[0]):
        plt.text(X_red[i, 0], X_red[i, 1], str(y[i]),
                 color=plt.cm.spectral(labels[i] / 10.),
                 fontdict={'weight': 'bold', 'size': 9})
    plt.xticks([])
    plt.yticks([])
    if title is not None:
        plt.title(title, size=17)
    plt.axis('off')
    plt.tight_layout()
print("Computing embedding")
X_red = manifold.SpectralEmbedding(n_components=2).fit_transform(X)
print("Done.")

from sklearn.cluster import AgglomerativeClustering
for linkage in ('ward', 'average', 'complete'):
    clustering = AgglomerativeClustering(linkage=linkage, n_clusters=10)
    t0 = time()
    clustering.fit(X_red)
    print("%s : %.2fs" % (linkage, time() - t0))
    plot_clustering(X_red, X, clustering.labels_, "%s linkage" % linkage)
plt.show()
```

- **Kết quả đạt được**:



❖ **Đánh giá kết quả chung**



- **Nhận xét**:

  + Thuật toán Kmean có kết quả tốt nhất

  + Thấp nhất là thuật toán DBSCAN

c. Ứng dụng 3: Ứng dụng Clustering với tập dữ liệu là face.

- **Phát biểu bài toán:**

+ **Input**: Tập dữ liệu là tập dữ liệu chuẩn LFW, gồm 7 người và 1228 hình. Feature được sử dụng là local binary pattern (LBP).

+Output: 7 cluster tương ứng với 7 lable(người), đánh giá kết quả của các giải thuật.

- **Hướng giải quyết**: Sử dụng các thuật toán: Kmeans, Spectral Clustering, DBSCAN, Agglomerative Clustering.

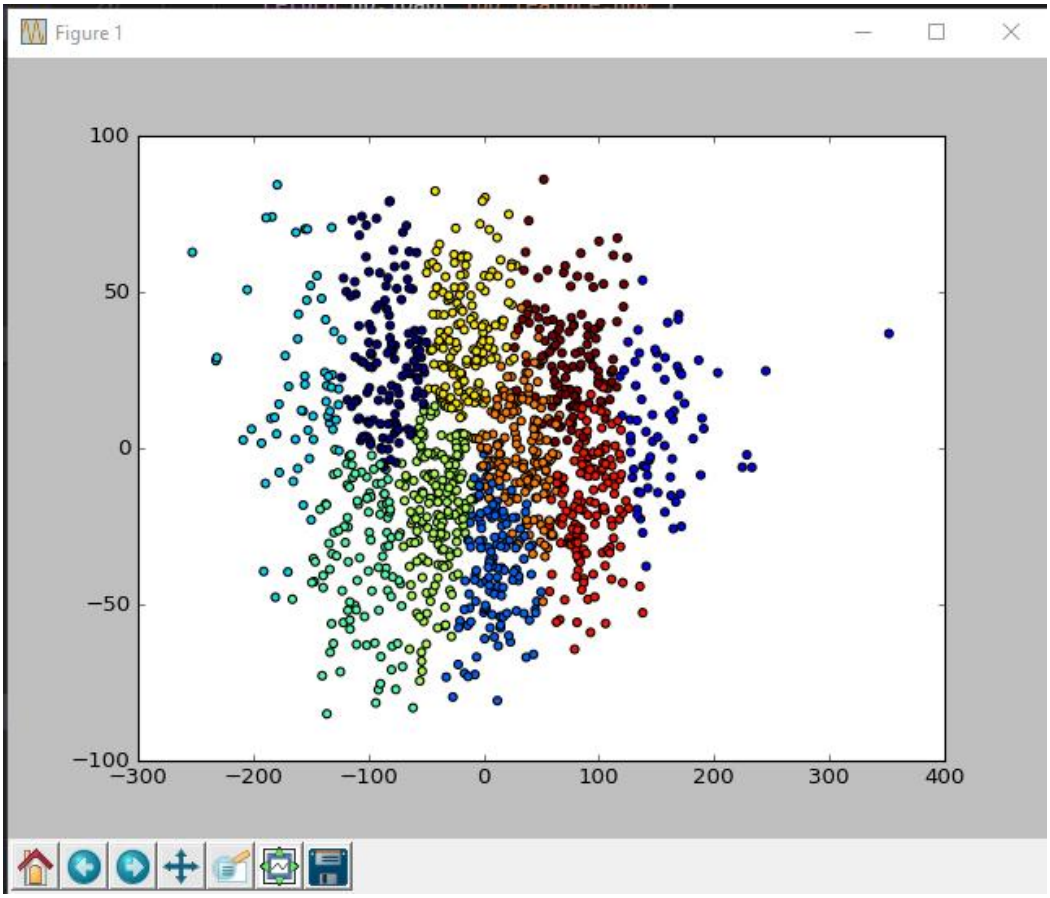
❖ **Kmean:**

- **Các hàm chính:**

```
import matplotlib.pyplot as plt
```

```
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
import training_data

data = training_data.load()
X = PCA(n_components=2).fit_transform(data)
y = KMeans(init='k-means++',
n_clusters=10).fit_predict(data)
plt.scatter(X[:, 0], X[:, 1], c=y)
plt.show()
```

- **Kết quả:**



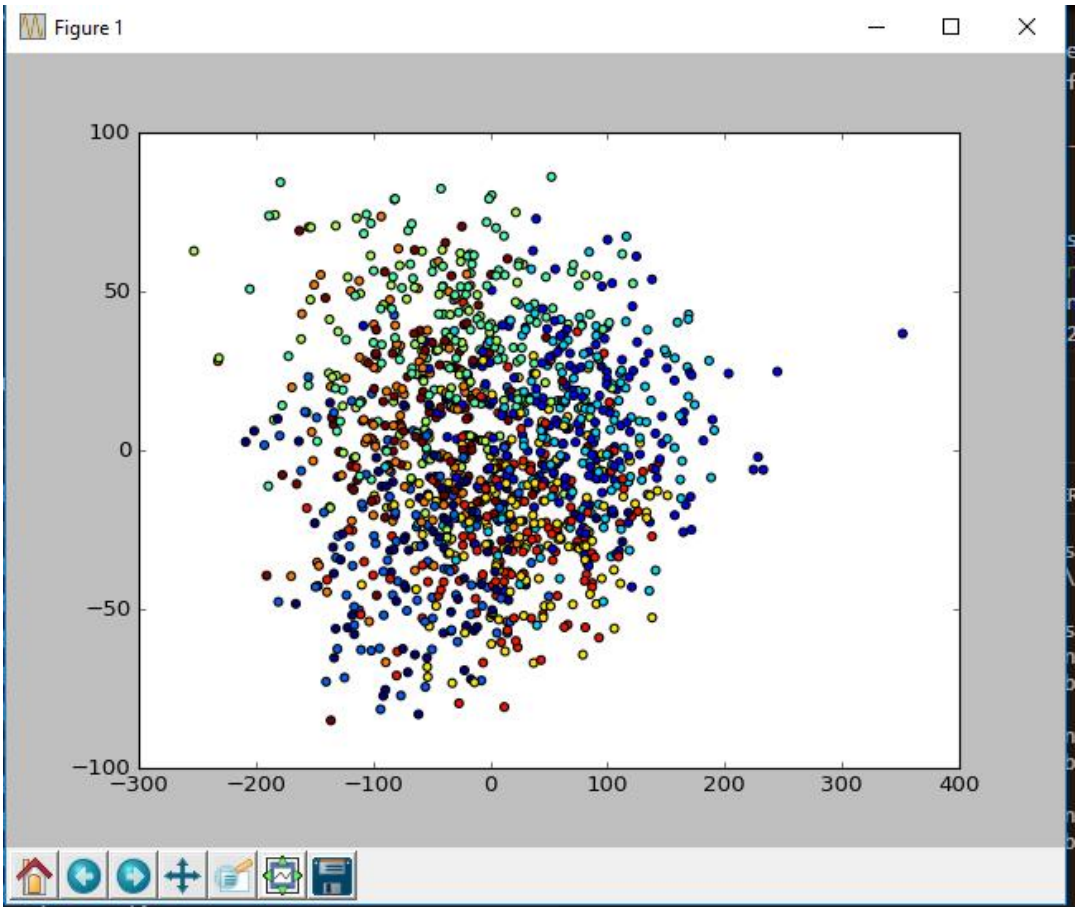
❖ **Spectral Clustering**

- Các hàm chính:

```
#MSSV: 14520178
```

```
# special clustering

print(__doc__)

from time import time
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn import metrics
from sklearn.cluster import KMeans
from sklearn.datasets import load_digits
from sklearn.decomposition import PCA
from sklearn.preprocessing import scale
from sklearn.cluster import spectral_clustering
np.random.seed(42)
digits = load_digits()
X = metrics.pairwise.cosine_similarity(digits.data)
labels = spectral_clustering(X, n_clusters=10, eigen_solver='arpack')
print("Result")
df = pd.DataFrame({'Labels':labels,'Truth labels':digits.target})
ct = pd.crosstab(df['Labels'],df['Truth labels'])
print(ct)
print("----------------------------------------------------------------------")
# visualize
pca_converter = PCA(n_components = 2)
# convert digits data to 2D points
data_2d = pca_converter.fit_transform(digits.data)
plt.scatter(data_2d[:,0], data_2d[:,1], c=labels)
plt.show()
```

- **Kết quả:**



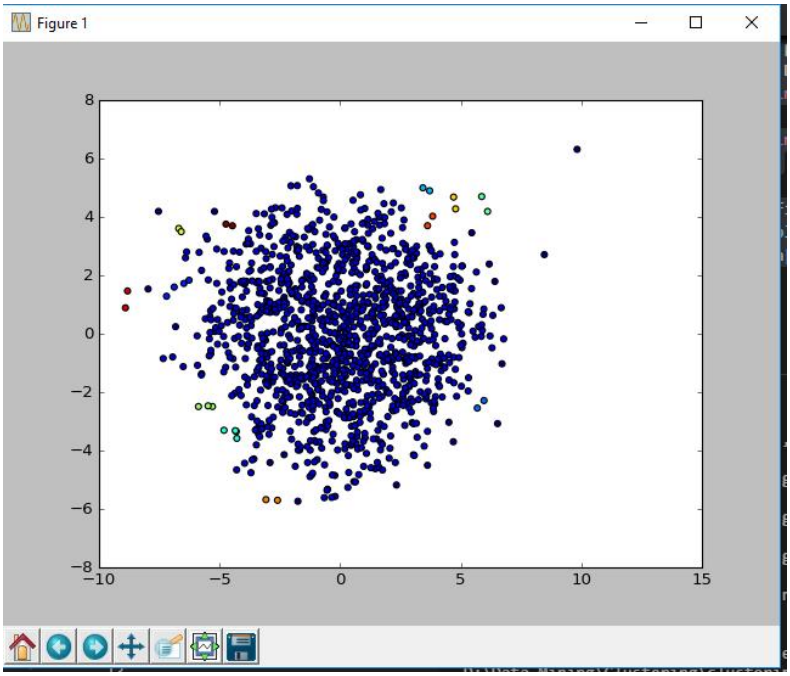
❖ **DBSCAN**

- **Các hàm chính**

```
import matplotlib.pyplot as plt
```

```
from sklearn.cluster import DBSCAN
from sklearn.decomposition import PCA
import training_data
from sklearn.preprocessing import scale
data = training_data.load()
data = scale(data)
data = PCA(n_components=2).fit_transform(data)
y = DBSCAN(eps=0.6, min_samples=2).fit_predict(data)
plt.scatter(data[:, 0], data[:, 1], c=y)
plt.show()
```

- **Kết quả:**



❖ **Agglomerative Clustering**

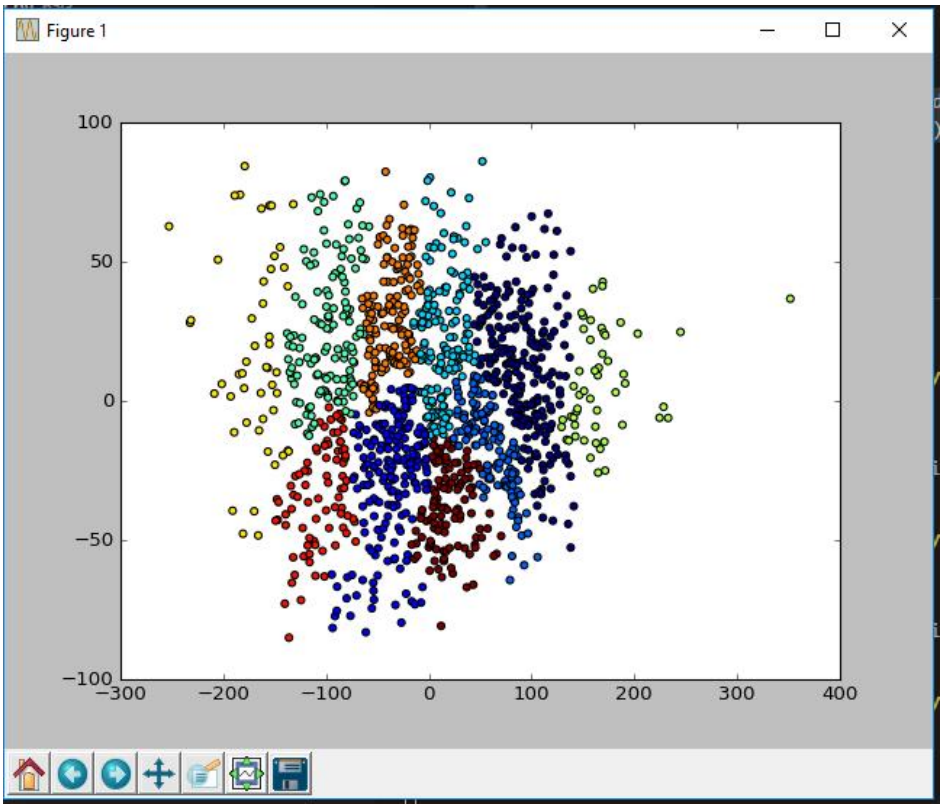
- **Các hàm chính**

```
from matplotlib import pyplot as plt
```

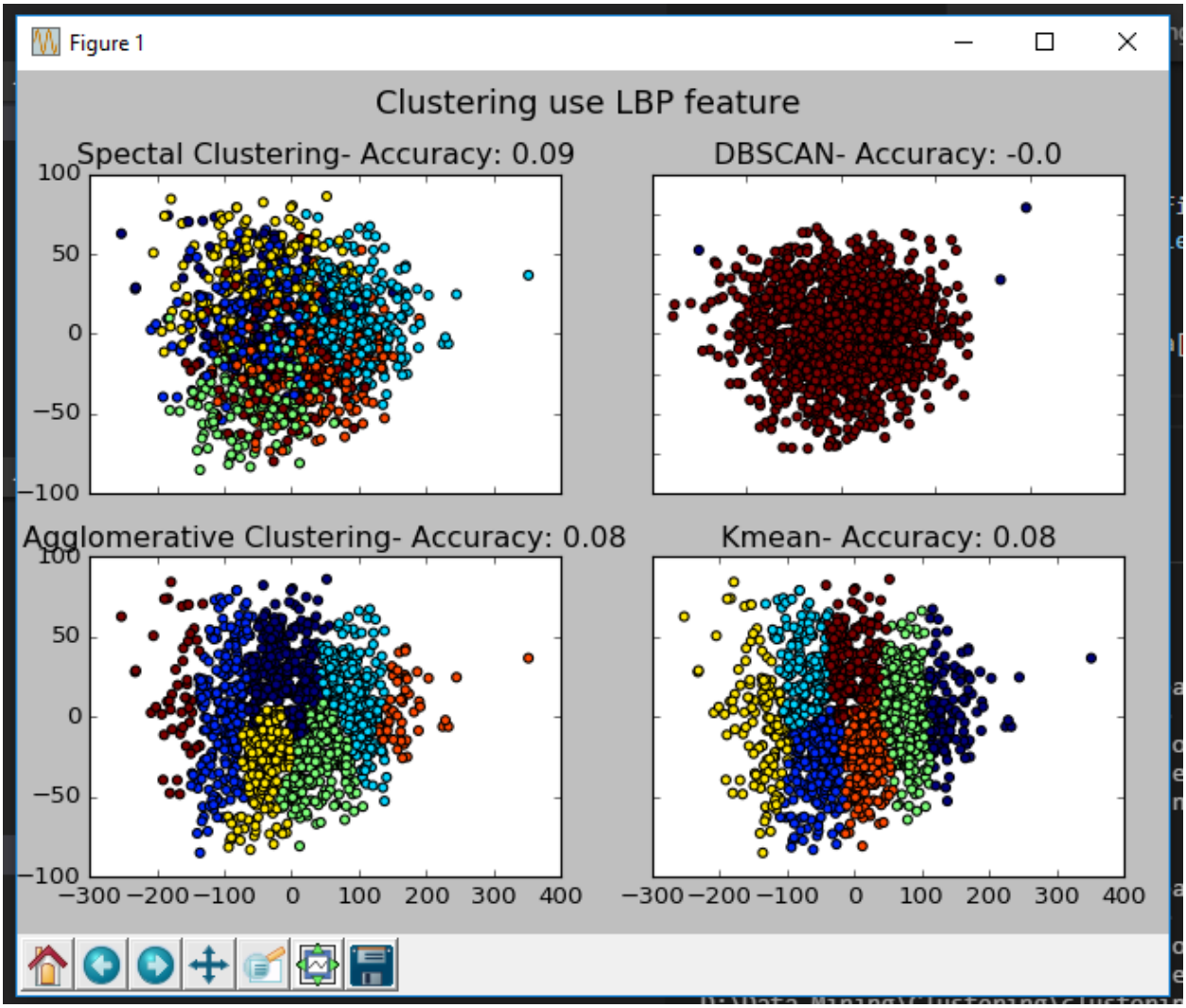
```
import training_data
from sklearn.decomposition import PCA
from sklearn.cluster import AgglomerativeClustering
```

```
data = training_data.load()
data = PCA(n_components=2).fit_transform(data)
y = AgglomerativeClustering(n_clusters=10).fit_predict(data)
plt.scatter(data[:, 0], data[:, 1], c=y)
plt.show()
```

- Kết quả:



❖ **Đánh giá chung vào nhận xét:**



- Nhận xét :

  + Kết quả cáo nhất là Spectral Clustering

  + Kế quả thấp nhất là DBSCAN

d. Ứng dụng 4: Ứng dụng Clustering với tập dữ liệu là face.

- **Phát biểu bài toán:**

+ Input: Tập dữ liệu là tập dữ liệu chuẩn LFW, gồm 7 người và 1228 hình. Feature được sử dụng là Histogram of Oriented Gradients.

+ Output: 7 cluster tương ứng với 7 lable(người), đánh giá kết quả của các giải thuật.

- **Hướng giải quyết**: Sử dụng các thuật toán: Kmeans, Spectral Clustering, DBSCAN, Agglomerative Clustering.

❖ **Kmean:**

- Các hàm chính:

```
#Kmean
```

```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
import hog_feature

data = hog_feature.load()
X = PCA(n_components=2).fit_transform(data)
y = KMeans(init='k-means++', n_clusters=7).fit_predict(data)
plt.scatter(X[:, 0], X[:, 1], c=y)
plt.show()
```

- **Kết quả:**

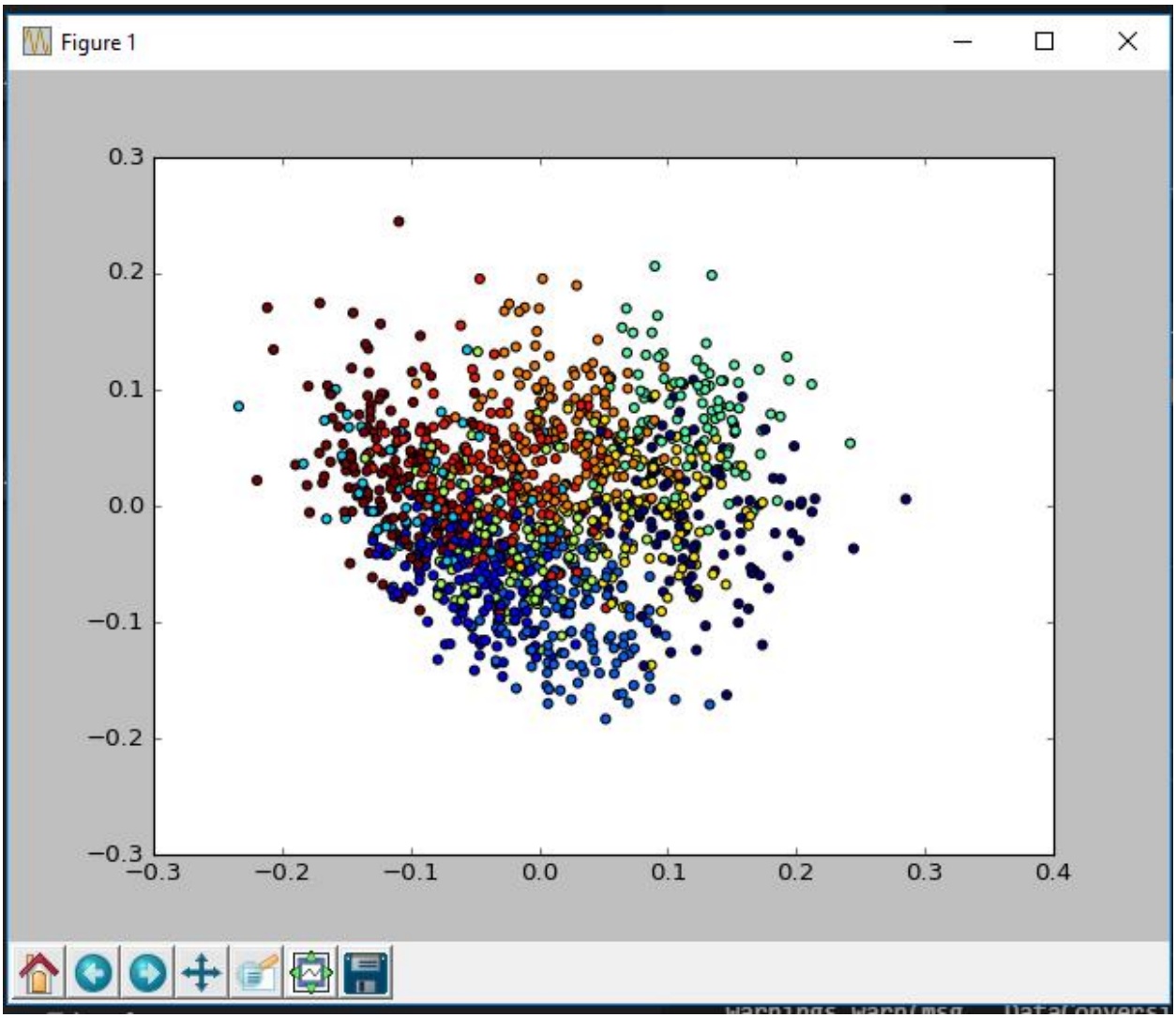


Figure 1
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.3
−0.2
−0.1
0.0
0.1
0.2
0.3
0.4

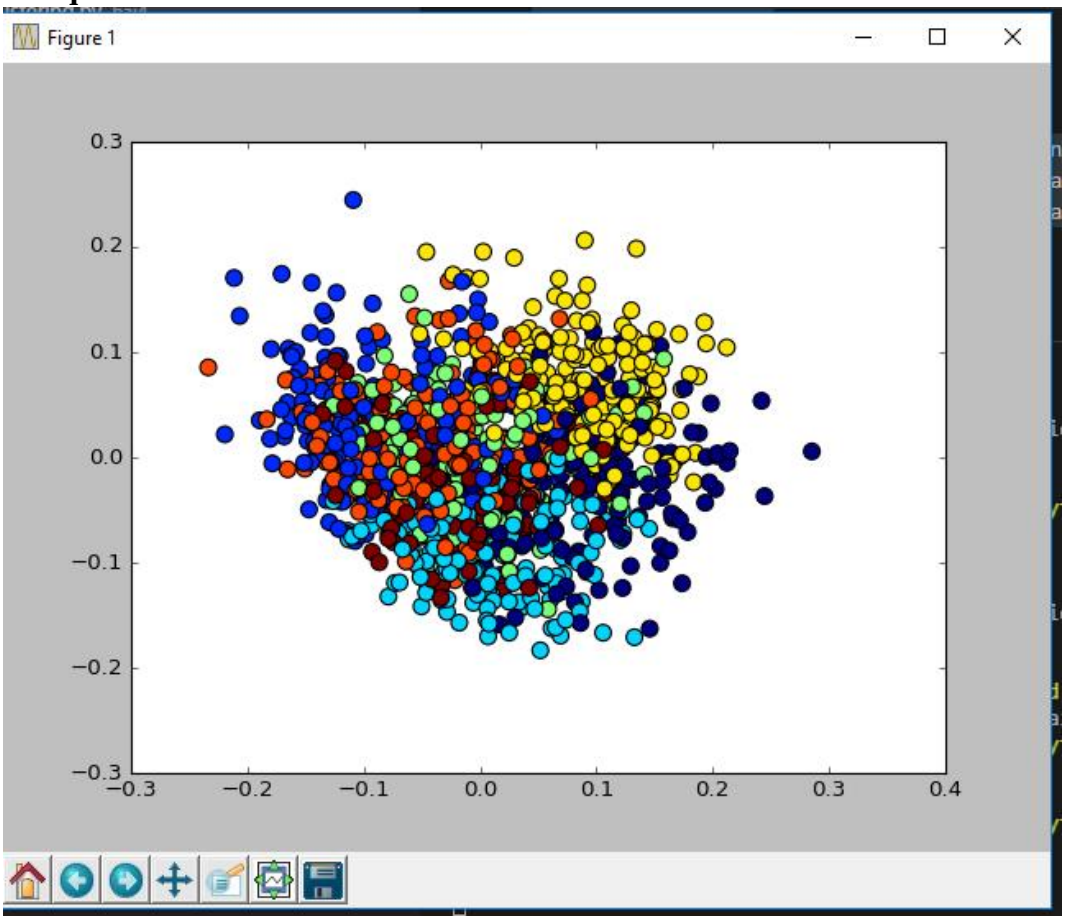
❖ **Spectral Clustering**

- Các hàm chính

```
import matplotlib.pyplot as plt
```

```
from sklearn.cluster import SpectralClustering
from sklearn.decomposition import PCA
from sklearn.metrics.pairwise import cosine_similarity
import hog_feature

data = hog_feature.load()
graph_data = cosine_similarity(data)
y = SpectralClustering(n_clusters=7, eigen_solver='arpack',
affinity='precomputed').fit_predict(graph_data)
reduced_data = PCA(n_components=2).fit_transform(data)
plt.scatter(reduced_data[:, 0], reduced_data[:, 1],s=80, c=y)
plt.show()
```

- **Kết quả**



❖ **DBSCAN**

- **Các hàm chính:**

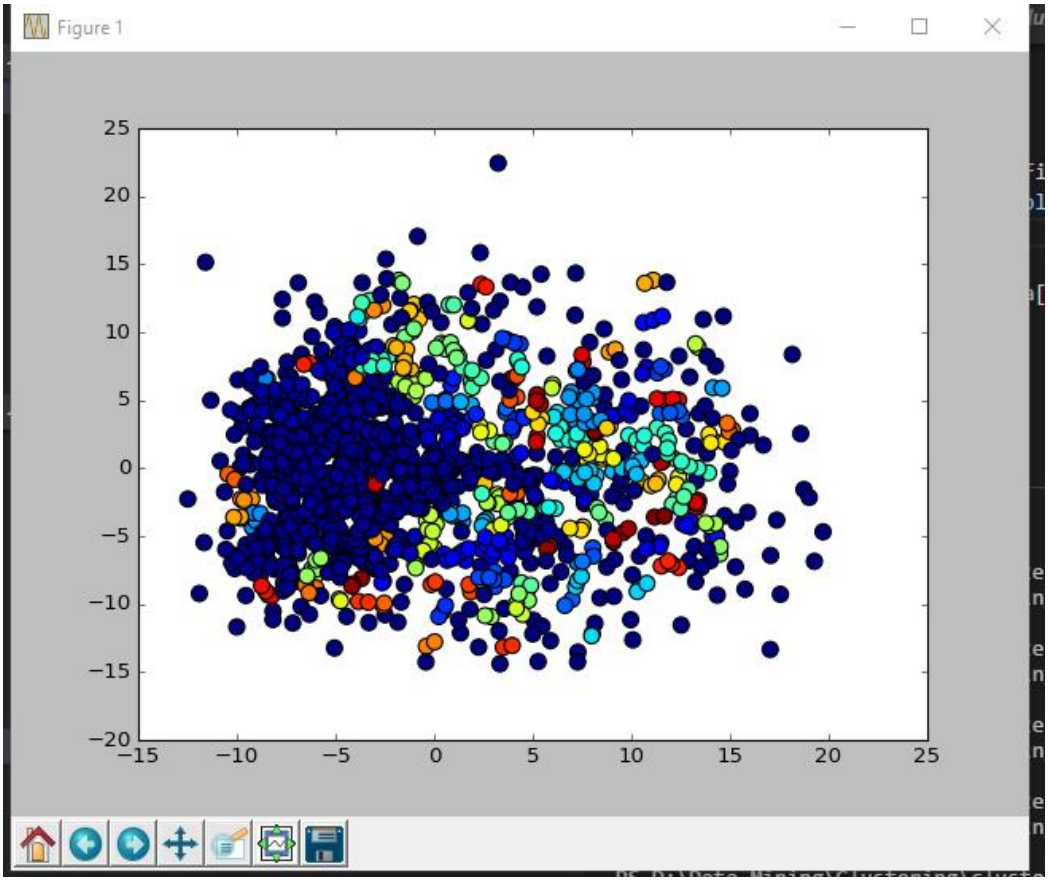
```
import matplotlib.pyplot as plt
```

```
from sklearn.cluster import DBSCAN
from sklearn.decomposition import PCA
import hog_feature
from sklearn.preprocessing import scale
```

```
data = hog_feature.load()
data = scale(data)
data = PCA(n_components=2).fit_transform(data)
y = DBSCAN(eps=40, min_samples=2).fit_predict(data)
```

```
plt.scatter(data[:, 0], data[:, 1],s=80, c=y)
plt.show()
```

- **Kết quả**:



❖ **Agglomerative Clustering**

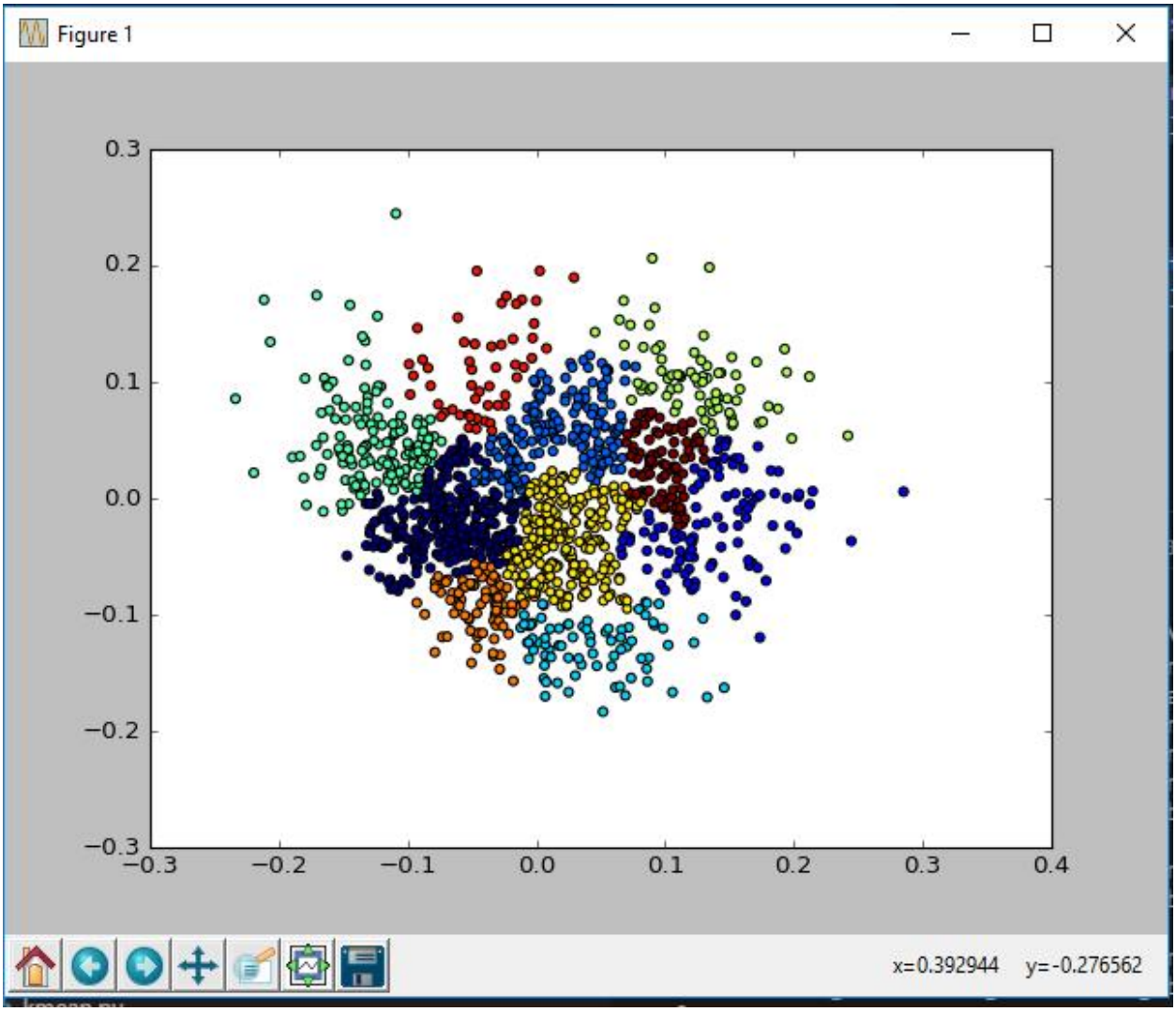
- Các hàm chính

```
from matplotlib import pyplot as plt
```

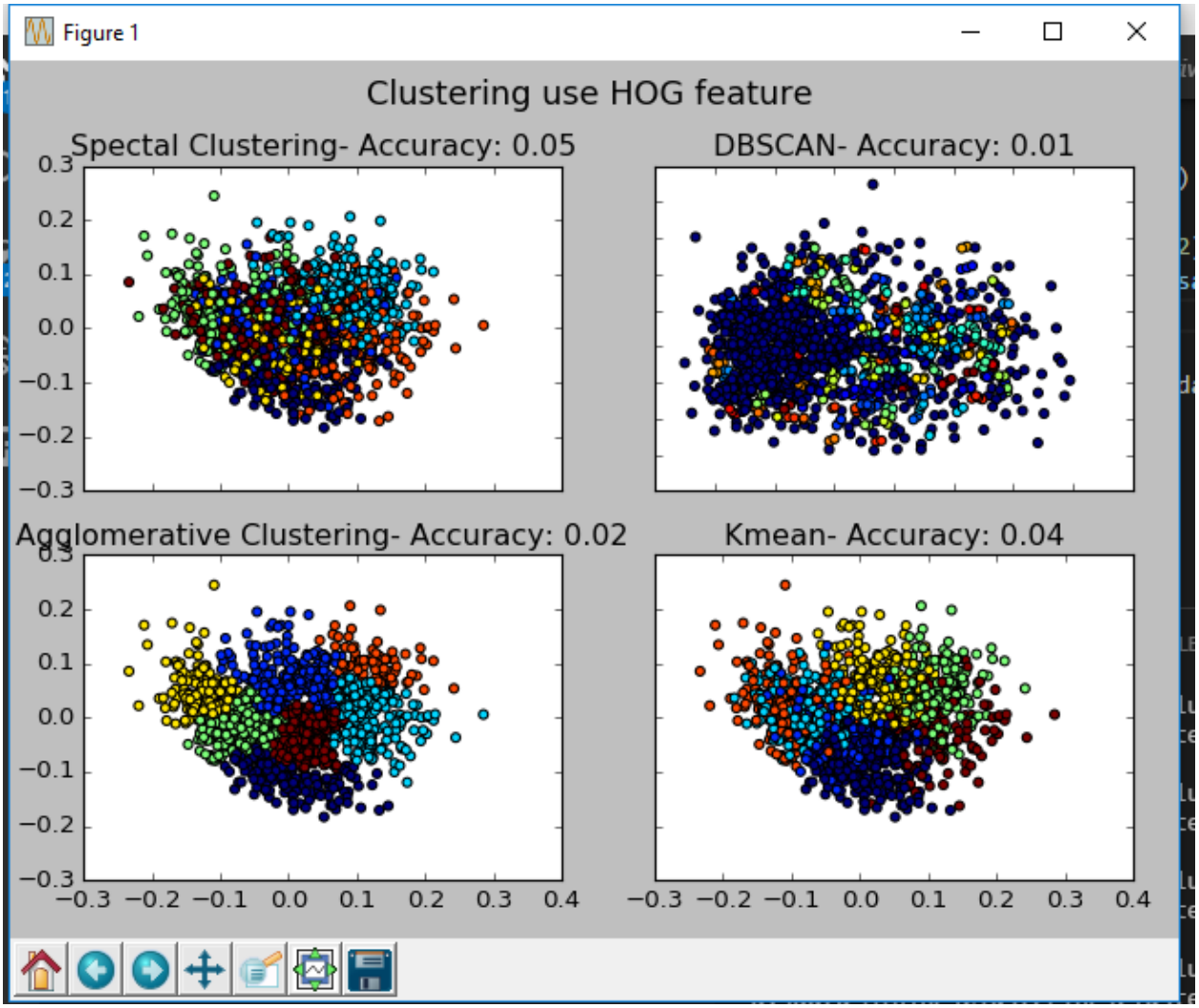
```
import hog_feature
from sklearn.decomposition import PCA
from sklearn.cluster import AgglomerativeClustering
```

```
data = hog_feature.load()
data = PCA(n_components=2).fit_transform(data)
y = AgglomerativeClustering(n_clusters=10).fit_predict(data)
plt.scatter(data[:, 0], data[:, 1], c=y)
plt.show()
```

- **Kết quả:**



❖ **Đánh giá chung và nhận xét:**



- Nhận xét: Với HOG feature trên tập dữ liêu face. Thuật toán Spectral Clustering có hiệu xuất cao nhất và thấp nhất là DBSCAN

5. TỔNG KẾT VÀ ĐÁNH GIÁ ƯU ĐIỂM VÀ NHƯỢC ĐIỂM CỦA CÁC THUẬT TOÁN CLUSTERING

a. KMEAN

- Ưu điểm:
- Dễ thực hiện
- Thời gian tính toán nhanh, chỉ O(n), với n là số điểm.
- Có khả năng thực hiện tính toán trên data lớn
- Nhược điểm: Nhạy cảm với hình dạng của các điểm
- Yêu cầu nhập số cluster trước.
- Hiệu suất phụ thuộc vào điểm khởi tạo ban đầu của mỗi cluster

b. Spectral Clustering

- Ưu điểm:
- Dễ thực hiện
- Thường cho kết quả tốt
- Nhược điểm:
- Phụ thuộc vào việc chọn các biến đầu vào : số cluster,…
- Chi phí tính toán cao

c. DBSCAN

- Ưu điểm
- Không cần nhập trước số cluster
- Sử lý tốt với các cluster có hình dạng không bình thường.
- Nhược điểm
- Phụ thuộc vào công thức tính khoảng cách.
- Với những cluster có sự khác nhau quá lớn về mật độ

d. Agglomerative Clustering

- Ưu điểm
- Nhược điểm:
- Chi phí tính toán cao, O(n^2).

6. SOURCE CODE

Link github: https://github.com/oscardoan/clustering.git

7. TÀI LIỆU THAM KHẢO:

https://www.slideshare.net/ishmecse13/hierarchical-clustering-38276870

https://www.learnopencv.com/histogram-of-oriented-gradients/

http://scikit-learn.org/stable/modules/generated/sklearn.metrics.adjusted_rand_score.html

https://123doc.org/document/2637050-ky-thuat-gom-nhom-van-ban-trong-linh-vuc-khai-pha-tri-thuc.htm

https://github.com/HoangTrinh/Machine-Leaning-in-Computer-Vision

http://scholar.vimaru.edu.vn/sites/default/files/thinhnv/files/dm_-_chapter_5_-_clustering.pdf

https://www.google.com.vn/url?sa=t&rct=j&q=&esrc=s&source=web&cd=5&cad=rja&uact=8&ved=0ahUKEwiT6dma2P_WAhWBMpQKHU-LAksQFgg-MAQ&url=http%3A%2F%2Fwww1.napa.vn%2Fepa%2Fwp-

content%2Fuploads%2Fsites%2F3%2F2016%2F01%2FNhan-dang-bien-so-xe-su-dung-dac-trung-LBP1.doc&usg=AOvVaw3k87VIXDNDeqZnQCCMdC4Q

http://www.picvietnam.com/forum/showthread.php?p=84771