CC282
Unsupervised Learning
(Clustering)

Lecture 07 – Outline

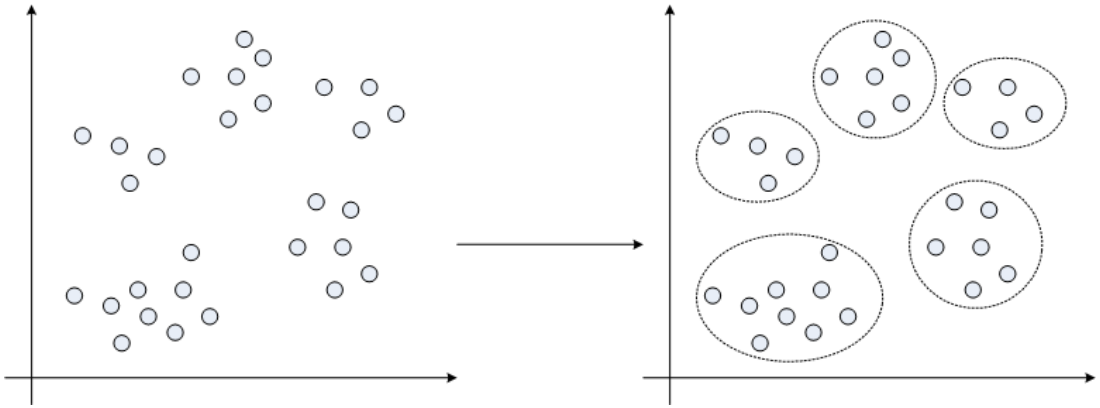
- Clustering: introduction
- Clustering approaches
- Exclusive clustering: *K*-means algorithm
- Agglomerative clustering: Hierarchical algorithm
- Overlapping clustering: Fuzzy C-means algorithm
- Cluster validity problem
- Cluster quality criteria: Davies-Bouldin index

Clustering (introduction)

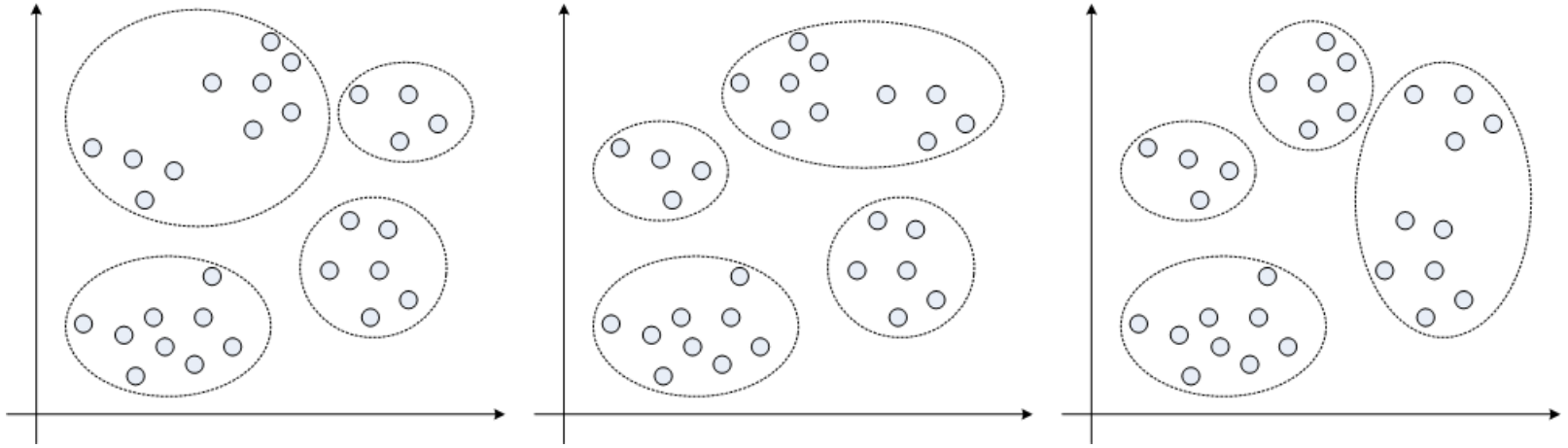
- Clustering is a type of unsupervised machine learning
- It is distinguished from supervised learning by the fact that there is not a priori output (i.e. no labels)
  - The task is to learn the classification/grouping from the data
- A cluster is a collection of objects which are *similar* in some way
- Clustering is the process of grouping *similar* objects into groups
- Eg: a group of people clustered based on their height and weight
- Normally, clusters are created using distance measures
  - Two or more objects belong to the same cluster if they are "close" according to a given distance (in this case geometrical distance like Euclidean or Manhattan)
- Another measure is conceptual
  - Two or more objects belong to the same cluster if this one defines a concept *common* to all that objects
  - In other words, objects are grouped according to their fit to descriptive concepts, not according to simple similarity measures

Clustering (introduction)

- Example: using distance based clustering



- This was easy but how if you had to create 4 clusters?
- Some possibilities are shown below but which is correct?



Clustering (introduction – ctd)

- So, the goal of clustering is to determine the intrinsic grouping in a set of unlabeled data
- But how to decide what constitutes a good clustering?
- It can be shown that there is no absolute “best” criterion which would be independent of the final aim of the clustering
- Consequently, it is the user which must supply this criterion, to suit the application
- Some possible applications of clustering
  - data reduction – reduce data that are homogeneous (similar)
  - find “natural clusters” and describe their unknown properties
  - find useful and suitable groupings
  - find unusual data objects (i.e. outlier detection)

Clustering – an early application example

- Hertzsprung-Russell diagram clustering stars by temperature and luminosity
  - Two astonomers in the early 20th century clustered stars into three groups using scatter plots
  - Main sequence: 80% of stars spending active life converting hydrogen to helium through nuclear fusion
  - Giants: Helium fusion or fusion stops: generates great deal of light
  - White dwarf: Core cools off

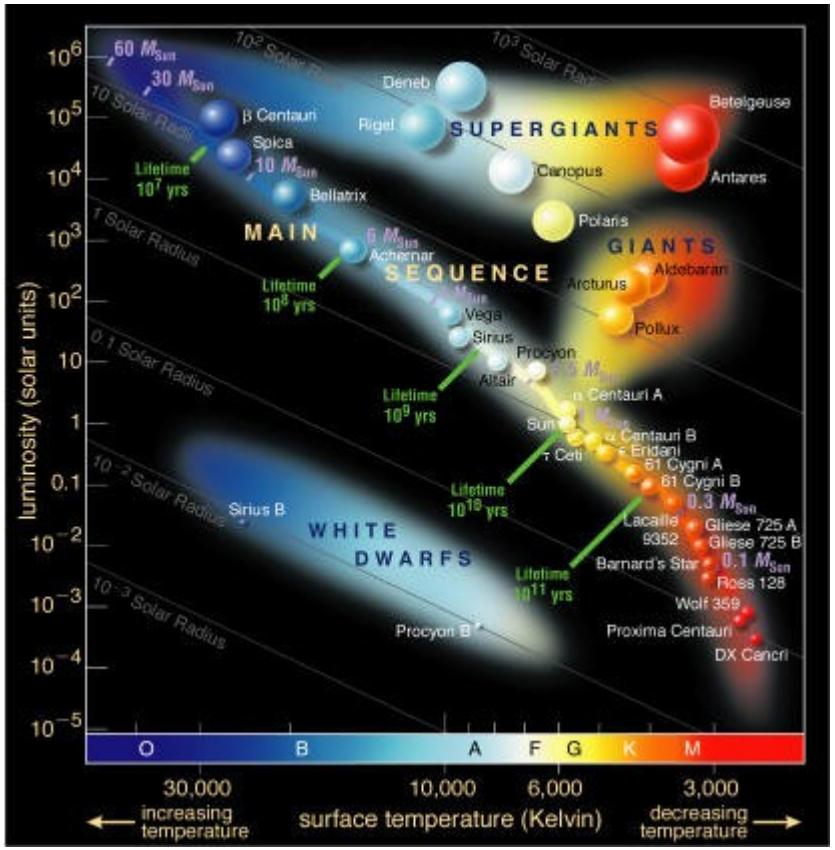


Diagram from Google Images

Clustering – Major approaches

- Exclusive (partitioning)
  - Data are grouped in an exclusive way, one data can only belong to one cluster
  - Eg: *K-means*
- Agglomerative
  - Every data is a cluster initially and iterative unions between the two nearest clusters reduces the number of clusters
  - Eg: Hierarchical clustering
- Overlapping
  - Uses fuzzy sets to cluster data, so that each point may belong to two or more clusters with different degrees of membership
  - In this case, data will be associated to an appropriate membership value
  - Eg: Fuzzy C-Means
- Probabilistic
  - Uses probability distribution measures to create the clusters
  - Eg: Gaussian mixture model clustering, which is a variant of *K*-means
  - Will not be discussed in this course

Exclusive (partitioning) clustering

- Aim: Construct a partition of a database *D* of *N* objects into a set of *K* clusters
- Method: Given a *K*, find a partition of *K* clusters that optimises the chosen partitioning criterion

- *K*-means (MacQueen'67) is one of the commonly used clustering algorithm
- It is a heuristic method where each cluster is represented by the centre of the cluster (i.e. the centroid)

- *Note: One and two dimensional (i.e. with one and two features) data are used in this lecture for simplicity of explanation*
- *In general, clustering algorithms are used with much higher dimensions*

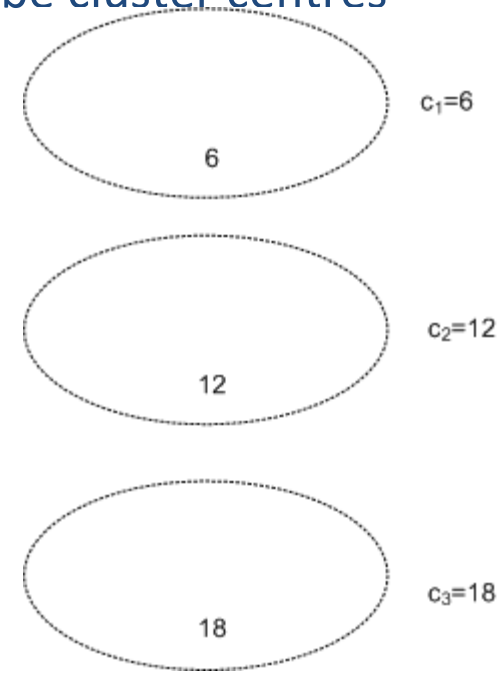
K-means clustering algorithm

Given *K*, the *K-means* algorithm is implemented in four steps:

1. Choose *K* points at random as cluster centres (centroids)
2. Assign each instance to its closest cluster centre using certain distance measure (usually Euclidean or Manhattan)
3. Calculate the centroid of each cluster, use it as the new cluster centre (one measure of centroid is mean)
4. Go back to Step 2, stop when cluster centres do not change any more

K-means – an example

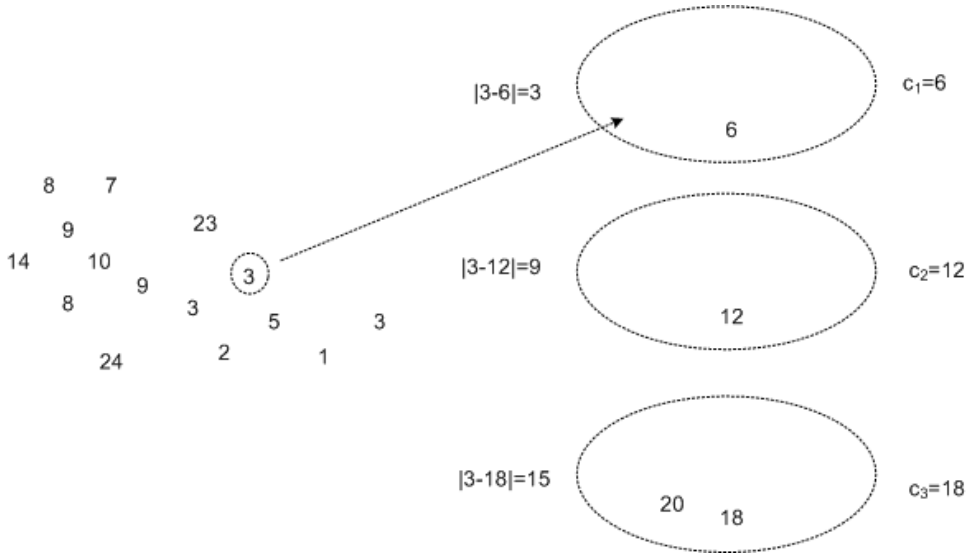
- Say, we have the data: {20, 3, 9, 10, 9, 3, 1, 8, 5, 3, 24, 2, 14, 7, 8, 23, 6, 12, 18} and we are asked to use K-means to cluster these data into 3 groups
- Assume we use Manhattan distance*
- Step one: Choose *K* points at random to be cluster centres
- Say 6, 12, 18 are chosen

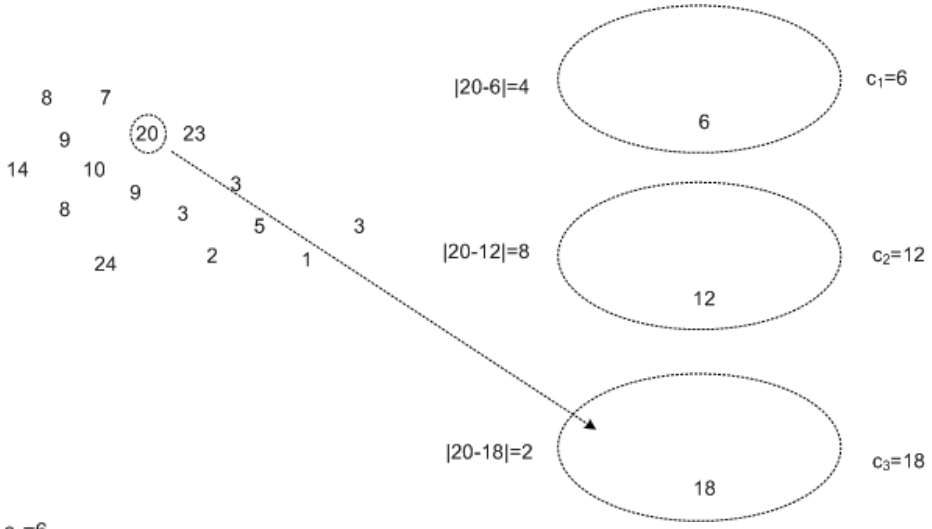


*note for one dimensional data, Manhattan distance=Euclidean distance

K-means – an example (ctd)

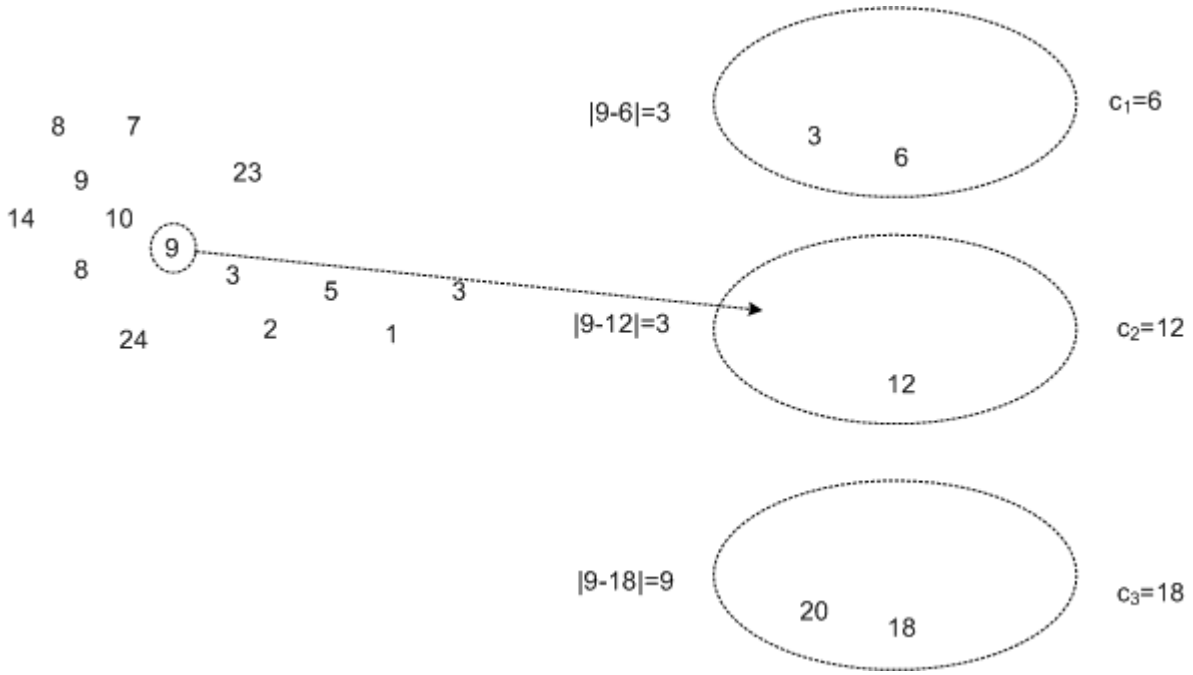
- Step two: Assign each instance to its closest cluster centre using Manhattan distance
- For instance:
  - 20 is assigned to cluster 3
  - 3 is assigned to cluster 1



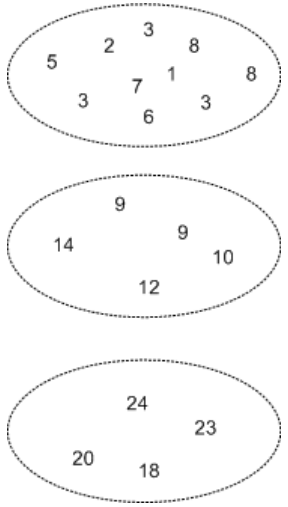


K-means – Example (ctd)

- Step two continued: 9 can be assigned to cluster 1, 2 but let us say that it is arbitrarily assigned to cluster 2
- Repeat for all the rest of the instances

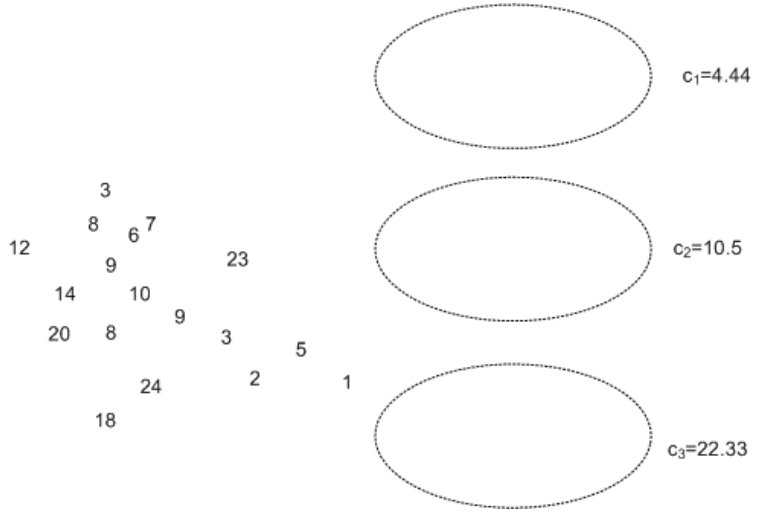


K-Means – Example (ctd)



- And after exhausting all instances…

- Step three: Calculate the centroid (i.e. mean) of each cluster, use it as the new cluster centre



- End of iteration 1
- Step four: Iterate (repeat steps 2 and 3) until the cluster centres do not change any more

K - means

- Strengths
  - Relatively efficient: where *N* is no. objects, *K* is no. clusters, and *T* is no. iterations. Normally, *K, T << N.*
  - Procedure always terminates successfully (but see below)

- Weaknesses
  - Does not necessarily find the most optimal configuration
  - Significantly sensitive to the initial randomly selected cluster centres
  - Applicable only when mean is defined (i.e. can be computed)
  - Need to specify *K*, the number of clusters, in advance

K-means in MATLAB

- Use the built in ‘kmeans’ function
- Example, for the data that we saw earlier
- The ‘ind’ is the output that gives the cluster index of the data, while ‘c’ is the final cluster centres
- For Manhanttan distance, use …‘distance’, ‘cityblock’…
- For Euclidean (default), no need to specify distance measure

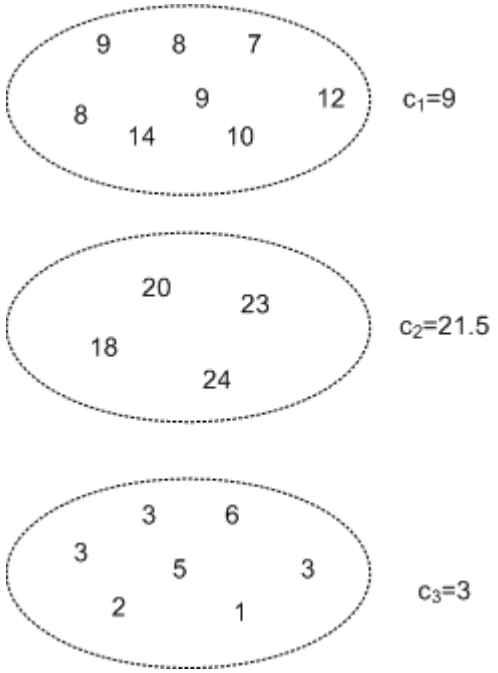
```
>> X=[20, 3, 9, 10, 9, 3, 1, 8, 5, 3, 24, 2, 14, 7, 8, 23, 6, 12, 18];
>> [ind, c]=kmeans(X',3,'distance','cityblock')

ind =

     2
     3
     1
     1
     1
     3
     3
     1
     3
     3
     2
     3
     1
     1
     1
     2
     3
     1
     2

c =

    9.0000
   21.5000
    3.0000
```

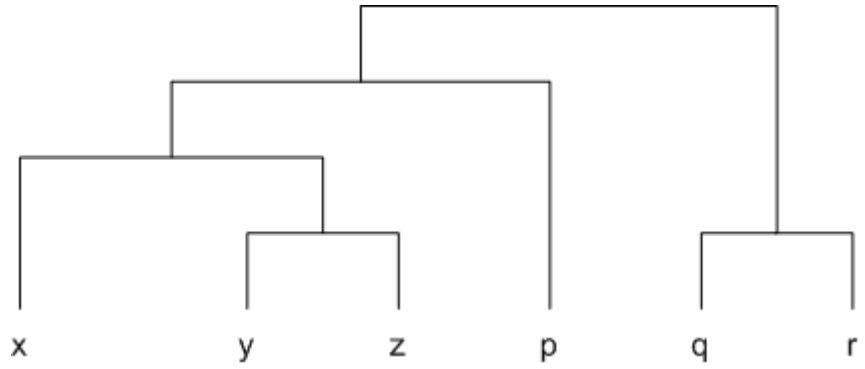


Agglomerative clustering

- K-means approach starts out with a fixed number of clusters and allocates all data into the exactly number of clusters
- But agglomeration does not require the number of clusters *K* as an input
- Agglomeration starts out by forming each data as one cluster
  - So, data of N object will have N clusters
- Next by using some distance (or similarity) measure, reduces the number so clusters (one in each iteration) by merging process
- Finally, we have one big cluster than contains all the objects
- But then what is the point of having one big cluster in the end?

Dendrogram (ctd)

- While merging cluster one by one, we can draw a tree diagram known as dendrogram
- **Dendrograms** are used to represent agglomerative clustering
- From dendrograms, we can get any number of clusters
- Eg: say we wish to have 2 clusters, then *cut the top one link*
  - Cluster 1: q, r
  - Cluster 2: x, y, z, p
- Similarly for 3 clusters, cut 2 top links
  - Cluster 1: q, r
  - Cluster 2: x, y, z
  - Cluster 3: p



A dendrogram example

Hierarchical clustering - algorithm

- **Hierarchical clustering algorithm is a type of agglomerative clustering**

- Given a set of *N* items to be clustered, hierarchical clustering algorithm:

  1. Start by assigning each item to its own cluster, so that if you have N items, you now have *N* clusters, each containing just one item

  2. Find the closest distance (most similar) pair of clusters and merge them into a single cluster, so that now you have one less cluster

  3. Compute pairwise distances between the new cluster and each of the old clusters

  4. Repeat steps 2 and 3 until all items are clustered into a single cluster of size *N*

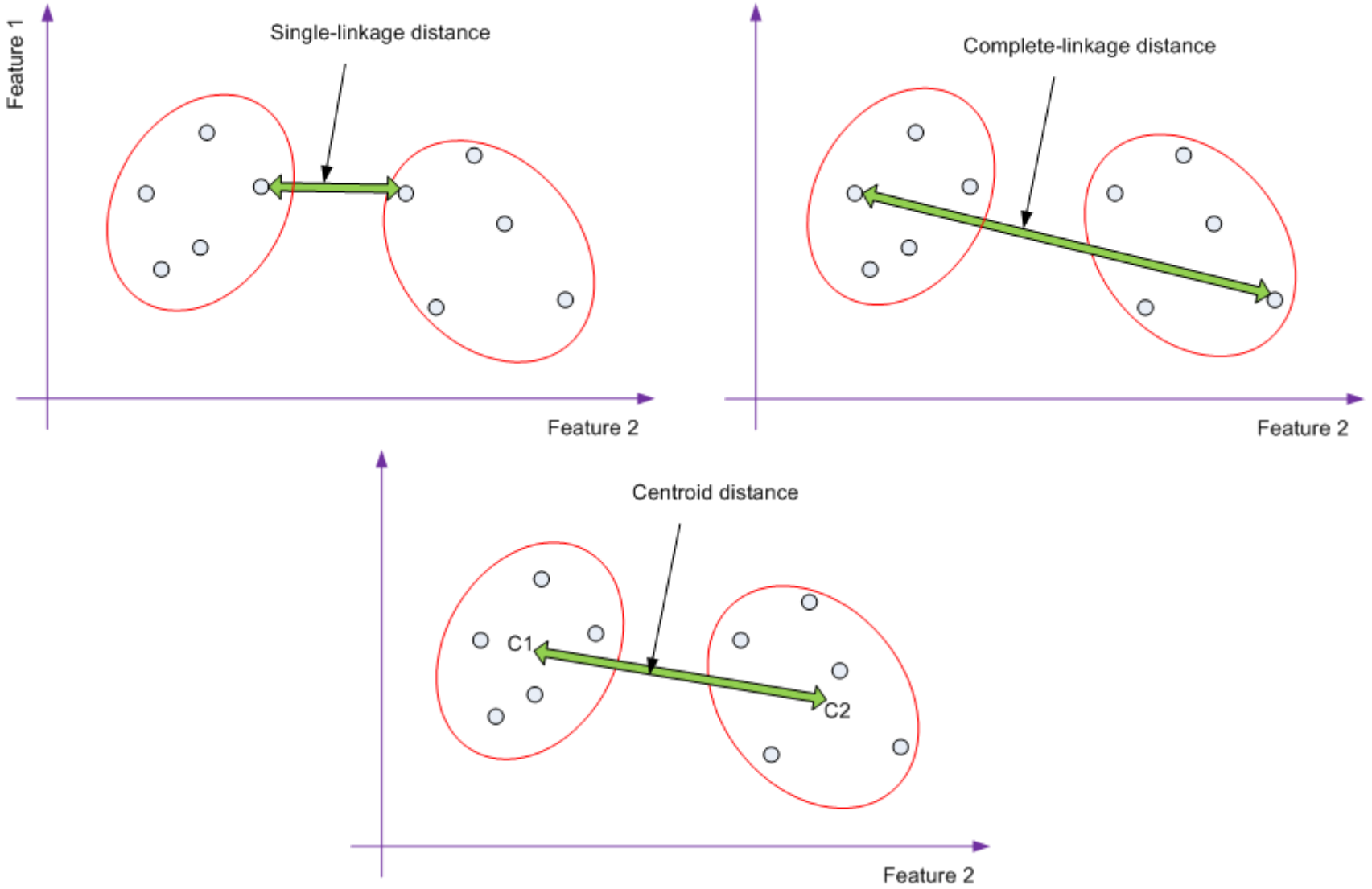
  5. Draw the dendogram, and with the complete hierarchical tree, if you want *K* clusters you just have to cut the *K*-1 top links

Note any distance measure can be used: Euclidean, Manhattan, etc

Hierarchical clustering algorithm– step 3

- *Computing distances between clusters for Step 3 can be implemented in different ways:*
  - Single-linkage clustering
  - The distance between one cluster and another cluster is computed as the shortest distance from any member of one cluster to any member of the other cluster
  - Complete-linkage clustering
  - The distance between one cluster and another cluster is computed as the greatest distance from any member of one cluster to any member of the other cluster
  - Centroid clustering
  - The distance between one cluster and another cluster is computed as the distance from one cluster centroid to the other cluster centroid

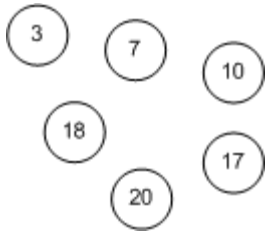
Hierarchical clustering algorithm– step 3



Hierarchical clustering – an example

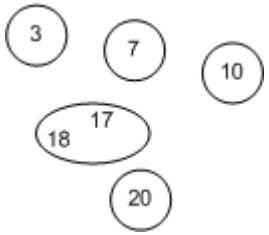
- Assume X=[3 7 10 17 18 20]

1. There are 6 items, so create 6 clusters initially



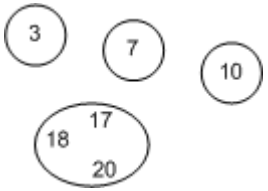
2. Compute pairwise distances of clusters (assume Manhattan distance)

The closest clusters are 17 and 18 (with distance=1), so merge these two clusters together



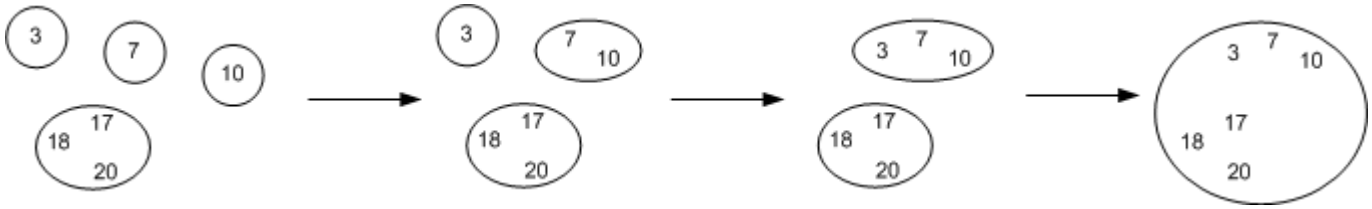
3. Repeat step 2 (assume single-linkage):

The closest clusters are $\text{cluster}_{17,18}$ to cluster_{20} (with distance |18-20|=2), so merge these two clusters together

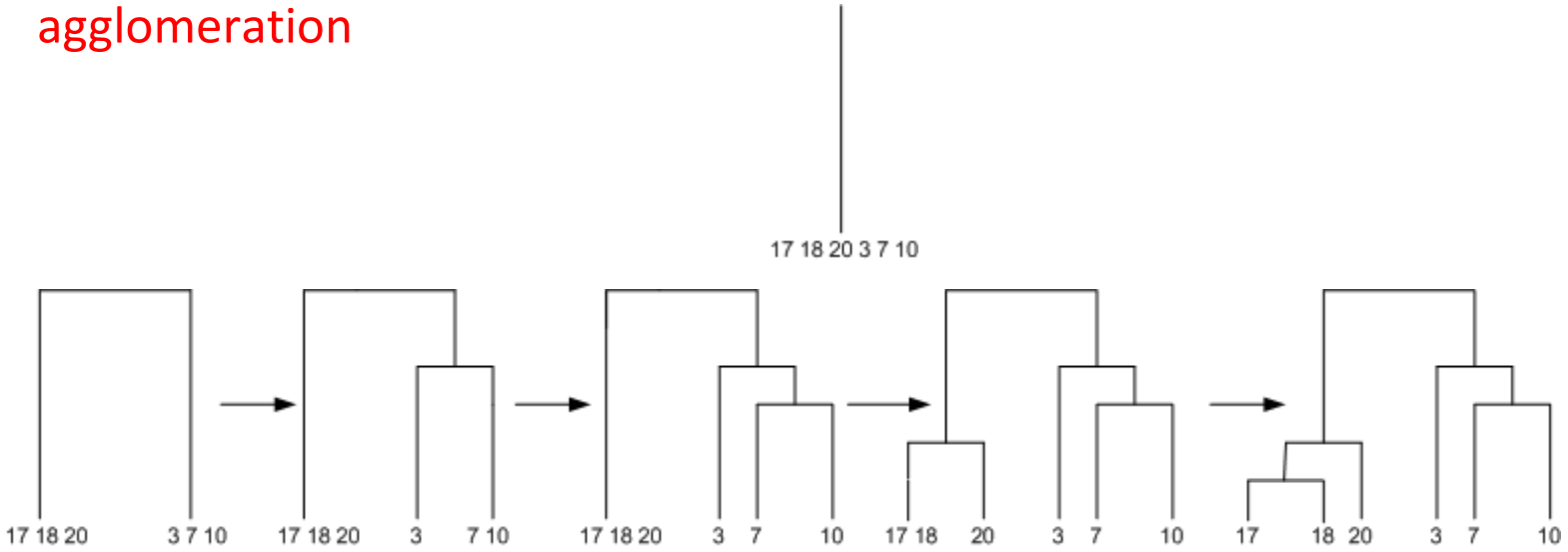


Hierarchical clustering – an example (ctd)

- Go on repeat cluster mergers until one big cluster remains

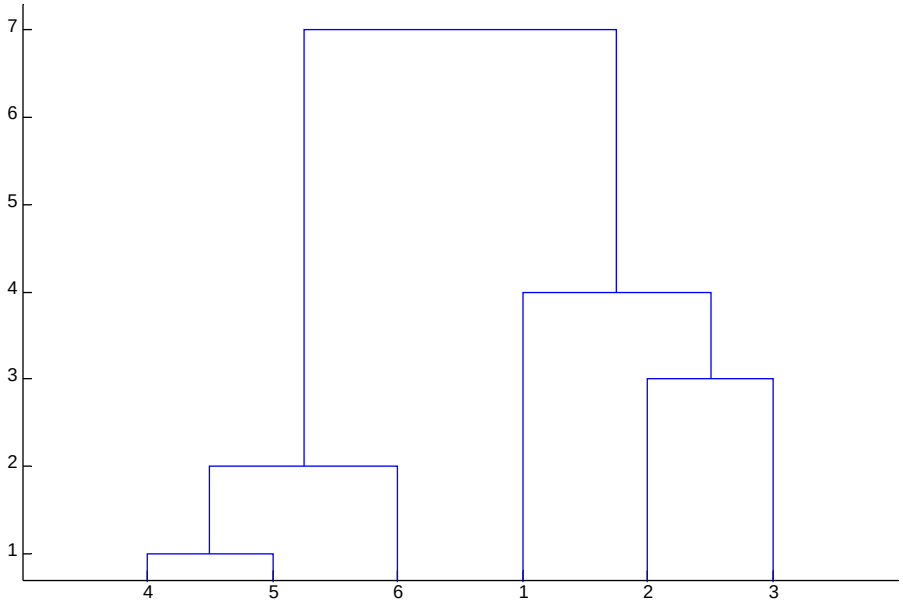


- Draw dendrogram (draw it in reverse of the cluster mergers) – remember the height of each cluster corresponds to the manner of cluster agglomeration



Hierarchical clustering – an example (ctd) –using MATLAB

- %Hierarchical clustering example
- X=[3 7 10 17 18 20]; %data
- Y=pdist(X', 'cityblock'); %compute pairwise Manhattan distances
- Z=linkage(Y, 'single'); %do clustering using single-linkage method
- dendrogram(Z); %draw dendrogram –note only indices are shown



Comparing agglomerative vs exclusive clustering

- Agglomerative - advantages
  - Preferable for detailed data analysis
  - Provides more information than exclusive clustering
  - We can decide on any number of clusters without the need to redo the algorithm –in exclusive clustering, *K* has to be decided first, if a different *K* is used, then need to redo the whole exclusive clustering algorithm
  - One unique answer
- Disadvantages
  - Less efficient than exclusive clustering
  - No backtracking, i.e. can never undo previous steps

Overlapping clustering –Fuzzy C-means algorithm

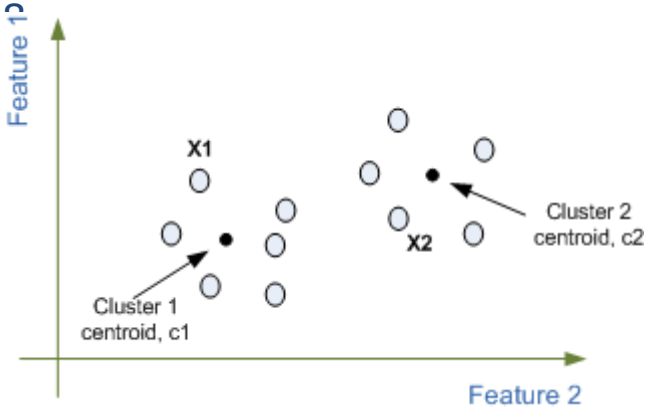
- Both agglomerative and exclusive clustering allows one data to be in one cluster only
- Fuzzy C-means (FCM) is a method of clustering which allows one piece of data to belong to more than one cluster
- In other words, each data is a member of every cluster but with a certain degree known as membership value
- This method (developed by Dunn in 1973 and improved by Bezdek in 1981) is frequently used in pattern recognition
- Fuzzy partitioning is carried out through an iterative procedure that updates membership u_{ij} and the cluster centroids c_j by

$$u_{ij} = \frac{1}{\sum_{k=1}^{C} \left(\frac{\| x_i - c_j \|}{\| x_i - c_k \|} \right)^{\frac{2}{m-1}}} \qquad c_j = \frac{\sum_{i=1}^{N} u_{ij}^m x_i}{\sum_{i=1}^{N} u_{ij}^m}$$

where m > 1, and represents the degree of fuzziness (typically, m=2)

Overlapping clusters?

- Using both agglomerative and exclusive clustering methods, data X_1 will be member of cluster 1 only while X_2 will be member of cluster 2 only
- However, using FCM, data X can be member of both clusters
- FCM uses distance measure too, so the further data is from that cluster centroid, the smaller the membership value will be
- For example, membership value for X_1 from cluster 1, u_{11}=0.73 and membership value for X_1 from cluster 2, u_{12}=0.27
- Similarly, membership value for X_2 from cluster 2, u_{22}=0.2 and membership value for X_2 from cluster 1, u_{21}=0.8



- Note: membership values are in the range 0 to 1 and membership values for each data from all the clusters will add to 1

Fuzzy *C*-means algorithm

- Choose the number of clusters, *C* and m, typically *2*

  1. Initialise all u_{ij}, membership values randomly – matrix U^0
  2. At step *k*: Compute centroids, c_j using $$c_j = \frac{\sum_{i=1}^{N} u_{ij}^m x_i}{\sum_{i=1}^{N} u_{ij}^m}$$
  3. Compute new membership values, u_{ij} using $$u_{ij} = \frac{1}{\sum_{k=1}^{C} \left(\frac{\| x_i - c_j \|}{\| x_i - c_k \|} \right)^{\frac{2}{m-1}}}$$
  4. Update $U^{k+1} \leftarrow U^k$
  5. Repeat steps 2-4 until change of membership values is very small, $U^{k+1} - U^k < \varepsilon$ where ε is some small value, typically 0.01

Note: || means Euclidean distance, | means Manhattan

However, if the data is one dimensional (like the examples here), Euclidean distance = Manhattan distance

Fuzzy *C*-means algorithm – an example

- X=[3 7 10 17 18 20] and assume *C*=2
- Initially, set U randomly $U = \begin{matrix} 0.1 & 0.2 & 0.6 & 0.3 & 0.1 & 0.5 \\ 0.9 & 0.8 & 0.4 & 0.7 & 0.9 & 0.5 \end{matrix}$
- Compute centroids, c_j using $c_j = \frac{\sum_{i=1}^{N} u_{ij}^m x_i}{\sum_{i=1}^{N} u_{ij}^m}$, assume *m*=2
- c_1=13.16; c_2=11.81
- Compute new membership values, u_{ij} using $u_{ij} = \frac{1}{\sum_{k=1}^{C} \left(\frac{\| x_i - c_j \|}{\| x_i - c_k \|} \right)^{\frac{2}{m-1}}}$
- New U: $U = \begin{matrix} 0.43 & 0.38 & 0.24 & 0.65 & 0.62 & 0.59 \\ 0.57 & 0.62 & 0.76 & 0.35 & 0.38 & 0.41 \end{matrix}$
- Repeat centroid and membership computation until changes in membership values are smaller than say 0.01

Fuzzy *C*-means algorithm –using MATLAB

- Using 'fcm' function in MATLAB
- The final membership values, U gives an indication on similarity of each item to the clusters
- For eg: item 3 (no. 10) is more similar to cluster 1 compared to cluster 2 but item 2 (no. 7) is even more similar to cluster 1

```
>> X=[3 7 10 17 18 20]

X =

     3     7    10    17    18    20

>> [centroid,U] = fcm(X',2)
Iteration count = 1, obj. fcn = 121.924664
Iteration count = 2, obj. fcn = 81.218073
Iteration count = 3, obj. fcn = 31.707888
Iteration count = 4, obj. fcn = 27.045564
Iteration count = 5, obj. fcn = 26.882882
Iteration count = 6, obj. fcn = 26.873648
Iteration count = 7, obj. fcn = 26.873043
Iteration count = 8, obj. fcn = 26.873002
Iteration count = 9, obj. fcn = 26.873000

centroid =

    6.4287
   18.2453


U =

    0.9519    0.9974    0.8420    0.0137    0.0004    0.0164
    0.0481    0.0026    0.1580    0.9863    0.9996    0.9836

>>
```

Fuzzy *C*-means algorithm –using MATLAB

- 'fcm' function requires Fuzzy Logic toolbox
- So, using MATLAB but without 'fcm' function:

$$c_j = \frac{\sum_{i=1}^{N} u_{ij}^m x_i}{\sum_{i=1}^{N} u_{ij}^m}$$

$$u_{ij} = \frac{1}{\sum_{k=1}^{C} \left(\frac{\| x_i - c_j \|}{\| x_i - c_k \|} \right)^{\frac{2}{m-1}}}$$

```
%fcm
clc; clear all; close all;

X=[3 7 10 17 18 20];

Unew=[0.1 0.2 0.6 0.3 0.1 0.4; 0.9 0.8 0.4 0.7 0.9 0.6];
U=[0 0 0 0 0 0; 0 0 0 0 0 0];

iteration=0;
while (sum(abs(U-Unew))>0.01)

U=Unew;

for k=1:2,
C(k)=sum(power(U(k,:),2)*X(:))/sum(power(U(k,:),2));
end

for j=1:6,
    for k=1:2,
    temp=0;
        for kk=1:2,
        temp=temp+power((abs(X(j)-C(k))/abs(X(j)-C(kk))),2);
        end
    Unew(k,j)=1/temp;
    end
end

iteration=iteration+1;
end % end of do loop
```

Clustering validity problem

- *Problem 1*
- A problem we face in clustering is to decide the optimal number of clusters that fits a data set
- *Problem 2*
- The various clustering algorithms behave in a different way depending on
  - the features of the data set the features of the data set (geometry and density distribution of clusters)
  - the input parameters values (eg: for *K*-means, initial cluster choices influence the result)
- So, how do we know which clustering method is better/suitable?
- *We need a clustering quality criteria*

Clustering validity problem

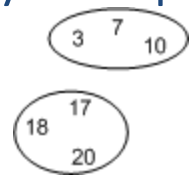
- In general, good clusters, should have
  - High intra–cluster similarity, i.e. low variance among intra-cluster members

  where variance for *x* is defined by

$$\text{var}(x) = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2$$

  with $\bar{x}$ as the mean of *x*
- For eg: if *x*=[2 4 6 8], then $\bar{x} = 5$ so *var(x)=6.67*

- Computing intra-cluster similarity is simple
- For eg: for the clusters shown



- *var(cluster1)*=2.33 while *var(cluster2)*=12.33
- So, cluster 1 is better than cluster 2
- *Note: use 'var' function in MATLAB to compute variance*

Clustering Quality Criteria

- But this does not tell us anything about how good is the overall clustering or on the suitable number of clusters needed!
- To solve this, we also need to compute inter-cluster variance
- Good clusters will also have low inter–cluster similarity, i.e. high variance among inter-cluster members in addition to high intra–cluster similarity, i.e. low variance among intra-cluster members
- One good measure of clustering quality is Davies-Bouldin index
- The others are
  - Dunn's Validity Index
  - Silhouette method
  - C–index
  - Goodman–Kruskal index
- So, we compute DB index for different number of clusters, *K* and the best value of DB index tells us on the appropriate *K* value or on how good is the clustering method

Davies-Bouldin index

- It is a function of the ratio of the sum of within-cluster (i.e. intra-cluster) scatter to between cluster (i.e. inter-cluster) separation
- Because a low scatter and a high distance between clusters lead to low values of *Rij , a minimization* of *DB index is desired*
- Let $C=\{C_1,\ldots..,C_k\}$ be a clustering of a set of N objects:

$$DB = \frac{1}{k}.\sum_{i=1}^{k} R_i$$

with $R_i = \max_{j=1,..k,i\neq j} R_{ij}$ and $R_{\substack{ij \\ i\neq j}} = \frac{\mathrm{var}(C_i)+\mathrm{var}(C_j)}{\| c_i - c_j \|}$

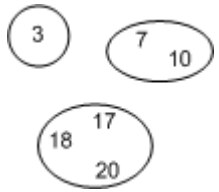
where C_i is the i^{th} cluster and c_i is the centroid for cluster i

Numerator of R_{ij} is a measure of intra-cluster similarity while the denominator is a measure of inter-cluster separation

Note, $R_{ij}=R_{ji}$

Davies-Bouldin index example

- For eg: for the clusters shown



- Compute $R_{\substack{ij \\ i \neq j}} = \frac{\text{var}(C_i) + \text{var}(C_j)}{\| c_i - c_j \|}$

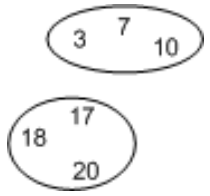
Note, variance of one element is zero and centroid is simply the element itself

- *var*(C_1)=0, *var*(C_2)=4.5, *var*(C_3)=2.33
- Centroid is simply the mean here, so c_1=3, c_2=8.5, c_3=18.33
- *So,* R_{12}=1, R_{13}=0.152, R_{23}=0.797
- Now, compute $R_i = \max_{j=1,..k, i \neq j} R_{ij}$

- R_1=1 (max of R_{12} and R_{13}); R_2=1 (max of R_{21} and R_{23}); R_3=0.797 (max of R_{31} and R_{32})
- Finally, compute $DB = \frac{1}{k} \cdot \sum_{i=1}^{k} R_i$
- DB=0.932

Davies-Bouldin index example (ctd)

- For eg: for the clusters shown



- Compute $R_{\substack{ij \\ i \neq j}} = \frac{\text{var}(C_i) + \text{var}(C_j)}{\| c_i - c_j \|}$

- Only 2 clusters here
- *var(C_1)*=12.33 while *var(C_2)*=2.33; c_1=6.67 while c_2=18.33
- R_{12}=1.26
- Now compute $R_i = \max_{j=1,..k, i \neq j} R_{ij}$
- *Since we have only 2 clusters here, R_1=R_{12}=1.26; R_2=R_{21}=1.26*
- Finally, compute $DB = \frac{1}{k} . \sum_{i=1}^{k} R_i$
- DB=1.26

Davies-Bouldin index example (ctd)

- DB with 2 clusters=1.26, with 3 clusters=0.932
- *So, K=3 is better than K=2 (since DB smaller, better clusters)*
- In general, we will repeat DB index computation for all cluster sizes from 2 to *N-1*
- So, if we have 10 data items, we will do clustering with *K*=2, …..9 and then compute DB for each value of *K*
  - K=10 is not done since each item is its own cluster
- Then, we decide the best clustering size (and the best set of clusters) would be the one with minimum values of DB index

Lecture 7: Study Guide

At the end of this lecture, you should be able to

- Define clustering
- Name major clustering approaches and differentiate between them
- State the *K*-means algorithm and apply it on a given data set
- State the hierarchical algorithm and apply it on a given data set
- Compare exclusive and agglomerative clustering methods
- State FCM algorithm and apply it to a given data set
- Identify major problems with clustering techniques
- Define and use cluster validity measures such as DB index on a given data set