OTHER CLUSTERING ALGORITHMS

- The following types of algorithms will be considered:
  - Graph theory based clustering algorithms.
  - Competitive learning algorithms.
  - Valley seeking clustering algorithms.
  - Cost optimization clustering algorithms based on:
    - Branch and bound approach.
    - Simulated annealing methodology.
    - Deterministic annealing.
    - Genetic algorithms.
  - Density-based clustering algorithms.
  - Clustering algorithms for high dimensional data sets.

GRAPH THEORY BASED CLUSTERING ALGORITHMS

- In principle, such algorithms are capable of detecting clusters of various shapes, at least when they are well separated.

In the sequel we discuss algorithms that are based on:

- The Minimum Spanning Tree (MST).
- Regions of influence.
- Directed trees.

Minimum Spanning Tree (MST) algorithms

Preliminaries:

Let

- G be the complete graph, each node of which corresponds to a point of the data set X.
- $e=(\underline{x}_i,\underline{x}_j)$ denote an edge of G connecting $\underline{x}_i$ and $\underline{x}_j$.
- $w_e \equiv d(\underline{x}_i, \underline{x}_j)$ denote the weight of the edge e.

Definitions:

- Two edges e_1 and e_2 are k steps away from each other if the minimum path that connects a vertex of e_1 and a vertex of e_2 contains k-1 edges.
- A Spanning Tree of G is a connected graph that:
  - Contains all the vertices of the graph.
  - Has no loops.
- The weight of a Spanning Tree is the sum of weights of its edges.
- A Minimum Spanning Tree (MST) of G is a spanning tree with minimum weight (when all w_e's are different from each other, the MST is unique).

❖ Minimum Spanning Tree (MST) algorithms (cont)

Sketch of the algorithm:

- Determine the MST of G.
- Remove the edges that are "unusually" large compared with their neighboring edges (inconsistent edges).
- Identify as clusters the connected components of the MST, after the removal of the inconsistent edges.

Identification of inconsistent edges.

For a given edge e of the MST of G:

- Consider all the edges that lie k steps away from it.
- Determine the mean m_e and the standard deviation σ_e of their weights.
- If w_e lies more than q (typically $q=2$) standard deviations σ_e from m_e, then:
  - e is characterized as inconsistent.
- Else
  - e is characterized as consistent.
- End if

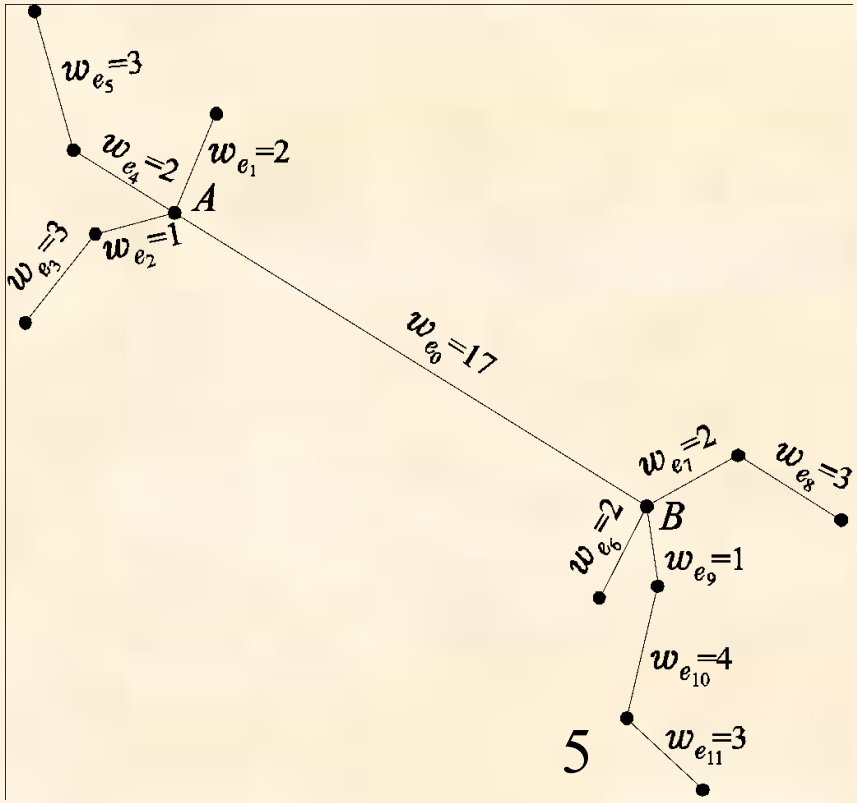
❖ Minimum Spanning Tree (MST) algorithms (cont)

Example 1:

For the MST in the figure and for k=2 and q=3 we have:

For e_0: w_{e0}=17, m_{e0}=2.3, σ_{e0}=0.95. w_{e0} lies $15.5*\sigma_{e0}$ away from m_{e0}, hence it is inconsistent.

For e_{11}: w_{e11}=3, m_{e11}=2.5, σ_{e11}=2.12. w_{e11} lies $0.24*\sigma_{e11}$ away from m_{e11}, hence it is consistent.



❖ Minimum Spanning Tree (MST) algorithms (cont)

- Remarks:
  - The algorithm depends on the choices of k and q.
  - The algorithm is insensitive to the order in which the data points are considered.
  - No initial conditions are required, no convergence aspects are involved.
  - The algorithm works well for many cases where the clusters are well separated.
  - A problem may occur when a "large" edge e has another "large" edge as its neighbor. In this case, e is likely not to be characterized as inconsistent and the algorithm may fail to unravel the underlying clustering structure correctly.

❖ Algorithms based on Regions of Influence (ROI)

Definition: The region of influence of two distinct vectors $\underline{x}_i, \underline{x}_j \in X$ is defined as:

$$R(\underline{x}_i, \underline{x}_j)=\{\underline{x}: \text{cond}(d(\underline{x}, \underline{x}_i), d(\underline{x}, \underline{x}_j), d(\underline{x}_i, \underline{x}_j)), \underline{x}_i \neq \underline{x}_j\}$$

where $\text{cond}(d(\underline{x}, \underline{x}_i), d(\underline{x}, \underline{x}_j), d(\underline{x}_i, \underline{x}_j))$ may be defined as:

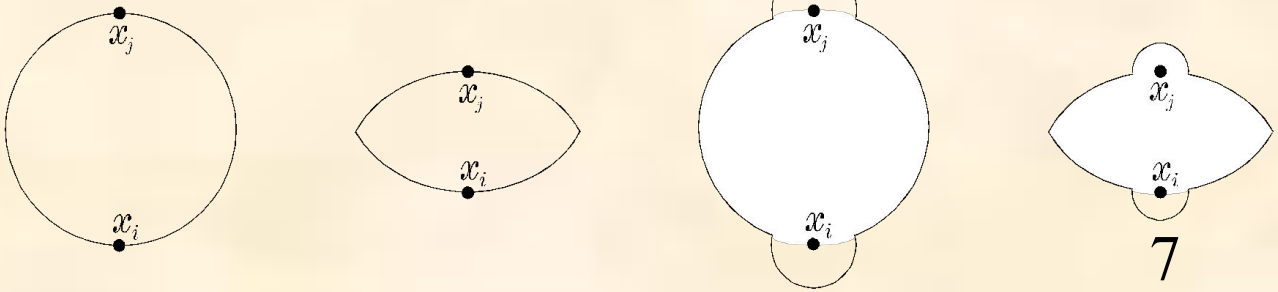
a) $\max\{d(\underline{x}, \underline{x}_i), d(\underline{x}, \underline{x}_j)\} < d(\underline{x}_i, \underline{x}_j)$,

b) $d^2(\underline{x}, \underline{x}_i) + d^2(\underline{x}, \underline{x}_j) < d^2(\underline{x}_i, \underline{x}_j)$,

c) $(d^2(\underline{x}, \underline{x}_i) + d^2(\underline{x}, \underline{x}_j) < d^2(\underline{x}_i, \underline{x}_j)\ )\ OR\ (\ \sigma \min\{d(\underline{x}, \underline{x}_i), d(\underline{x}, \underline{x}_j)\} < d(\underline{x}_i, \underline{x}_j)\ )$,

d) $(\max\{d(\underline{x}, \underline{x}_i), d(\underline{x}, \underline{x}_j)\} < d(\underline{x}_i, \underline{x}_j)\ )\ OR\ (\ \sigma \min\{d(\underline{x}, \underline{x}_i), d(\underline{x}, \underline{x}_j)\} < d(\underline{x}_i, \underline{x}_j))$,

where σ affects the size of the ROI defined by $\underline{x}_i, \underline{x}_j$ and is called relative edge consistency.



❖ Algorithms based on Regions of Influence (cont)

Algorithm based on ROI

- For $i=1$ to N
  - For $j=i+1$ to N
    - Determine the region of influence $R(\underline{x}_i,\underline{x}_j)$
    - If $R(\underline{x}_i,\underline{x}_j) \cap (X-\{\underline{x}_i,\underline{x}_j\}) = \varnothing$ then
      - Add the edge connecting $\underline{x}_i,\underline{x}_j$.
    - End if
  - End For
- End For
- Determine the connected components of the resulted graph and identify them as clusters.

In words:

- The edge between $\underline{x}_i$ and $\underline{x}_j$ is added to the graph if no other vector in X lies in $R(\underline{x}_i,\underline{x}_j)$.
- Since for $\underline{x}_i$ and $\underline{x}_j$ close to each other it is likely that $R(\underline{x}_i,\underline{x}_j)$ contains no other vectors in X, it is expected that close to each other points will be assigned to the same cluster.

❖ Algorithms based on Regions of Influence (cont)

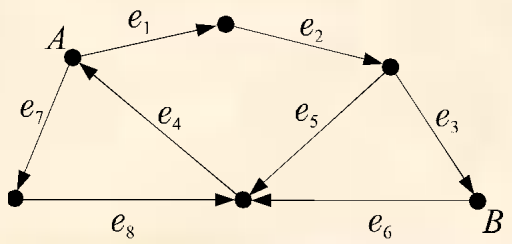
- Remarks:
  - The algorithm is insensitive to the order in which the pairs are considered.
  - In the choices of $cond$ in (c) and (d), σ must be chosen *a priori*.
  - For the resulting graphs:
    - if the choice (a) is used for $cond$, they are called relative neighborhood graphs (RNGs)
    - if the choice (b) is used for $cond$, they are called Gabriel graphs (GGs)
  - Several results show that better clusterings are produced when (c) and (d) conditions are used in the place of $cond$, instead of (a) and (b).

❖ Algorithms based on Directed Trees

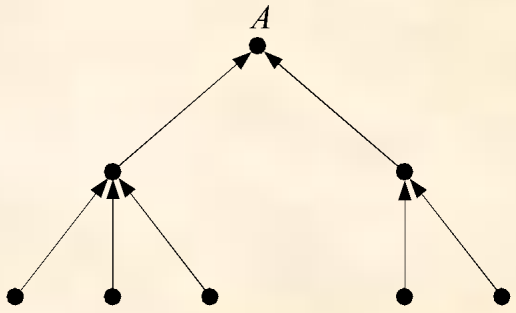
Definitions:

- A directed graph is a graph whose edges are directed.

- A set of edges $e_{i1},\ldots,e_{iq}$ constitute a directed path from a vertex A to a vertex B, if,
  - A is the initial vertex of e_{i1}
  - B is the final vertex of e_{iq}
  - The destination vertex of the edge e_{ij}, $j = 1,\ldots,q\text{-}1$, is the departure vertex of the edge e_{ij+1}.
  - (In figure (a) the sequence e_1, e_2, e_3 constitute a directed path connecting the vertices A and B).



(a)



(b)

❖ Algorithms based on Directed Trees (cont)

- A directed tree is a directed graph with a specific node A, known as root, such that,
  - Every node B≠A of the tree is the initial node of exactly one edge.
  - No edge departs from A.
  - No circles are encountered (see figure (b)).

- The neighborhood of a point $\underline{x}_i \in X$ is defined as

$$\rho_i(\theta)=\{\underline{x}_j \in X: d(\underline{x}_i,\underline{x}_j) \leq \theta,\ \underline{x}_j \neq \underline{x}_i\}$$

  where θ determines the size of the neighborhood.

- Also let
  - $n_i=|\rho_i(\theta)|$ be the number of points of X lying within $\rho_i(\theta)$
  - $g_{ij}=(n_j-n_i)/d(\underline{x}_i,\underline{x}_j)$

Main philosophy of the algorithm

Identify the directed trees in a graph whose vertices are points of X, so that each directed tree corresponds to a cluster.

❖ Algorithms based on Directed Trees (cont

Clustering Algorithm based on Directed Trees

- Set θ to a specific value.
- Determine n_i, $i=1,\ldots,N$.
- Compute g_{ij}, $i,j=1,\ldots,N$, $i\neq j$.
- For $i=1$ to N
  - If $n_i=0$ then
    - $\underline{x}_i$ is the root of a new directed tree.
  - Else
    - Determine $\underline{x}_r$ such that $g_{ir}=\max_{\underline{x}_j\in\rho i(\theta)} g_{ij}$
    - If $g_{ir}<0$ then
      - $\underline{x}_i$ is the root of a new directed tree.
    - Else if $g_{ir}>0$ then
      - $\underline{x}_r$ is the parent of $\underline{x}_i$ (there exists a directed edge from $\underline{x}_i$ to $\underline{x}_r$).

❖ Algorithms based on Directed Trees (cont)

Clustering Algorithm based on Directed Trees (cont)

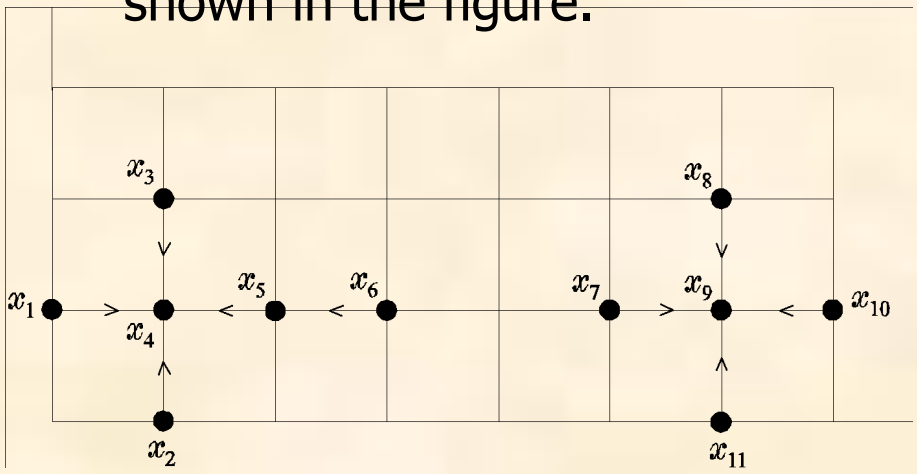
- Else if $g_{ir}=0$ then
  - Define $T_i=\{\underline{x}_j: \underline{x}_j \in \rho_i(\theta), g_{ij}=0\}$.
  - Eliminate all the elements $\underline{x}_j \in T_i$, for which there exists a directed path from $\underline{x}_j$ to $\underline{x}_i$.
  - If the resulting T_i is empty then
    - * $\underline{x}_i$ is the root of a new directed tree
  - Else
    - * The parent of $\underline{x}_i$ is $\underline{x}_q$ such that $d(\underline{x}_i, \underline{x}_q)=\min_{\underline{x}_s \in T_i} d(\underline{x}_i, \underline{x}_s)$.
  - End if
- End if

- End if

- End for
- Identify as clusters the directed trees formed above.

❖ Algorithms based on Directed Trees (cont)

- Remarks:
  - The root $\underline{x}_i$ of a directed tree is the point in $\rho_i(\theta)$ with the most dense neighborhood.
  - The branch that handles the case $g_{ir} = 0$ ensures that no circles occur.
  - The algorithm is sensitive to the order of consideration of the data points.
  - For proper choice of θ and large N, this scheme behaves as a mode-seeking algorithm (see a later section).
- Example 2: In the figure below, the size of the edge of the grid is 1 and θ=1.1. The above algorithm gives the directed trees shown in the figure.



COMPETITIVE LEARNING ALGORITHMS

- The main idea
  - Employ a set of representatives $\underline{w}_j$ (in the sequel we consider only point representatives).
  - Move them to regions of the vector space that are "dense" in vectors of X.
- Comments
  - In general, representatives are updated each time a new vector $\underline{x} \in X$ is presented to the algorithm (pattern mode algorithms).
  - These algorithms do not necessarily stem from the optimization of a cost function.
- The strategy
  - For a given vector $\underline{x}$
    - All representatives compete to each other
    - The winner (representative that lies closest to $\underline{x}$) moves towards $\underline{x}$.
    - The losers (the rest of the representatives) either remain unchanged or move towards $\underline{x}$ but at a much slower rate.

Generalized Competitive Learning Scheme (GCLS)

- $t=0$
- $m=m_{init}$ (initial number of iterations)
- (A) Initialize any other necessary parameters (depending on the specific scheme).
- Repeat
  - $t=t+1$
  - Present a new randomly selected $\underline{x} \in X$ to the algorithm.
  - (B) Determine the winning representative $\underline{w}_j$.
  - (C) If $((\underline{x}$ *is not "similar" to* $\underline{w}_j)$ *OR (other condition)) AND* $(m<m_{max})$ (maximum allowable number of clusters) then
    - $m=m+1$
    - $\underline{w}_m=\underline{x}$
  - Else
    - (D) *Parameter updating*

$$\underline{w}_j(t)=\begin{cases}\underline{w}_j(t-1)+\eta h(\underline{x},\underline{w}_j(t-1)), & \text{if } \underline{w}_j \text{ is thewinne}\\ \underline{w}_j(t-1)+\eta' h(\underline{x},\underline{w}_j(t-1)), & \text{otherwise}\end{cases}$$

  - End
- (E) Until (convergence occurred) OR $(t>t_{max})$ (max allowable no of iterations)
- Assign each $\underline{x} \in X$ to the cluster whose representative $\underline{w}_j$ lies closest to $\underline{x}$.

- Remarks:
  - $h(\underline{x}, \underline{w}_i)$ is an appropriately defined function (see below).
  - η and η' are the learning rates controlling the updating of the winner and the losers, respectively (η' may differ from looser to looser).
  - A threshold of similarity Θ (carefully chosen) controls the similarity between $\underline{x}$ and a representative $\underline{w}_j$.
    - If $d(\underline{x}, \underline{w}_j) > \Theta$, for some distance measure, $\underline{x}$ and $\underline{w}_j$ are considered as dissimilar.
  - The termination criterion is $\|\underline{W}(t)-\underline{W}(t-1)\| < \varepsilon$, where $\underline{W}=[\underline{w}_1^T,\ldots,\underline{w}_m^T]^T$.
  - With appropriate choices of (A), (B), (C) and (D), most competitive learning algorithms may be viewed as special cases of GCLS.

❖ Basic Competitive Learning Algorithm

Here the number of representatives m is constant.

The algorithm

- $t=0$
- Repeat
  - $t=t+1$
  - Present a new randomly selected $\underline{x} \in X$ to the algorithm.
  - (B) Determine the winning representative $\underline{w}_j$ on $\underline{x}$ as the one for which

$$d(\underline{x},\underline{w}_j)=\min_{k=1,\ldots,m} d(\underline{x},\underline{w}_k) \ (*).$$

  - (D) *Parameter updating*

$$\underline{w}_j(t)=\begin{cases}\underline{w}_j(t-1)+\eta(\underline{x}-\underline{w}_j(t-1)), & \text{if } \underline{w}_j \text{ is the winner} \quad (**)\\ \underline{w}_j(t-1), & \text{otherwise}\end{cases}$$

  - End
- (E) Until (convergence occurred) OR ($t>t_{max}$) (max allowable no of iterations)
- Assign each $\underline{x} \in X$ to the cluster whose representative $\underline{w}_j$ lies closest to $\underline{x}$.

(*) $d(.)$ may be the Euclidean distance or other distances (e.g., Itakura-Saito distortion). In addition similarity measures may be used (in this case $\min$ is replaced by $\max$).

(**) η is the learning rate and takes values in $[0, 1]$.

❖ Basic Competitive Learning Algorithm (cont)

- Remarks:
  - In this scheme losers remain unchanged. The winner, after the updating, lies in the line segment formed by $\underline{w}_j(t\text{-}1)$ and $\underline{x}$.
  - *A priori* knowledge of the number of clusters is required.
  - If a representative is initialized far away from the regions where the points of X lie, it will never win.

    Possible solution: Initialize all representatives using vectors of X.
  - Versions of the algorithm with variable learning rate have also been studied.

❖ Leaky Learning Algorithm

The same with the Basic Competitive Learning Algorithm except part (D), the updating equation of the representatives, which becomes

$$\underline{w}_j(t) = \begin{cases} \underline{w}_j(t-1) + \eta_w(\underline{x} - \underline{w}_j(t-1)), & \text{if } \underline{w}_j \text{ is thewinne} \\ \underline{w}_j(t-1) + \eta_l(\underline{x} - \underline{w}_j(t-1)), & \text{if } \underline{w}_j \text{ is a loser} \end{cases}$$

where η_w and η_l are the learning rates in $[0, 1]$ and $\eta_w >> \eta_l$.

Remarks:

- All representatives move towards $\underline{x}$ but the losers move at a much slower rate than the winner does.
- The algorithm does not suffer from the problem of poor initialization of the representatives (why?).
- An algorithm in the same spirit is the "neural-gas" algorithm, where η_l varies from loser to loser and decays as the corresponding representatives lie away from $\underline{x}$. This algorithm results from the optimization of a cost function.

❖ Conscientious Competitive Learning Algorithms

Main Idea: Discourage a representative from winning if it has won many times in the past. Do this by assigning a "conscience" to each representative.

A simple implementation

- Equip each representative $\underline{w}_j$, $j=1,\ldots,m$, with a counter f_j that counts the times that $\underline{w}_j$ wins.
- At part (A) (initialization stage) of GCLS set $f_j=1$, $j=1,\ldots,m$.
- Define the distance $d^*(\underline{x},\underline{w}_j)$ as

$$d^*(\underline{x},\underline{w}_j)= d(\underline{x},\underline{w}_j)\, f_j.$$

  (the distance is penalized to discourage representatives that have won many times)

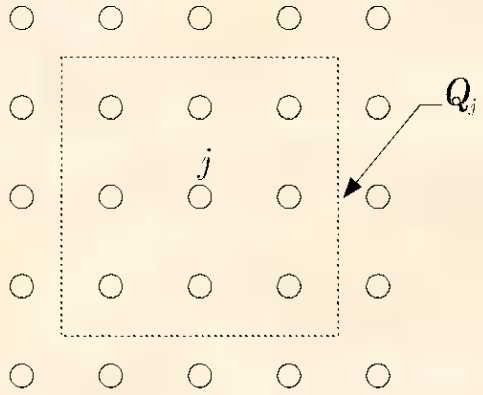
- Part (B) becomes
  - The representative $\underline{w}_j$ is the winner on $\underline{x}$ if

$$d^*(\underline{x},\underline{w}_j)=\min_{k=1,\ldots,m} d^*(\underline{x},\underline{w}_k)$$

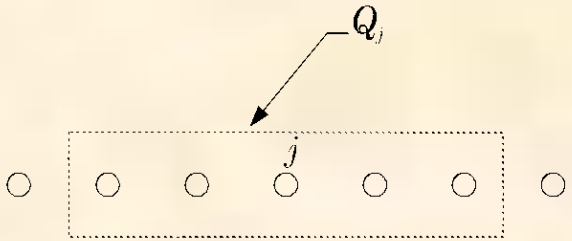
  - $f_j(t)=f_j(t\text{-}1)+1$
- Parts (C) and (D) are the same as in the Basic Competitive Learning Algorithm
- Also $m=m_{init}=m_{max}$

Self-Organizing Maps

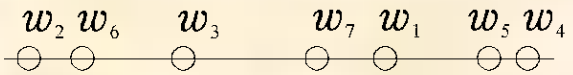
- Here interrelation between representatives is assumed.
- For each representative $\underline{w}_j$ a neighborhood of representatives $Q_j(t)$ is defined, centered at $\underline{w}_j$.
- As t (number of iterations) increases, $Q_j(t)$ shrinks and concentrates around $\underline{w}_j$.
- The neighborhood is defined with respect to the indices j and it is independent of the distances between representatives in the vector space.



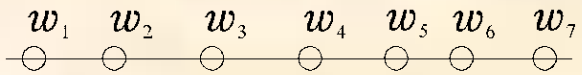
(a)



(b)



(c)



(d)

Assuming that in fig. (c) the neighborhood of $\underline{w}_j$ constitutes of $\underline{w}_{j-1}$ and $\underline{w}_{j+1}$, $\underline{w}_2$ and $\underline{w}_4$ are the neighbors of $\underline{w}_3$ although $\underline{w}_6$ and $\underline{w}_7$ are closer in terms of the geometrical distance to $\underline{w}_3$)

❖ Self Organizing Maps (cont.)

- If $\underline{w}_j$ wins on the current input $\underline{x}$ all the representatives in $Q_j(t)$ are updated (Self Organizing Map (SOM) scheme).
- SOM (in its simplest version) may be viewed as a special case of GCLS if

  - Parts (A), (B) and (C) are defined as in the basic competitive learning scheme.
  - In part (D), if $\underline{w}_j$ wins on $\underline{x}$, the updating equation becomes:

$$\underline{w}_k(t) = \begin{cases} \underline{w}_k(t-1) + \eta(t)(\underline{x} - \underline{w}_k(t-1)), & \text{if } \underline{w}_k \in Q_j(t) \\ \underline{w}_k(t-1), & \text{otherwise} \end{cases}$$

  where $\eta(t)$ is a variable learning rate satisfying certain conditions.

- After convergence, neighboring representatives also lie "close" in terms of their distance in the vector space (topographical ordering) (see fig. (d)).

❖ Supervised Learning Vector Quantization (VQ)

In this case

- each cluster is treated as a class (m compact classes are assumed)
- the available vectors have known class labels.

The goal:

Use a set of m representatives and place them in such a way so that each class is "optimally" represented.

The simplest version of VQ (LVQ1) may be obtained from GCLS as follows:

- Parts (A), (B) and (C) are the same with the basic competitive learning scheme.
- In part (D) the updating for $\underline{w}_j$' s is carried out as follows

$$\underline{w}_j(t) = \begin{cases} \underline{w}_j(t-1) + \eta(t)(\underline{x} - \underline{w}_j(t-1)), & \text{if } \underline{w}_j \text{ correctly wins on } \underline{x} \\ \underline{w}_j(t-1) - \eta(t)(\underline{x} - \underline{w}_j(t-1)), & \text{if } \underline{w}_j \text{ wrongly wins on } \underline{x} \\ \underline{w}_j(t-1), & \text{otherwise} \end{cases}$$

❖ Supervised Learning Vector Quantization (cont.)

In words:

- $\underline{w}_j$ is moved:
  - Toward $\underline{x}$ if $\underline{w}_j$ wins and $\underline{x}$ belongs to the j-th class.
  - Away from $\underline{x}$ if $\underline{w}_j$ wins and $\underline{x}$ does not belong to the j-th class.
- All other representatives remain unaltered.

VALLEY-SEEKING CLUSTERING ALGORITHMS

- Let $p(\underline{x})$ be the density function describing the distribution of the vectors in X.

Clusters may be viewed as peaks of $p(\underline{x})$ (clusters) separated by valleys.

Thus one may

- Identify these valleys and
- Try to move the borders of the clusters in these valleys.

A simple method in this spirit.

Preliminaries

- Let the distance $d(\underline{x},\underline{y})$ be defined as

$$d(\underline{x},\underline{y})=(\underline{y}-\underline{x})^T A(\underline{y}-\underline{x}),$$

where A is a positive definite matrix

- Let the local region of $\underline{x}$, $V(\underline{x})$, be defined as

$$V(\underline{x})=\{\underline{y}\in X-\{\underline{x}\}: d(\underline{x},\underline{y})\le a\},$$

where a is a user-defined parameter

- k_j^i be the number of vectors of the j cluster that belong to $V(\underline{x}_i)-\{\underline{x}_i\}$.
- $c_i\in\{1,\ldots,m\}$ denote the cluster to which $\underline{x}_i$ belongs.

❖ Valley-Seeking Clustering Algorithms (cont.)

Valley-Seeking algorithm

- Fix a.
- Fix the number of clusters m.
- Define an initial clustering X.
- Repeat
  - For $i=1$ to N
    - Find j: $k_j^i = \max_{q=1,\ldots,m} k_q^i$
    - Set $c_i = j$
  - End For
  - For $i=1$ to N
    - Assign $\underline{x}_i$ to cluster C_{ci}.
  - End For
- Until no reclustering of vectors occurs.

❖ Valley-Seeking Clustering Algorithms (cont.)

The algorithm

- Moves a window $d(\underline{x},\underline{y}) \leq a$ at $\underline{x}$ and counts the points from different clusters in it.
- Assigns $\underline{x}$ to the cluster with the larger number of points in the window (the cluster that corresponds to the highest local pdf).

In other words

- The boundary is moved away from the "winning" cluster (close similarity with Parzen windows).

- Remarks:
  - The algorithm is sensitive to a. It is suggested to perform several runs, for different values of a.
  - The algorithm is of a mode-seeking nature (if more than enough clusters are initially appointed, some of them will be empty).

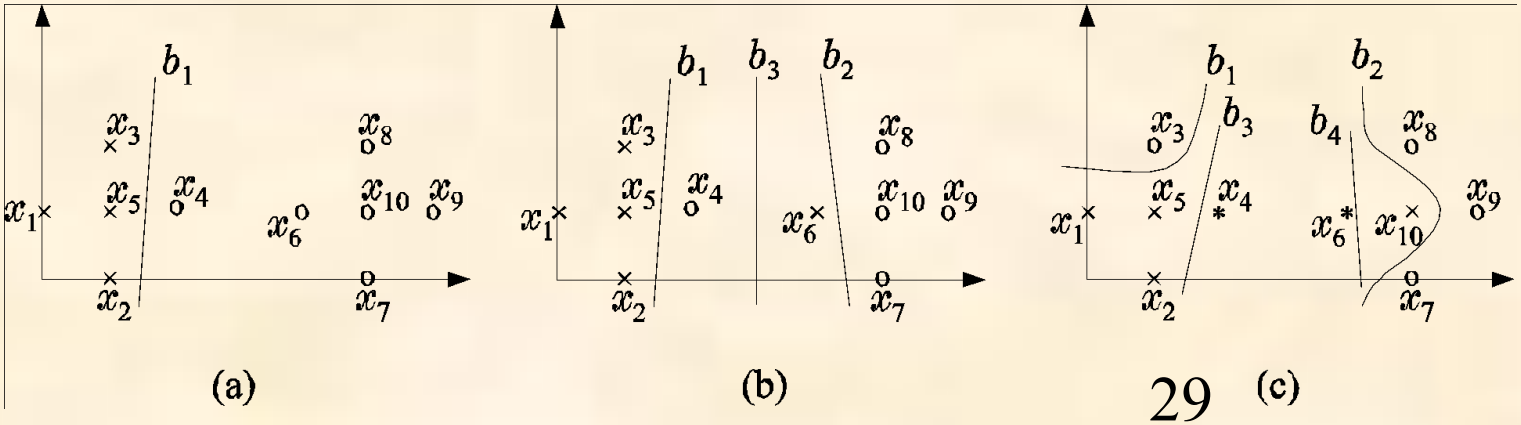
❖ Valley-Seeking Clustering Algorithms (cont.)

- Example 3: Let $X=\{\underline{x}_1,\ldots,\underline{x}_{10}\}$ and $a=1.1415$. X contains two clusters $C_1=\{\underline{x}_1,\ldots,\underline{x}_5\}$, $C_2=\{\underline{x}_6,\ldots,\underline{x}_{10}\}$.

(a) Initially two clusters are considered separated by b_1. After the convergence of the algorithm, C_1 and C_2 are identified (equivalently b_1 is moved between $\underline{x}_4$ and $\underline{x}_6$).

(b) Initially two clusters are considered separated by b_1, b_2 and b_3. After the convergence of the algorithm, C_1 and C_2 are identified (equivalently b_1 and b_2 are moved to the area where b_3 lies).

(c) Initially three clusters are considered separated by b_1, b_2, b_3, b_4. After the convergence of the algorithm, only two clusters are identified, C_1 and C_2 (equivalently b_1, b_2, b_3 and b_4 are moved between $\underline{x}_4$ and $\underline{x}_6$).



CLUSTERING VIA COST OPTIMIZATION (REV.)

- Branch and Bound Clustering Algorithms
  - They compute the globally optimal solution to combinatorial problems.
  - They avoid exhaustive search via the employment of a monotonic criterion J.

Monotonic criterion J: *if k vectors of X have been assigned to clusters, the assignment of an extra vector to a cluster does not decrease the value of J.*

Consider the following 3-vectors, 2-class case

121: 1^{st}, 3^{rd} vectors belong to class 1

2^{nd} vector belongs to class 2.

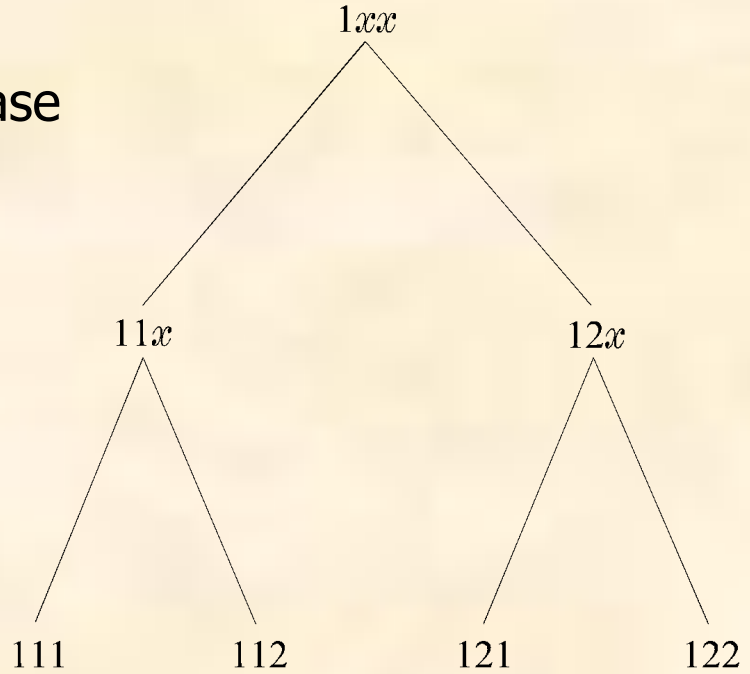
(leaf of the tree)

$12x$: 1^{st} vector belongs to class 1

2^{nd} vector belongs to class 2

3^{rd} vector is unassigned

(Partial clustering- node of the tree).



❖ Branch and Bound Clustering Algorithms

How exhaustive search is avoided

- Let B be the best value for criterion J computed so far.
- If at a node of the tree, the corresponding value of J is greater than B, no further search is performed for all subsequent descendants springing from this node.
- Let $\mathbf{C}_r=[c_1,\ldots,c_r]$, $1\le r\le N$, denotes a partial clustering where $c_i\in\{1,2,\ldots,m\}$, $c_i=j$ if the vector $\underline{x}_i$ belongs to cluster C_j and $\underline{x}_{r+1},\ldots,\underline{x}_N$ are yet unassigned.

For compact clusters and fixed number of clusters, *m*, a suitable cost function is

$$J(\mathbf{C}_r)=\sum_{i=1}^{r}||\underline{x}_i-\underline{m}_{c_i}(\mathbf{C}_r)||^2$$

where $\underline{m}_{ci}$ is the mean vector of the cluster C_{ci}

$$\underline{m}_j(\mathbf{C}_r)=\frac{1}{n_j(\mathbf{C}_r)}\sum_{\{q=1,\ldots r:\ c_q=j\}}\underline{x}_q,\quad j=1,\ldots,m$$

with $n_j(\mathbf{C}_r)$ being the number of vectors $\underline{x}\in\{\underline{x}_1,\ldots,\underline{x}_r\}$ that belong to cluster C_j.

❖ Branch and Bound Clustering Algorithms (cont.)

- *Initialization*
  - Start from the initial node and go down to a leaf. Let B be the cost of the corresponding clustering (initially set $B=+\infty$).
- *Main stage*
  - Start from the initial node of the tree and go down until either
    - (i) A leaf is encountered.
      - If the cost B' of the corr. clustering C' is smaller than B then
        - $B=B'$
        - C' is the best clustering found so far
      - End if
    - Or (ii) a node q with value of J greater than B is encountered. Then
      - No subsequent clustering branching from q is considered.
      - Backtrack to the parent of q, q^{par}, in order to span a different path.
      - If all paths branching from q^{par} have been considered then
        - Move to the grandparent of q.
      - End if
    - End if
- Terminate when all possible paths have been considered explicitly or implicitly.

❖ Branch and Bound Clustering Algorithms (cont.)

- Remarks:
  - Variations of the above algorithm, where much tighter bounds of B are used (that is many more clusterings are rejected without explicit consideration) have also been proposed.
  - A disadvantage of the algorithm is the excessive (and unpredictable) amount of required computational time.

❖ Simulated Annealing

- It guarantees (under certain conditions) in probability, the computation of the globally optimal solution of the problem at hand via the minimization of a cost function J.
- It may escape from local minima since it allows moves that temporarily may increase the value of J.

Definitions

- An important parameter of the algorithm is the "temperature" T, which starts at a high value and reduces gradually.
- A sweep is the time the algorithm spents at a given temperature so that the system can enter the "thermal equilibrium".

Notation

- T_{max} is the initial value of the temperature T.
- $\mathbf{C}_{init}$ is the initial clustering.
- $\mathbf{C}$ is the current clustering.
- t is the current sweep.

- The algorithm:
  - Set $T=T_{max}$ and $\mathbf{C}=\mathbf{C}_{init}$.
  - $t=0$
  - Repeat
    - $t=t+1$
    - Repeat
      - Compute $J(\mathbf{C})$
      - Produce a new clustering, $\mathbf{C}'$, by assigning a randomly chosen vector from X to a different cluster.
      - Compute $J(\mathbf{C}')$
      - If $\Delta J=J(\mathbf{C}')-J(\mathbf{C})<0$ then
        - * (A) $\mathbf{C}=\mathbf{C}'$
      - Else
        - * (B) $\mathbf{C}=\mathbf{C}'$, with probability $P(\Delta J)=e^{-\Delta J/T}$.
      - End if
    - Until an equilibrium state is reached at this temperature.
    - $T=f(T_{max},t)$
  - Until a predetermined value T_{min} for T is reached.

❖ Simulated Annealing (cont.)

- Remarks:
  - For $T \to \infty$, $p(\Delta J) \approx 1$. Thus almost all movements of vectors between clusters are allowed.
  - For lower values of T fewer moves of type (B) (from lower to higher cost clusterings) are allowed.
  - As $T \to 0$ the probability of moves of type (B) tends to zero.
  - Thus as T decreases, it becomes more probable to reach clusterings that correspond to lower values of J.
  - Keeping T positive, we ensure a nonzero probability for escaping from a local minimum.
  - We assume that the equilibrium state has been reached

    "If for k successive random reassignments of vectors, $\mathbf{C}$ remains unchanged."
  - A schedule for lowering T that guarantees convergence to the global minimum with probability 1, is

$$T = T_{max} / ln(1+t)$$

❖ Deterministic Annealing (DA)

- It is inspired by the phase transition phenomenon observed when the temperature of a material changes. It involves the parameter $\beta=1/T$, where T is defined as in simulated annealing.

- The Goal of DA: Locate a set of representatives w_j, $j=1,\ldots,m$ (m is fixed) in appropriate positions so that a distortion function J is minimized.

  J is defined as

$$J = -\frac{1}{\beta}\sum_{i=1}^{N}\ln\left(\sum_{j=1}^{m} e^{-\beta d(\underline{x}_i,\underline{w}_j)}\right)$$

- By defining P_{ir} as the probability that $\underline{x}_i$ belongs to C_r, $r=1,\ldots,m$, we can write:

$$P_{ir} = \frac{e^{-\beta d(\underline{x}_i,\underline{w}_r)}}{\sum_{j=1}^{m} e^{-\beta d(\underline{x}_i,\underline{w}_j)}}$$

- Then, the optimal value of $\underline{w}_r$ satisfies the following condition:

$$\frac{\partial J}{\partial \underline{w}_r} = \sum_{i=1}^{N} P_{ir}\frac{\partial d(\underline{x}_i,\underline{w}_r)}{\partial \underline{w}_r} = 0$$

❖ Deterministic Annealing (cont.)

Assumption: $d(\underline{x},\underline{w})$ is a convex function of $\underline{w}$ for fixed $\underline{x}$.

- *Stages of the algorithm*
  - For $\beta=0$, all P_{ij}'s are equal to $1/m$, for all $\underline{x}_i$'s, $i=1,\ldots,N$. Thus

$$\sum_{i=1}^{N} \frac{\partial d(\underline{x}_i,\underline{w}_r)}{\partial \underline{w}_r} = 0$$

    Since $d(\underline{x},\underline{w})$ is a convex function, $d(\underline{x}_1,\underline{w}_r)+\ldots+d(\underline{x}_N,\underline{w}_r)$ is a convex function. All representatives coincide with its unique global minimum (all the data belong to a single cluster).

  - As β increases, it reaches a critical value where P_{ir} "depart sufficiently" from the uniform model. Then the representatives split up in order to provide an optimal presentation of the data set at the new phase.

  - The increase of β continues until P_{ij} approach the hard clustering model (for all $\underline{x}_i$, $P_{ir}\approx 1$ for some r, and $P_{ij}\approx 0$ for $j\neq r$).

❖ Deterministic Annealing (cont.)

- Remarks:
  - It is not guaranteed that it reaches the globally optimum clustering.
  - If m is chosen greater than the "actual" number of clusters, the algorithm has the ability to represent the data properly.

❖ Clustering using genetic algorithms (GA)

A few hints concerning genetic algorithms

- They have been inspired by the natural selection mechanism (Darwin).
- They consider a population of solutions of the problem at hand and they perform certain operators to this, so that the new population is improved compared to the previous one (in terms of a criterion function J).
- The solutions are coded and the operators are applied on the coded versions of the solutions.

The most well-known operators are:

- Reproduction:
  - It ensures that, in probability, the better (worse) a solution in the current population is, the more (less) replicates it has in the next population.
- Crossover:
  - It applies to the temporary population produced after the application of the reproduction operator.
  - It selects pairs of solutions randomly, splits them at a random position and exchanges their second parts.

❖ Clustering using genetic algorithms (cont.)

- Mutation:
  - It applies after the reproduction and the crossover operators.
  - It selects randomly an element of a solution and alters it with some probability.
  - It may be viewed as a way out of getting stuck in local minima.

Aspects/Parameters that affect the performance of the algorithm

- The coding of the solutions.
- The number of solutions in a population, p.
- The probability with which we select two solutions for crossover.
- The probability with which an element of a solution is mutated.

❖ Clustering using genetic algorithms (cont.)

GA Algorithmic scheme

- $t=0$
- Choose an initial population $\wp_t$ of solutions.
- Repeat
  - Apply reproduction on $\wp_t$ and let $\wp_t'$ be the resulting temporary population.
  - Apply crossover on $\wp_t'$ and let $\wp_t''$ be the resulting temporary population.
  - Apply mutation on $\wp_t''$ and let $\wp_{t+1}$ be the resulting population.
  - $t=t+1$
- Until a termination condition is met.
- Return
  - either the best solution of the last population,
  - or the best solution found during the evolution of the algorithm.

❖ Clustering using genetic algorithms (cont.)

Genetic Algorithms in Clustering

The characteristics of a simple GA hard clustering algorithm suitable for compact clusters, whose number m is fixed, is discussed next.

- A (not unique) way to code a solution is via the cluster representatives.

$$[\underline{w}_1, \underline{w}_2, \ldots, \underline{w}_m]$$

- The cost function in use is

where $$J = \sum_{i=1}^{N} u_{ij} d(\underline{x}_i, \underline{w}_j)$$

$$u_{ij} = \begin{cases} 1, & if \ d(\underline{x}_i, \underline{w}_j) = \min_{k=1,\ldots,m} d(\underline{x}_i, \underline{w}_k) \\ 0, & \text{otherwise} \end{cases}, \quad i = 1,\ldots,N$$

- The allowable cut points for the crossover operator are between different representatives.
- The mutation operator selects randomly a coordinate and decides randomly to add a small random number to it.

❖ Clustering using genetic algorithms (cont.)

- Remark:
  - An alternative to the above scheme results if prior to the application of the reproduction operator, the hard clustering algorithm (GHAS), described in the previous chapter, runs p times, each time using a different solution of the current population as the initial state. The p resulting solutions constitute the population on which the reproduction operator will be applied.

DENSITY-BASED ALGORITHMS FOR LARGE DATA SETS

These algorithms:

- Consider clusters as regions in the l-dimensional space that are "dense" in points of X.
- Have, in principle, the ability to recover arbitrarily shaped clusters.
- Handle efficiently outliers.
- Have time complexity less than $O(N^2)$.

Typical density-based algorithms are:

- The DBSCAN algorithm.
- The DBCLASD algorithm.
- The DENCLUE algorithm.

❖ The Density-Based Spatial Clustering of Applications with Noise (DBSCAN) Algorithm.

The "density" around a point $\underline{x}$ is estimated as the number of points in X that fall inside a certain region of the l-dimensional space surrounding $\underline{x}$.

Notation

- $V_\varepsilon(\underline{x})$ is the hypersphere of radius ε (user-defined parameter) centered at $\underline{x}$.
- $N_\varepsilon(\underline{x})$ the number of points of X lying in $V_\varepsilon(\underline{x})$.
- q is the minimum number of points of X that must be contained in $V_\varepsilon(\underline{x})$, in order for $\underline{x}$ to be considered an "interior" point of a cluster.

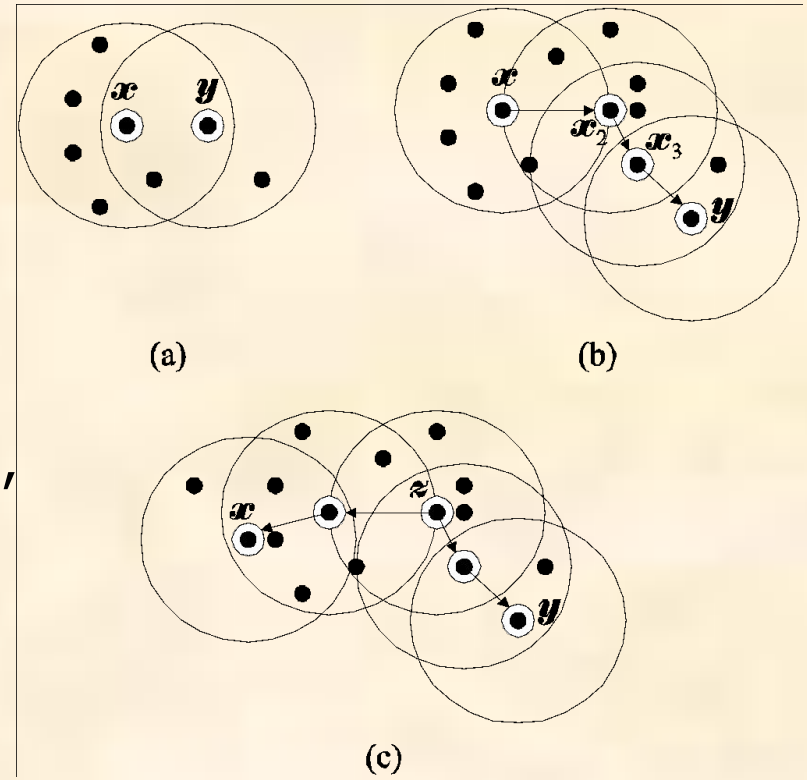
Definitions

1. A point $\underline{y}$ is directly density reachable from a point $\underline{x}$ if
   (i) $\underline{y} \in V_\varepsilon(\underline{x})$
   (ii) $N_\varepsilon(\underline{x}) \geq q$ (fig. (a)).
2. A point $\underline{y}$ is density reachable from a point $\underline{x}$ in X if there is a sequence of points $\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_p \in X$, with $\underline{x}_1 = \underline{x}$, $\underline{x}_p = \underline{y}$, such that $\underline{x}_{i+1}$ is directly density reachable from $\underline{x}_i$ (fig. (b)).

❖ The DBSCAN Algorithm (cont.)

3. A point x is density connected to a point $y \in X$ if there exists $z \in X$ such that both x and y are density reachable from z (fig. (c)).

Example 4: Assuming that $q=5$, (a) y is directly density reachable from x, but not vice versa, (b) y is density reachable from x, but not vice versa, and (c) x and y are density connected (in addition, y is density reachable from x, but not vice versa).



❖ The DBSCAN Algorithm (cont.)

4. A cluster C in DBSCAN is defined as a nonempty subset of X satisfying the following conditions:
   - If $\underline{x}$ belongs to C and $\underline{y} \in X$ is density reachable from $\underline{x}$, then $\underline{y} \in C$.
   - For each pair $(\underline{x}, \underline{y}) \in C$, $\underline{x}$ and $\underline{y}$ are density connected.

5. Let $C_1, \ldots, C_m$ be the clusters in X. The set of points that are not connected in any of the $C_1, \ldots, C_m$ is known as noise.

6. A point $\underline{x}$ is called a core (noncore) point if it has at least (less than) q points in its neighborhood. A noncore point may be either
   - a border point of a cluster (that is, density reachable from a core point) or
   - a noise point (that is, not density reachable from other points in X).

❖ The DBSCAN Algorithm (cont.)

Proposition 1: If x is a core point and D is the set of points in X that are density reachable from x, then D is a cluster.

Proposition 2: If C is a cluster and x is a core point in C, then C equals to the set of the points $y \in X$ that are density reachable from x.

Therefore: A cluster is uniquely determined by any of its core points.

Notation

- X_{un} is the set of points in X that have not been considered yet.
- m denotes the number of clusters.

The DBSCAN Algorithm (cont.)

DBSCAN Algorithm

- Set $X_{un}=X$
- Set $m=0$
- While $X_{un} \neq \varnothing$ do
  - Arbitrarily select a $\underline{x} \in X_{un}$
  - If $\underline{x}$ is a noncore point then
    - Mark $\underline{x}$ as noise point
    - $X_{un}=X_{un}-\{\underline{x}\}$
  - End if
  - If $\underline{x}$ is a core point then
    - $m=m+1$
    - Determine all density-reachable points in X from $\underline{x}$.
    - Assign $\underline{x}$ and the previous points to the cluster C_m. The border points that may have been marked as noise are also assigned to C_m.
    - $X_{un}=X_{un}-C_m$
  - End {if}
- End {while}

❖ The DBSCAN Algorithm (cont.)

- Important notes:
  - If a border point y of a cluster C is selected, it will be marked initially as a noise point. However, when a core point x in C will be selected later on, y will be identified as a density-reachable point from x and it will assigned to C.
  - If a noise point y is selected, it will be marked as such and since it is not density reachable by any of the core points in X, its "noise" label will remain unaltered.

- Remarks:
  - The parameters ε and q influence significantly the performance of DBSCAN. These should selected such that the algorithm is able to detect the least "dense" cluster (experimentation with several values for ε and q should be carried out).
  - Implementation of DBSCAN using R*-tree data structure can achieve time complexity of $O(Nlog_2N)$ for low-dimensional data sets.
  - DBSCAN is not well suited for cases where clusters exhibit significant differences in density as well as for applications of high-dimensional data.

The Distribution-Based Clustering of LArge Spatial Databases (DBCLASD) Algorithm

Key points of the algorithm:

- The distance d of a point $\underline{x} \in X$ to its nearest neighbor is considered as a random variable.
- The "density" is quantified in terms of the probability distribution of d.
- Under the assumption that the points in each cluster are uniformly distributed, the distribution of d can be derived analytically.
- A cluster C is defined as a nonempty subset of X with the following properties:
  - The distribution of the distances, d, between points in C and their nearest neighbors is in agreement with the distribution of d derived theoretically, within some confidence interval.
  - It is the maximal set with this property (insertion of additional points neighboring to points in C will cause (a) not to hold anymore).
  - It is connected. Having applied a grid of cubes on the feature space, this property implies that for any pair of points $(\underline{x},\underline{y})$ from C there exists a path of adjacent cubes that contains at least one point in C that connects $\underline{x}$ and $\underline{y}$.

The DBCLASD Algorithm (cont.)

- Points are considered in a sequential manner.
- A point that has been assigned to a cluster may be reassigned to another cluster at a later stage of the algorithm.
- Some points are not assigned to any of the clusters determined so far, but they are tested again at a later stage.

- Remarks:
  - DBCLASD is able to determine arbitrary shaped clusters of various densities in X.
  - It requires no parameter definition.
  - Experimental results show that:
    - the runtime of DBCLASD is roughly twice the runtime of DBSCAN.
    - DBCLASD outperforms CLARANS by a factor of at least 60.

❖ The DENsity-based CLUstEring (DENCLUE) Algorithm

Definitions

The influence function $f^{\underline{y}}(\underline{x})$ for a point $\underline{y} \in X$ is a positive function that decays to zero as $\underline{x}$ "moves away" from $\underline{y}$ ($d(\underline{x},\underline{y}) \to \infty$). Typical examples are:

$$f^{\underline{y}}(\underline{x}) = \begin{cases} 1, & \text{if } d(\underline{x},\underline{y}) < \sigma \\ 0, & \text{otherwise} \end{cases}, \qquad f^{\underline{y}}(\underline{x}) = e^{-\frac{d(\underline{x},\underline{y})^2}{2\sigma^2}}$$

where σ is a user-defined function.

The density function based on X is defined as (Remember the Parzen windows):

$$f^X(\underline{x}) = \sum_{i=1}^{N} f^{\underline{x}_i}(\underline{x})$$

The Goal:

(a) Identify all "significant" local maxima, $\underline{x}_j^*$, $j=1,\ldots,m$ of $f^X(\underline{x})$

(b) Create a cluster C_j for each $\underline{x}_j^*$ and assign to C_j all points of X that lie within the "region of attraction" of $\underline{x}_j^*$.

❖ The DENCLUE Algorithm (cont.)

Two clarifications

- The region of attraction of $\underline{x}_j^*$ is defined as the set of points in $\underline{x} \in \Re^l$ such that if a "hill-climbing" (such as the steepest ascent) method is applied, initialized by $\underline{x}$, it will terminate arbitrarily close to $\underline{x}_j^*$.
- A local maximum is considered as significant if $f^X(\underline{x}_j^*) \geq \xi$ (ξ is a user-defined parameter).

Approximation of $f^X(\underline{x})$

$$\hat{f}^X(\underline{x}) = \sum_{\underline{x}_i \in Y(\underline{x})} f^{\underline{x}_i}(\underline{x})$$

where $Y(\underline{x})$ is the set of points in X that lie "close" to $\underline{x}$.

The above framework is used by the DENCLUE algorithm.

The DENCLUE Algorithm (cont.)

DENCLUE algorithm

- *Preclustering stage* (identification of regions dense in points of X)
  - Apply an l-dimensional grid of edge-length 2σ in the $\Re^l$ space.
  - Determine the set D_p of the hypercubes that contain at least one point of X.
  - Determine the set D_{sp} ($\subset D_p$) that contains the "highly populated" cubes of D_p (that is, cubes that contain at least $\xi_c > 1$ points of X).
  - For each $c \in D_{sp}$ define a connection with all neighboring cubes c_j in D_p for which $d(\underline{m}_c, \underline{m}_{cj})$ is no greater than 4σ, where $\underline{m}_c$, $\underline{m}_{cj}$ are the means of c and c_j, respectively.
- *Main stage*
  - Determine the set D_r that contains:
    - the highly populated cubes and
    - the cubes that have at least one connection with a highly populated cube.
  - For each point $\underline{x}$ in a cube $c \in D_r$ determine $Y(\underline{x})$ as the set of points that belong to cubes c_j in D_r such that the mean values of c_js lie at distance less than $\lambda\sigma$ from $\underline{x}$ (typically λ=4).

❖ The DENCLUE Algorithm (cont.)

DENCLUE algorithm (cont.)

- For each point $\underline{x}$ in a cube $c \in D_r$
  - Apply a hill climbing method starting from $\underline{x}$ and let $\underline{x}^*$ be the local maximum to which the method converges.
  - If $\underline{x}^*$ is a significant local maximum ($f^X(\underline{x}^*) \geq \xi$) then
    - If a cluster C associated with $\underline{x}^*$ has already been created then
      - $\underline{x}$ is assigned to C
    - Else
      - Create a cluster C associated with $\underline{x}^*$
      - Assign $\underline{x}$ to C
    - End if
  - End if
- End for

❖ The DENCLUE Algorithm (cont.)

- Remarks:
  - Shortcuts allow the assignment of points to clusters, without having to apply the hill-climbing procedure.
  - DENCLUE is able to detect arbitrarily shaped clusters.
  - The algorithm deals with noise very satisfactory.
  - The worst-case time complexity of DENCLUE is $O(Nlog_2N)$.
  - Experimental results indicate that the average time complexity is $O(log_2N)$.
  - It works efficiently with high-dimensional data.

CLUSTERING ALGORITHMS FOR HIGH-DIMENSIONAL DATA SETS

- What is a High-dimensionality space?

  Dimensionality l of the input space with

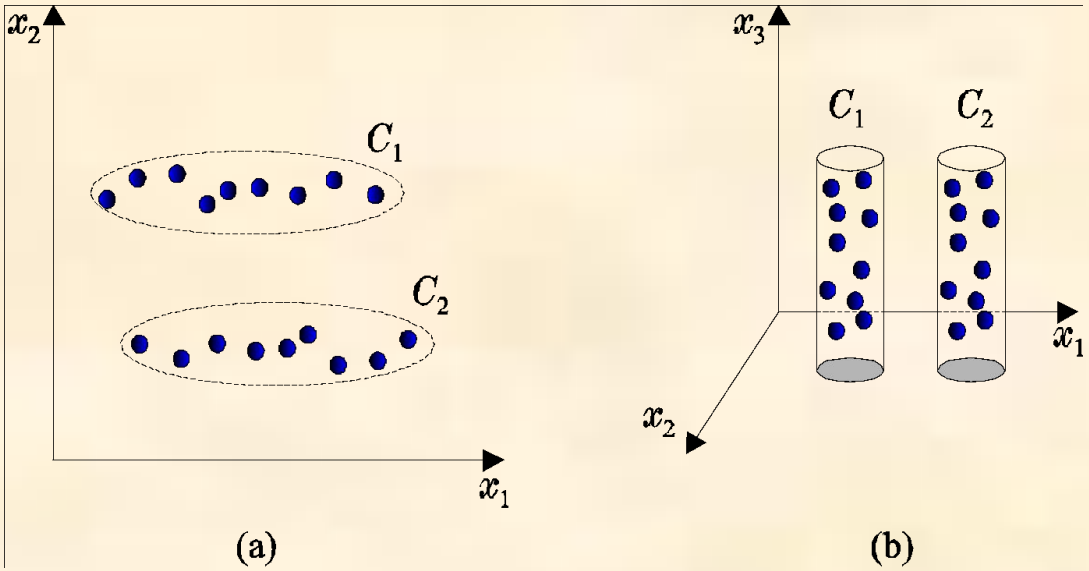
  $$20 \leq l \leq \text{few thousands}$$

  indicate high-dimensional data sets.

- Problems of considering simultaneously all dimensions in high-dimensional data sets:
  - "Curse of dimensionality". As a fixed number of points spread out in high-dimensional spaces, they become almost equidistant (that is, the terms similarity and dissimilarity tend to become meaningless).
  - Several dimensions may be irrelevant to the identification of the clusters (that is, the clusters usually are identified in subspaces of the original feature space).

- A way out: *Work on subspaces of dimension lower than l.*
  - Main approaches:
    - Dimensionality reduction clustering approach.
    - Subspace clustering approach.

- An example:



❖ Dimensionality Reduction Clustering Approach

Main idea

- Identify an appropriate l'-dimensional space $H_{l'}$ ($l' < l$).
- Project the data points in X into $H_{l'}$.
- Apply a clustering algorithm on the projections of the points of X into $H_{l'}$.

Identification of $H_{l'}$ may be carried out using either by:

- Feature generation methods,
- Feature selection methods,
- Random projections.

❖ Dimensionality Reduction Clustering Approach (cont.)

Feature generation methods

- In general, these methods preserve the distances between the points in the high-dimensional space, when these are mapped to the lower-dimensional space.
- They are very useful in producing compact representations of the original high-dimensional feature space.
- Some of them may be integrated within a clustering algorithm (k-means, EM).
- They are useful in cases where a significant number of features contributes to the identification of the clusters.
- Typical Methods in this category are:
  - Principal component analysis (PCA).
  - Singular value decomposition (SVD).
  - Nonlinear PCA.
  - Independent component analysis (ICA).

❖ Dimensionality Reduction Clustering Approach (cont.)

Feature selection methods

- They identify the features that are the main contributors to the formation of the clusters.
- The criteria used to evaluate the "goodness" of a specific subset of features follow either the
  - Wrapper model (The clustering algorithm is first chosen and a set of features F_i is evaluated through the results obtained from the application of the algorithm to X, where for each point only the features in F_i are taken into account).
  - Filter model (The evaluation of a subset of features is carried out using intrinsic properties of the data, prior to the application of the clustering algorithm).
- They are useful when all clusters lie in the same subspace of the feature space.

❖ Dimensionality Reduction Clustering Approach (cont.)

Clustering using Random Projections:

Here $H_{l'}$ is identified in a random manner.

Note: The projection of an l-dimensional space to an l'-dimensional space ($l' < l$) is uniquely defined via an $l' \times l$ projection matrix A.

Issues to be addressed:

(a) The proper estimate of l'. Estimates of l' guarantee that the distances between the points of X, in the original data set will be preserved (with some distortion) after the projection to a randomly chosen l'-dimensional space, whose projection matrix is constructed via certain probabilistic rules (Preservation of distances does not necessarily preserve clusters).

(b) The definition of the projection matrix A. Possible rules for constructing A are:

1. Set each of its entries equal to a value stemming from an i.i.d. zero mean, unit variance Gaussian distribution and then normalize each row to the unit length.
2. Set each entry of A equal to -1 or $+1$, with probability 0.5.
3. Set each entry of A equal to $+\sqrt{3}$, $-\sqrt{3}$ or 0, with probabilities $1/6$, $1/6$, $2/3$, respectively.

❖ Dimensionality Reduction Clustering Approach (cont.)

Having defined A:

- Project the points of X into $H_{l'}$
- Perform a clustering algorithm on the projections of the points of X into $H_{l'}$.

Problem: Different random projections may lead to totally different results.

Solution:

- Perform several random projections $H_{l'}$.
- Apply a clustering algorithm on the projections of X to each $H_{l'}$.
- Combine the clustering results and produce the final clustering.

A method in the above spirit is described next ($O(N^2)$).

Clustering using Random Projections

- Select l'.
- Generate $A_1,\ldots,A_r$ different projection matrices using the (b.1) rule given above.
- For $s=1$ to r
  - Run GMDAS for the s-th random projection of X.
  - Compute the probability that $\underline{x}_i$ belongs to the j-th cluster of the s projection, $P(C_j^s|\underline{x}_i)$, $i=1,\ldots,N$, $j=1,\ldots,m_s$.
  - Create the similarity matrix P^s, where $P_{ij}^{\ s}$ is the probability that $\underline{x}_i$ and $\underline{x}_j$ belong to the same cluster,

$$P_{ij}^s = \sum_{q=1}^{m_s} P(C_q^s|\underline{x}_i)P(C_q^s|\underline{x}_j)$$

- End for
- Compute the average proximity matrix P, so that P_{ij} is the average of $P_{ij}^{\ s}$ s, $s=1,\ldots,r$.
- Apply GAS (actually its complete link version) on P.
- Plot the similarity between the closest pair of clusters at each iteration versus the number of iterations.
- Select the clustering that corresponds to the most abrupt decrease in the plot.

❖ Subspace Clustering Approach

- This approach deals with the problem where clusters are formed in different subspaces of the feature space.
- The subspace clustering algorithms (SCA) reveal clusters as well as the subspaces where they reside.
- SCA can be divided into two main categories
  - Grid-based SCAs
  - Point-based SCAs.

Grid-based Subspace Clustering Algorithms (GBSCAs)

Main strategy:

1) Identification of the subspaces of the feature space that are likely to contain clusters.
2) Determination of the clusters lying in each of these subspaces.
3) Description of the resulting clusters.

Grid-based Subspace Clustering Algorithms (cont.)

1) Identification of subspaces

- Here an l-dimensional grid is applied on the feature space and the subspaces that are likely to contain clusters are identified based on the k-dimensional units (boxes) ($k \leq l$) defined by the grid.
- In order to avoid to consider explicitly all the possible subspaces, the algorithms establish certain criteria that comply with the so-called downward closure property.
- Downward closure (DC) property: If a criterion is satisfied in a k-dimensional space, it is also satisfied in **all** of its $(k\text{-}1)$-dimensional subspaces.
- Due to the DC property, identification of subspaces is carried out in an **iterative** bottom-up fashion (from lower to higher dimensional subspaces).

2) Determination of the clusters

Clusters are identified as maximally connected components of units in each of the subspaces defined in the previous step.

Grid-based Subspace Clustering Algorithms (cont.)

Two popular GBSCAs are:

- The CLIQUE.
- The ENCLUS.

- The CLIQUE (CLustering In QUEst) Algorithm

  Preliminaries – Definitions

  Assume that an l-dimensional grid of edge-size ξ is applied on the feature space.

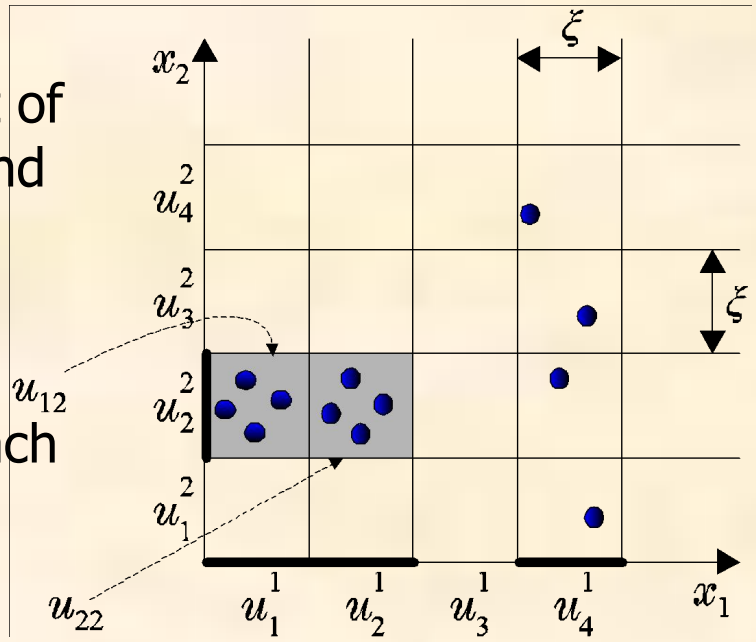
  - A unit u of the grid is written as $u_{t1}\text{x}\ldots\text{x}\,u_{tk} \equiv (u_{t1}, \ldots, u_{tk})$ ($t_1<\ldots<t_k$, $k \leq l$), where $u_{ti}=[a_{ti},b_{ti})$ is a right-open interval in the partitioning of the t_i-th dimension of the feature space (e.g., $t_1=2$, $t_2=5$, $t_3=7$ indicates a unit lying in the subspace spanned by $\underline{x}_2$, $\underline{x}_5$ and $\underline{x}_7$ dimensions).
  - A point $\underline{x}$ is contained in a k-dimensional unit $u=(u_{t1}, \ldots, u_{tk})$ if $a_{ti} \leq x_{ti} < b_{ti}$ for all t_i.
  - The selectivity of a unit u is defined as the fraction of the total number of data points (N) contained in u.
  - A unit u is called dense if its selectivity is greater than a user-defined threshold τ.

The CLIQUE Algorithm (cont.)

- Two k-dimensional units $u=(u_{t1},\ldots,u_{tk})$ and $u'=(u_{t1}',\ldots,u_{tk}')$ share a face if there are $(k\text{-}1)$-dimensions (e.g., $\underline{x}_{t1},\ldots,\underline{x}_{tk\text{-}1}$), such that $u_{tj}=u_{tj}'$, $j=1,2,\ldots,k\text{-}1$ and either $a_{tk}=b_{tk}'$ or $b_{tk}=a_{tk}'$.
- Two k-dimensional units are said to be directly connected if they have in common a $(k\text{-}1)$-dimensional face.
- Two k-dimensional units u_1 and u_2 are said to be connected if there exists a sequence of k-dimensional units, $v_1,\ldots,v_s$, with $v_1=u_1$ and $v_s=u_2$, such that each pair (v_i,v_{i+1}) of units is directly connected.
- In CLIQUE, a cluster is defined as a maximal set of connected dense units in k dimensions.
- Downward closure property of the density: "If there is a dense unit u in a k-dimensional space, there are also dense units in the projections of u in all $(k\text{-}1)$-dimensional subspaces of the k-dimensional space".

- Example 5: A two dimensional grid of lines of edge size ξ applied in the two-dimensional feature space.
  - Two-dimensional and one-dimensional units are defined.
  - u_i^q denotes the i-th one dimensional unit along x_q
  - u_{ij} denotes the two dimensional unit resulting from the Cartesian product of the i-th and j-th intervals along x_1 and x_2, respectively.
  - For τ=3:
    - u_1^1, u_2^1, u_4^1, u_2^2 are one-dimensional dense units, each containing 4, 4, 4 and 9 points, respectively.
    - u_{12} and u_{22} are two-dimensional dense units each containing 4 points.
  - u_{12} and u_{22} are directly connected.

- The downward closure property of the density is shown for units u_{12}, u_{22}.



The CLIQUE Algorithm (cont.)

Main stages of CLIQUE

1. Identification of subspaces.
2. Identification of clusters.
3. "Description" of clusters.

1. *Identification of subspaces*

A. Determination of dense units

- Determine the set D_1 of all one-dimensional dense units.
- $k=1$
- While $D_k \neq \varnothing$ do
  - $k=k+1$
  - Determine the set D_k as the set of all the k-dimensional dense units all of whose $(k\text{-}1)$-dimensional projections, belong to $D_{k\text{-}1}$.
- End while

The CLIQUE Algorithm (cont.)

B. Determination of high coverage subspaces.

- Determine all the subspaces that contain at least one dense unit.
- Sort these subspaces in descending order according to their coverage

  (fraction of the num. of points of the original data set they contain).
- Optimize a suitably defined Minimum Description Length criterion function and determine a threshold under which a coverage is considered "low".
- Select the subspaces with "high" coverage.

The CLIQUE Algorithm (cont.)

2. *Identification of clusters*

- For each high coverage subspace S do
  - Consider the set E of all the dense units in S.
  - While $E \neq \varnothing$
    - $m' = 1$
    - Select a randomly chosen unit u from E.
    - Assign to $C_{m'}$, u and all units of E that are connected to u.
    - $E = E - C_{m'}$
  - End while
- End for

The clusters in the data set are all clusters identified in all high coverage subspaces (they are consisted of units).

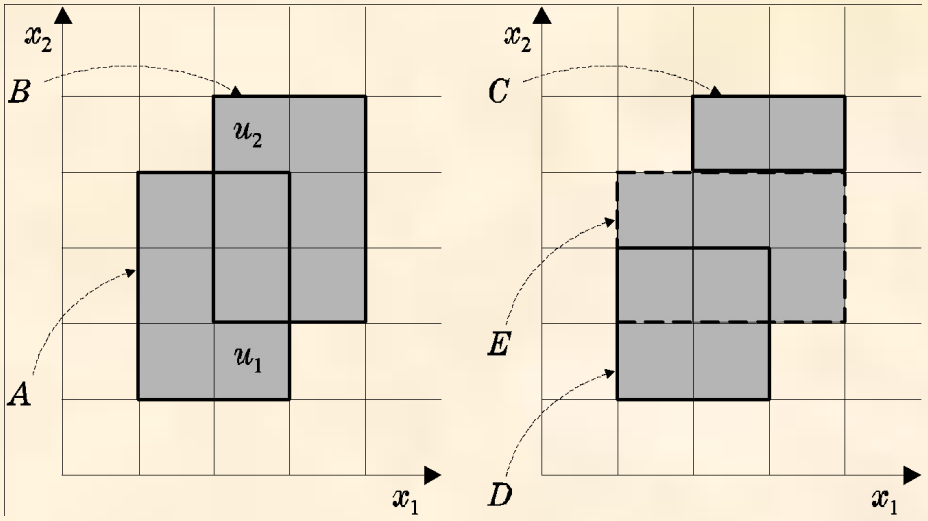
The CLIQUE Algorithm (cont.)

3. *Minimal description of clusters*

The minimal description of a cluster C, produced by the above procedure, is the minimum possible union of hyperrectangular regions.

For example

- $A \cup B$ is the minimum cluster description of the shaded region.
- $C \cup D \cup E$ is a non-minimal cluster description of the same region.



The CLIQUE Algorithm (cont.)

3. *Minimal description of clusters (algorithm)*

For each cluster C do

1st stage

- $c=0$
- While $C \neq \varnothing$
  - $c=c+1$
  - Choose a dense unit in C
  - For $i=1$ to l
    - Grow the unit in both directions along the i-th dimension, trying to cover as many units in C as possible (boxes that are not belong to C should not be covered).
  - End for
  - Define the set I containing all the units covered by the above procedure
  - $C=C-I$
- End while

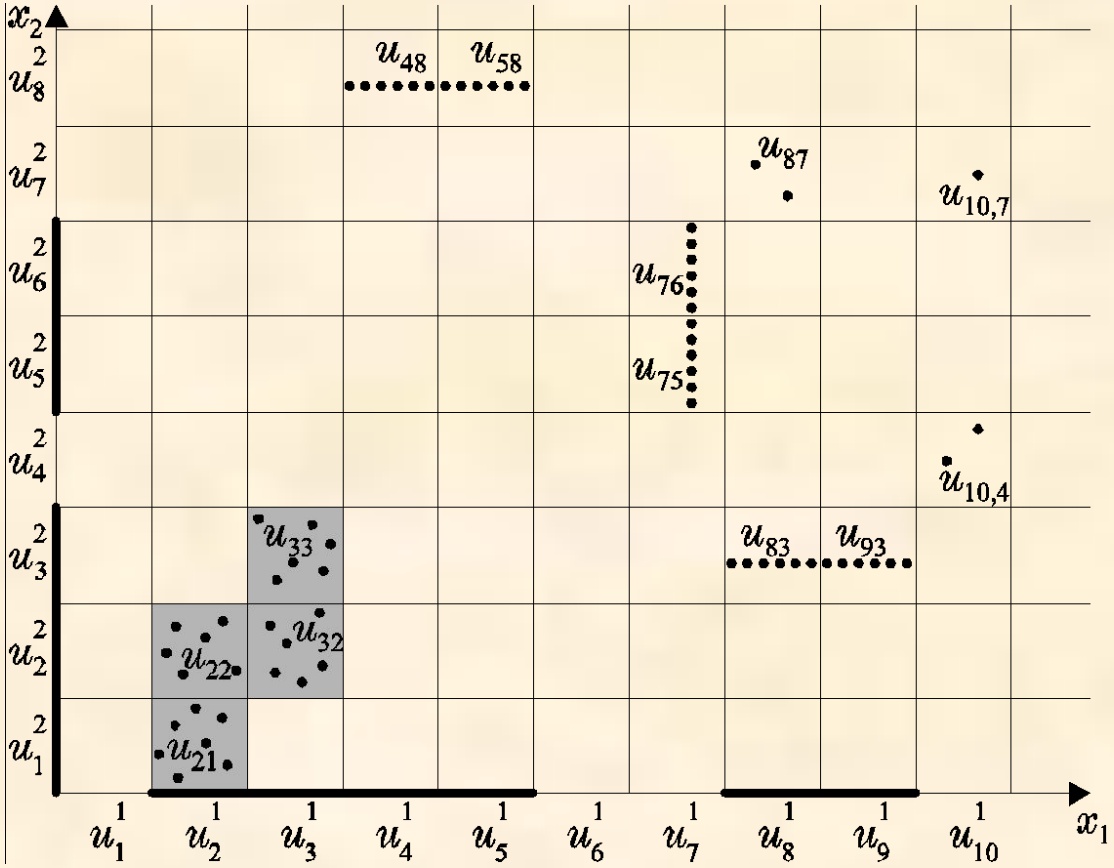
2nd stage

- Remove all covers whose units are covered by at least another cover.

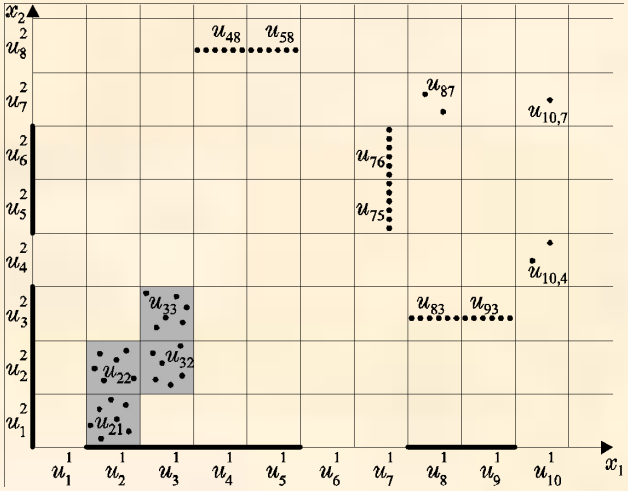
The CLIQUE Algorithm (cont.)

Example 6:

- u_i^q, u_{ij} are defined as in example 5.
- ξ=1 and τ=8% (thus, each unit containing more than 5 points is considered to be dense).
- The points in u_{48} and u_{58}, u_{75} and u_{76}, u_{83} and u_{93} are collinear.



Example 6 (cont.)



Identification of subspaces

One-dimensional dense units:

$D_1=\{u_2^1, u_3^1, u_4^1, u_5^1, u_8^1, u_9^1, u_1^2, u_2^2, u_3^2, u_5^2, u_6^2\}$

Two-dimensional dense units:

$D_2=\{u_{21}, u_{22}, u_{32}, u_{33}, u_{83}, u_{93}\}$

Notes:

- Although each one of the u_{48}, u_{75}, u_{76} contains more that 5 points, they are not included in D_2.
- Although it seems unnatural for u_{83} and u_{93} to be included in D_2, they are included since u_3^2 is dense.
- All subspaces of the two-dimensional space contain clusters.

Identification of clusters

One-dimensional clusters:

$C_1=\{u_2^1, u_3^1, u_4^1, u_5^1\}$, $C_2=\{u_8^1, u_9^1\}$, $C_3=\{u_1^2, u_2^2, u_3^2\}$, $C_4=\{u_5^2, u_6^2\}$.

Two-dimensional clusters:

$C_5=\{u_{21}, u_{22}, u_{32}, u_{33}\}$, $C_6=\{u_{83}, u_{93}\}$.

Example 6 (cont.):

Description of clusters

$C_1=\{(x_1)\colon 1\le x_1<5\}$

$C_2=\{(x_1)\colon 7\le x_1<9\}$

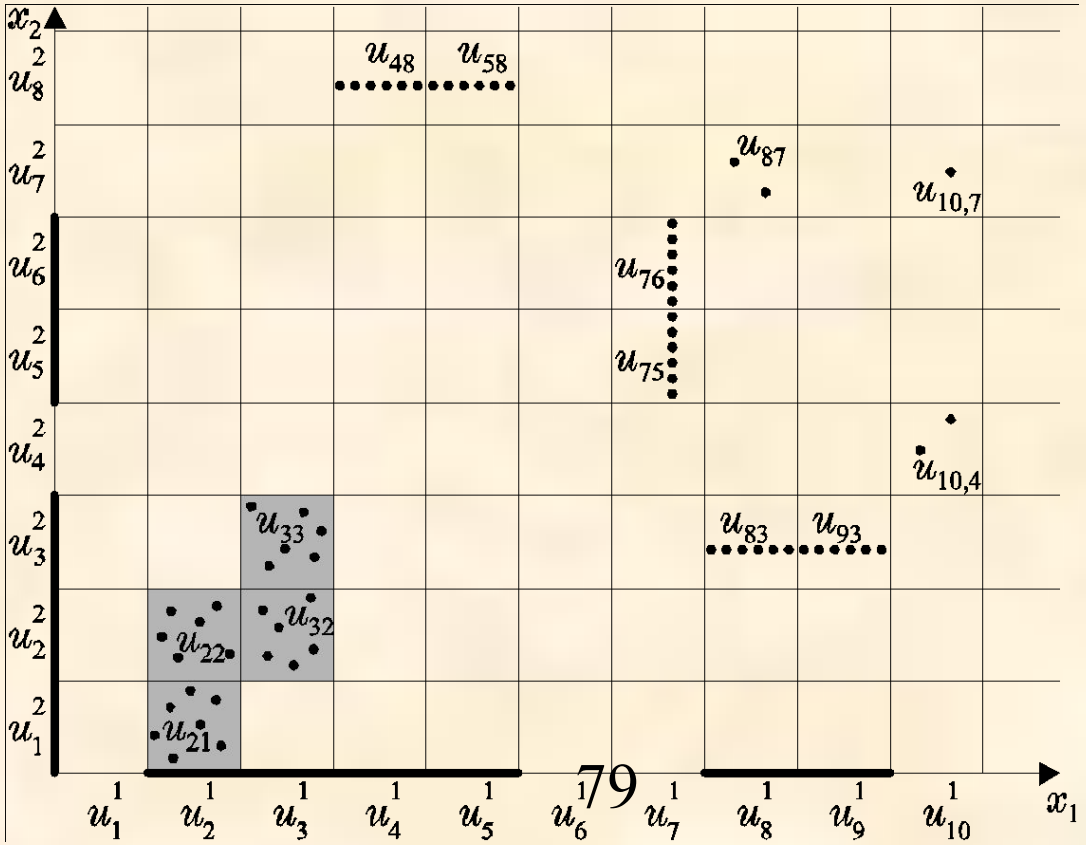
$C_3=\{(x_2)\colon 0\le x_2<3\}$

$C_4=\{(x_2)\colon 4\le x_2<6\}$

$C_5=\{(x_1, x_2)\colon 1\le x_1<2,\ 0\le x_2<2\}\cup\{(x_1, x_2)\colon 2\le x_1<3,\ 1\le x_2<3\}$

$C_6=\{(x_1, x_2)\colon 7\le x_1<9,\ 2\le x_2<3\}$

Note that C_2 and C_6 are essentially the same cluster, which is reported twice by the algorithm.



- The CLIQUE Algorithm (cont.)
- Remarks:
  - CLIQUE determines automatically the subspaces of the feature space where high-density clusters exist.
  - It is insensitive to the order of the presentation of the data.
  - It does not impose any data distribution hypothesis on the data set.
  - It scales linearly with N but scales exponentially with l.
  - The accuracy of the determined clusters may be degraded because the clusters are given not in terms of points of X but in terms of dense units.
  - The performance of the algorithm depends heavily on the choices of ξ and τ.
  - There is a large overlap among the reported clusters.
  - There is a risk of losing small but meaningful clusters, after the pruning of subspaces based on their coverage.

Grid-based Subspace Clustering Algorithms (cont.)

The ENCLUS Algorithm

- It follows the three-stage philosophy of CLIQUE (identification of subspaces, determination of clusters, description of clusters).
- The last two stages of ENCLUS are the same with those of CLIQUE.

1. *Identification of subspaces that contain clusters*

Definitions

- Entropy $H(X^k)$ of a k-dimensional subspace X^k ($k \le l$) of X:
  - Apply a k-dimensional grid on X^k.
  - Measure the percentage of points that fall within each one of the n grid units.
  - Compute the entropy as

$$H(X^k) = -\sum_{i=1}^{n} p_i \log_2 p_i$$

where n is the total number of units.

- Downward closure property of entropy: "If a k-dimensional space is of low entropy all of its $(k\text{-}1)$-dimensional subspaces are also of low entropy".

The ENCLUS Algorithm (cont.)

- Interest $\text{interest}(X^k)$ of a k-dimensional subspace X^k ($k \le l$) of X:

$$\text{interest}(X^k) = \sum_{j=1}^{k} H(x_{t_j}^k) - H(X^k)$$

where $t_1 < \ldots < t_k$ ($k \le l$) and $x_{tj}^{\ k}$ denotes the j dimension of X^k.

- It measures the degree of correlation among the dimensions of X^k.
- The higher the interest the stronger the correlation among the dimensions of X^k.
- The minimum value of the interest is zero. In this case, the dimensions of X^k are independent to each other.

- A subspace with "low" entropy (below a user-defined threshold ω) is considered to have a good clustering.
- A significant subspace is a subspace with good clustering **and** interest above a user-defined threshold ε.

The ENCLUS Algorithm (cont.)

ENCLUS seeks for subspaces X^k with:

- High coverage (high percentage of points covered by all dense units in X^k).
- High density of points in the dense units in the subspace.
- High correlation among the dimensions of the subspace.

Remarks:

- The above requirements are indicative of a subspace with nonrandom structure.
- Also, a subspace with a strong clustering structure has lower entropy than a subspace where data do not show any clustering tendency.

Notation

B_k: the set of significant k-dimensional subspaces.

D_k: the set of subspaces that have good clustering but they are of low interest.

A_1: the set of all one-dimensional subspaces

- The ENCLUS Algorithm (cont.)

2. *Determination of significant subspaces*

- Based on A_1, determine B_1 and D_1.
- k=1
- Repeat
  - Determine A_{k+1} as the set containing all the $(k+1)$-dimensional subspaces E such that
    - E is the result of the union of two k-dimensional subspaces in D_k sharing $(k\text{-}1)$ dimensions and
    - All k-dimensional projections of E belong to D_k
  - k=k+1
- Until A_k=$\varnothing$
- Return the significant subspaces $B_1 \cup \ldots \cup B_{k\text{-}1}$.

The rest stages of the algorithm are applied on the significant subspaces extracted above.

The ENCLUS Algorithm (cont.)

Remarks:

- ENCLUS is insensitive to the order of consideration of the data.
- It can unravel arbitrarily shaped clusters.
- The computational time required by ENCLUS scales linearly with N.
- There is a significant overlap among the reported clusters.
- ENCLUS depends heavily on the choice of the edge of the grid as well as on the parameters ω and ε.
- Other Grid-based subspace clustering algorithms are
  - MAFIA
  - Cell-based clustering method (CBF)
  - CLTree algorithm.

❖ Subspace Clustering Approach (cont.)

Point-based Subspace Clustering Algorithms (PBSCA)

In these algorithms

- The clusters are defined in terms of data points of X.
- Each data point contribute to a single cluster.
- The clusters as well as the subspaces in which they live are simultaneously determined in an iterative fashion.

Typical PBSCAs

- PROCLUS
- ORCLUS

- The PROCLUS algorithm
  - Its main idea is to generate an initial set of medoids and to iterate until an "optimum" set of medoids results, estimating at the same time the subspace where each cluster resides.
  - Inputs of the algorithm: number of clusters, m, average dimensionality, s.
  - It consists of three stages: the initialization stage, the iterative stage and the refinement stage.

The PROCLUS algorithm (cont.)

1. *Initialization stage* (a, b are user-defined parameters).

- Generate a sample X' of size am from X, via random selection.
- Generate, via random selection, a subset X'' of X' of size bm ($b<a$) such that each of the points in X'' lies as far as possible from the other points in X''.
- Generate, via random selection, the subset Θ of X'' of size m and let I_{Θ} denoting the corresponding index set.
- The elements of Θ are taken as the initial estimates of the medoids of the m clusters.

The PROCLUS algorithm (cont.)

2. *Iterative stage*

- Set $cost=\infty$
- $iter=0$
- Repeat
  - $iter=iter+1$
  - (A) For each $i \in I_{\Theta}$ determine the set of dimensions D_i of the subspace where the cluster C_i lives.
  - (B) For each $i \in I_{\Theta}$ determine the corresponding cluster C_i.
  - (C) Compute the cost $J(\Theta)$ associated with Θ.
  - If $J(\Theta) < cost$ then
    - $\Theta_{best}=\Theta$
    - $cost=J(\Theta_{best})$
  - End if
  - (D) Determine the "bad" medoids of Θ_{best}.
  - Set $\Theta=\Theta_{best}$ and replace its bad medoids with randomly selected points from X''.
- Until a termination condition is satisfied.

Iterative stage (cont.)

(A) Determination of cluster subspaces

- For each point $\underline{x}_i$, $i \in I_\Theta$,
  - the set of points L that consists of points in X that lie in a sphere around $\underline{x}_i$ of appropriate radius is determined.
  - The "concentration" of the distances between $\underline{x}_i$ and each $\underline{x} \in L$ along each direction is measured.
- Among the total number of ml dimensions, the ms dimensions with the lowest concentration are selected for the identification of the subspaces D_is that correspond to the clusters C_is

(B) Determination of the clusters

For each point $\underline{x} \in X$

- Its one-dimensional distances from each $\underline{x}_i$, $i \in I_\Theta$, along each dimension of D_i are computed and their mean, denoted by $dist_i$ is determined.
- $\underline{x}$ is assigned to the cluster with the minimum $dist_i$.

Iterative stage (cont.)

(C) Computation of $J(\Theta)$

- For each $\underline{x} \in X$ the lowest $dist_i$, denoted by $dist$, is computed.
- $J(\Theta)$ is computed as the average of these $dist$ s.

(D) Determination of "bad" medoids

A medoid is considered "bad" if either

(a) Its corresponding cluster has the least number of points or,

(b) its corresponding cluster has less than $(N/m)q$ points, where q is a user-defined constant (typically $q=0.1$).

3. *Refinement stage*

After the completion of the *iteration stage*,

- D_i s are recomputed based on more precise information.
- C_i s are recomputed based on the above defined D_i s.

- Remarks:
  - PROCLUS is biased toward hyperspherically shaped clusters.
  - Cluster subspaces must be of similar size (since the average dimensionality is an input to the algorithm).
  - The Initialization stage is crucial since it is desirable to obtain representative points from all physical clusters.
  - PROCLUS is somewhat faster than CLIQUE on large data sets.
  - Its required computational effort increases linearly with the dimension of the feature space l.

Point-based Subspace Clustering Algorithms (cont.)

- The ORCLUS algorithm

Key points

- This is a point based SCA of agglomerative hierarchical nature.
- At each iteration
  - the number of clusters decreases from an initial value m_0 to a final user-defined value m.
  - the dimensionality decreases from l_0 (the dimensionality of the feature space) to a final user-defined value l.
- The reduction in the number of clusters as well as in the dimensionality must be carried out in the same number of iterations.
- The subspace where each cluster lies is represented by vectors that are not necessarily parallel to the axes of the original feature space.

- Remarks:
  - ORCLUS is biased toward hyperspherical clusters.
  - The required computational time is $O(m_0^3+m_0Nl_0+m_0^2l_0^3)$. That is,
    - It increases linearly with N.
    - It increases cubically with l_0.
  - Computational time may be reduced by adopting random sampling techniques.
  - Increased values of m_0 may improve the quality of the final clustering.
  - The subspaces of all clusters are restricted to the same dimensionality.

Point-based Subspace Clustering Algorithms (cont.)

- Remarks (GBSCAs vs PBSCAs):
  - In GBSCAs, all clusters are represented as unions of dense units (rough description), while in PBSCAs they are represented in terms of data points (exact description).
  - In GBSCAs a point may contribute to more than one cluster in different subspaces through its projections, while in PBSCAs each point contributes to a single cluster.
  - In GBSCAs the identification of clusters is carried out only after the determination of the appropriate subspaces. In PBSCAs the clusters as well as the appropriate subspaces are simultaneously determined in an iterative fashion.
  - GBSCAs are able, in principle, to unravel arbitrarily shaped clusters, while PBSCAs are biased toward hyperspherically-based clusters.
  - The computational time required by most of the GBSCAs and PBSCAs scales linearly with N, the number of points.
  - The computational time required by the above GBSCAs increases exponentially with the dimensionality of the input space, l, whereas in PBSCAs it exhibits a polynomial dependence.

- Remarks (GBSCAs vs PBSCAs) (cont.):
  - In GBSCAs there are no restrictions concerning the dimensionality of the subspaces, whereas the PBSCAs pose constraints on it.
  - In GBSCAs there exists a large overlap in the resulting clusters. On the contrary, most of the PBSCAs produce disjoint clusters.
  - Both GBSCAs and PBSCAs are sensitive to the choice of the involved user-defined parameters.

MISCELLANEOUS CLUSTERING ALGORITHMS

Clustering algorithms based on a variety of ideas not included in the above main categories have also been developed. Such algorithms are:

- Algorithms based on the so-called tabu search method, where the next state of the algorithm is selected from a set of candidate clusterings resulting from the current state of the algorithm.
- Algorithms inspired by physical laws.
- Algorithms that combine ideas from fuzzy clustering and agglomerative algorithms.
- Algorithms based on conceptual distances.
- Algorithms that employ graph theory-based concepts in a probabilistic framework.
- Algorithms that combine data partitions resulting from different clustering algorithms.
- Algorithms that state the clustering problem in information theoretic terms, such as entropy.

- Algorithms that employ the wavelet transform (for large data sets with low dimensionality).
- Algorithms that impose constraints on the data points or on specific parameters.
- Algorithms using Hidden Markov Models (HMMs).