Clustering

1 k-means

1. Randomly pick k centroids.
2. Repeat until the algorithm converges (when no example changes cluster in two subsequent iterations):
   (a) For each example find the nearest centroid and assign the example to its cluster.
   (b) For each cluster, calculate the new centroid (the average of all vectors in the cluster).

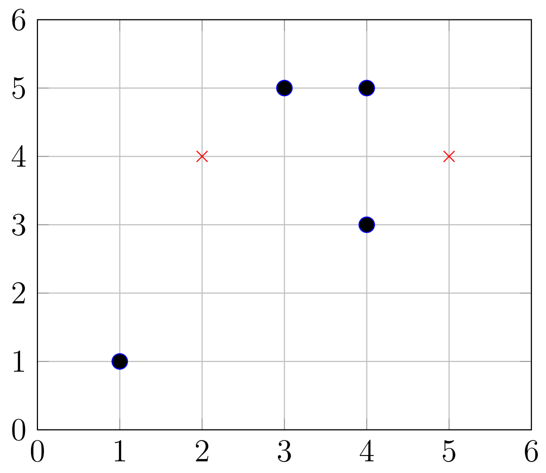
2 Hierarchical agglomerative clustering

1. Begin with each example in its own cluster.
2. Identify two nearest clusters according to a given metric and join them.
3. Repeat step 2 until all examples are in the one cluster.

Questions

Question 1.

Cluster the following dataset using k-means, starting with the centroids $c_1(2, 4)$ and $c_2(5, 4)$:
A(1, 1), B(3, 5), C(4, 3), D(4, 5).



Question 2.

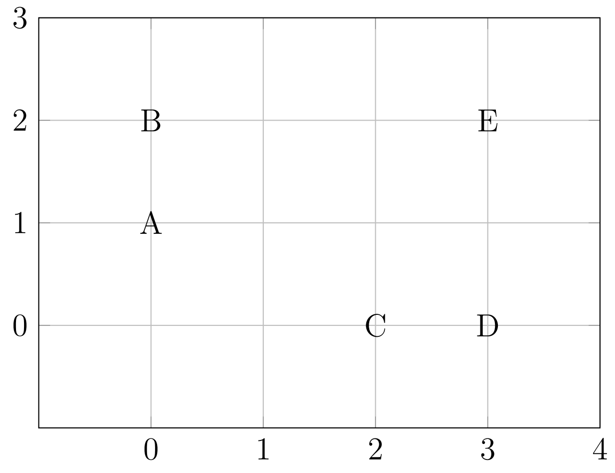
Cluster the following dataset using k-means, starting with the centroids $c_1(0, 1, 0, 1)$ and $c_2(0, 2, 1, 0)$:

A(0, 1, 0, 1), B(0, 0, 2, 0), C(3, 2, -1, 1), D(0, 2, 1, 0).

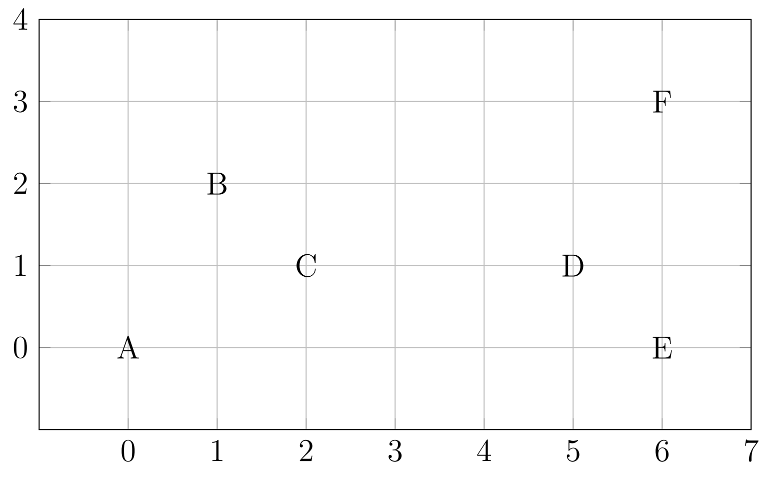
Question 3.

Cluster the following datasets using hierarchical agglomerative clustering. Draw the dendrograms.

A(0,1), B(0,2), C(2,0), D(3,0), E(3,2).



A(0,0), B(1,2), C(2,1), D(5,1), E(6,0), F(6,3).



Mini-project: *k*-means

Implement the *k*-means algorithm. Cluster the iris dataset in in `iris.data` (ignore the decision attribute).

Additional requirements:

- Allow the user to pick *k*.
- After every iteration: print the sum of distances from each point from its centroid. This value should decrease with every iteration. Note: print the total for all points, not each cluster separately. Example:
  ```
  Iteration 1: 126.38
  Iteration 2: 86.34
  Iteration 3: 49.91
  ...
  ```
- At the end: print the members of each cluster.
- Optional: print measures of cluster homogeneity, eg. percentage of each iris class per cluster, or entropy.