

HW4 – VINH NGUYEN

Q1 (6 points). Chapter 8 – Exercise 20

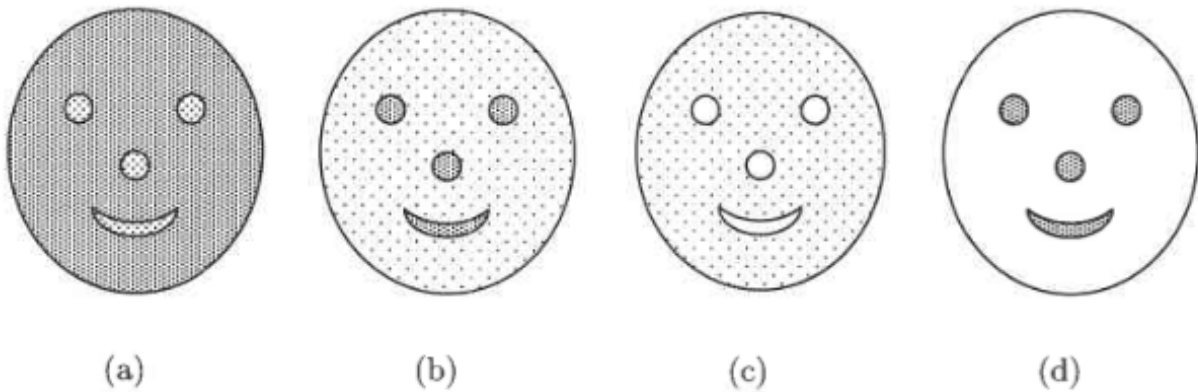


Figure 8.39. Figure for Exercise 20.

- a. For each figure, could you use single link to find the patterns represented by the nose, eyes and mouth? Explain?

First, a **single link** is a MIN version of hierarchical clustering. We start with singleton clusters, add a link one at a time, shortest link first. Then these links will be combined to form clusters. From this understanding, we predict that closer points will be grouped first. With this notion in mind we will look at every Figure to answer questions.

- Figure 8.39 (a). **NO**. Because points in the other areas are closer to each other than points in eyes, nose and mouths. In this case, it only can find one cluster.
- Figure 8.39 (b). **YES**. Because points in eyes, nose, months are dense (or closer) to each other than points in other areas (background).
- Figure 8.39 (c). **NO**. Because these areas contain no data. So, no information to represent
- Figure 8.39 (d). **YES**. Because points in eyes, nose, months are dense (or closer). This is the ideal data for clustering.

- b. For each figure, could you use K-means to find the pattern represented by nose, eyes, and mouth? Explain?

- Figure 8.39 (a). **NO**. K-means is a center based clustering algorithm. It may identify four clusters but those clusters may not represent the patterns of eyes, noses and mouths, because all points are included into one of the four clusters.

- Figure 8.39 (b). Maybe **YES**. Depending how purely patterns we want to get. Because these clusters contain noise.
- Figure 8.39 (c). **NO**. Because the desired patterns do not contain any data.
- Figure 8.39 (d). **YES**. If we set $K = 4$. Again, this is an ideal case.


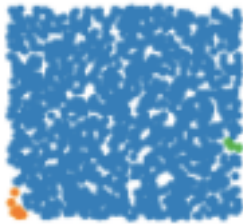
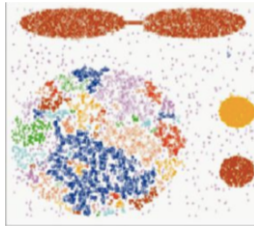
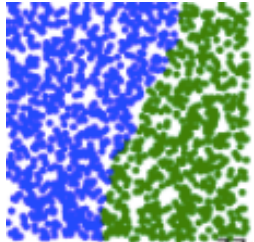
c. What limitation does clustering have in detecting all the patterns formed by the points in Figure 8.39 (c)?

Clustering techniques (K-means, Hierarchical Clustering, or DBSCAN) mostly use distance between data points for clustering. Meaning that they need data and cannot work with empty dataset. This is why Figure 8.39 (c) has problem with clustering.

Q2 (8 points).

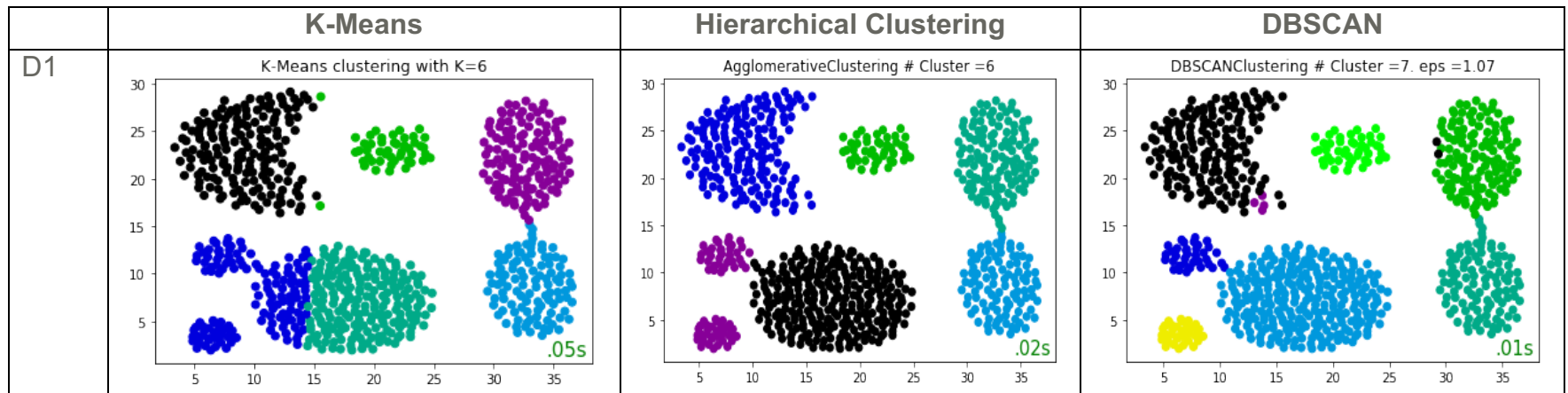
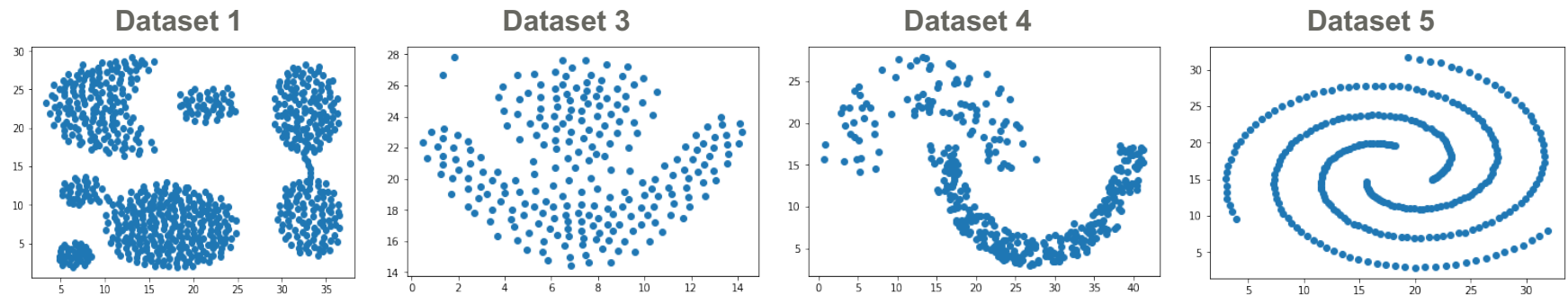
What is the idea/algorithm of K-means clustering, Hierarchical clustering and DBSCAN and Spectral clustering? (In other words, how do they work?). What are their strength and weakness? You may use some example.

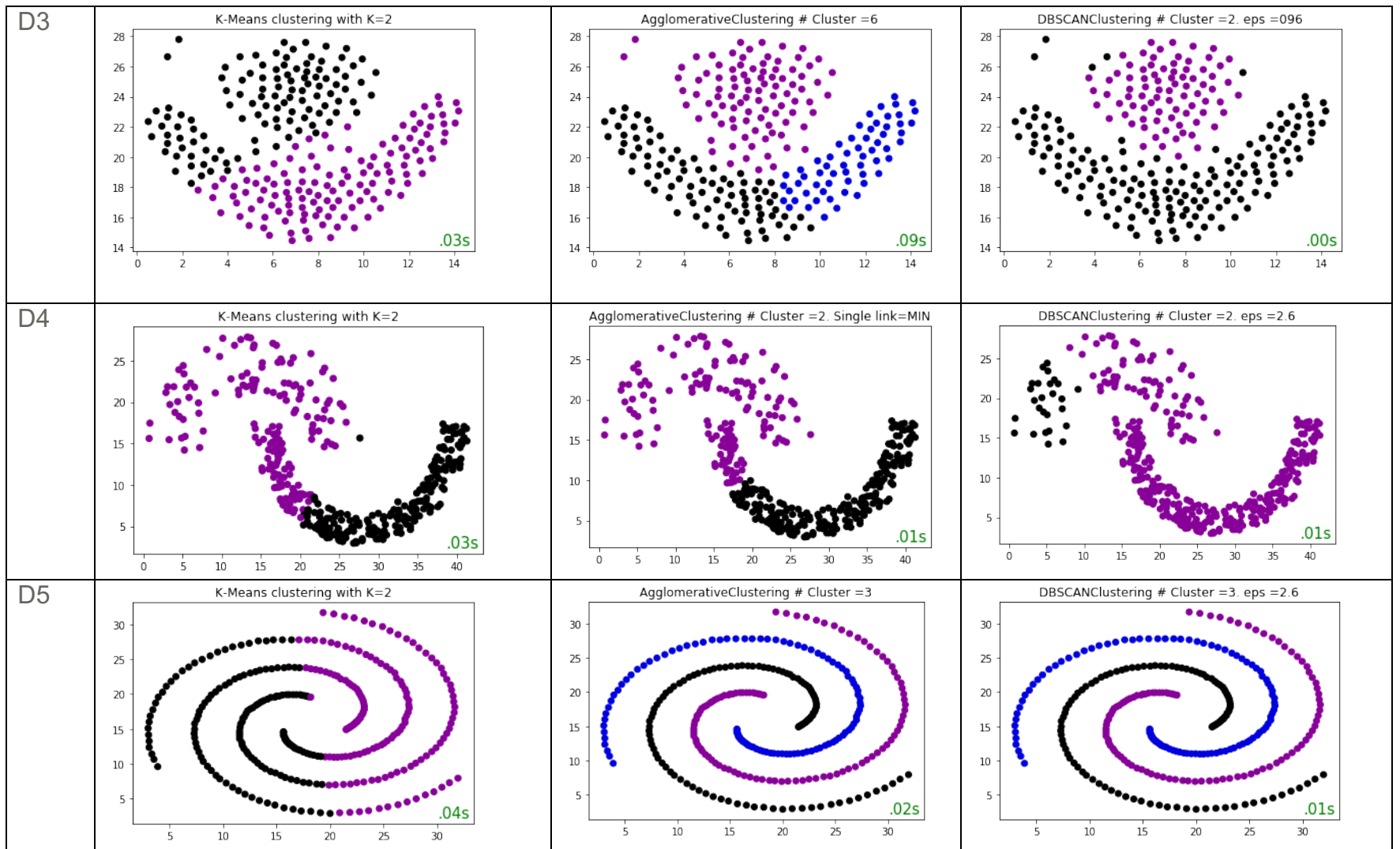
Indicator	K-means	Hierarchical Clustering	DBSCAN	Spectral Clustering
Type of clusters	Prototype-based	Graph-based/prototype-base	Density-based	Graph-based or contiguity-based
How it works? (Note: I don't want to copy algorithms because they are in textbooks)	Partition clustering based on user-defined # clusters K .	Bottom up approach. Each point is considered as a singleton cluster. The algorithm recursively merges two closet clusters into one until all points belong to one cluster.	Partition clustering where # of clusters defined by the algorithm. Points in low density areas are omitted.	It transforms data into similarity space and then clustering data in that space. The purpose of transformation is to reduce dimension of data. The result of transformation is a similarity matrix. Each block will correspond to a cluster.
Strength	Simple and widely used.	Don't have to assume # of clusters. Desired clusters can be defined by cutting dendrogram.	Resistant to noise. Can handle clusters with different shapes and sizes.	No assumption about the # of clusters. Easy implementation. Good cluster results. Fast for

		Meaningful to taxonomies.		sparse dataset of several thousand obs.
Weakness	<p>Not suitable for all data types. Cannot handle non-globular clusters of different sizes and shapes. Problem with outliers and handling empty dataset. Eg.</p> 	<p>Cannot be viewed as global objective function. Expensive in terms of time and space. Cannot redo clusters once they're combined. Breaking large clusters, sensitive to noise and outliers. Problem with clusters' shapes and sizes.</p> 	<p>Problem with clusters that have widely varying densities. Problem with high-dimensional data. Sensitive to parameters. Expensive when required computing all pairwise proximities (high-dimensional data)</p> 	<p>Sensitive to choose of parameters. Expensive in terms of time and space for large dataset.</p> 

Q3. Choose three clustering algorithms, choose one or more dataset in Blackboard folder: Homework->clustering-dataset.zip, implement the three clustering algorithms and visualize them

Plot all datasets to see distribution.





Implementation in Python

K-means: Repeat this code to dataset 1->5. Change value of K to get desired clustering results.

```
from sklearn.cluster import KMeans
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib
import time
dataset = pd.read_csv('~\\Desktop\\Clustering-dataset\\Clustering5.txt', sep='\\t')
kmeans = KMeans(n_clusters=2, random_state=0)
start_time = time.time()
kmeans.fit(dataset.values)
end_time = time.time()
running_time=end_time-start_time;

plt.scatter(dataset.values[:,0],dataset.values[:,1], c=[matplotlib.cm.spectral(float(i) /10) for i in kmeans.labels_]);
plt.text(.99, .01, ('%.2fs' % running_time).rstrip('0'),
        verticalalignment='bottom', horizontalalignment='right',
        transform=plt.gca().transAxes,
        color='green', fontsize=15)
plt.title('K-Means clustering with K=2')
plt.show()
```

Hierarchical Clustering: Repeat this code to dataset 1->5. Change 3 parameters: *n_clusters*, *linkage*, and *n_neighbours* to achieve desired clustering results.


```

from sklearn.cluster import KMeans, AgglomerativeClustering, DBSCAN
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib
import time
from sklearn.neighbors import kneighbors_graph
dataset = pd.read_csv('~\\Desktop\\Clustering-dataset\\Clustering1.txt', sep='\\t')
connectivity = kneighbors_graph(dataset.values, n_neighbors=14, include_self=False)
connectivity = 0.5 * (connectivity + connectivity.T)
AgglomerativeClustering = AgglomerativeClustering(linkage="ward", n_clusters=6)
start_time = time.time()
AgglomerativeClustering.fit(dataset.values)
end_time = time.time()
running_time=end_time-start_time;

plt.scatter(dataset.values[:,0],dataset.values[:,1], c=[matplotlib.cm.spectral(float(i) /10) for i in AgglomerativeClus
plt.text(.99, .01, ('%.2fs' % running_time).rstrip('0'),
        verticalalignment='bottom', horizontalalignment='right',
        transform=plt.gca().transAxes,
        color='green', fontsize=15)
plt.title('AgglomerativeClustering # Cluster =6')
plt.show()

```

DBSCAN. Repeat this code for dataset 1->5. Change eps to get desired clustering results.

```

dataset = pd.read_csv('~\\Desktop\\Clustering-dataset\\Clustering5.txt', sep='\\t')
DBSCANClustering = DBSCAN(eps=2.6)
start_time = time.time()
DBSCANClustering.fit(dataset.values)
end_time = time.time()
running_time=end_time-start_time;

plt.scatter(dataset.values[:,0],dataset.values[:,1], c=[matplotlib.cm.spectral(float(i) /10) for i in DBSCANClustering.
plt.text(.99, .01, ('%.2fs' % running_time).rstrip('0'),
        verticalalignment='bottom', horizontalalignment='right',
        transform=plt.gca().transAxes,
        color='green', fontsize=15)
plt.title('DBSCANClustering # Cluster =3. eps =2.6')
plt.show()

```