K-means

- Datasets
  - Iris
  - Iris_Initial_Centroids

	A	B
1	5.4,3.9,1.7,0.4	
2	6.4,3.1,5.5,1.8	
3	4.6,3.6,1.0,0.2	
4	5.5,4.2,1.4,0.2	
5	5.1,3.5,1.4,0.2	
6	4.3,3.0,1.1,0.1	
7	7.7,2.6,6.9,2.3	
8	6.4,2.7,5.3,1.9	
9	6.0,2.9,4.5,1.5	

	A	B
1	4.4,3,1.3,0.2	
2	5.9,3,5.1,1.8	
3	4,3,4,1.2	
4		

K-means

- assignCluster `def assignCluster(dataSet, k, centroids)`
- For each data point, the function is to assign it to the closest centroid.
- Input
  - dataSet: each row represents an observation and each column represents an attribute;
  - k: number of clusters
  - centroids: initial centroids or centroids of last iteration
- Output
  - clusterAssment: a list, which contains assigned cluster id for each data point

K-means

- assignCluster
- To implement this function, you can follow the following pseudo-code

```
for data in dataSet:
        minDist = Inf, minIndex = -1
        for center in centroids:
                d = distance(data, center)
                if d<minDist:
                        minDist = d, minIndex = index of center
        clusterAssment.append(minIndex)
```

K-means

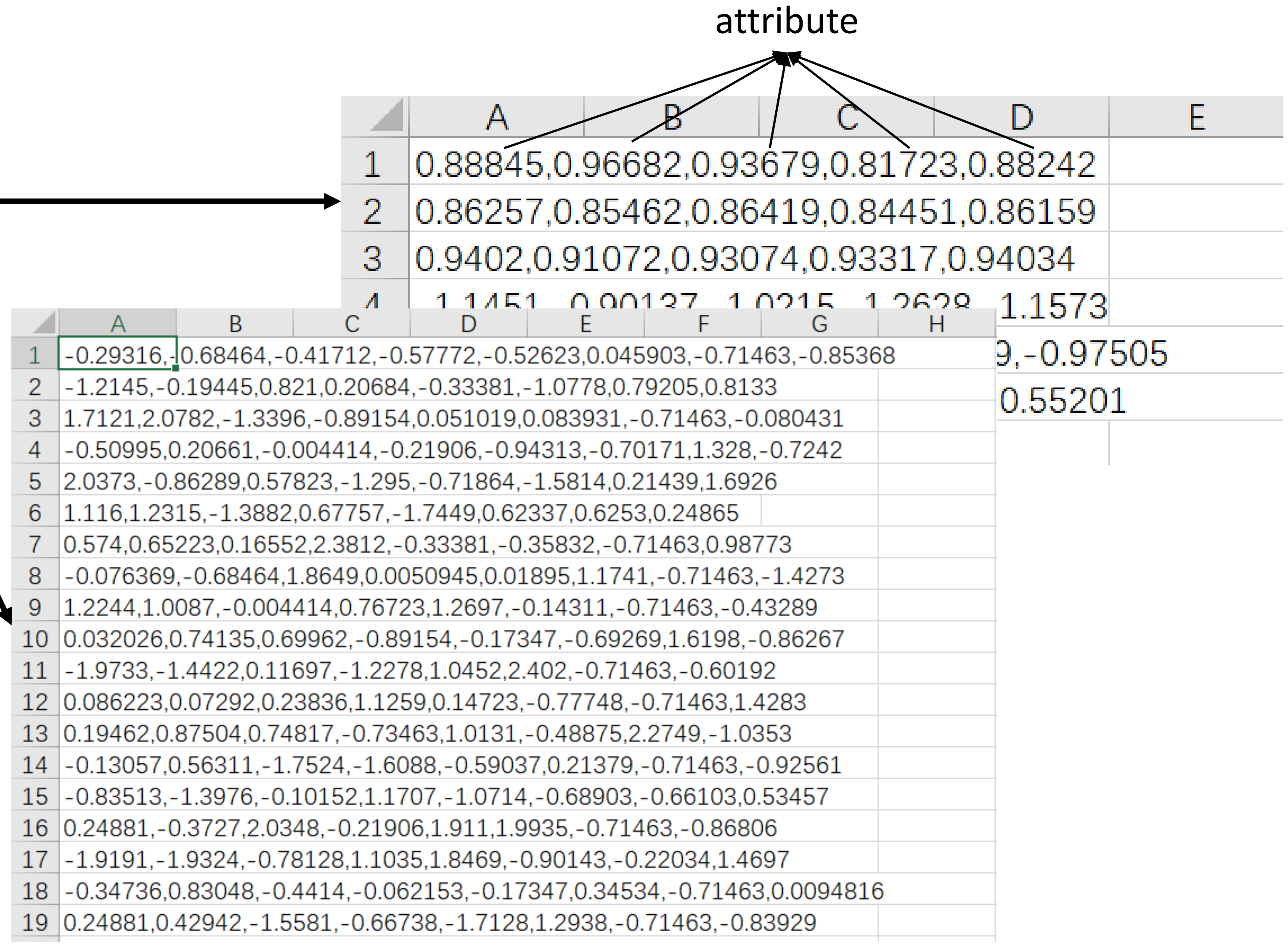
- getCentroid

```
def getCentroid(dataSet, k, clusterAssment)
```

- The function is to recalculate centroids.
- Input
  - dataSet: each row represents an observation and each column represents an attribute
  - k: number of clusters
  - clusterAsment: a list, which contains assigned cluster id for each data point
- Output
  - centroids: cluster centroids

Hierarchical clustering

- Datasets
  - Example
  - Utilities



	A	B	C	D	E
1	0.88845,0.96682,0.93679,0.81723,0.88242				
2	0.86257,0.85462,0.86419,0.84451,0.86159				
3	0.9402,0.91072,0.93074,0.93317,0.94034				
4	[illegible]1.1573				
5	[illegible]9,-0.97505				
6	[illegible]0.55201				

	A	B	C	D	E	F	G	H
1	-0.29316,-0.68464,-0.41712,-0.57772,-0.52623,0.045903,-0.71463,-0.85368							
2	-1.2145,-0.19445,0.821,0.20684,-0.33381,-1.0778,0.79205,0.8133							
3	1.7121,2.0782,-1.3396,-0.89154,0.051019,0.083931,-0.71463,-0.080431							
4	-0.50995,0.20661,-0.004414,-0.21906,-0.94313,-0.70171,1.328,-0.7242							
5	2.0373,-0.86289,0.57823,-1.295,-0.71864,-1.5814,0.21439,1.6926							
6	1.116,1.2315,-1.3882,0.67757,-1.7449,0.62337,0.6253,0.24865							
7	0.574,0.65223,0.16552,2.3812,-0.33381,-0.35832,-0.71463,0.98773							
8	-0.076369,-0.68464,1.8649,0.0050945,0.01895,1.1741,-0.71463,-1.4273							
9	1.2244,1.0087,-0.004414,0.76723,1.2697,-0.14311,-0.71463,-0.43289							
10	0.032026,0.74135,0.69962,-0.89154,-0.17347,-0.69269,1.6198,-0.86267							
11	-1.9733,-1.4422,0.11697,-1.2278,1.0452,2.402,-0.71463,-0.60192							
12	0.086223,0.07292,0.23836,1.1259,0.14723,-0.77748,-0.71463,1.4283							
13	0.19462,0.87504,0.74817,-0.73463,1.0131,-0.48875,2.2749,-1.0353							
14	-0.13057,0.56311,-1.7524,-1.6088,-0.59037,0.21379,-0.71463,-0.92561							
15	-0.83513,-1.3976,-0.10152,1.1707,-1.0714,-0.68903,-0.66103,0.53457							
16	0.24881,-0.3727,2.0348,-0.21906,1.911,1.9935,-0.71463,-0.86806							
17	-1.9191,-1.9324,-0.78128,1.1035,1.8469,-0.90143,-0.22034,1.4697							
18	-0.34736,0.83048,-0.4414,-0.062153,-0.17347,0.34534,-0.71463,0.0094816							
19	0.24881,0.42942,-1.5581,-0.66738,-1.7128,1.2938,-0.71463,-0.83929							

Hierarchical clustering

- merge_cluster
- The function is to merge two closest clusters according to min distances.
- Input
  - distance_matrix: a 2-D array distance matrix
  - cluster_candidate: a dictionary. Key is the cluster id, and value is point ids in the cluster.
  - T: current cluster index
- Output
  - cluster_candidate: a dictionary. We update cluster dictionary after merging two clusters. Key is the cluster id, and value is a list of point ids in the cluster
  - merge_list: list of tuples. It records the two old clusters' id and points that have just been merged.

    [(cluster_one_id, point_ids_in_cluster_one),

    (cluster_two_id, point_ids_in_cluster_two)]

Hierarchical clustering

- merge_cluster
- You can implement this function by two steps:
  - Find the smallest entry in the distance matrix—suppose the entry is i-th row and j-th column.
  - Merge the clusters that correspond to the i-th row and j-th column of the distance matrix as a new cluster with index T

Hierarchical clustering

- merge_cluster
- When you find minimum value indices in distance matrix, you may use the following methods in NumPy:
  - .flatten()
    - Return a copy of the array collapsed into one dimension.
  - np.unravel_index()
    - Converts a flat index or array of flat indices into a tuple of coordinate arrays

```
>>> a = np.array([[1,2], [3,4]])
>>> a.flatten()
array([1, 2, 3, 4])
```

```
>>> np.unravel_index([22, 41, 37], (7,6))
(array([3, 6, 6]), array([4, 5, 1]))
```

Hierarchical clustering

- merge_cluster
- Merge the clusters that correspond to the i-th row and j-th column of the distance matrix as a new cluster with index T

- To implement this function, you may use the following method in python:
    - .pop
        - It can remove data in dictionary.

```
>>> a={1:2,3:4,5:6}
>>> a
{1: 2, 3: 4, 5: 6}
>>> a.pop(5)
6
>>> a
{1: 2, 3: 4}
>>>
```

Hierarchical clustering

- update_distance
- This function is to update the distance matrix.
- Input
  - distance_matrix: 2-D array
  - cluster_candidate: a dictionary. Key is the updated cluster id, value is point ids in the cluster.
  - merge_list: list of tuples. It records the two old clusters' id and points that have just been merged.

    [(cluster_one_id, point_ids_in_cluster_one),

    (cluster_two_id, point_ids_in_cluster_two)]

- Output
  - distance_matrix: 2-D array. Updated distance matrix.

Hierarchical clustering

- You need to mark all distance between points in two clusters in merge_list to be a large value.
- You can use “merge_list[0][1]” and “merge_list[1][1]” to get points indices in two clusters in merge_list .
- The large value can be set as 100000 or other big number.