COMP3308/3608, Lecture 12

ARTIFICIAL INTELLIGENCE

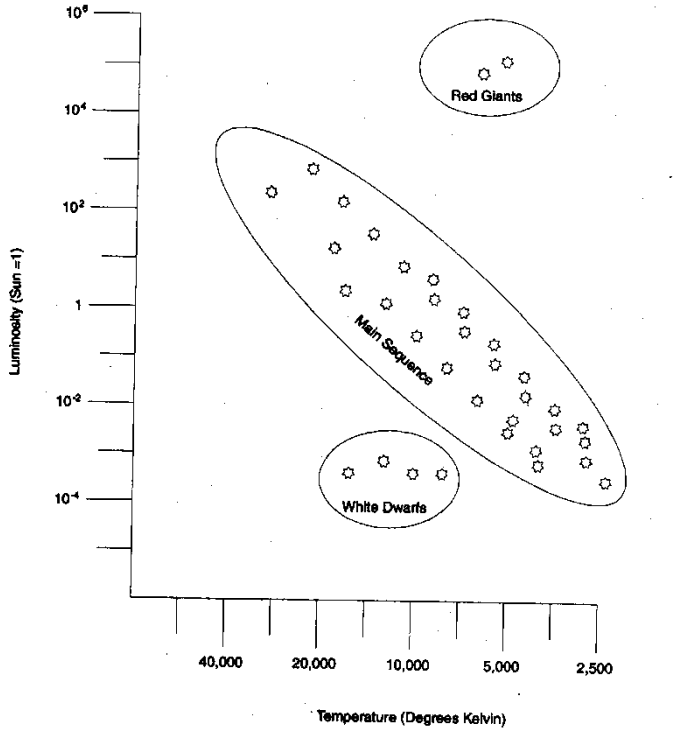
Unsupervised Learning (Clustering)

Outline

- **Introduction to clustering**
- **Clustering algorithms**
  - **K-Means**
  - **K-Medoids (COMP3608 only)**
  - **Nearest neighbour**
  - **Hierarchical**

What is Clustering?

- **Clustering – the process of partitioning the data into a set of groups (called *clusters*) so that the items from the same cluster are:**
  - **similar to each other**
  - **dissimilar to the items in other clusters**
- **Similarity is defined in terms of distance measure**

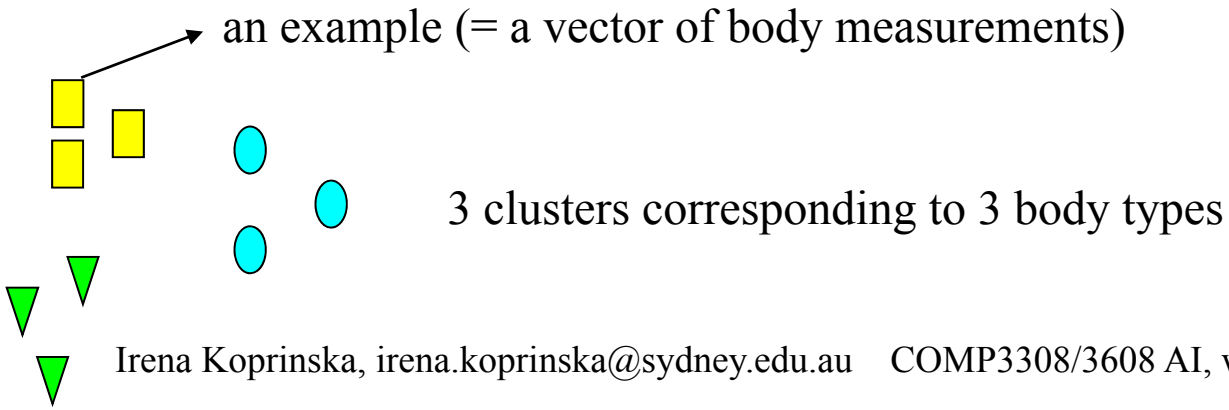


- **Ex. Star clustering based on temperature and brightness (Hertzsprung-Russel diagram)**
  - **3 well-defined clusters**
  - **now we understand that the groups represent stars in 3 different phases of their life**

from "Data Mining Techniques", M. Berry, G. Linoff, John Wiley and Sons Publ.

Clustering Example – Fitting Troops

- **Re-designing the uniforms for female soldiers in the US army**
  - **Goal: reduce the number of uniform sizes to be kept in inventory while still providing good fit**
- **Researchers from Cornell University used clustering and designed a new set of sizes**
  - **Traditional clothing system: ordered set of sizes where all dimensions increase together**
  - **The new system: sizes that fit *body types***
    - **E.g. one size for women with short legs, small waist, long torsos, average arms, broad shoulders and skinny necks**



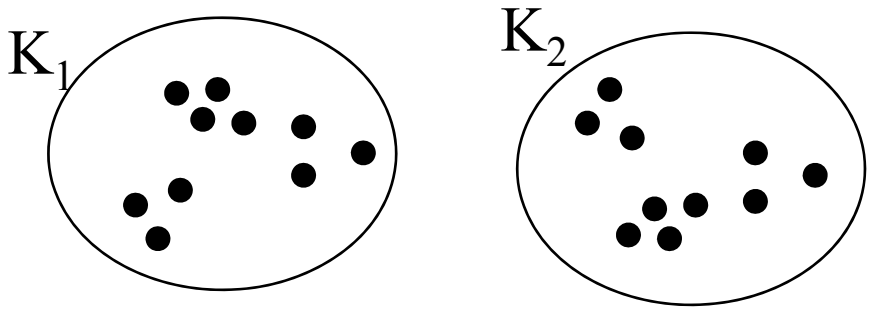
Supervised vs Unsupervised Learning

- **Clustering is *unsupervised learning*: no labels**
  - **Given:**
    - **A set of unlabeled examples (input vectors) x_i**
    - ***k* – desired number of clusters (may not be given)**
  - **Task: Cluster (group) the examples into *k* clusters (groups)**

- **Supervised vs unsupervised learning**
  - **Supervised: We know the class labels and the number of classes. We want to build a classifier that can be used to predict the class of new (unlabelled) examples.**
  - **Unsupervised: We do not know the class labels and may not know the number of classes. We want to group similar examples together.**

Clustering Task – Formal Definition

- **Given:**
  - **a dataset P={p_1,…, p_n} of input vectors (examples, instances, points, data objects, items, records)**
  - **an integer k - the number of clusters we are looking for**
- **Find:**
  - **a mapping f: P->{1,…,k} where each p_i is assigned to 1 cluster K_j, 1≤j ≤ k**
- **Result:**
  - **a set of clusters K={K_1,K_2,…,K_k}**



- **Note: According to this definition:**
  - **each example is assigned to exactly 1 cluster. Some clustering algorithms (e.g. fuzzy and probabilistic) assign each example to each class with a certain probability.**
  - **not all clustering algorithms require k to be specified (e.g. agglomerative, SOM)**

Typical Clustering Applications

- **As a *stand-alone tool* to group data**
- **As a *building block* for other algorithms**
  - **e.g. pre-processing tool for dimensionality reduction – using the cluster center to represent all data points in the cluster (also called vector quantization)**

More Examples of Clustering Applications

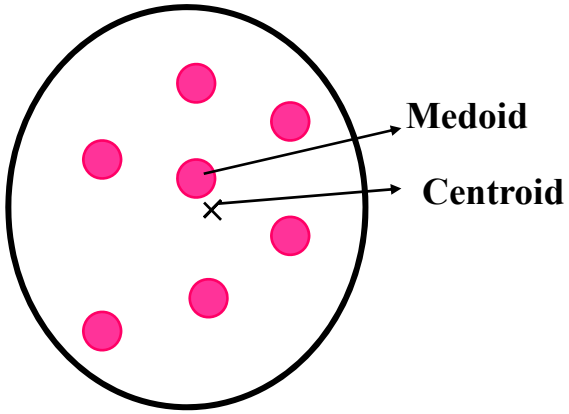
- **Marketing**
  - **find distinct groups of customers, and then use this knowledge to develop targeted marketing programs (e.g. potential customers of a new product)**
- **Biology**
  - **derive plant and animal taxonomies**
  - **find genes with similar expression patterns; often they have similar function**
- **Land use**
  - **find areas of similar land use in an earth observation database**
- **Insurance**
  - **find groups of insurance policy holders with a high average claim cost**
- **City-planning**
  - **identify groups of houses according to their house type, value, and geographical location**

Centroid and Medoid of a Cluster

- **Consider a cluster K of N points $\{p_1,..,p_N\}$**
- ***Centroid (means)* – the "middle" of the cluster**
  - **no need to be an actual data point in the cluster**

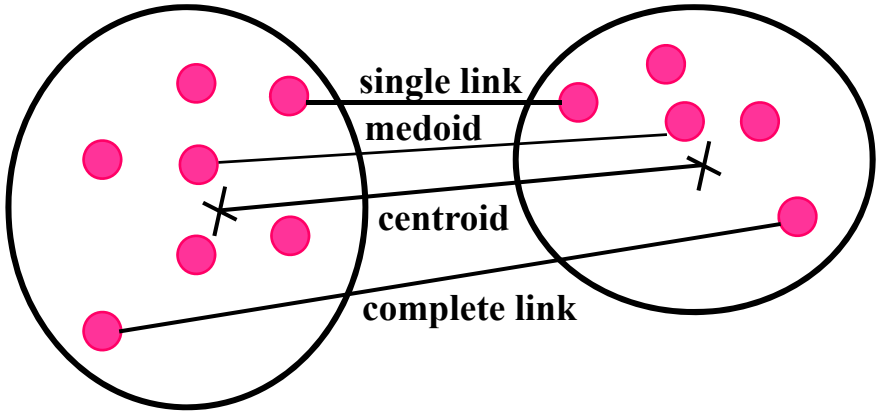
$$C = \frac{\sum_{i=1}^{n} p_i}{N}$$

- ***Medoid M* – the centrally located data point in the cluster**



Distance Between Clusters

- **Different ways to define it:**

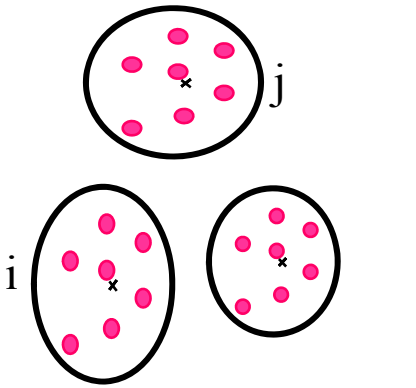


- **Centroid – the distance between the centroids**
- **Medoid – the distance between the medoids**
- **Single link (MIN) – The smallest pairwise distance between elements from each cluster**
- **Complete link (MAX) – the largest pairwise distance between elements from each cluster**
- **Average link – the average pairwise distance between elements from each cluster**

What is a Good Clustering?

- **A *good clustering* will produce clusters with**
  - **High *cohesion* (i.e. high similarity within the cluster)**
  - **High *separation* (i.e. low similarity between the clusters)**
- **Cohesion and separation are measured with a distance function**
- **Various ways to combine them in 1 measure**
  - **Davies-Bouldin (DB) index**
  - **Silhouette coefficient**

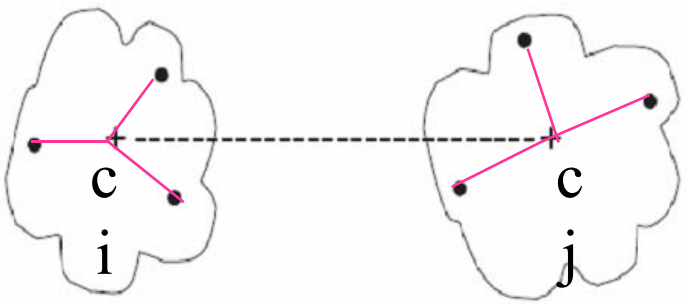
Davies-Bouldin (DB) index



a clustering consisting of 3 clusters

- **A heuristic measure of the quality of the *clustering***
- **Combines *cohesion* and *separation***
- **Each pair of clusters *i* and *j* (from the resulting clustering) are compared in pairs:**

$$DB = \frac{1}{k}\sum_{i=1}^{k} \max_{j, j\neq i}\left[\frac{dist(\mathbf{x}, c_i) + dist(\mathbf{x}, c_j)}{dist(c_i, c_j)}\right]$$



***k* – number of clusters**

c_i and c_j – centroids of clusters *i* and *j*

***dist*(x, c_i) – mean-squared distance from each item x in cluster *i* to its centroid**

dist*(ci,c_j) – distance between the centroids of cluster *i* and *j

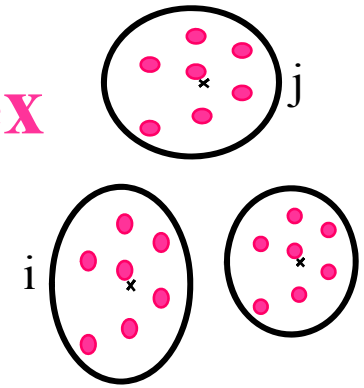
What is the DB index for a good clustering – big or small?

A Closer Look at the DB-index

Spread of the items within cluster i

$$DB = \frac{1}{k}\sum_{i=1}^{k} \max_{j, j\neq i}\left[\frac{dist(\mathbf{x}, c_i) + dist(\mathbf{x}, c_j)}{dist(c_i, c_j)}\right]$$

Distance between the clusters (centroid distance)



- **The summation is pair-wise (for pairs of clusters from the clustering)**
- **meaning of "max" - for each cluster *i*, find the cluster *j* that maximizes the fraction in brackets:**

  - denominator is low: small distance between the centroids of *i* and *j*, i.e. possibly overlapping clusters *i* and *j* AND/OR

  - nominator is high: big spread within each of the clusters

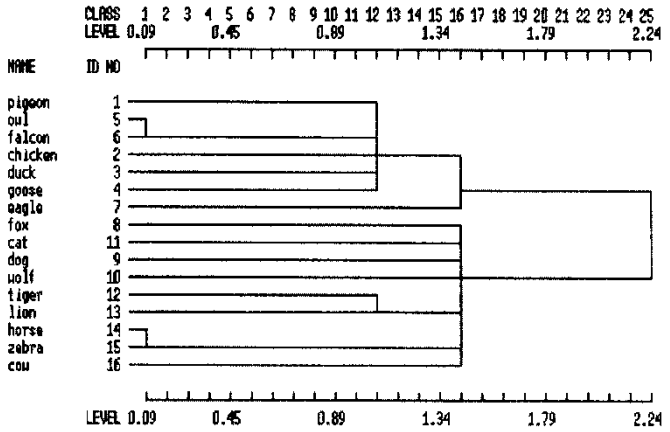
- **=> DB index is sum of max-es (worst pairing for each cluster) and we would like to keep it minimum**
  - **Small DB index = clusters have small spread and are far from each other =good clustering**

Taxonomy of Clustering Algorithms

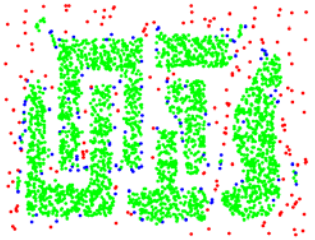
- **Partitional – k-means, k-medoids, nearest neighbor, SOM**
  - create only one set of clusters
  - The number of clusters *k* is required for most algorithms (e.g. k-means and k-medoids) and not required for some (k-nearest neighbor and SOM)

Eagle	Cow
Falcon	Zebra
Owl	Horse
Goose	Lion
Duck	Tiger
Chicken	Cat
Pigeon	Wolf
	Dog
	Fox

- **Hierarchical - agglomerative and divisive**
  - Creates a nested set of clusters
  - k does not need to be specified



- **Density-based - DBSCAN**
  - regions of high density form clusters
  - *k* does not need to be specified directly
- **Model-based (generative) – EM**
- **Fuzzy clustering**

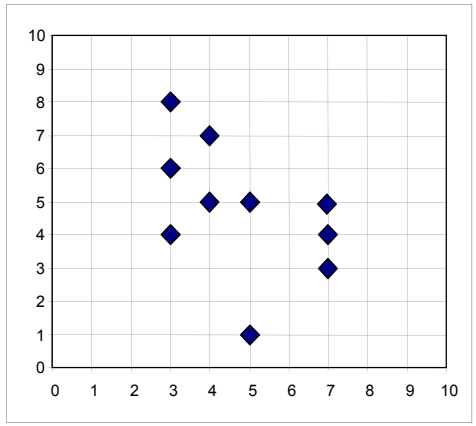


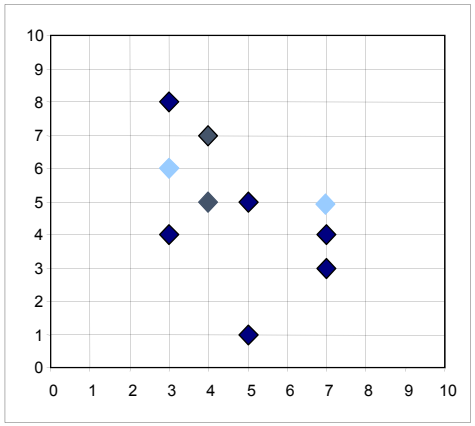
K-Means Clustering Algorithm

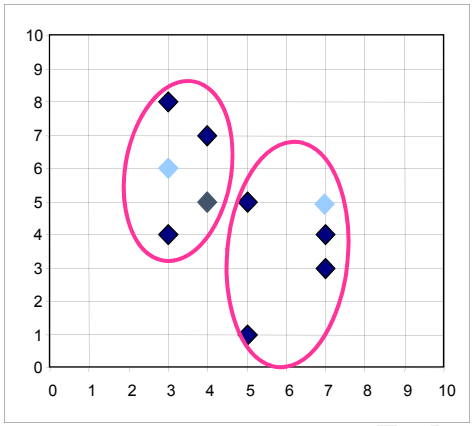
K-Means Clustering Algorithm

- **Partitional clustering algorithm; simple and very popular**
- **Iterative and distance-based**
- **Requires the number of clusters *k* to be specified in advance**
- **Can be implemented in 3 steps:**
  1. **Choose *k* examples as the initial centroids (seeds) of the clusters**
  2. **Form *k* clusters by assigning each example to the closest centroid**
  3. **At the end of each epoch:**
     - **Re-compute the centroid of the clusters**
     - **Check if the stopping criterion is satisfied: centroids do not change. If yes – stop; otherwise, repeat steps 2 and 3 using the new centroids.**

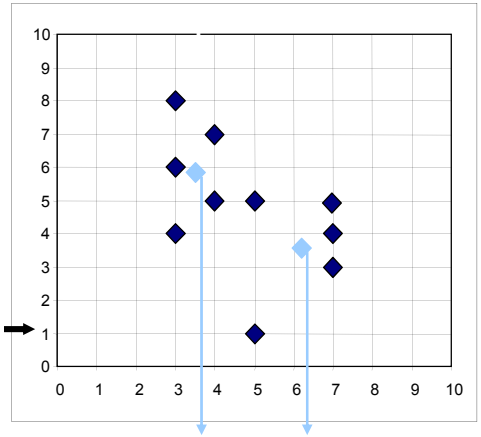
K-Means - Example





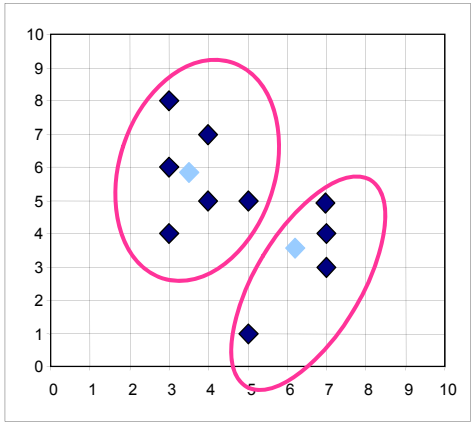


End of epoch 1

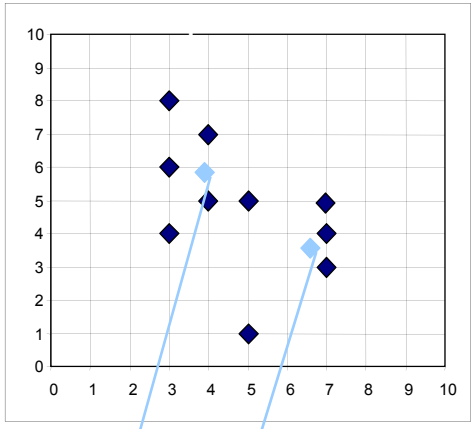


(3.4,6) (6.2,3.6)

Stop? No

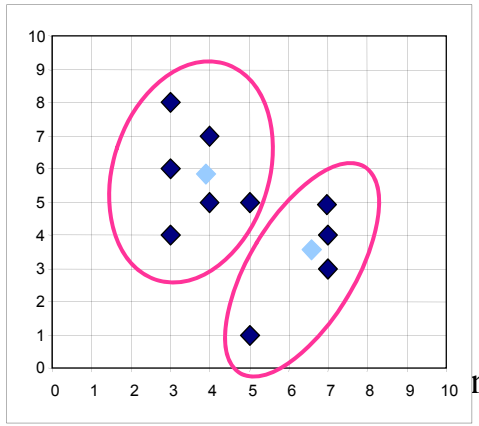


End of epoch 2

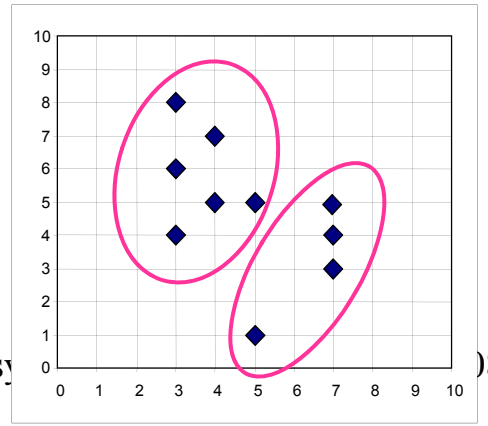


(3.7,5.8) (6.5,3.3)

Stop? No



End of epoch 3; recompute centroids; no change; stop



K-means Clustering - Example

- **Given: 5 items with the distance between them**
- **Task: Cluster them into 2 groups using the k-means algorithm. Assume that the initial centroids (seeds) are B and C. Show the clusters after the first epoch.**

Item	A	B	C	D	E
A	0	1	2	2	3
B	1	0	2	4	3
C	2	2	0	1	5
D	2	4	1	0	3
E	3	3	5	3	0

From Tan, Steinbach & Kumar, Introduction to Data Mining

Algorithm 1 Basic K-means Algorithm.

1: Select K points as the initial centroids.
2: **repeat**
3: Form K clusters by assigning all points to the closest centroid.
4: Recompute the centroid of each cluster.
5: **until** The centroids don't change

Solution

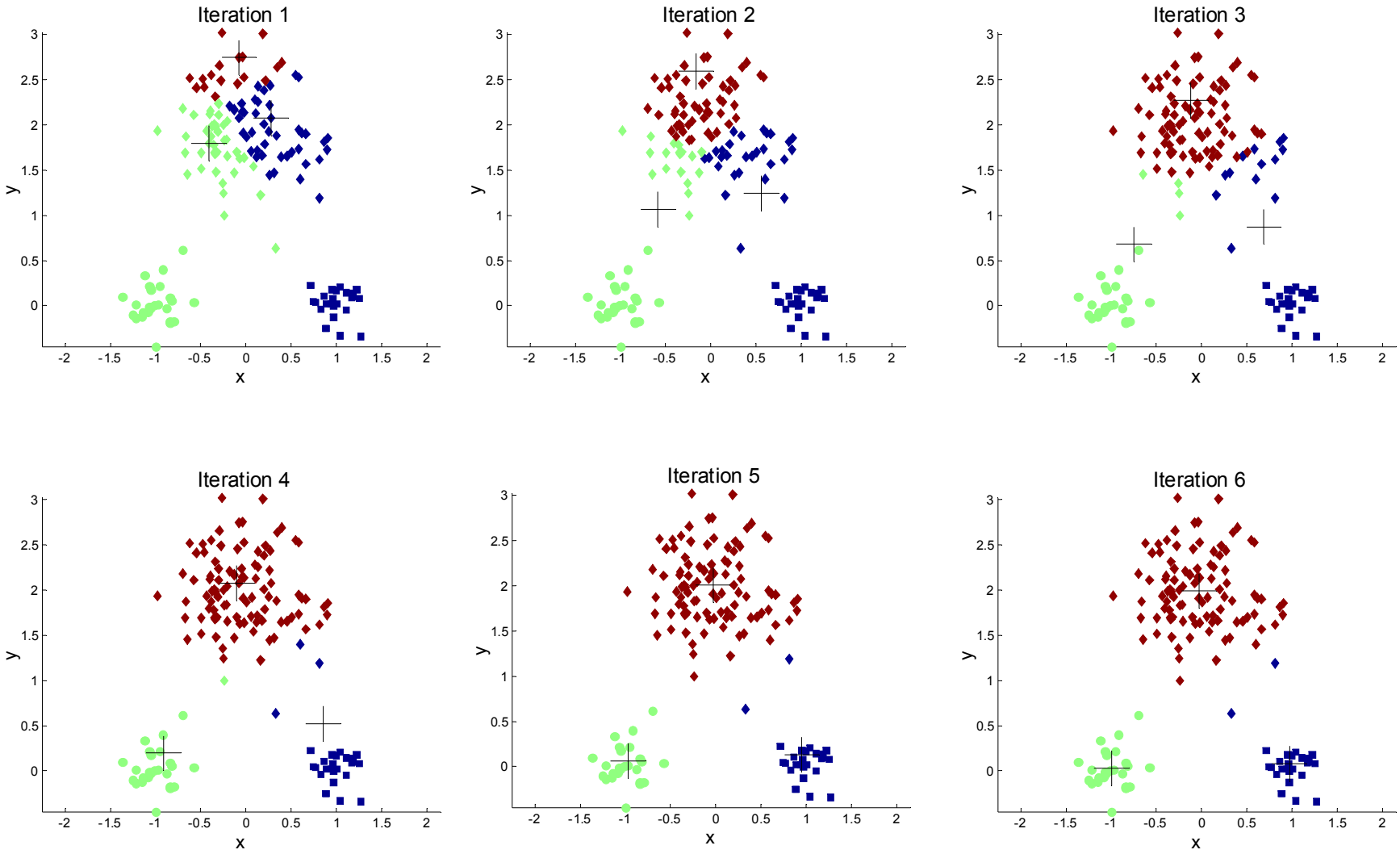
Item	A	B	C	D	E
A	0	1	2	2	3
B	1	0	2	4	3
C	2	2	0	1	5
D	2	4	1	0	3
E	3	3	5	3	0

K-means - Issues

- **Typical values of k: 2 to 10**
- **Different distance measures can be used**
  - **typically Euclidean distance**
- **Data should be normalized**
- **Nominal data need to be converted into numeric**
- **The number of epochs for convergence is typically much smaller than the number of points, i.e. converges quickly**
- **Often the stopping criterion is changed to 'Until relatively few points change clusters", e.g. <1%**
- **Sensitive to the choice of initial seeds (centroids)**
  - **different runs of k-means produce different clusters (see next slide)**
  - **there are several approaches for choosing "good" initial centroids**

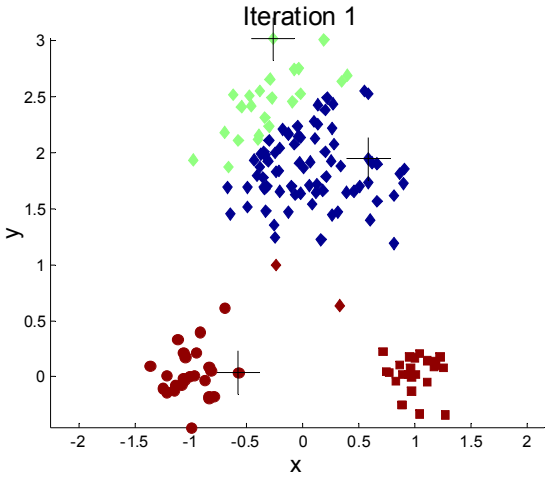
Images from Tan, Steinbach & Kumar, Introduction to Data Mining

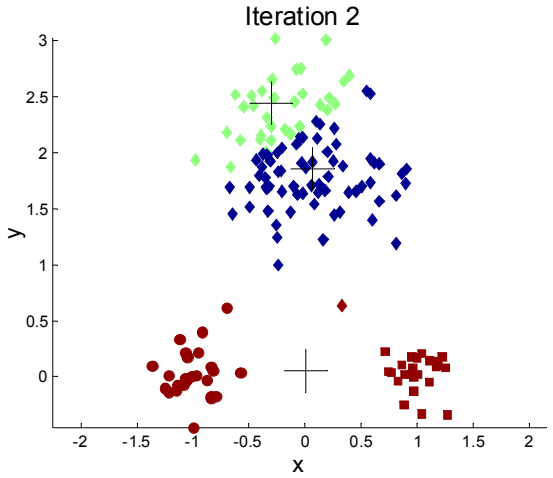
Good Initial Centroids

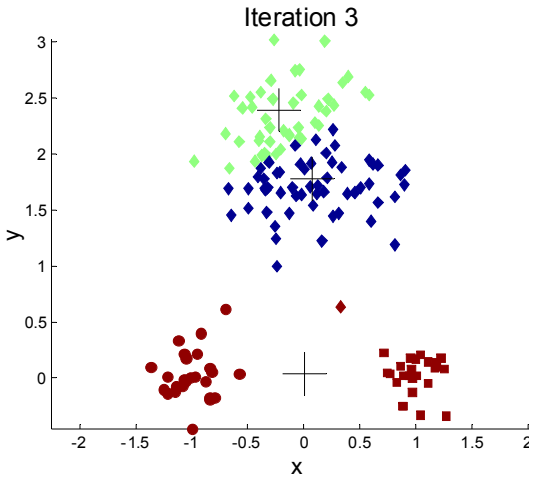


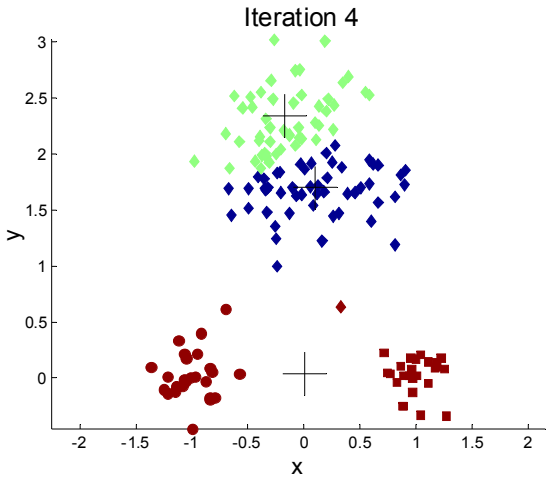
Images from from Tan, Steinbach & Kumar, Introduction to Data Mining

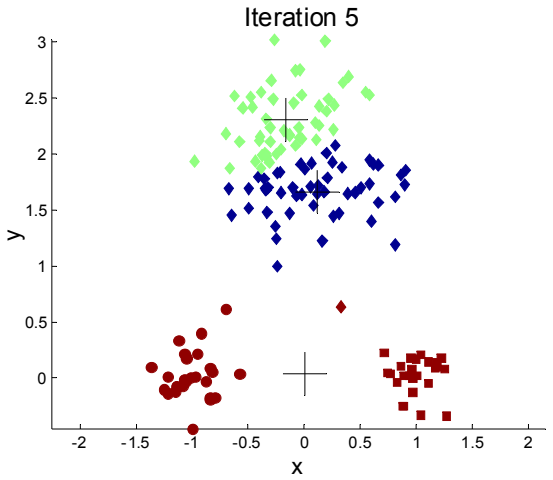
Poor Initial Centroids











Time and Space Complexity

- **Space complexity - modest**
  - **All examples and centroids need to be stored**
  - ***O((m+k)n)*** **where: *m*** **– number of examples, *n*** **– number of attributes, *k*** **– number of clusters**
- **Time complexity – expensive**
  - ***O(tkmn), t*** **- number of iterations**
  - **Involves finding the distance from each example to each cluster center at each iteration**
  - ***t*** **is often small and can be safely bounded as most changes occur in the first few iterations**
- **Not practical for large datasets**

K-means as an Optimization Problem

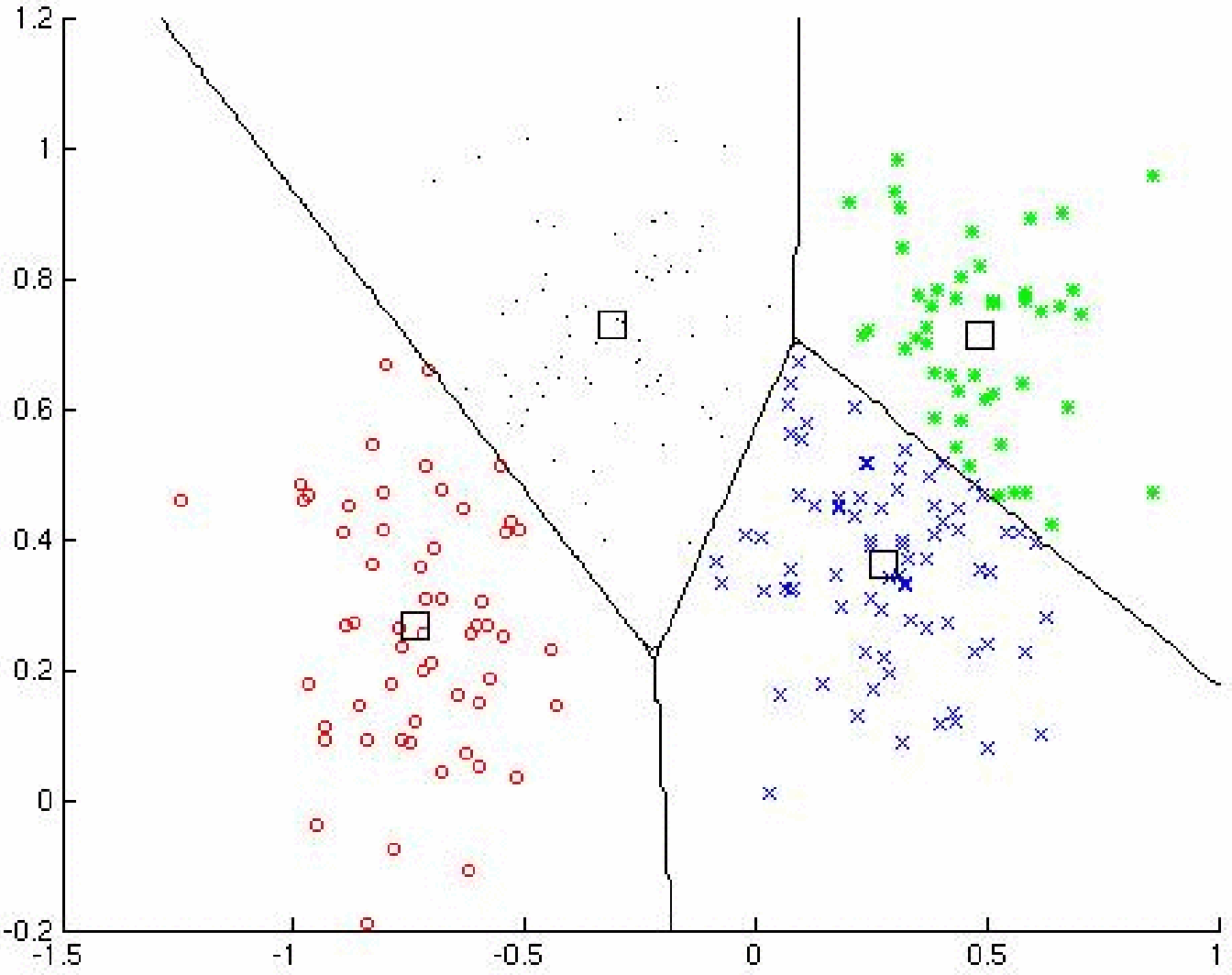
- **k-means clustering can be viewed as an optimization problem: find k clusters that minimize the sum-squared error (SSE)**

$$SSE = \sum_{i=1}^{k} \sum_{\mathbf{x} \in K_i} dist(c_i, \mathbf{x})^2$$

  - **c_i are the centroids, k- number of clusters, x – items**
- **Not guaranteed to find the global minimum, i.e. may find a local minimum**

Voronoi diagram for k-means – example

www.math.unimaas.nl/personal/ronaldw/DAM/voordracht-DAM_2.ppt



K-means – Strengths and Weaknesses

- **Simple and very popular**
- **Relatively efficient**
  - **Modest space complexity**
  - **Fast convergence (typically)**
- **Not guaranteed to find the optimal solution**
- **Sensitive to initialization**
  - **In practice: run it several times with different initialization, select the best clustering found**
- **Not sensitive to the order of input examples**
- **Does not work well for clusters with**
  - **non-spherical shape**
  - **non-convex shape**
- **Does not work well with data containing outliers**
  - **Pre-processing is needed - outlier detection and removal**

K-means – Other Issues

- **The best number of clusters *k* is not known in advance**
  - **There is more than one correct answer to a clustering problem**
  - **Domain expert may be required to suggest a good number of clusters and also to evaluate a solution**
- **Interpreting the semantic meaning of each cluster is difficult**
  - **Applies for all clustering algorithms, not only k-means**
  - **Why are the items in the same cluster – what are their common characteristics? What are the distinct characteristics of the items in each cluster?**
  - **Domain expert is needed**

K-means – Variations

- **Improving the chances of k-means to find the global minimum**
  - **Different ways to initialize the centroids (seeds)**
  - **Using weights based on how close the example is to the cluster center – Gaussian mixture models**
  - **Allowing clusters to split and merge**
    - **Split if the variance within a cluster is large**
    - **Merge if the distance between cluster centers is smaller than a threshold**
- **Make it scale better**
  - **Save distance information from one iteration to the next, thus reducing the number of calculations**
- **K-means can be used for hierarchical clustering**
  - **Start with k=2 and repeat recursively within each cluster**

Example and images from Hastie, Tibshirani, Friedman, The Elements of Statistical Learning

K-means – Image Compression Example

- **Compression by Vector Quantization (VQ) using k-means**
- **Original image 1024 x 1024 pixels**
- **Each pixel is a grayscale value [0, 255]**
- **How many bytes are needed to store the original image?**

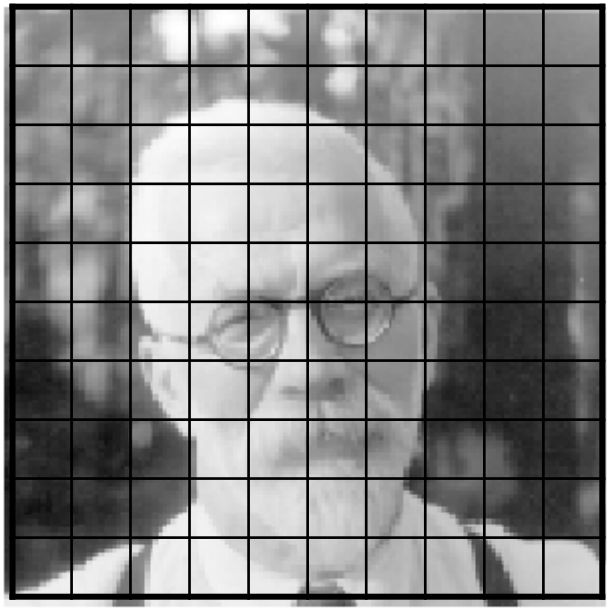
original

Sir Ronald A. Fisher (1890-1962)

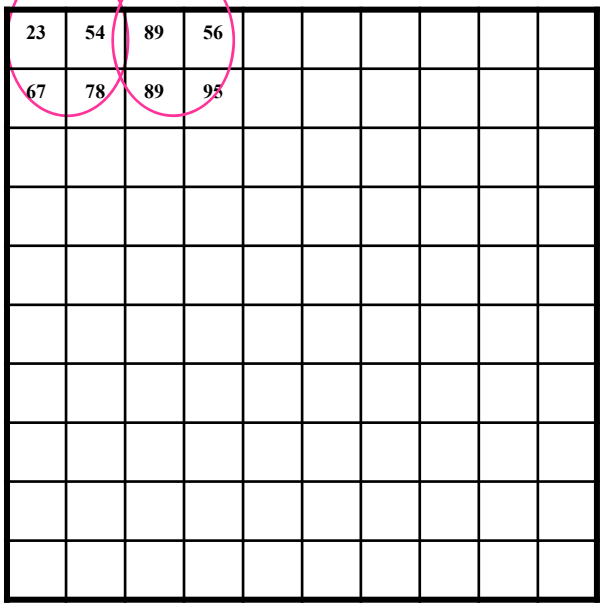
Image Compression Example (2)

- **Break the image into 2x2 blocks of pixels**
- **Each of the 512x512 blocks of 4 numbers is regarded as a vector (4 dim)**
- **Cluster these vectors using k-means (e.g. k=10)**
- **In the compressed image represent each block with the centroid of its cluster (i.e. replace each vector with the centroid)**

original



Sir Ronald A. Fisher (1890-1962)



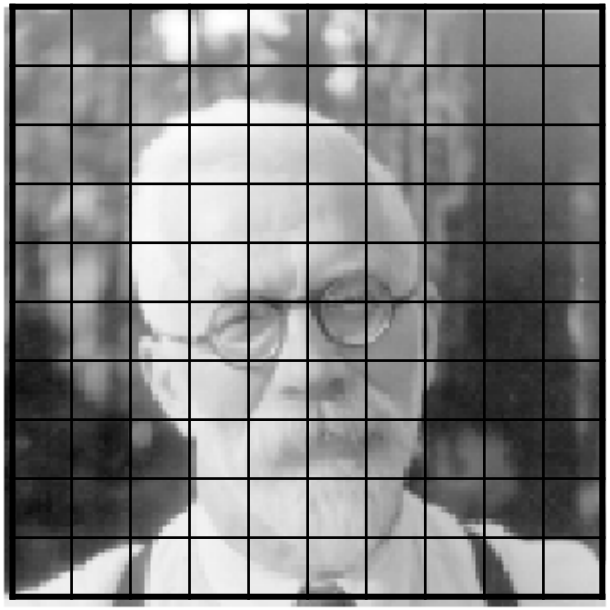
[23, 54, 47, 73]
[39, 56, 89, 95]
…
512 x 512 input vectors for clustering

Clustering into 10 clusters:
centroid1: [95, 56, 78, 56]
centroid2: [256, 250, 196, 175]
centroid3: [115, 168, 130, 178]
…
centroid10: [23, 45, 67, 89]

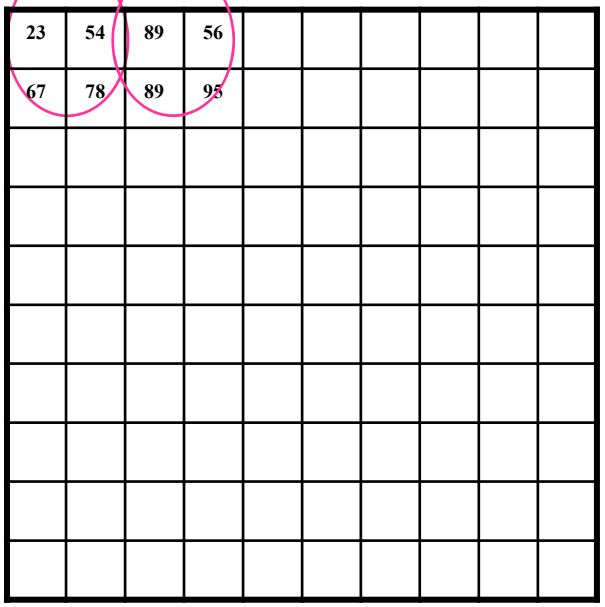
Image Compression Example (3)

- **For k=10**
  - **What is the number of centroids?**
  - **In the new image, what is the maximum number of**
    - **unique grayscale vectors?**
    - **unique grayscale values?**

original



Sir Ronald A. Fisher (1890-1962)



[23, 54, 47, 73]
[39, 56, 89, 95]
…
512 x 512 input vectors for clustering

Clustering into 10 clusters:
centroid1: [95, 56, 78, 56]
centroid2: [256, 250, 196, 175]
centroid3: [115, 168, 130, 178]
…
centroid10: [23, 45, 67, 89]

Image Compression Example (4)

- **The clustering process is called *encoding***
- **The centroids are called *codewords***
- **The collection of all centroids is called *codebook***

original

Sir Ronald A. Fisher (1890-1962)

compressed1 (k=200)

k=200

compressed2 (k=4)

k=4

Storage Requirements for the Compressed Image

For our example (when the image is split into 2x2 blocks):

1) All centroids

- ***k* centroids with dimensionality 4 = k x 4 real numbers = k x 4 bytes**
- **If *k* is small (as in this example), this is a very small storage requirement and we can ignore it**

2) For each block, the index of the closest centroid that will replace its values: $\log_2 k$ bits per block

- **Why $\log_2 k$ bits? (we need to store k integers => $\log_2 k$ bits to encode them)**

- **Compression rate (compressed/original)=**
- **$=(k*4*8+ 512*512*\log_2 k)/(1024*1024*8)$ bits**

  =0.239 for k=200

  =0.063 for k=4

K-Medoids Clustering Algorithm

K-medoids

- **Similar to K-means, reduces the sensitivity to outliers**
- **Uses cluster *medoid* instead of cluster *means***
- **Similarly to K-means, K-medoids minimizes the distance between a point in a cluster and the reference point (*medoid* in K-medoids and *means* in K-means)**

K-medoids – Pseudo code

- **Select K points (items) as the initial medoids**
- **Repeat**
  - **Form K clusters by assigning all items to the closest medoid**
  - **Re-compute the medoids for each cluster (search for better medoids):**
    - **Initial (parent) state = current set of medoids**
    - **Generate the children states by swapping each medoid with each other non-medoid**
    - **Evaluate the children states – are they better than the parent?**

      evaluation function: cost=sum of absolute distances (item, closest medoid).
    - **Choose the best state (set of medoids with the smallest cost)**
- **Until medoids don't change or cost doesn't decrease**

Hill-climbing search for the set of medoids minimizing the distance from an example to a medoid

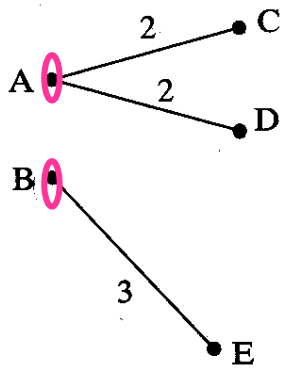
K-medoids – Example

- **Given: 5 items with the distance between them**
- **Task: Cluster them using the K-medoids, k=2**
  - **(Note: randomly place objects in clusters if they have identical distances to the medoids)**

Item	A	B	C	D	E
A	0	1	2	2	3
B	1	0	2	4	3
C	2	2	0	1	5
D	2	4	1	0	3
E	3	3	5	3	0

K-medoids – Solution

1) Randomly select 2 medoids as initial medoids,
   - e.g. A (K1) and B (K2)
2) Assign all items to the closest medoid:
   - K1= {A,C,D} , K2={B,E}



Cost=7

3) Consider swapping medoids A and B with each non-medoid

A->C, B=B (swap A with C, don't change B; now the medoids are C and B)

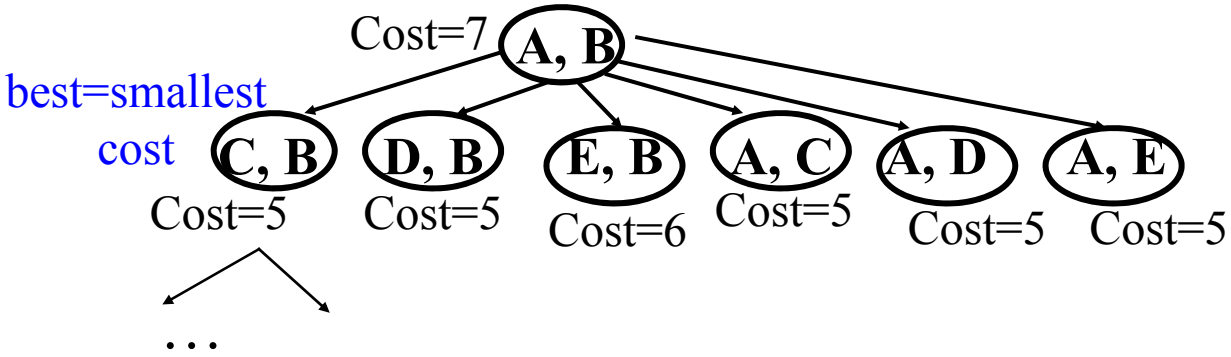
A->D, B=B

A->E, B=B

A=A, B->C

A=A, B->D

A=A, B->E



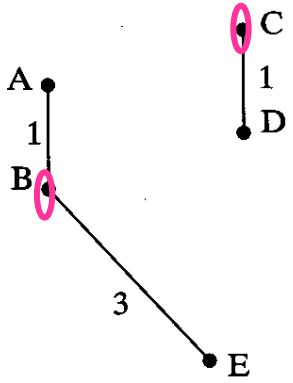
Given *n* items and *k* desired clusters , how many swaps need to be considered?

k(n-k)

Evaluating Children States

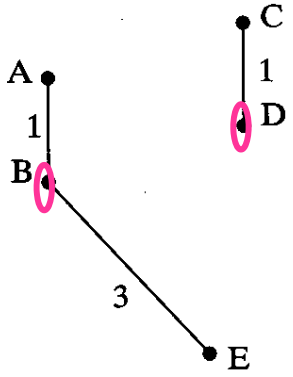
- **Evaluate child 1 (C, B) – the new medoids are C and B, resulting from the swap: A ->C, B=B**
- **Assign each non-medoid to the closest medoid and calculate the total cost:**
  - **From A to the new medoids: dist(A,C)=2 > dist(A,B)=1**
  - **From D to the new medoids: dist(D,C)=1 < dist(D,B)=4**
  - **From E to the new medoids: dist(E,C)=5 > dist(E,B)=3**

  cost=1+1+3=5



- **Evaluate child 2 (D, B) – the new medoids are D and B, resulting from the swap: A->D, B=B**
  - **From A to the new medoids: dist(A,D)=2 > dist(A,B)=1**
  - **From C to the new medoids: dist(C,D)=1 < dist(C,B)=2**
  - **From E to the new medoids: dist(E,D)=3 = dist(E,B)=3 (tie, random choice)**

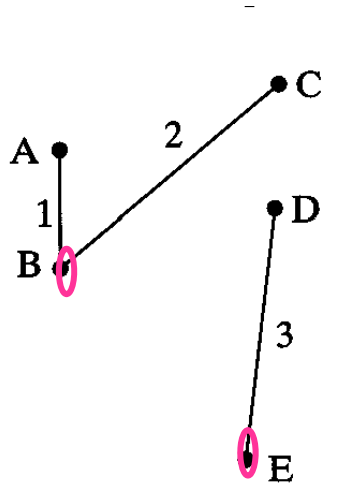
  cost=1+2+3=5



Evaluating Children States (2)

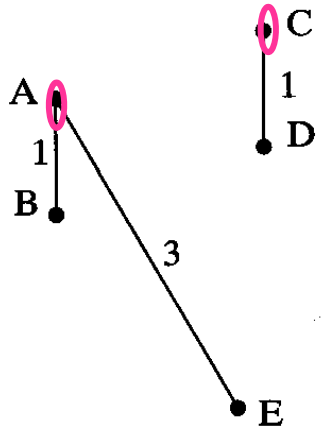
- **Evaluate child 3 (E, B) – the new medoids are E and B, resulting from the swap: A -> E, B=B**
  - **From A to the new medoids: dist(A,E)=3 > dist(A,B)=1**
  - **From C to the new medoids: dist(C,E)=5 > dist(C,B)=2**
  - **From D to the new medoids: dist(D,E)=3 < dist(D,B)=4**

  cost=1+2+3=6



- **Evaluate child 4 (C, A) – the new medoids are C and A, resulting from the swap: B->C, A=A**
  - **From B to the new medoids: dist(B,C)=2 > dist(B,A)=1**
  - **From C to the new medoids: dist(D,C)=1 < dist(D,A)=2**
  - **From E to the new medoids: dist(E,C)=5 > dist(E,A)=3**

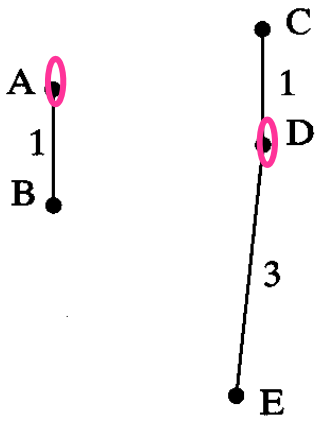
  cost=1+1+3=5



Evaluating Children States (3)

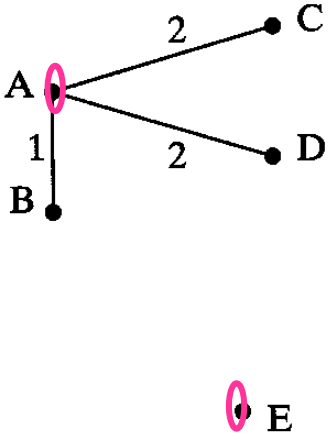
- **Evaluate child 5 (D, A) – the new medoids are D and A, resulting from the swap: B-> D, A=A**
  - **From B to the new medoids: dist(B,D)=4 > dist(B,A)=1**
  - **From C to the new medoids: dist(C,D)=1 < dist(C,A)=2**
  - **From E to the new medoids: dist(E,D)=3 = dist(E,A)=3 (tie, random choice)**

  cost=1+1+3=5



- **Evaluate child 6 (E, A) – the new medoids are E and A, resulting from the swap: B->E, A=A**
  - **From B to the new medoids: dist(B,E)=3 > dist(B,A)=1**
  - **From C to the new medoids: dist(C,E)=2 = dist(C,A)=2 (tie, random choice)**
  - **From D to the new medoids: dist(D,E)=3 > dist(D,A)=2**

  cost=1+2+2=5



Select the Best Child

4) Select the clustering with the best (smallest) cost (in this case cost=5)

- **there is a tie for our example, select the first child (random choice)**

5) 1 epoch has finished; check stopping criterion (no change in the medoids or cost doesn't decrease) – not met

K-medoids: Country Dissimilarities Example

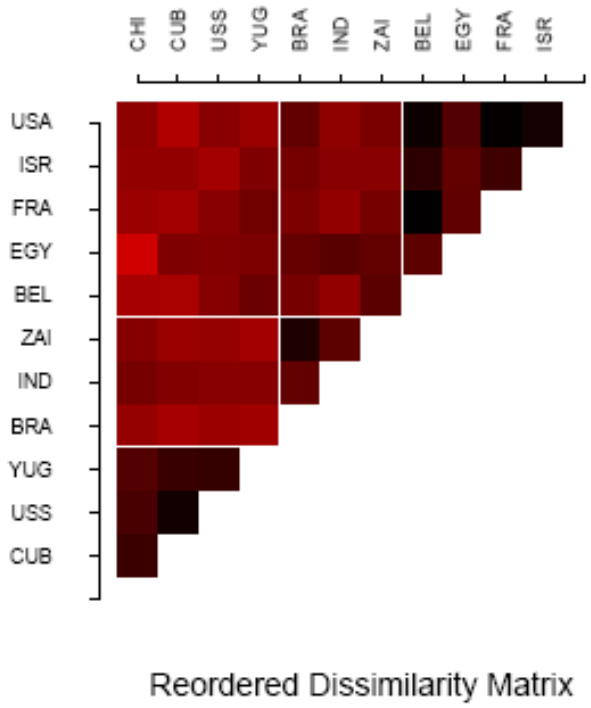
- A study by Kaufman and Rousseeuw (1990)
- Political science students were asked to provide pairwise dissimilarity measures for 12 countries: Belgium, Brazil, Chile, Cuba, Egypt, France, Israel, US, USSR, Yugoslavia and Zaire
- The average scores are:

	BEL	BRA	CHI	CUB	EGY	FRA	IND	ISR	USA	USS	YUG
BRA	5.58										
CHI	7.00	6.50									
CUB	7.08	7.00	3.83								
EGY	4.83	5.08	8.17	5.83							
FRA	2.17	5.75	6.67	6.92	4.92						
IND	6.42	5.00	5.58	6.00	4.67	6.42					
ISR	3.42	5.50	6.42	6.42	5.00	3.92	6.17				
USA	2.50	4.92	6.25	7.33	4.50	2.25	6.33	2.75			
USS	6.08	6.67	4.25	2.67	6.00	6.17	6.17	6.92	6.17		
YUG	5.25	6.83	4.50	3.75	5.75	5.42	6.08	5.83	6.67	3.67	
ZAI	4.75	3.00	6.08	6.67	5.00	5.58	4.83	6.17	5.67	6.50	6.92

K-medoids: Country Dissimilarities Example (2)

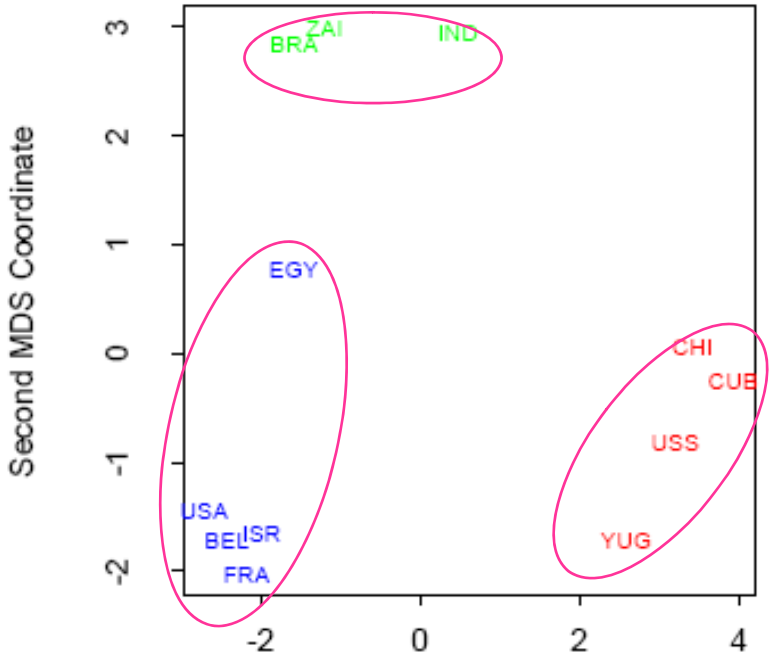
- The 3-medoids algorithm was used to cluster the data
- Note: k-means cannot be applied as we have only distances and not raw data

Color coded and reordered dissimilarity matrix



Reordered Dissimilarity Matrix

• 3 well separated clusters were found, only Egypt is halfway between 2 clusters
• MDS=multidimensional scaling; used to convert the data into 2 dimensions for plotting



First MDS Coordinate

K-Medoids - Summary

- **Less sensitive to outliers than k-means**
- **Computationally expensive and not suitable for large databases**
  - **Time complexity: O(n(n-k))**
  - **Space complexity: $O(n^2)$– needs the proximity matrix**
- **Does not depend on the order of examples**
- **Unlike k-means, k-medoids can be used when only the distances are given and not the raw data**

Nearest Neighbor Clustering Algorithm

Nearest Neighbor Clustering Algorithm

- **Partitional algorithm**
- **Suitable for *dynamic* data (data that changes over time) – e.g. clustering web logs to find patterns of usage**
- **Idea: Items are iteratively merged into clusters**
  - **The first item forms a cluster of itself**
  - **A new item is either merged with an existing cluster or forms a new cluster of itself depending on how close it is to the existing clusters**

    if distance(new_item, existing_cluster)< threshold t -> merge

    else new_item forms a cluster of itself
  - **Typically used distance measure: single link (MIN)**
- **Space and time complexity: $O(n^2)$, n-number of items**
  - **Each item is compared to each item already in the cluster**

Nearest Neighbor Clustering -Pseudocode

```
Input:
   D = {t1, t2, ..., tn}   //Set of elements
   A      //Adjacency matrix showing distance between elements
Output:
   K      //Set of clusters
Nearest neighbor algorithm:
   K1 = {t1}; // t1 is placed in a cluster by itself
   K = {K1};
   k = 1;
   for i = 2 to n do      // t2-tn items: – add to an existing cluster or create a new cluster?
      find the tm in some cluster Km in K such that dis(ti, tm) is
         the smallest;
      if dis(ti, tm), ≤ t then
         Km = Km ∪ ti
      else
         k = k + 1;
         Kk = {ti};
```

From Dunham, Data Mining

Nearest Neighbor Clustering - Example

- **Given: 5 items with the distance between them**
- **Task: Cluster them using the Nearest Neighbor algorithm with a threshold t=2**

Item	A	B	C	D	E
A	0	1	2	2	3
B	1	0	2	4	3
C	2	2	0	1	5
D	2	4	1	0	3
E	3	3	5	3	0

Solution

- t=2

Item	A	B	C	D	E
A	0	1	2	2	3
B	1	0	2	4	3
C	2	2	0	1	5
D	2	4	1	0	3
E	3	3	5	3	0

-A: K1={A}
-B: d(B,A)=1 ≤ t => K1={A,B}
-C: d(C,A}=d(C,B)=2≤t=>K1={A,B,C}
-D: d(D,A)=2, d(D,B)=4, d(D,C)=1 =dmin ≤t => K1={A,B,C,D}
-E: d(E,A)=3, d(E,B)=3, d(E, C)=5, d(E, D)=3=dmin>t=>K2={E}

1. Is this algorithm sensitive to the order of examples?
2. How many passes through the data does it require?

Hierarchical Clustering Algorithm

Hierarchical Clustering Algorithms

- **Create a hierarchical tree representation, called a dendrogram**
  - **At the leaf level: each example is in a cluster of itself**
  - **At the root level: all examples are in the same cluster**
  - **The clusters at each level are produced by merging clusters from the previous level**

- **Produces not a single set of clusters but several sets – each hierarchy level is associated with a set of clusters**
  - **in red: 5 clusters**
- **Does not require the number of clusters *k* to be specified**

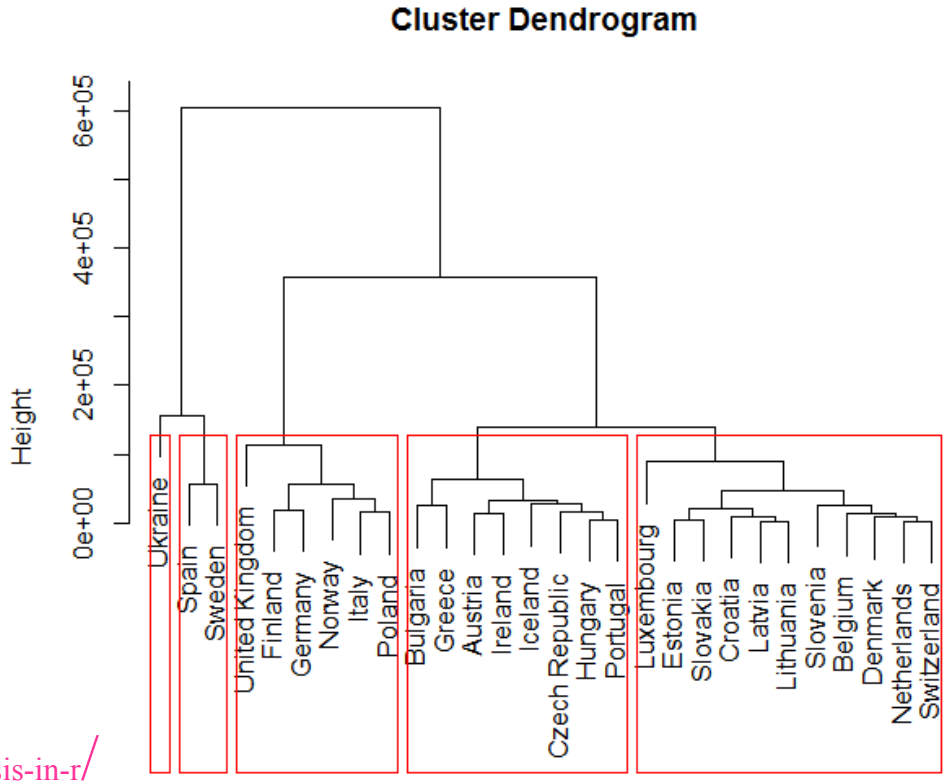
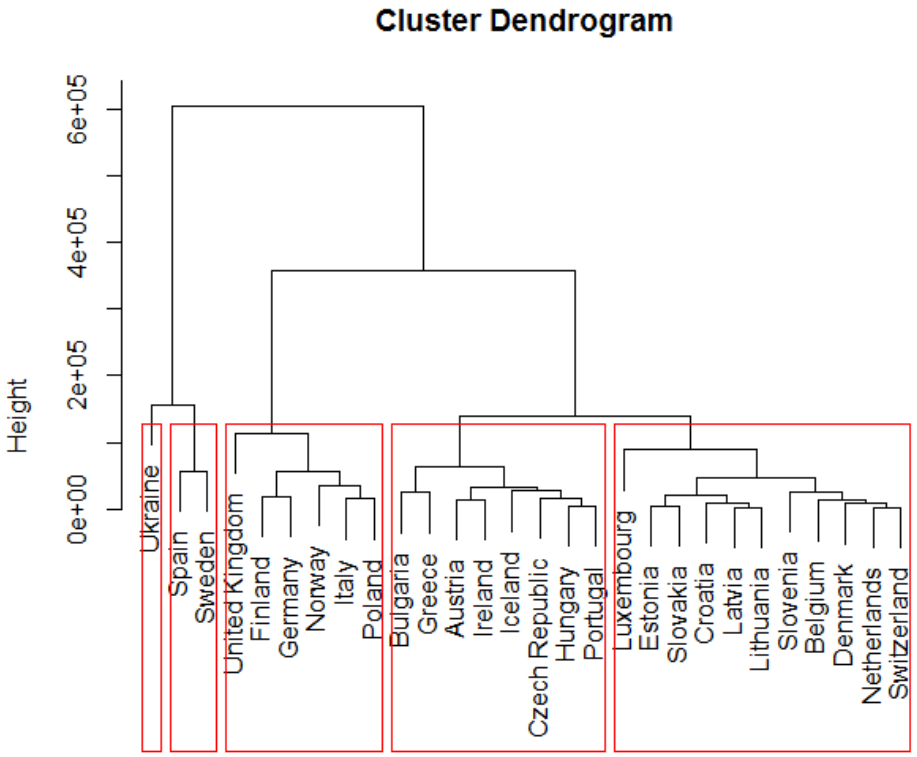


Image: http://www.instantr.com/2013/02/12/performing-a-cluster-analysis-in-r/

Hierarchical Clustering Algorithms (2)

- **The dendrogram provides a highly interpretable description of the clustering => this is one of the main reasons for the popularity of hierarchical clustering**



- **2 versions: agglomerative and divisive**

Agglomerative Clustering

- **Agglomerative (bottom-up) – *merges* clusters iteratively**
  - **Start with each item in its own cluster; iteratively merge clusters until all items belong to one cluster**
  - **Merging is based on how close the clusters are to each other**
    - **Distance threshold *d*: if the distance between two clusters is smaller or equal to *d*, merge them**

      => more than 2 clusters may be merged
    - **Initially *d* is set to a small value that is incremented at each level**

Agglomerative Clustering - Algorithm

1) d=0; // initial distance threshold for merging

2) Compute the distance matrix

3) Let each data point be a cluster

4) Repeat

- **Increment the distance threshold with 1 // or another value**
- **Merge all clusters with distance <=d**
- **Update the distance matrix // this is the key operation**

Until only a single cluster remains

Hierarchical Agglomerative Single Link - Example

- **Example:**
  - **Given: 5 items with the distance between them**
  - **Task: Cluster them using agglomerative single link clustering**

	A	B	C	D	E
A	0	1	2	2	3
B		0	2	4	3
C			0	1	5
D				0	3
E					0

1) d=0; // initial distance threshold for merging

2) Compute the distance matrix

3) Let each data point be a cluster

4) Repeat

Increment the distance threshold with 1

Merge all clusters with distance <=d

Update the distance matrix

Until only a single cluster remains

Solution

- **Distance Level 0: each item is in a cluster of itself**

Level 0

	A	B	C	D	E
A	0	1	2	2	3
B		0	2	4	3
C			0	1	5
D				0	3
E					0

- **Distance Level 1: merge A&B, C&D; update the distance matrix:**

Level 1

	AB	CD	E
AB	0	2	3
CD		0	3
E			0

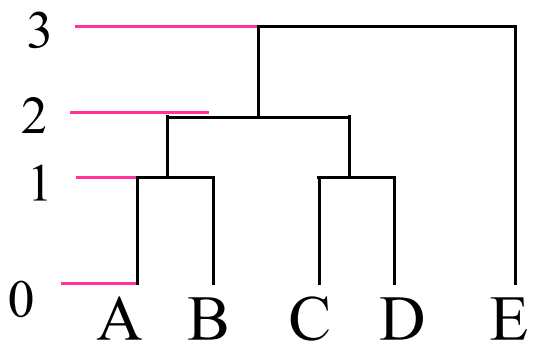
- **Distance Level 2: merge AB&CD; update the distance matrix:**

Level 2

	ABCD	E
ABCD	0	3
E		0

- **Distance Level 3: merge ABCD&E; all items are in one cluster; stop**

- **Dendrogram:**

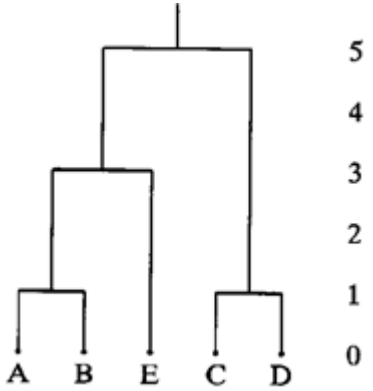


Single Link vs. Complete Link Algorithm

- **Single link (MIN) suffers from the so called *chain effect***
  - **2 clusters are merged if only 2 of their elements are close to each other**
  - **there may be elements in the 2 clusters that are far from each other but this has no effect on the algorithm**
  - **Thus, the clusters may contain points that are not related to each other but simply happen to be near points that are close to each other**

  - **More sensitive to noise and outliers**

Dendrogram for our example



- **Complete link (MAX)– the distance between 2 clusters is the largest distance between an element in one cluster and an element in another**
  - **Generates more compact clusters**
  - **Less sensitive to noise and outliers**

Divisive Clustering

- **Divisive (top-down) – splits a cluster iteratively**
  - **Place all items in one cluster; iteratively split clusters in two until all items are in their own cluster**
    - **Splitting is based on the distance between clusters – split if the distance is smaller or equal to the threshold *d***
    - **Initially *d* is set to a big value that is decremented at each level**
- **Less popular than agglomerative**

Applicability and Complexity

- **Hierarchical clustering algorithms are suitable for tasks with natural nesting relationships between clusters (taxonomies, hierarchies)**
  - **biology tasks - plant and animal taxonomies**
- **Computationally expensive which limits its applicability to high dimensional data**
  - **Space complexity: $O(n^2)$, n - number of items**
    - **The space required to store the proximity distance matrix and the dendrogram**
  - **Time complexity of the algorithm: $O(n^3)$ – n levels; at each of them n^2 proximity matrices must be searched and updated**
  - **can be reduced to $O(n^2 \log n)$ if the distances are stored in a sorted list**
- **Not incremental – assumes all data is present**

Hierarchical Clustering as a Graph Search Problem

Agglomerative Algorithm – Another Pseudo Code

- **This version: starts with distance threshold d=0 and increments it with 1; at each level merges all clusters with distance <=d**

```
Input:
   D = {t1, t2, ..., tn}   //Set of elements
   A      //Adjacency matrix showing distance between elements
Output:
   DE     // Dendrogram represented as a set of ordered triples
Agglomerative algorithm:
   d = 0;
   k = n;
   K = {{t1}, ..., {tn}};
   DE = {(d, k, K)}; // Initially dendrogram contains each element
                     in its own cluster.
   repeat
      oldk = k;
      d = d + 1;
      Ad = Vertex adjacency matrix for graph with threshold
           distance of d;
      (k, K) = newClusters(Ad, D);
      if oldk ≠ k then
         DE = DE ∪ (d, k, K); // New set of clusters added to dendrogram.
         until k = 1
```

A[i,j]=distance(t_i,t_j)

Threshold distance

Num. of clusters

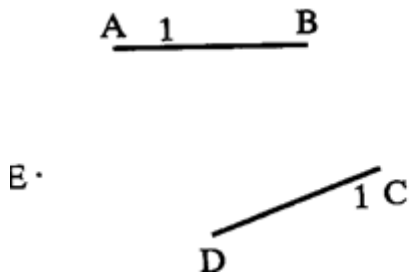
Set of clusters

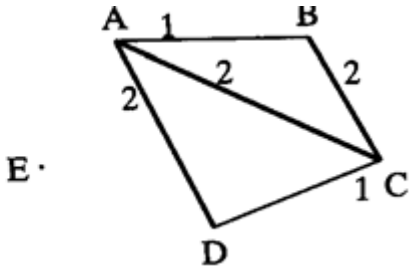
May be different than 1

Finds all clusters within d, merges them & updates the distance matrix A

Single Link Agglomerative as a Graph Problem

- **`NewClusters` (see the previous slide) can be replaced with a procedure for finding connected components in a graph**
  - **two components of a graph are connected if there exists a path between any 2 vertices**
  - **Examples:**





A and B are connected

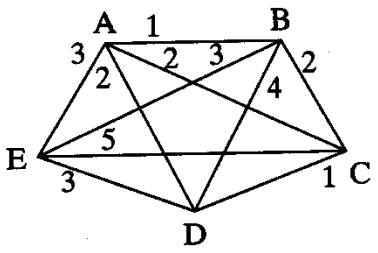
A, B, C and D are connected

C and D are connected

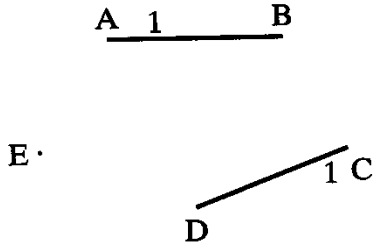
- **Show the graph edges with a distance of *d* or below**
- **Merge 2 clusters if there is at least 1 edge that connects them (i.e. if the minimum distance between any 2 points is <= *d*)**
- **Increment d**

Example – Solution 2

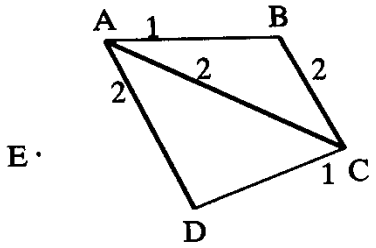
- **Procedure `NewClusters`**
  - **Input: graph defined by a set of vertices + vertex adjacency matrix**
  - **Output: a set of connected components defined by a number of these components (i.e. number of clusters *k*) and an array with the membership of these components (i.e. *K* - the set of clusters)**



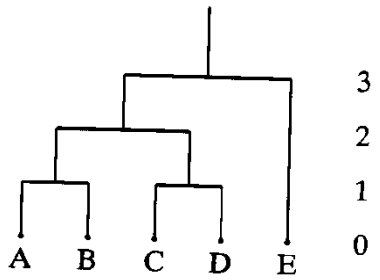
(a) Graph with all distances



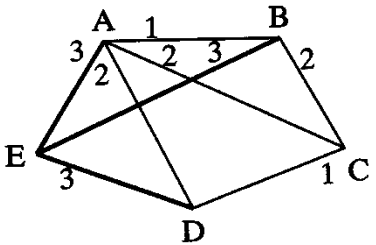
(b) Graph with threshold of 1



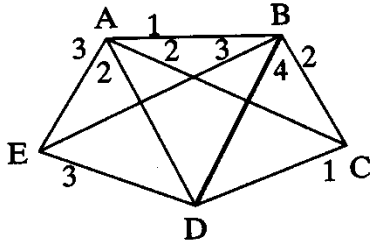
(c) Graph with threshold of 2



Single link dendrogram



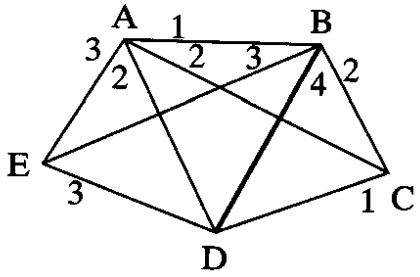
(d) Graph with threshold of 3



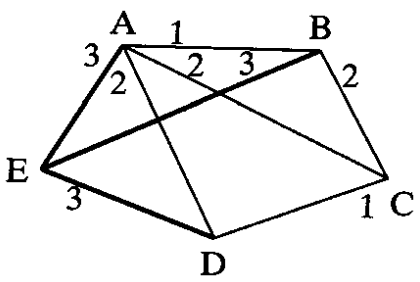
(e) Graph with threshold of 4

Divisive Clustering as a Graph Problem

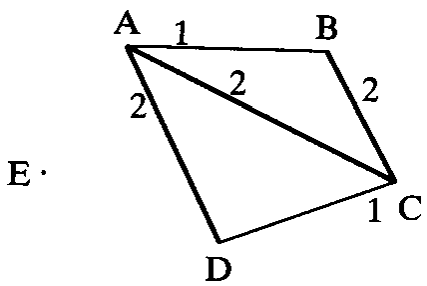
- **All items are initially placed in one cluster**
- **Clusters are iteratively split in two until all items are in their own cluster**
- **In reverse order: from *e* to *b***



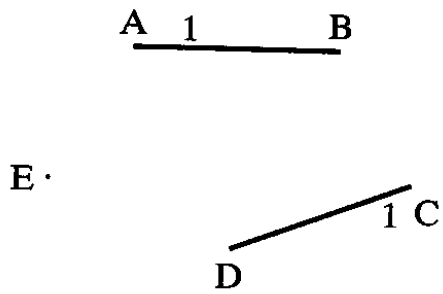
(e) Graph with threshold of 4



(d) Graph with threshold of 3



(c) Graph with threshold of 2



(b) Graph with threshold of 1

(4, 1, {A,B,C,D,E}}) **(3, 1, {A,B,C,D,E}})** **(2, 2, {{A,B,C,D}, {E}})** **(1, 3, {{A,B}, {C,D}, {E}})**

(0, 5, {{A},{B},{C}, {D}, {E}})

References

- **Dunham (2003). *Data Mining – Introductory and Advanced Topics*, Pearson Education**
- **Tan, Steinbach and Kumar (2006). *Introduction to Data Mining*, Addison Wesley**
- **Hastie, Tibshirani and Friedman (2001). *The Elements of Statistical Learning*, Springer-Verlag.**