Assignment 5

Jiarong Ye

November 28, 2018

Q1

1. (30%) Perform K-means on the data shown in Table 1. Use K=2 and suppose A and C are randomly selected as initial means. Please show the clustering results of nodes of each round until converged.

Table 1

i	X_1	X_2
A	1	1
B	1	0
C	0	2
D	2	4
E	3	5

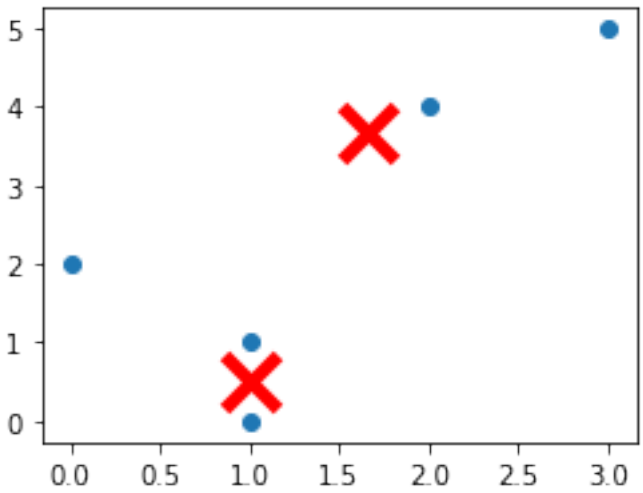
In [43]:
```
# slightly adjust the programming assignment

import numpy as np
from collections import Counter
import matplotlib.pyplot as plt
from sklearn.metrics import f1_score, normalized_mutual_info_score

THRESHOLD = 1e-5

X = np.array([[1,1],
              [1,0],
              [0,2],
              [2,4],
              [3,5]])
```

```
def norm(x):
    max_val = np.max(x, axis=0)
    x = x / max_val
    return x

def converged(centroids1, centroids2):
    diff = np.sum(np.abs(np.sum(centroids1 - centroids2, axis=1)), axis=0)
    if np.equal(centroids1, centroids2).all():
        return True
    elif diff < THRESHOLD:
        return True
    else:
        return False

def euclidean_dist(x1, x2):
    return np.sqrt(np.sum(np.square(x1 - x2), axis=1))


def closest_centroid(val, centroids):
    return np.argmin(np.sqrt(np.sum(np.square(val - centroids), axis=1)))


def update_centroids(data, centroids, k=2):
    n_samples = np.shape(data)[0]
    n_features = np.shape(data)[1]
    clusters = [[] for _ in range(k)]
    labels = np.zeros((n_samples))

    for idx, val in enumerate(data):
        val_label = closest_centroid(val, centroids)
        clusters[val_label].append(val)
        labels[idx] = val_label
    print(">>> clusters:")
    print(clusters)
    centroids = np.zeros((k, n_features))
    for idx, cluster_val in enumerate(clusters):
        centroid = np.mean(cluster_val, axis=0)
        centroids[idx] = centroid
    return centroids, labels


def kmeans(data, k=2):
    centroids = [[1,1], [0,2]]
    converge = False
    iteration = 0
    while not converge:
        print(">>> iteration: ", iteration)
```

```
            old_centroids = np.copy(centroids)
            # step 2 & 3
            centroids, label = update_centroids(data, old_centroids)
            plot_result(X, centroids)
            # step 4
            converge = converged(old_centroids, centroids)
            iteration += 1
        print('number of iterations to converge: ', iteration)
        print(">>> final centroids")
        print(centroids)
        return centroids, label


    def plot_result(data, centroids):
        fig = plt.figure(1, figsize=(4, 3))
        plt.scatter([i[0] for i in data], [i[1] for i in data])
        plt.plot(centroids[0][0], centroids[0][1], color='red', marker='x', ms=20, mew=5)
        plt.plot(centroids[1][0], centroids[1][1], color='red', marker='x', ms=20, mew=5)
        plt.show()
```

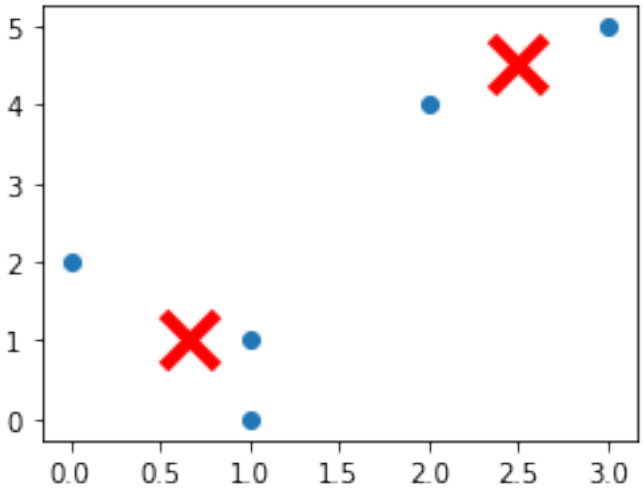
```
In [44]: centroids, label = kmeans(X,2)
         centroids, label
         plot_result(X, centroids)
```

```
>>> iteration:  0
>>> clusters:
[[array([1, 1]), array([1, 0])], [array([0, 2]), array([2, 4]), array([3, 5])]]
```

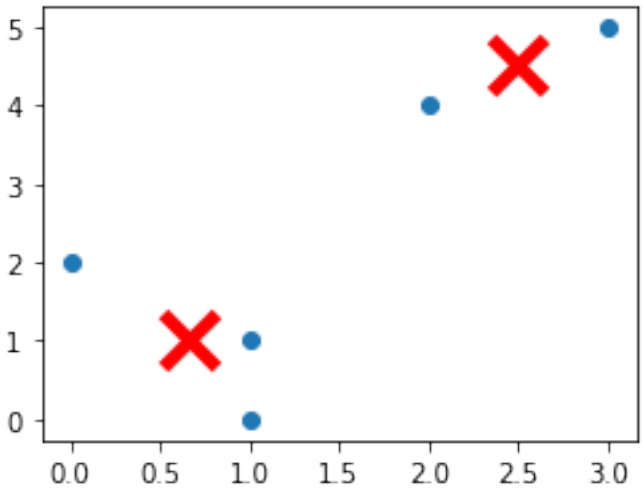



```
>>> iteration:  1
>>> clusters:
```

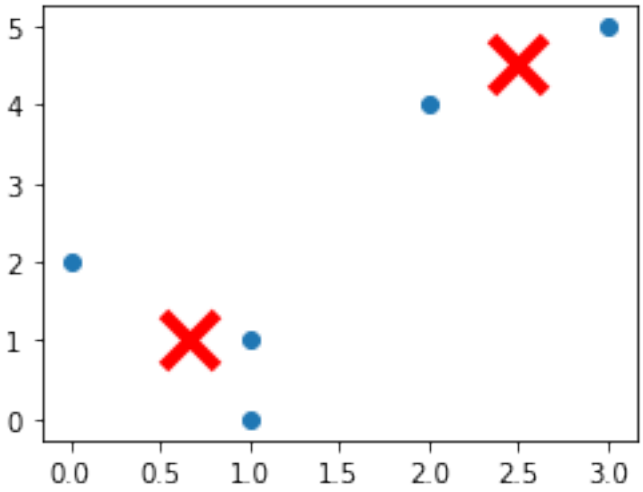
```
[[array([1, 1]), array([1, 0]), array([0, 2])], [array([2, 4]), array([3, 5])]]
```




```
>>> iteration:  2
>>> clusters:
[[array([1, 1]), array([1, 0]), array([0, 2])], [array([2, 4]), array([3, 5])]]
```




```
number of iterations to converge:  3
>>> final centroids
[[0.66666667 1.        ]
 [2.5        4.5       ]]
```



Q2

2. (50%) Use the Similarity Matrix in Table 2 to perform MIN (Single Link) and MAX (Complete Link) hierarchical clustering. Note that MIN/MAX Hierarchical Clustering schemes are named based on dissimilarity but here the matrix below contains the similarity measure. Show your results by drawing a dendrogram. The dendrogram should clearly show the order in which the points are merged.

Table 2 Similarity Measure

	p1	p2	p3	p4	p5
p1	1.00	0.10	0.41	0.55	0.35
p2	0.10	1.00	0.64	0.47	0.98
p3	0.41	0.64	1.00	0.44	0.85
p4	0.55	0.47	0.44	1.00	0.76
p5	0.35	0.98	0.85	0.76	1.00

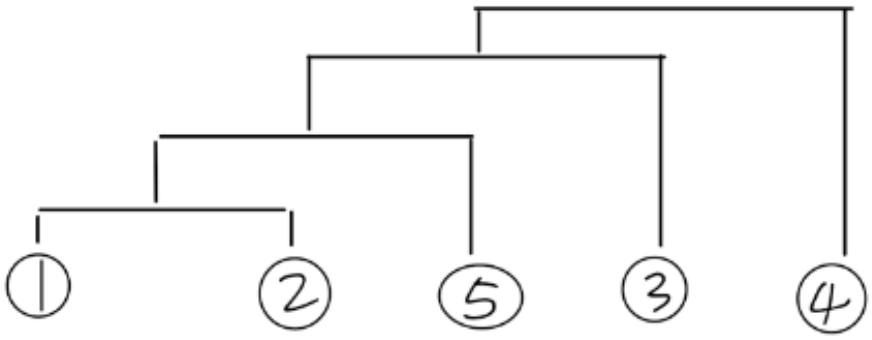
MIN (Single Link)

	1	2	3	4	5
1	1	0.1	0.41	0.55	0.35
2	0.1	1	0.64	0.47	0.98
3	0.41	0.64	1	0.44	0.85
4	0.55	0.47	0.44	1	0.76
5	0.35	0.98	0.85	0.76	1

	1 2	3	4	5
12	1	0.41	0.47	0.35
3	0.41	1	0.44	0.85
4	0.47	0.44	1	0.76
5	0.35	0.85	0.76	1

	125	3	4
125	1	0.41	0.47
3	0.41	1	0.44
4	0.47	0.44	1

	1253	4
1253	1	0.44
3	0.44	1



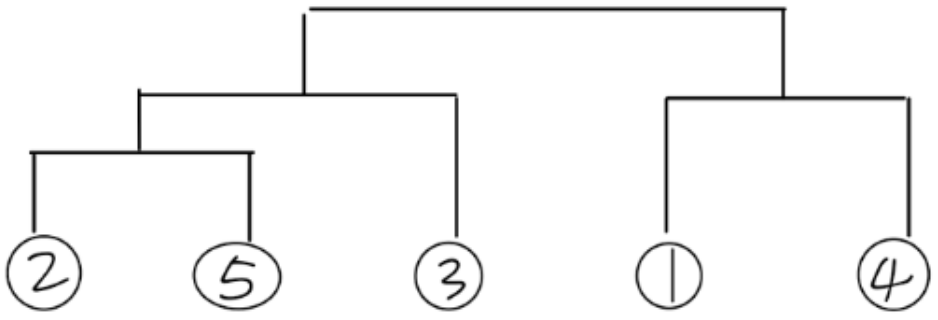
MAX (Complete Link)

	1	2	3	4	5
1	1	0.1	0.41	0.55	0.35
2	0.1	1	0.64	0.47	0.98
3	0.41	0.64	1	0.44	0.85
4	0.55	0.47	0.44	1	0.76
5	0.35	0.98	0.85	0.76	1

	25	1	3	4
25	1	0.1	0.64	0.47
1	0.1	1	0.41	0.55
3	0.64	0.41	1	0.44
4	0.47	0.55	0.44	1

	253	1	4
253	1	0.1	0.44
1	0.1	1	0.55
4	0.44	0.55	1

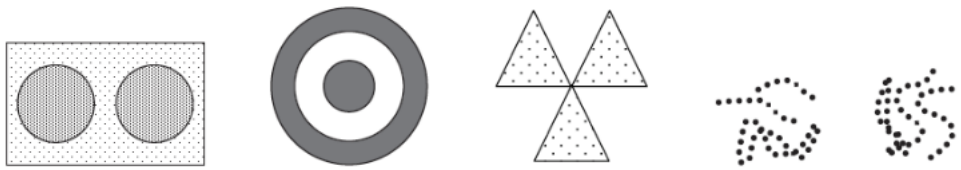
	253	14
253	1	0.1
14	0.1	1



Q3

3. (20%) Identify possible clusters in the figure below using DBSCAN. Please indicate the number of clusters for each case and briefly explain your reasoning. Note that darkness or the

number of dots in an area visually indicates its density.



(a) (b) (c) (d)

- 1.
  - Clusters: 2
  - Density-based using DBSCAN
  - reason: The inner points in circle are core points and the exterior are noise
- 2.
  - Clusters: 2
  - Density-based using DBSCAN
  - reason: The points in inner circle and outer ring are core points.
- 3.
  - Clusters: 3
  - Center-based not using DBSCAN (probably using KMeans)
  - reason: The clustering is partitional clustering with centroids as the mean of points in each individual cluster (in this case, each triangle).
- 4.
  - Clusters: 2
  - Contiguity-based using DBSCAN or hierarchal clustering
  - reason: The clustering is contiguity-based, the two clusters are formed because the data points in each cluster are continuous and also the density of the two clusters is higher than the region between them.