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

BÁO CÁO BÀI THỰC HÀNH 1

Lớp: KHTN2014

GVLT: Lê Đình Duy

GVHDTH: Mai Tiến Dũng

SVTH: Lê Thị Ngọc Thúy - 14520932

TP HCM, ngày 20 tháng 10 năm 2017

Bài thực hành số 1

Mục lục

Github: https://github.com/thuyltn2408/homework1

Bài tập 1: KMeans trên tập dữ liệu ngẫu nhiên

** Bước 1: Phát sinh dữ liệu

- Sử dụng sklearn.dataset.make_bobs để phát sinh dữ liệu ngẫu nhiên với số cluster là 2
- Visualize dữ liệu đã phát sinh

** Bước 2: Áp dụng KMeans

- Dùng sklearn.cluster.KMeans để clustering dữ liệu trên với số cluster bằng 2

** Bước 3: Visualize

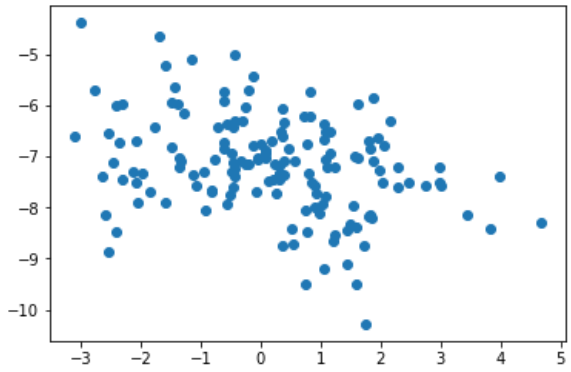
- Dùng pandas.scatter để visualize dữ liệu và centerPoints của clusters

In [119]:

```
# Import thư viện
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs
import matplotlib.pyplot as plt
import sklearn.datasets as datasets
import pandas as pd
%matplotlib inline
#plt.gray()
```

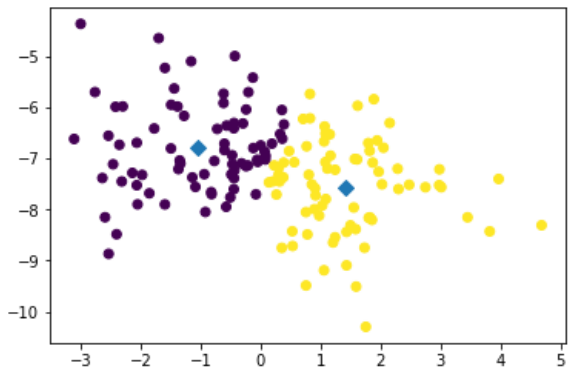
In [120]:

```
# Bước 1
#      1.1: Phát sinh dữ liệu với 150 mẫu, số lớp: 2
X,Y = make_blobs(n_samples=150, n_features=2, centers=2)
```



In [122]:

```
# Bước 2: Áp dụng KMeans clustering trên dữ liệu đã phát sinh
model = KMeans(n_clusters=2)
labels1 = model.fit_predict(X)
```



Bài tập 2: KMeans, Spectral, DBSCAN, Agglomerative clustering trên Digits data

In [124]:

```
# Import thư viện
from sklearn.cluster import spectral_clustering
from sklearn.cluster import AgglomerativeClustering
from sklearn.feature_extraction import image
import numpy as np
from sklearn.neighbors import DistanceMetric
from sklearn.metrics.pairwise import cosine_similarity

import numpy as np
from sklearn.decomposition import PCA

from sklearn.cluster import DBSCAN
```

2.1 Load dataset

In [125]:

```
# Load dataset
digits = datasets.load_digits()
print(digits.data.shape)
```

```
(1797, 64)
```

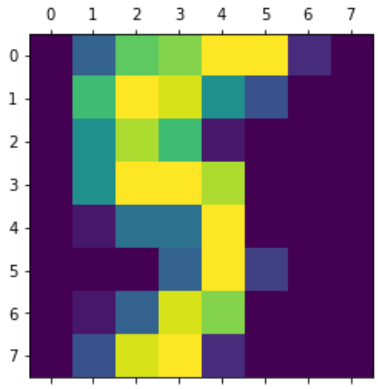
2.2 KMeans trên dữ liệu

In [126]:

```
# KMeans trên digits data
model2 = KMeans(n_clusters=10)
labels = model2.fit_predict(digits.data)
```

- Thống kê kết quả
- Cột đầu tiên: nhãn mà KMeans clustering đã gán vào dữ liệu
- Hàng đầu tiên: nhãn đúng của dữ liệu
- Ý nghĩa phần tử hàng i, cột j , i và j > 0, phần tử có giá trị là value:
  - Có value ảnh mà clustering gán nhãn i có nhãn thật sự là j

Truth labels	0	1	2	3	4	5	6	7	8	9
labels										
0	0	24	148	0	0	0	0	0	3	0
1	0	0	3	6	10	0	0	168	4	8
2	0	2	0	0	0	1	177	0	2	0
3	0	54	2	0	2	0	0	9	10	20
4	0	100	8	7	2	0	3	2	101	1
5	0	0	2	12	0	48	0	0	48	139
6	176	0	1	0	0	0	1	0	0	0
7	0	1	13	156	0	2	0	0	2	7
8	0	1	0	2	0	129	0	0	4	5
9	2	0	0	0	167	2	0	0	0	0



2.3 Spectral clustering

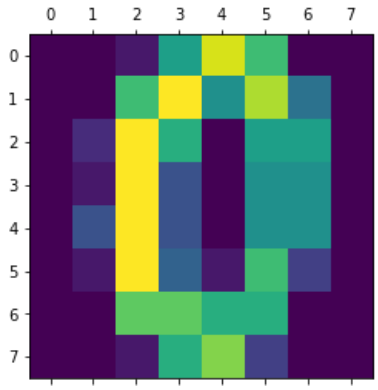
In [201]:

```
# Tính ma trận độ tương đồng của data
graph = cosine_similarity(digits.data)

# Áp dụng Spectral clustering cho data
labels_spectral = spectral_clustering(graph, n_clusters=10)
```

Truth labels	0	1	2	3	4	5	6	7	8	9
labels										
0	0	36	115	4	0	0	0	0	1	0
1	0	2	0	0	0	2	172	0	13	0
2	0	0	2	2	11	0	0	154	2	2
3	0	0	1	147	0	0	0	0	6	2
4	177	0	1	0	1	1	0	0	0	3
5	1	0	0	0	163	2	0	0	0	0
6	0	58	5	5	1	0	0	15	40	36
7	0	0	0	16	0	20	2	0	7	133
8	0	86	53	5	5	0	7	10	102	1
9	0	0	0	4	0	157	0	0	3	3

```
lables_predict: 4
 True:  0
```



2.4 DBSCAN clustering

In [132]:

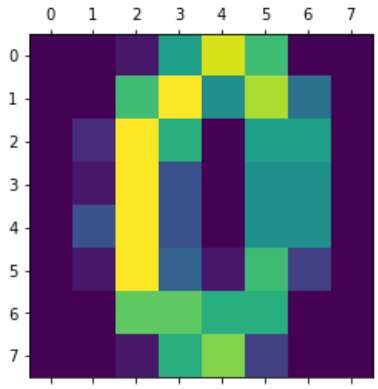
```
# Gán thông số cho DBSCAN clustering
eps = 0.0595
min_samples = 10

# Áp dụng DBSCAN cho data
dbscan= DBSCAN(eps=eps, min_samples =min_samples,metric='cosine')
labels_dbscan = dbscan.fit_predict(digits.data)

# Thống kê kết quả
df_dbscan = pd.DataFrame({'labels':labels_dbscan,'Truth labels':digits.target})
ct_dbscan=pd.crosstab(df_dbscan['labels'],df_dbscan['Truth labels'])
print(ct_dbscan)
```

```
Truth labels    0    1    2    3    4   5    6    7   8    9
labels
-1              7   13   41   49   35  68    5   52  81   78
 0            171    0    0    0    0   0    0    0   0    0
 1              0  143    0    0    0   0    1    0  93    1
 2              0    0    0    0    0   0  175    0   0    0
 3              0    0    0  134    0   1    0    0   0  101
 4              0    0    0    0  146   0    0    0   0    0
 5              0    0  136    0    0   0    0    0   0    0
 6              0    0    0    0    0  65    0    0   0    0
 7              0    0    0    0    0   0    0  127   0    0
 8              0    0    0    0    0  48    0    0   0    0
 9              0   26    0    0    0   0    0    0   0    0
```

```
lables_predict: 0
 True:  0
```



2.5 AgglomerativeClustering

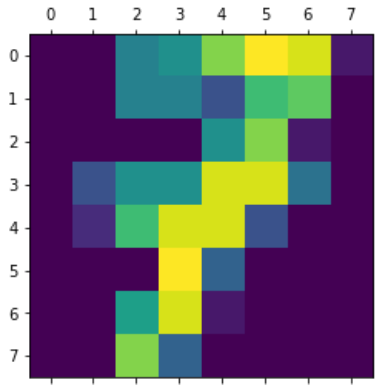
In [203]:

```
# Áp dụng Agglomerative clustering
model = AgglomerativeClustering(n_clusters = 10, affinity = 'euclidean')
labels_Agg = model.fit_predict(digits.data)

# Thống kê kết quả
df=pd.DataFrame({'labels':labels_Agg,'Truth labels':digits.target})
ct = pd.crosstab(df['labels'],df['Truth labels'])
print(ct)
```

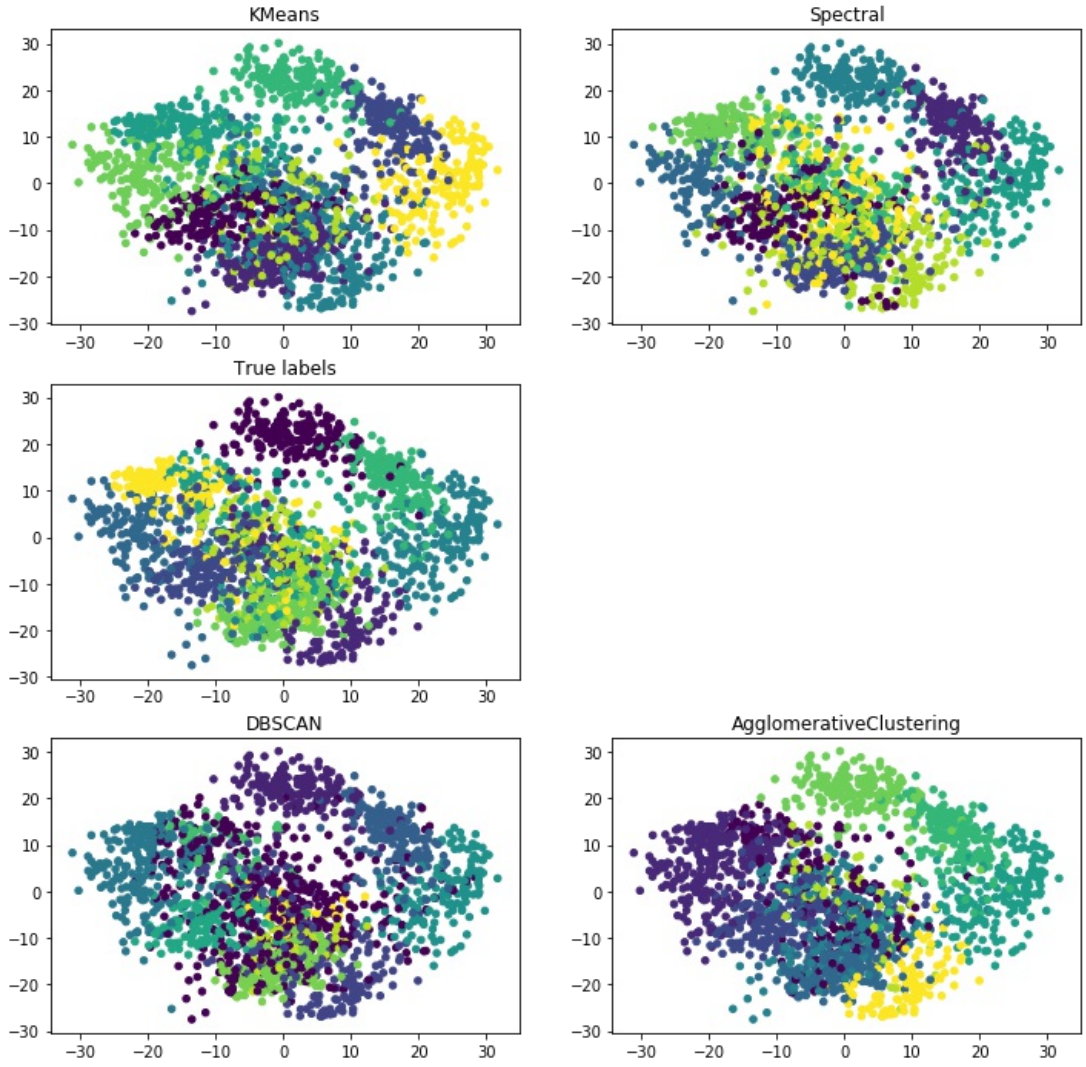
```
Truth labels    0   1    2    3    4    5    6    7    8    9
labels
0               0   0    0    0    0  179    0    0    0    2
1               0   0    0  169    0    2    0    0    1  145
2               0  27  166    0    0    0    0    0    4    0
3               0   0    1    1    3    0    0  179    1   11
4               0   0   10   13    0    0    1    0  165    2
5               0   0    0    0  178    0    0    0    0    0
6               0   0    0    0    0    1  180    0    0    0
7             178   0    0    0    0    0    0    0    0    0
8               0  59    0    0    0    0    0    0    1   20
9               0  96    0    0    0    0    0    0    2    0
```

```
lables_predict: 3
 True:  7
```



2.6 Visualize

```
<matplotlib.text.Text at 0xf543668>
```



2.7 Evaluation

- Sử dụng sklearn.metrics để đánh giá kết quả của các phương pháp clustering trên tập dữ liệu là các ảnh số viết tay (Digits data)

In [137]:

```
# Import thư viện để đánh giá kết quả
from sklearn.metrics.cluster import adjusted_mutual_info_score
from sklearn.metrics.cluster import mutual_info_score
from sklearn.metrics.cluster import homogeneity_completeness_v_measure
```

2.7.1 Thực hiện đáng giá theo adjusted_mutual_info_score

```
KMeans evaluation:  0.732275121833
Spectral evaluation:  0.710701587724
DBSCAN evaluation:  0.699846081661
AgglomerativeClustering evaluation:  0.856084675987
```

2.7.2 Thực hiện đáng giá theo mutual_info_score

```
KMeans evaluation:  1.69215855823
Spectral evaluation:  1.6429746714
DBSCAN evaluation:  1.61911644797
AgglomerativeClustering evaluation:  1.97440556938
```

2.7.3 Thực hiện đáng giá theo homogeneity_completeness_v_measure

- Giá trị trả về trong khoảng 0 >> 1
- Càng về 1 thì độ khớp của True labels và cluster labels càng cao.

```
KMeans evaluation:  (0.7349289161099819, 0.74309789241362245, 0.73899082951097061)
Spectral evaluation:  (0.71356764327722144, 0.71787431828502479, 0.71571450219165855)
DBSCAN evaluation:  (0.70320567205291673, 0.73876562855409511, 0.72054718439720333)
AgglomerativeClustering evaluation:  (0.85751287195047232, 0.87909558517241981, 0.86817011269090827)
```

2.8 Nhận xét

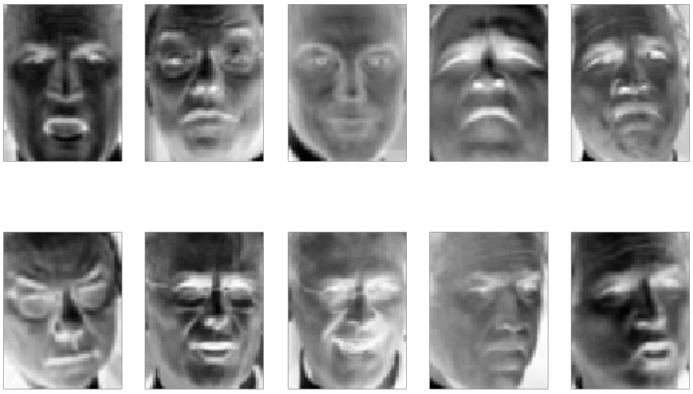
- Đối với data là chữ số viết tay (Digits data) thì Agglomerative clustering hiệu quả hơn hẳn so với KMeans, Spectral, DBSCAN clustering
- DBSCAN clustering: khó sử dụng bởi parameters: eps và min_samples. Thử nhiều lần giá trị của eps và min_sample mới cho kết quả khả quan.

Bài tập 3: KMeans, Spectral, DBSCAN, Agglomerative clustering trên Faces data

- 3.1 Load dataset: Faces data
- 3.2 Rút trích LBP feature
- 3.3 KMeans clustering
- 3.4 Spectral clustering
- 3.5 DBSCAN clustering
- 3.6 Agglomerative clustering
- 3.7 Visualize
- 3.8 Evaluation
- 3.9 Nhận xét

3.1 Load Faces datsaset

```
Number of images:  (1140, 62, 47)
Number of classes:  5
```

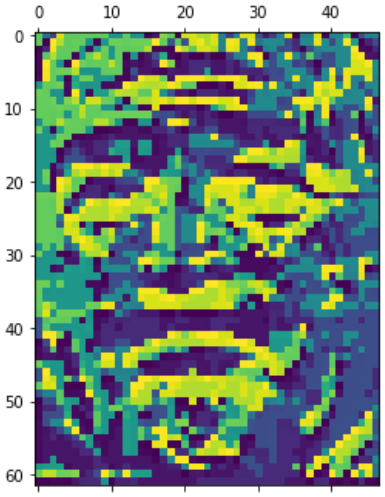


3.2 Rút trích feature LBP

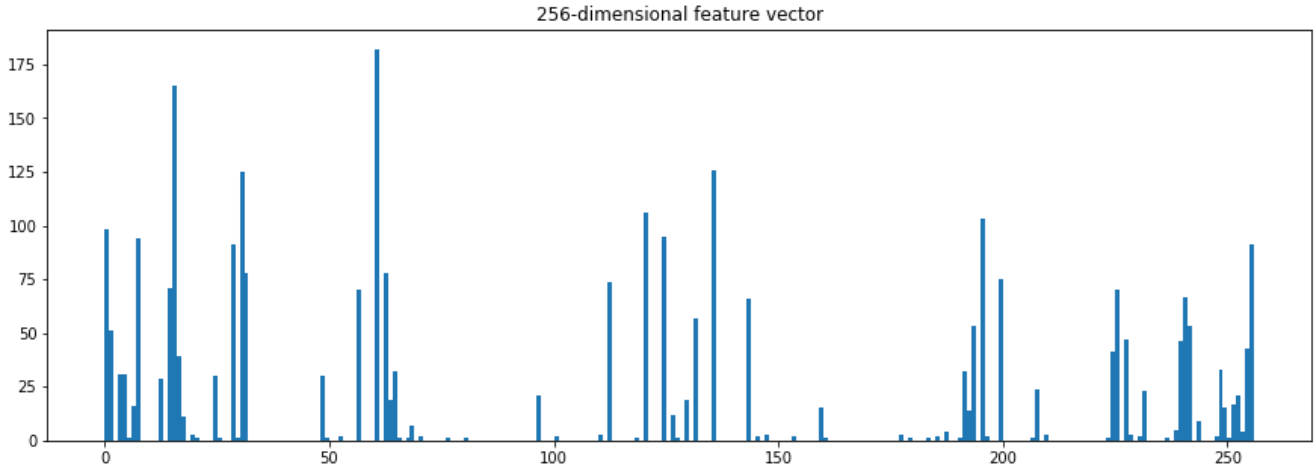
In [143]:

```
# Rút trích LBP feature cho ảnh đầu tiên của dataset
featureLBP = local_binary_pattern(faces_data.images[0], P=8, R=0.5)
```

```
<matplotlib.image.AxesImage at 0xb2729b0>
```




```
[[ 193.   96.   48. ...,   64.  112.  112.]
 [ 193.    0.  255. ...,    0.   24.   56.]
 [   0.   28.  251. ...,    7.    4.   28.]
 ...,
 [ 135.  143.  143. ...,  254.  255.   96.]
 [ 199.  231.  231. ...,  160.  243.   96.]
 [   1.    1.    3. ...,   31.    3.    0.]]
```



In [147]:

```
# Xây dựng hàm rút trích LBP feature cho 1 ảnh
def extractFeatureLBP(image):
    featureLBP = local_binary_pattern(image, P=8, R=0.5)
    return np.histogram(featureLBP, bins=list(range(257)))[0]
```

In [148]:

```
# Rút trích feature LBP trên tập data
featureLBP = list(map(extractFeatureLBP, faces_data.images))
```

In [149]:

```
featureLBP = np.array(featureLBP)
type(featureLBP)
print(featureLBP.shape)
print(featureLBP[0:2])
```

```
(1140, 256)
[[ 98  51   0  31  31   1  16  94   0   0   0   0  29   0  71 165  39  11
    0   3   1   0   0   0  30   1   0   0  91   1 125  78   0   0   0   0
    0   0   0   0   0   0   0   0   0   0   0   0  30   1   0   0   2   0
    0   0  70   0   0   0 182   0  78  19  32   1   0   1   7   0   2   0
    0   0   0   0   1   0   0   0   1   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0  21   0   0   0   2   0   0   0   0   0   0   0
    0   0   3   0  74   0   0   0   0   0   1   0 106   0   0   0  95   0
   12   1   0  19   0  57   0   0   0 126   0   0   0   0   0   0   0  66
    0   2   0   3   0   0   0   0   0   2   0   0   0   0   0  15   1   0
    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   3   0   1
    0   0   0   1   0   2   0   4   0   0   1  32  14  53   0 103   2   0
    0  75   0   0   0   0   0   0   1  24   0   3   0   0   0   0   0   0
    0   0   0   0   0   0   0   1  41  70   0  47   3   0   2  23   0   0
    0   0   1   0   5  46  67  53   0   9   0   0   0   2  33  15   1  17
   21   4  43  91]
 [ 83  41   1  25  39   2  14  85   1   0   0   0  17   0  67 131  38   5
    0   3   1   0   0   1  19   1   0   2  95   0 159  78   0   0   0   0
    0   0   0   0   0   0   0   0   1   0   0   0  17   1   0   0   0   0
    0   0  46   2   0   0 225   0  66   8  51   2   0   0   7   0   1   0
    0   0   0   0   1   0   1   2   2   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0  18   0   0   0   2   0   0   0   0   0   0   0
    2   0   2   0  80   1   0   0   0   0   0   0 100   1   0   0  58   0
   22   3   1  17   0  54   0   0   0 134   0   0   0   0   1   0   1  36
    0   1   0   2   0   0   0   0   0   0   0   2   0   0   0  23   0   0
    0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   2
    0   0   0   0   0   2   0   3   0   1   1  29  20  87   0 152   0   0
    3  70   0   0   0   0   1   0   1  15   0   1   0   0   0   0   0   0
    0   1   0   1   0   0   0   6  34 109   0  44   2   0   0  16   0   1
    0   2   0   0   4  38  91  54   0  15   0   0   0   0  48  14   0  18
   16   1  24  84]]
```

3.3 KMeans cluster

In [150]:

```
# Áp dụng KMeans trên Faces dataset
# Với dữ liệu đầu vào là LBP feature
model = KMeans(n_clusters=5)
labels = model.fit_predict(featureLBP)
print(labels)

# Thống kê kết quả
df = pd.DataFrame({'label':labels, 'True Label':faces_data.target})
ct = pd.crosstab(df['label'], df['True Label'])
print(ct.tail(10))
```

```
[2 1 3 ..., 0 0 1]
True Label   0   1    2   3   4
label
0           73   8   79  23  32
1           35  23   95  25  21
2           75  43  101  24  38
3           26  16  148  13  28
4           27  31  107  24  25
```

3.4 Spectral cluster

In [151]:

```
# Tính ma trận độ tương đồng của data
graph = cosine_similarity(featureLBP)

# Áp dụng Spectral clustering cho data
labels_spectral = spectral_clustering(graph, n_clusters=5)

# Thống kê kết quả
df1=pd.DataFrame({'labels':labels_spectral,'Truth labels':faces_data.target})
ct2=pd.crosstab(df1['labels'],df1['Truth labels'])
print(ct2)
```

```
Truth labels   0   1    2   3   4
labels
0             58  17   63  40  43
1             32  32  121  18  24
2             52  11   86  28  35
3             15  14  144  11  23
4             79  47  116  12  19
```

3.5 DBSCAN

In [152]:

```
# Gán thông số cho cluster
eps = 50
min_samples = 10

# Áp dụng DBSCAN cho data
dbscan= DBSCAN(eps=eps, min_samples =min_samples,metric='cosine')
labels_dbscan = dbscan.fit_predict(featureLBP)

# Thống kê kết quả
df_dbscan = pd.DataFrame({'labels':labels_dbscan,'Truth labels':faces_data.target})
ct_dbscan=pd.crosstab(df_dbscan['labels'],df_dbscan['Truth labels'])
print(ct_dbscan)
```

```
Truth labels    0    1    2    3    4
labels
0             236  121  530  109  144
```

3.6 AgglomerativeClustering

In [153]:

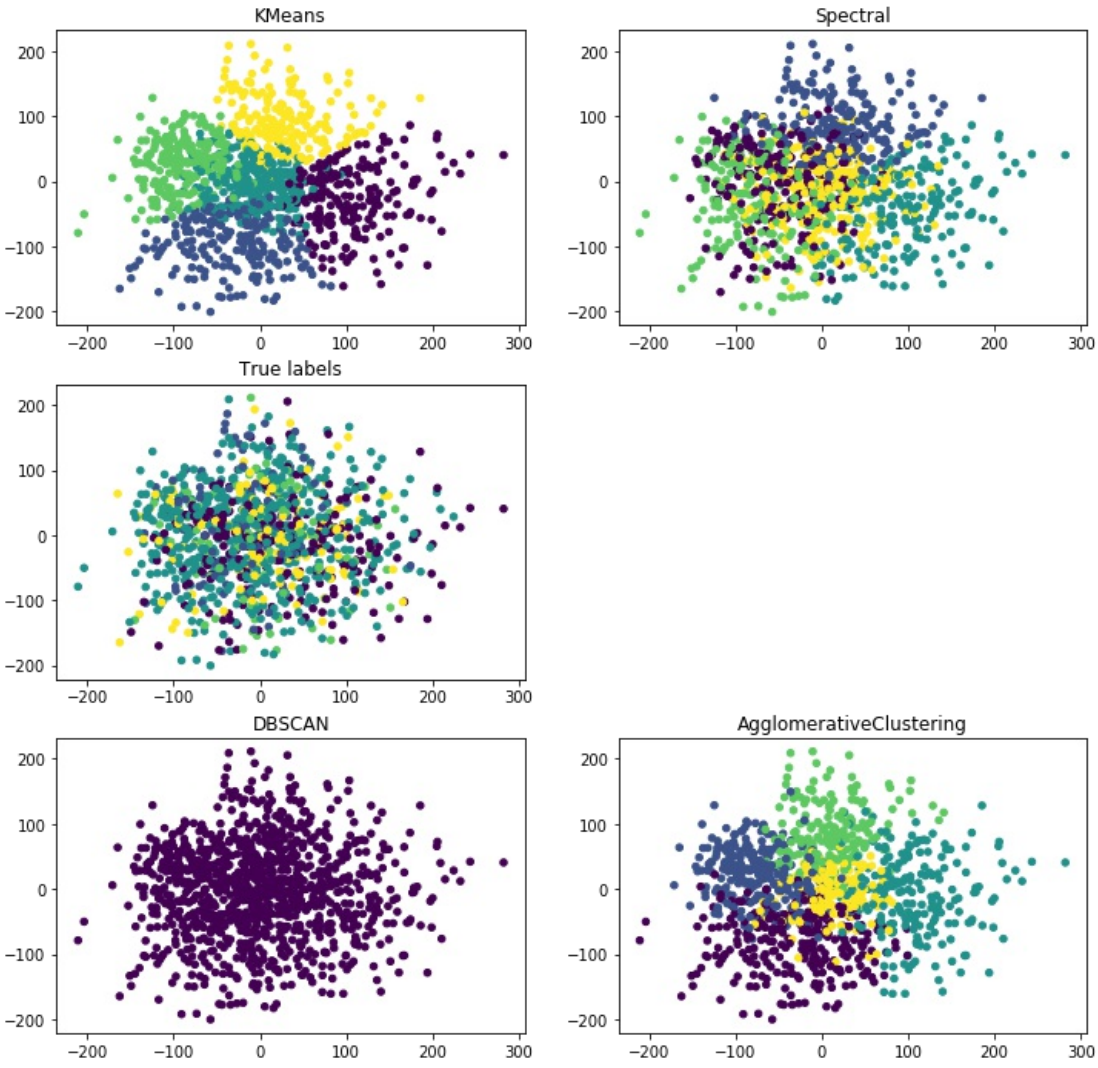
```
model = AgglomerativeClustering(n_clusters = 5, affinity = 'euclidean')
labels_Agg = model.fit_predict(featureLBP)

# Thống kê kết quả
df=pd.DataFrame({'labels':labels_Agg,'Truth labels':faces_data.target})
ct = pd.crosstab(df['labels'],df['Truth labels'])
print(ct)
```

```
Truth labels   0   1    2   3   4
labels
0             52  35  111  26  32
1             58  28  129  25  30
2             73  11   79  26  34
3             25  30  102  21  30
4             28  17  109  11  18
```

3.7 Visualize

```
<matplotlib.text.Text at 0xf3479b0>
```



3.8 Evaluation

In [155]:

```
#Thực hiện đáng giá theo adjusted_mutual_info_score

# KMeans
print("KMeans evaluation: ",adjusted_mutual_info_score(faces_data.target, labels))

# Spectral cluster
print("Spectral evaluation: ",adjusted_mutual_info_score(faces_data.target, labels_spectral))

# DBSCAN
print("DBSCAN evaluation: ",adjusted_mutual_info_score(faces_data.target, labels_dbscan))

# AgglomerativeClustering
print("AgglomerativeClustering evaluation: ",adjusted_mutual_info_score(faces_data.target, labels_Agg))
```

```
KMeans evaluation:  0.0231504281237
Spectral evaluation:  0.039274635648
DBSCAN evaluation:  6.18551703224e-16
AgglomerativeClustering evaluation:  0.0119017358518
```

In [156]:

```
# Thực hiện đáng giá theo mutual_info_score

# KMeans
print("KMeans evaluation: ",mutual_info_score(faces_data.target, labels))

# Spectral cluster
print("Spectral evaluation: ",mutual_info_score(faces_data.target, labels_spectral))

# DBSCAN
print("DBSCAN evaluation: ",mutual_info_score(faces_data.target, labels_dbscan))

# AgglomerativeClustering

print("AgglomerativeClustering evaluation: ",mutual_info_score(faces_data.target, labels_Agg))
```

```
KMeans evaluation:  0.0439948214265
Spectral evaluation:  0.0698021480935
DBSCAN evaluation:  2.77555756156e-16
AgglomerativeClustering evaluation:  0.0260280438289
```

In [157]:

```
# Thực hiện đáng giá theo homogeneity_completeness_v_measure: giá trị trả về trong khoảng 0 >> 1
# - Càng về 1 thì độ khớp của True labels và cluster labels càng cao.

# KMeans
print("KMeans evaluation: ",homogeneity_completeness_v_measure(faces_data.target, labels))

# Spectral cluster
print("Spectral evaluation: ",homogeneity_completeness_v_measure(faces_data.target, labels_spectral))

# DBSCAN
print("DBSCAN evaluation: ",homogeneity_completeness_v_measure(faces_data.target, labels_dbscan))

# AgglomerativeClustering
print("AgglomerativeClustering evaluation: ",homogeneity_completeness_v_measure(faces_data.target, labels_Ag
g))
```

```
KMeans evaluation:  (0.031291092685954459, 0.027462685261817381, 0.029252159093381754)
Spectral evaluation:  (0.049646422348130027, 0.04350944085704829, 0.046375783608216932)
DBSCAN evaluation:  (1.9741011805011969e-16, 1.0, 3.9482023610023927e-16)
AgglomerativeClustering evaluation:  (0.018512313619525083, 0.016269929608157097, 0.017318839242322897)
```

3.9 Nhận xét

- Với data là Faces và sử dụng LBP feature:
  - Sau khi xem xét evaluation: cho ra kết quả không khả quan.

Bài 4: Rút trích feature trên dataset tự chọn

In [158]:

```
- 4.1 Load dataset: car dataset
- 4.2 Rút trích feature: chọn HoG feature
```

```
  File "<ipython-input-158-c90ffb4c7731>", line 1
    - 4.1 Load dataset: car dataset
            ^
SyntaxError: invalid syntax
```

4.1 Load car dataset

In [178]:

```
# import thư viện
import glob
from scipy import misc
import imageio
from skimage.feature import hog
from skimage import data, color, exposure

# Load car dataset
# Car dataset gồm 1614 ảnh các góc chụp của nhiều xe 4 bánh
carData = glob.glob('carDataset/*.jpg')
carData += (glob.glob('carDataset/*.png'))
print(len(carData))
```

```
1614
```

4.2 Rút trích HoG feature

In [180]:

```
# Rút trích HoG feature
featureHoG = []

for i in range(1614):
    image = imageio.imread(carData[i])
    image = color.rgb2gray(image)
    fd, hog_image = hog(image, orientations=8, pixels_per_cell=(16, 16),cells_per_block=(1, 1), visualise=Tr
ue)
    featureHoG.append(fd)
```

```
c:\python34\lib\site-packages\skimage\feature\_hog.py:119: skimage_deprecation: Default value of `block_norm
`=='L1` is deprecated and will be changed to `L2-Hys` in v0.15
  'be changed to `L2-Hys` in v0.15', skimage_deprecation)
```

```
(1614, 48)
```

In [182]:

```
# Áp dụng KMeans trên car dataset
# Với dữ liệu đầu vào là HoG feature
# Chia thành 2 nhóm: nhóm 1 là nhóm có xe, nhóm 2 là nhóm có người
model = KMeans(n_clusters=2)
labels = model.fit_predict(featureHoG)
trueLabels = [0 if i > 1000 else 1 for i in range(1614)]
```

```
Truth labels    0    1
labels
0             157  801
1             456  200
```

In [190]:

```
# KMeans evaluation
print("KMeans evaluation: ",adjusted_mutual_info_score(trueLabels, labels))
```

```
KMeans evaluation:  0.22060746659
```