Clustering algorithms

Konstantinos Koutroumbas

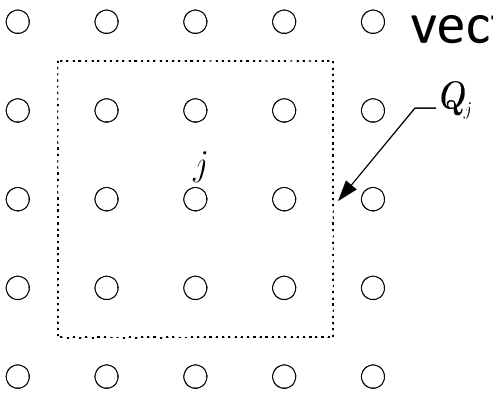
Unit 10

– SOM, LVQ
– Valley seeking clust. Algorithms
– Branch & bound clustering algorithms
– Simulated & Deterministic annealing
– Genetic algorithms
– Density-based algorithms for large data sets (DBSCAN)

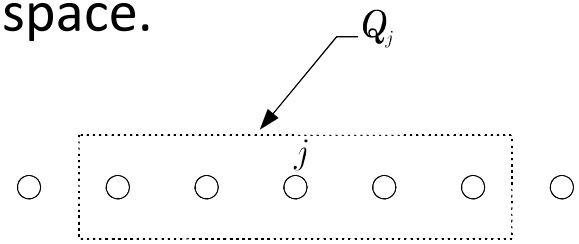
koutroum@noa.gr

Self-organizing maps

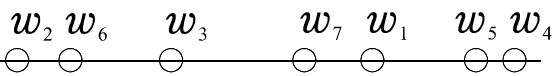
- It is **used** for data visualization (maps high dim. Data→ 1-d or 2-d maps) and ("loose") clustering.
- Here interrelation between representatives is assumed.
- For each representative $\boldsymbol{w}_j$ a topological neighborhood of representatives $Q_j(t)$ is defined, centered at $\boldsymbol{w}_j$.
- As t (no. of iterations) increases, $Q_j(t)$ shrinks and concentrates around $\boldsymbol{w}_j$.
- The neighborhood is defined with respect to the indices j and it is independent of the geometrical distances between representatives in the vector space.



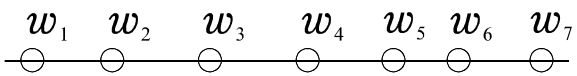
(a)



(b)



(c)



(d)

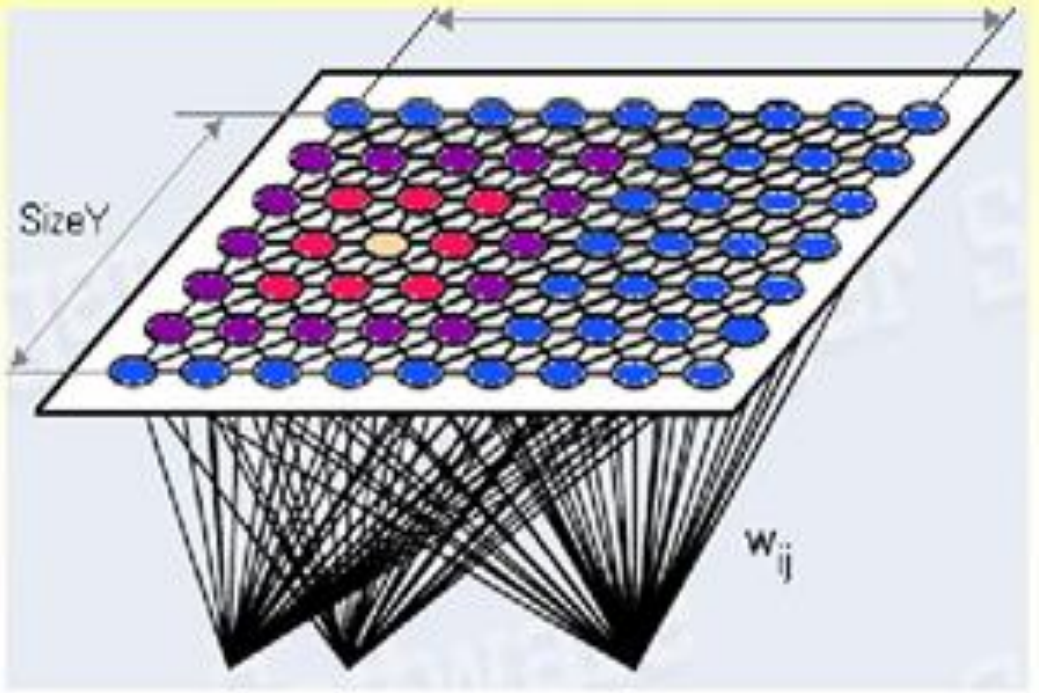
Assuming that in fig. (c) the neighborhood of $\boldsymbol{w}_j$ **constitutes** of $\boldsymbol{w}_{j-1}$ and $\boldsymbol{w}_{j+1}$,

$\boldsymbol{w}_2$ and $\boldsymbol{w}_4$ are the **topo-logical neighbors** of $\boldsymbol{w}_3$ **although**

$\boldsymbol{w}_6$ and $\boldsymbol{w}_7$ are closer in terms of the geometrical distance to $\boldsymbol{w}_3$)

Self-organizing maps

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



Self-organizing maps

- If $\boldsymbol{w}_j$ wins on the current input $\boldsymbol{x}$ 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 $\boldsymbol{w}_j$ wins on $\boldsymbol{x}$, the updating equation becomes:

$$\boldsymbol{w}_k(t) = \begin{cases} \boldsymbol{w}_k(t-1) + {\eta_t}^{k,j}\big(\boldsymbol{x} - \boldsymbol{w}_k(t-1)\big), & if\ \boldsymbol{w}_k \in Q_j(t) \\ \boldsymbol{w}_k(t-1), & otherwise \end{cases}$$

  where ${\eta_t}^{k,j}$ is a *variable* learning rate, which decreases with t and with the topological distance between the k-th and the j-th representatives.

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

Self-organizing maps

The algorithm

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

$$d\big(\boldsymbol{x}, \boldsymbol{w}_j(t-1)\big) = \min_{k=1,\dots,m} d(\boldsymbol{x}, \boldsymbol{w}_k(t-1))$$

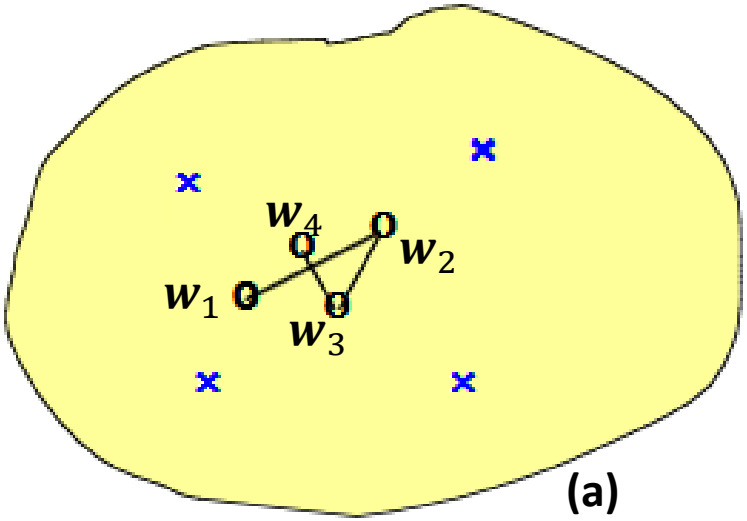
  - (D) *Parameter updating*

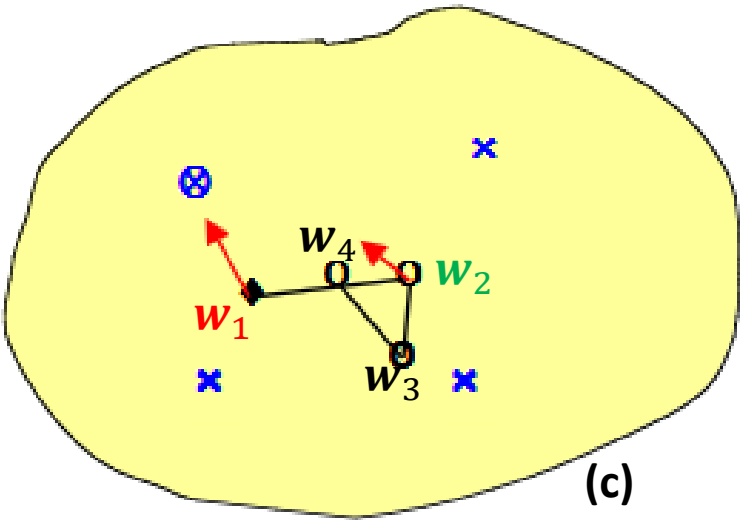
$$\boldsymbol{w}_k(t) = \begin{cases} \boldsymbol{w}_k(t-1) + \eta_t{}^{k,j}\big(\boldsymbol{x} - \boldsymbol{w}_k(t-1)\big), & if\ \boldsymbol{w}_k \in Q_j(t) \\ \boldsymbol{w}_k(t-1), & otherwise \end{cases}$$

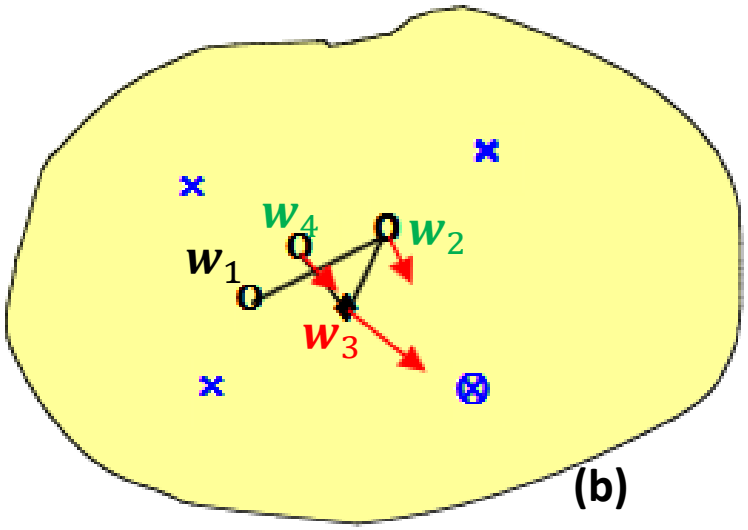
  - End
- (E) **Until** (convergence occurred) OR $(t > t_{max})$

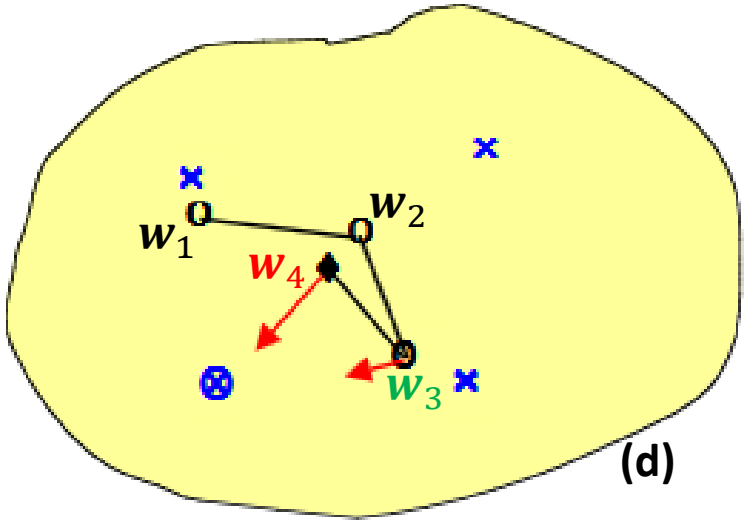
Self-organizing maps

Example





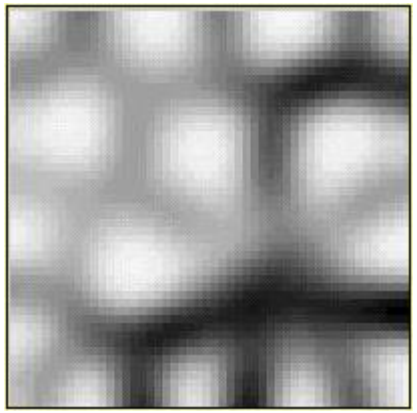




Self-organizing maps

How to represent the result of a SOM

NOTE: After SOM convergence the topological ordering of the m **representatives** will comply with their "geometrical ordering".



Produce an **image** A of size

- m for the 1-d case or
- $k \times k$ for the 2-d case ($m = k^2$)

As follows

For **each representative** (pixel of A):

Compute its **average distance** d_{avg} from its neighboring representatives

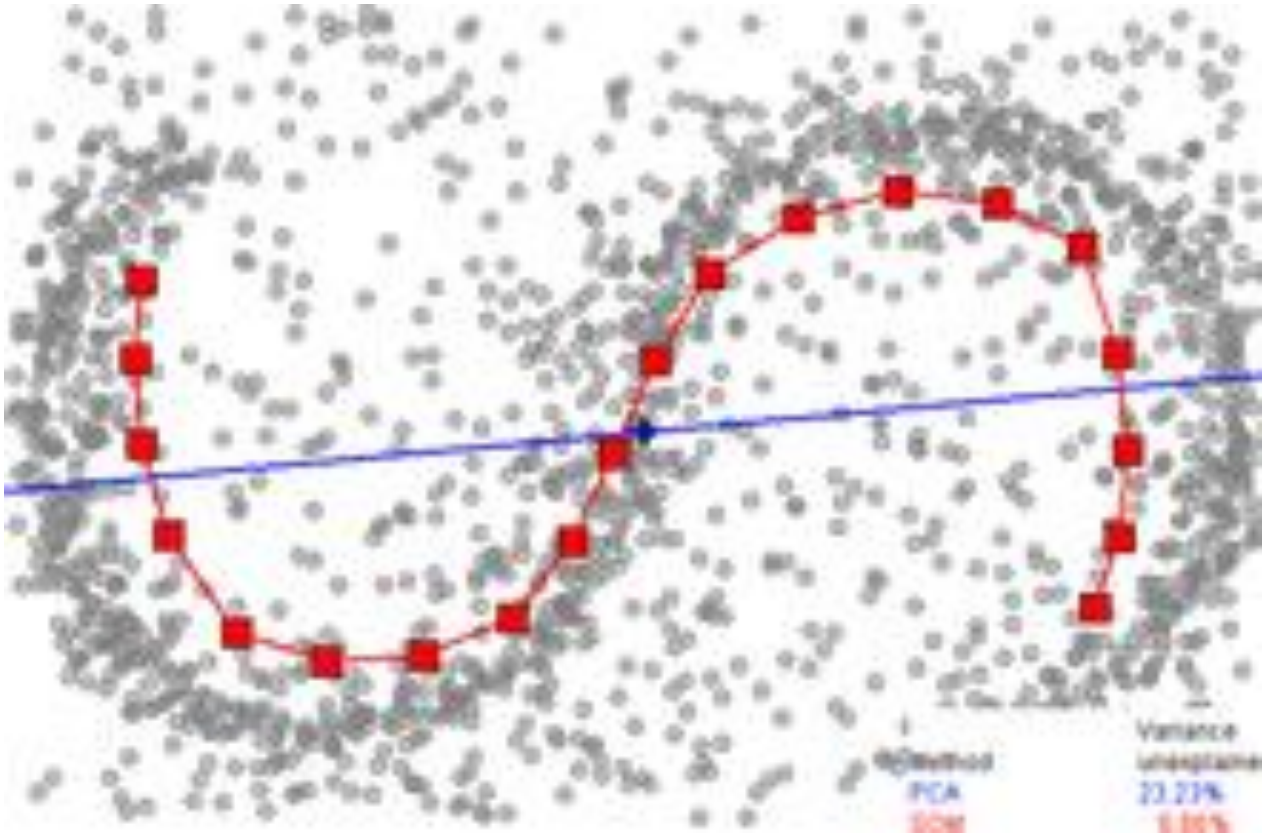
Draw the associate pixel of A with a color so that:

The larger the d_{avg}, the darker the color will be.

Then **lighter areas surrounded** by **darker areas** in A are indicative of clustering structure in the data.

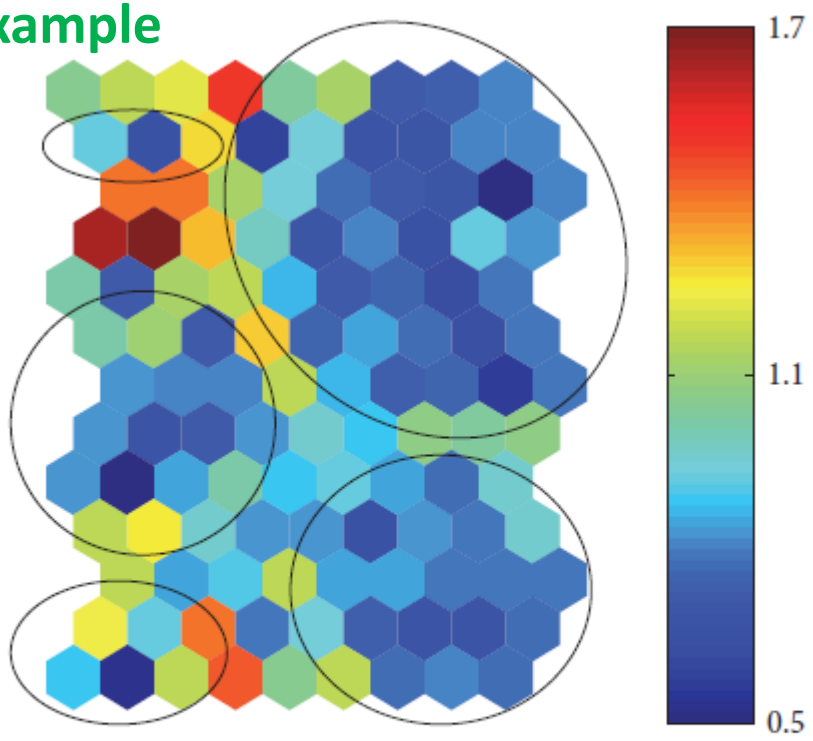
Self-organizing maps

Example



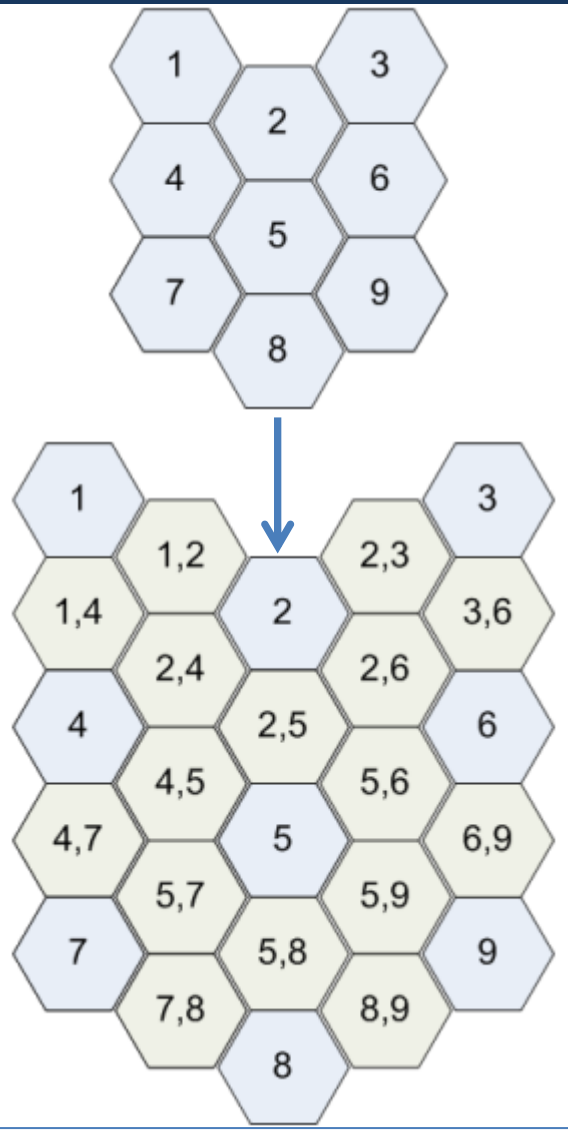
Self-organizing maps

Example



U-matrix of the 54 ERCs texts preprocessed through the VSM. Color scale is related to distances between map units. Red colors represent large distances and blue colors represent small distances.

From: Massimo Pacella, Antonio Grieco, Marzia Blaco, "On the Use of Self-Organizing Map for Text Clustering in Engineering Change Process Analysis: A Case Study", *Computational Intelligence and Neuroscience*, vol. 2016, Article ID 5139574, 11 pages, 2016. https://doi.org/10.1155/2016/5139574



Computation of the numbers in U-matrix via example:

$$\{2,3\} \rightarrow d(w_2, w_3)$$

$$\{3\} \rightarrow mean\{d(w_2, w_3), d(w_3, w_6)\}$$

Supervised Learning Vector Quantization (VQ)

In this case

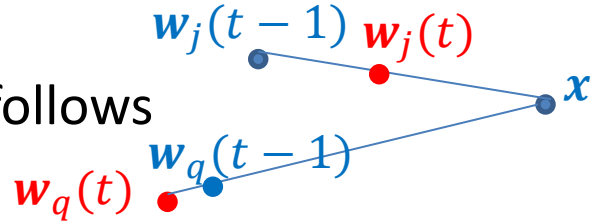
- each cluster is **treated** as a class (m compact classes are assumed)
- the available vectors have known class labels.

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 $\boldsymbol{w}_j$' s is carried out as follows



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

Supervised Learning Vector Quantization (VQ)

The algorithm

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

$$d\big(\boldsymbol{x}, \boldsymbol{w}_j(t-1)\big) = \min_{k=1,\dots,m} d(\boldsymbol{x}, \boldsymbol{w}_k(t-1))$$

  - (D) *Parameter updating*

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

- (E) **Until** (convergence occurred) OR $(t > t_{max})$ (max allowable no of iter.)

In words:

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

Valley seeking clustering algorithms

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

- Clusters may be **viewed** as peaks of $p(\boldsymbol{x})$ **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 $d(\boldsymbol{x}, \boldsymbol{y})$ be defined as

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

where A is a positive definite matrix

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

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

where a is a user-defined parameter

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

Valley seeking clustering algorithms

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** $\boldsymbol{x}_i$ **to** cluster C_{c_i}.
  - End For
- **Until** no reclustering of vectors occurs.

Valley seeking clustering algorithms

The algorithm

- **Centers** a window defined by $d(\boldsymbol{x}, \boldsymbol{y}) \leq a$ at $\boldsymbol{x}$ and **counts** the points from different clusters in it.
- **Assigns** $\boldsymbol{x}$ 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.

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 become empty*).

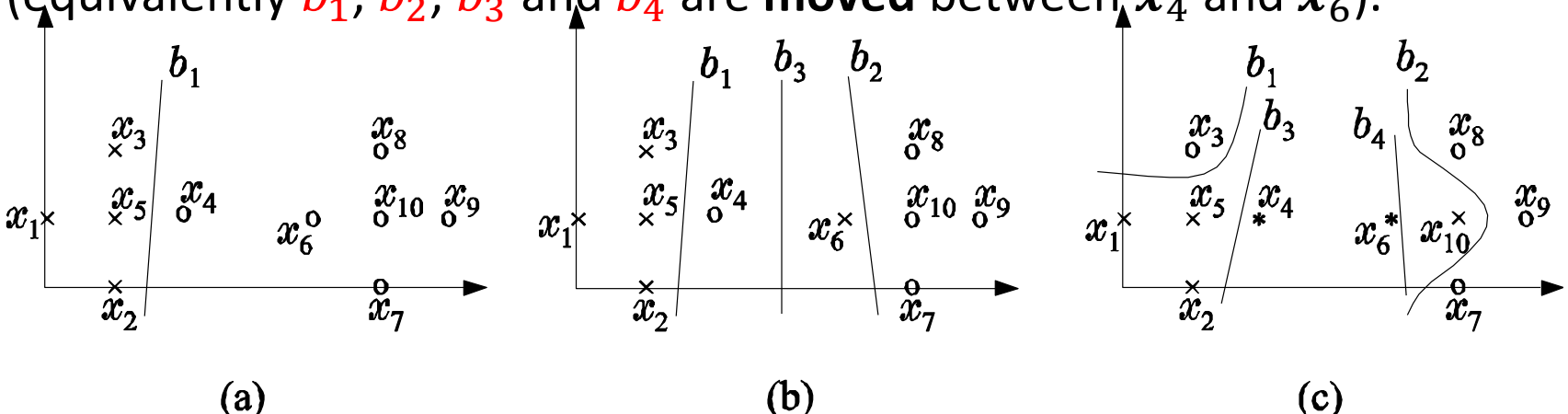
Valley seeking clustering algorithms

Example: Let $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{10}\}$ and $a = 1.1415\ (> \sqrt{2})$. X contains two physical clusters: $C_1 = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_5\}$, $C_2 = \{\boldsymbol{x}_6, \ldots, \boldsymbol{x}_{10}\}$.

(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 $\boldsymbol{x}_4$ and $\boldsymbol{x}_6$).

(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 $\boldsymbol{x}_4$ and $\boldsymbol{x}_6$).



Branch and Bound Clustering algorithms

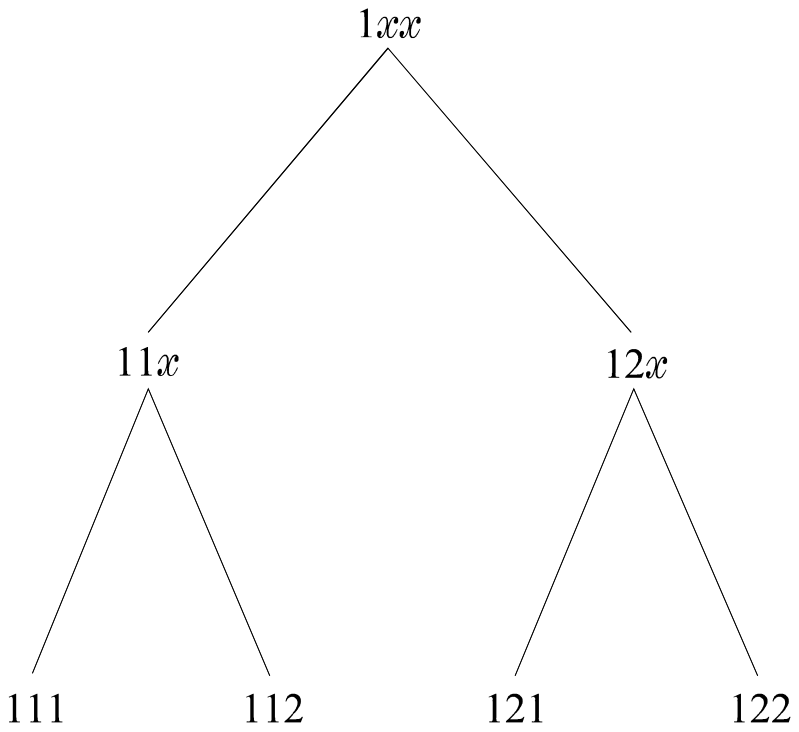
- They compute the globally optimal solution to combinatorial problems.
- They *avoid exhaustive search* via the employment of a monotonic criterion J.

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: 1st, 3rd vectors belong to class 1
2nd vector belongs to class 2.
(leaf of the tree)

$12x$: 1st vector belongs to class 1
2nd vector belongs to class 2
3rd 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 $\boldsymbol{C}_r = [c_1, \ldots, c_r]$, $1 \leq r \leq N$, denotes a partial clustering where $c_i \in \{1,2, \ldots, m\}$, $c_i = j$ if the vector $\boldsymbol{x}_i$ belongs to cluster C_j and $\boldsymbol{x}_{r+1}, \ldots, \boldsymbol{x}_N$ are yet unassigned.
- For compact clusters and fixed number of clusters, m, a suitable cost function is

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

where $\boldsymbol{m}_{c_i}$ is the mean vector of the cluster C_{c_i}

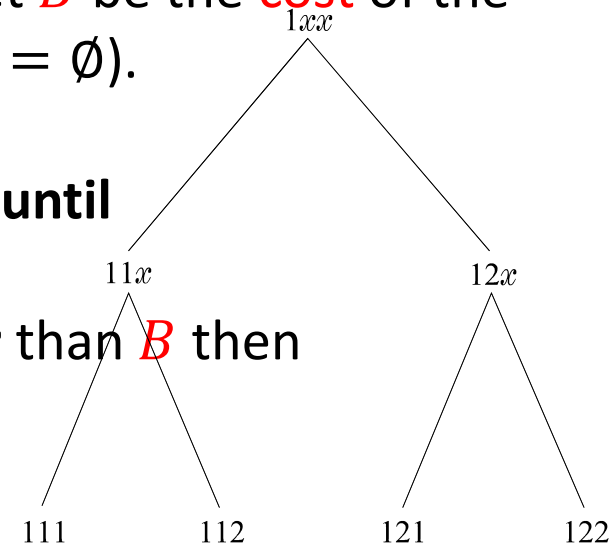
$$\boldsymbol{m}_j(\boldsymbol{C}_r) = \frac{1}{n_j(\boldsymbol{C}_r)} \sum_{\{q=1,\ldots,r, c_q=j\}} \boldsymbol{x}_q\,, \qquad j = 1, \ldots, m$$

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

Branch and Bound Clustering algorithms

Initialization

- **Start** from the initial node and go down to a leaf. Let B be the cost of the corresponding clustering $\boldsymbol{C}$ (initially set $B = +\infty$, $\boldsymbol{C} = \emptyset$).



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 $\boldsymbol{C}'$ is **smaller** than B then
      - $B = B'$
      - $\boldsymbol{C} = \boldsymbol{C}'$ 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

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 determination 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 spends at a given temperature so that the system can enter the "thermal equilibrium" in this temperature.

Notation

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

Simulated Annealing

The algorithm:

- **Set** $T = T_{max}$ and $\boldsymbol{C} = \boldsymbol{C}_{init}$.
- $t = 0$
- **Repeat**
  - $t = t + 1$
  - *Repeat*
    - **Compute** $J(\boldsymbol{C})$
    - **Produce** a new clustering, $\boldsymbol{C}'$, by **assigning** a randomly chosen vector from X to a different cluster.
    - **Compute** $J(\boldsymbol{C}')$
    - **If** $\Delta J = J(\boldsymbol{C}') - J(\boldsymbol{C}) < 0$ then
      - (A) $\boldsymbol{C} = \boldsymbol{C}'$
    - Else
      - (B) $\boldsymbol{C} = \boldsymbol{C}'$, 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

Remarks:

- For $T \to \infty$, it is $p(\Delta J) \approx 1$. 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, $\boldsymbol{C}$ remains unchanged."
- A schedule for lowering T that guarantees convergence to the global minimum with probability 1, is

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

- The method is computationally demanding.

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 $\boldsymbol{w}_j$, $j = 1, \ldots, m$ (m is fixed) in appropriate positions so that a distortion function J is minimized.

  J is defined as

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

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

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

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

where

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

- P_{ir} may be **interpreted** as the probability that $\boldsymbol{x}_i$ **belongs** to C_r, $r = 1, \ldots, m$.

Deterministic Annealing

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

Stages of the algorithm

- For $\beta \longrightarrow 0$, **all** P_{ij}'s are almost equal to $\frac{1}{m}$, for all $\boldsymbol{x}_i$'s, $i = 1, \ldots, N$. Thus

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

Since $d(\boldsymbol{x}, \boldsymbol{w})$ is a convex function, $d(\boldsymbol{x}_1, \boldsymbol{w}_r) + \cdots + d(\boldsymbol{x}_N, \boldsymbol{w}_r)$ 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}'s "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 $\boldsymbol{x}_i$, $P_{ir} \approx 1$ for a specific r, and $P_{ij} \approx 0$, for $j \neq r$).

Deterministic Annealing

Application: For the squared Euclidean distance $d(\boldsymbol{x}, \boldsymbol{w}) = (\boldsymbol{x} - \boldsymbol{w})^T(\boldsymbol{x} - \boldsymbol{w})$ it is

$$\frac{\partial J}{\partial \boldsymbol{w}_r} = \sum_{i=1}^{N} P_{ir} \frac{\partial d(\boldsymbol{x}_i, \boldsymbol{w}_r)}{\partial \boldsymbol{w}_r} = 2\sum_{i=1}^{N} P_{ir}(\boldsymbol{x}_i - \boldsymbol{w}_r) = 0 \Leftrightarrow \boldsymbol{w}_r = \frac{\sum_{i=1}^{N} P_{ir}\boldsymbol{x}_i}{\sum_{i=1}^{N} P_{ir}}$$

This is **coupled** wrt $\boldsymbol{w}_r$

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 on this, so that the new population **of the same size** is improved compared to the previous one (wrt a *criterion function* F).
- 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.
- A simple implementation:
  - For each solution s_i, out of the population of the p solutions, compute the associated criterion function value $F(s_i)$.
    (it is assumed that the **higher** the value of F, the **better** the solution)
  - Assign to each s_i a probability $p_i = F(s_i) / \sum_{j=1}^{p} F(s_j)$.
  - Perform sampling with replacement to produce the next solution population.

Clustering using genetic algorithms (GA)

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.

Mutation:

- It applies to the temporary population produced after the application of the crossover operator.
- 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 two solutions are **selected** for crossover.

The probability with which an element of a solution is **mutated**.

Clustering using genetic algorithms (GA)

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 (GA)

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.

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

The cost function in use is

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

The *criterion function* can be defined e.g., as $F(s_i) = e^{-J(s_i)}$

where

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

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 (GA)

Remark:

- An alternative to the above scheme results if prior to the application of the reproduction operator, the hard clustering algorithm (GHAS), described in a previous lecture, **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 (however, difficulties may arise in the case where the clusters differ in terms of their density).

- **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.

Density-based algorithms for large data sets

Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

Algorithm

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

Notation

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

Definitions

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

Density-based algorithms for large data sets

DBSCAN Algorithm (cont.)

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

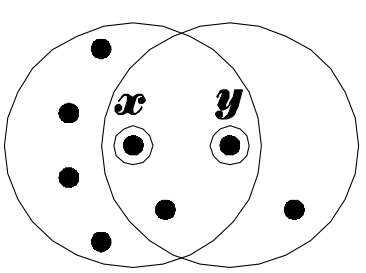
Example:

Assuming that $q = 5$,

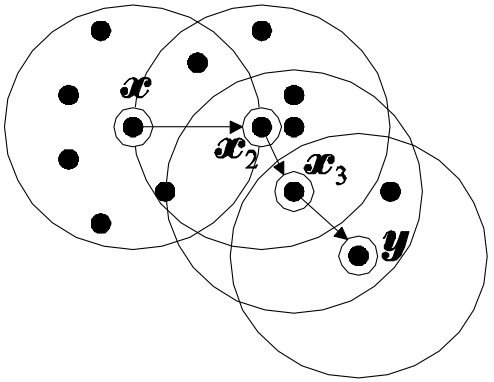
(a) $\boldsymbol{y}$ is directly density reachable from x, but not vice versa,

(b) $\boldsymbol{y}$ is density reachable from x, but not vice versa, and

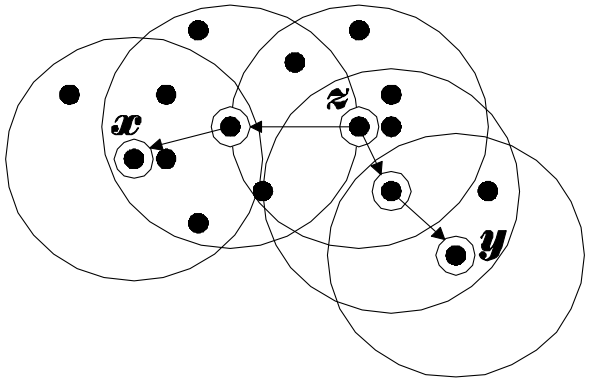
(c) $\boldsymbol{x}$ and $\boldsymbol{y}$ are density connected (in addition, $\boldsymbol{y}$ is density reachable from $\boldsymbol{x}$, but not vice versa).



(a)



(b)



(c)

Density-based algorithms for large data sets

DBSCAN Algorithm (cont.)

4. A cluster C in DBSCAN is defined as a nonempty subset of X satisfying the following conditions:
   - *If* $\boldsymbol{x}$ **belongs** to C and $\boldsymbol{y} \in X$ is density reachable from $\boldsymbol{x}$, *then* $\boldsymbol{y} \in C$.
   - For each pair $(\boldsymbol{x}, \boldsymbol{y}) \in C$, $\boldsymbol{x}$ and $\boldsymbol{y}$ 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 $\boldsymbol{x}$ 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 noisy point (that is, not density reachable from other points in X).

Density-based algorithms for large data sets

DBSCAN Algorithm (cont.)

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

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

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.

Density-based algorithms for large data sets

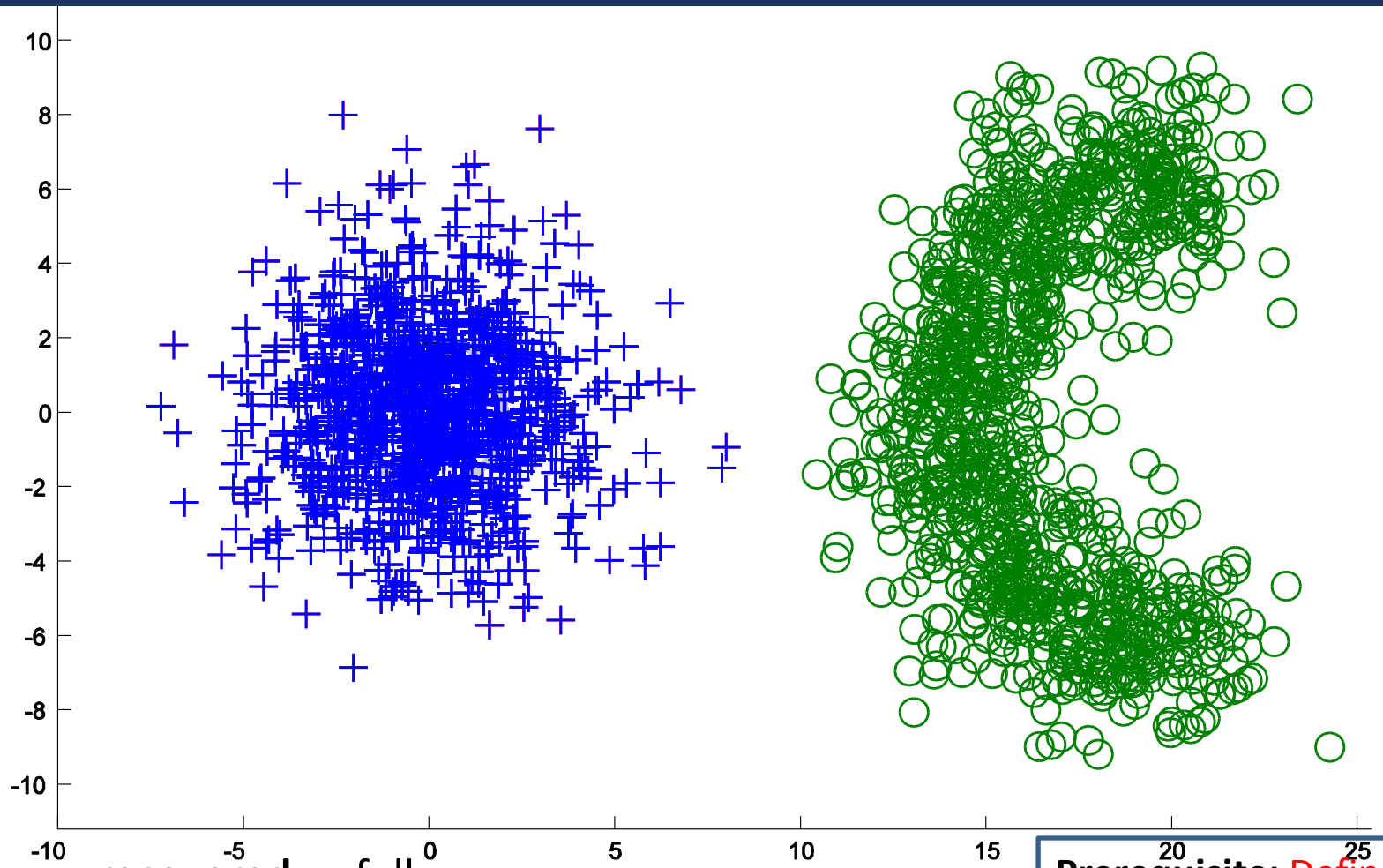
DBSCAN Algorithm (cont.)

DBSCAN Algorithm

- **Set** $X_{un} = X$
- **Set** $m = 0$
- **While** $X_{un} \neq \emptyset$ do
  - Arbitrarily **select** a $\boldsymbol{x} \in X_{un}$
  - *If* $\boldsymbol{x}$ is a noncore point then
    - **Mark** $\boldsymbol{x}$ as noise point
    - $X_{un} = X_{un} - \{\boldsymbol{x}\}$
  - *End if*

  - *If* $\boldsymbol{x}$ is a core point *then*
    - $m = m + 1$
    - **Determine all** density-reachable points $\boldsymbol{y} \in X$ from $\boldsymbol{x}$.
    - **Assign** $\boldsymbol{x}$ and the previous points to the cluster C_m. The border points among them that may have been **marked** as "noise" are also **assigned** to C_m.
    - $X_{un} = X_{un} - C_m$
  - End {if}
- **End** {while}

Clustering – Density-based algorithms



Clusters are **recovered** as follows:

- **Start** a new cluster C by **choosing** a data point $\boldsymbol{x}$.
- **Assign** all the data points that lie in the neighborhood of $\boldsymbol{x}$ to the same cluster.
- **Repeat** recursively the previous step **until** all neighboring points of ALL $\boldsymbol{x} \in C$ are **assigned** to C.

Prerequisite: Definition of the neighborhood size

Density-based algorithms for large data sets

DBSCAN Algorithm (cont.)

Important notes:

- If a border point $\boldsymbol{y}$ of a cluster C is selected, it will be **marked initially** as a noise point. However, ***when*** **(a)** a core point $\boldsymbol{x}$ in C is selected later on, and **(b)** $\boldsymbol{y}$ is identified as a density-reachable point from $\boldsymbol{x}$ ***then*** $\boldsymbol{y}$ will assigned to C.
- If an actual noise point $\boldsymbol{y}$ 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 $O(N \log_2 N)$ 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.