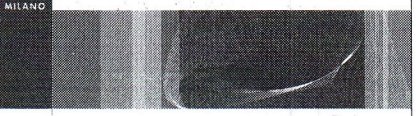
POLITECNICO DI MILANO



# Clustering

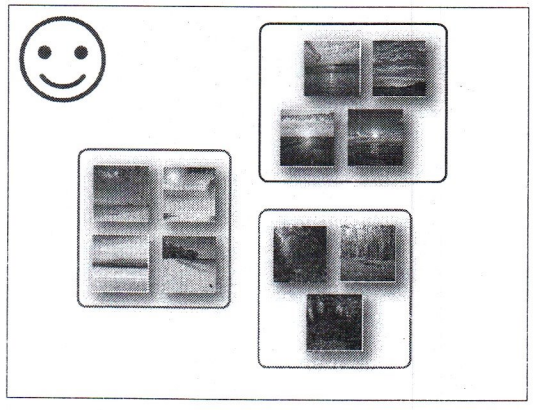
Data Mining and Text Mining

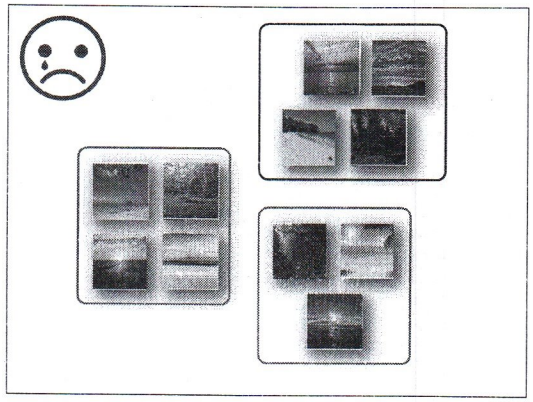
can be applied to **any type of data** (we don't need, for instance, the notion of "transaction" as we did with associations rules)

Prof. Pierluca Lanzi

Clustering searches for "natural" grouping/structure in un-labeled data

there is **no** target





Clustering algorithms group a collection of data points into "clusters" according to some distance measure

Data points in the same cluster should have a small distance from one another

Data points in different clusters should be at a large distance from one another

low distance = high similarity

## Clustering Applications



- **Marketing**
  - Help marketers discover distinct groups in their customer bases, and then use this knowledge to develop targeted marketing programs
- **Land use**
  - Identification of areas of similar land use in an earth observation database
- **Insurance**
  - Identifying groups of motor insurance policy holders with a high average claim cost
- **City-planning**
  - Identifying groups of houses according to their house type, value, and geographical location
- **Earth-quake studies**
  - Observed earthquake epicenters should be clustered along continent faults



## What is Cluster Analysis?



- **A cluster is a collection of data objects**
  - Similar to one another within the same cluster
  - Dissimilar to the objects in other clusters
- **Cluster analysis**
  - Given a set data points try to understand their structure
  - Finds similarities between data according to the characteristics found in the data
  - Groups similar data objects into clusters
  - It is unsupervised learning since there is no predefined classes



## What Is Good Clustering?



- A good clustering consists of high-quality clusters with
  - High intra-class similarity
  - Low inter-class similarity
- The quality of a clustering result depends on both
  - The similarity measure used by the method and
  - Its implementation (the algorithms used to find the clusters)
- The quality of a clustering method is also measured by its ability to discover some or all the hidden patterns
- Evaluation
  - Various measure of intra/inter cluster similarity
  - Manual inspection
  - Benchmarking on existing labels



different similarity measures give different results → influences a lot the quality of the clustering

also the implementation matters!

we can consider a labeled data (e.g. iris dataset), we remove the labels and we perform the clustering. if we obtain clusters similar to the existing labels then we can conclude that labels reflect an (underlying) existing structure of the data

## Measure the Quality of Clustering



- **Dissimilarity/Similarity metric**
  - Similarity expressed in terms of distance function, typically a metric, d(i, j)
  - Definitions of distance functions are usually very different for interval-scaled, Boolean, categorical, ordinal ratio, and vector variables
  - Weights can/should be associated with different variables based on applications and data semantics
- **Cluster quality measure**
  - Separate from distance, there is a "quality" function that measures the "goodness" of a cluster
  - It is hard to define "similar enough" or "good enough" as the answer is typically highly subjective

"There are 2 major groups, in each there are 3 sub-groups, in one of the 3 there are n subsubgroups, .."

## Clustering Strategies

11

- Hierarchical vs point assignment
- Numeric and/or symbolic data
- Deterministic vs. probabilistic
- Exclusive vs. overlapping
- Hierarchical vs. flat
- Top-down vs. bottom-up

"This is a good customer" vs. "This is a good customer at 80%"

we can start from all the data and then divide or we can start from one datum and then aggregate

Prof. Pierluca Lanzi POLITECNICO DI MILANO

## Data Structures

12

| Outlook | Temp | Humidity | Windy | Play |
|---|---|---|---|---|
| Sunny | Hot | High | False | No |
| Sunny | Hot | High | True | No |
| Overcast | Hot | High | False | Yes |
| ... | ... | ... | ... | ... |

Data Matrix

$$\begin{bmatrix} x_{11} & \cdots & x_{1f} & \cdots & x_{1p} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{i1} & \cdots & x_{if} & \cdots & x_{ip} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{n1} & \cdots & x_{nf} & \cdots & x_{np} \end{bmatrix}$$

Dis/Similarity Matrix

squared matrix $n \times n$ ($n$ = number of data)

$$\begin{bmatrix} 0 & & & \\ d(2,1) & 0 & & \\ d(3,1) & d(3,2) & 0 & \\ \vdots & \vdots & \vdots & \\ d(n,1) & d(n,2) & \cdots & \cdots & 0 \end{bmatrix}$$

this for instance is the distance between the observation 2 and the observation 3

if the distance is 0 it's because we're looking at the distance between an element and itself

Prof. Pierluca Lanzi POLITECNICO DI MILANO

## Distance and Similarity Measures

independently on the algorithm we're going to use a distance/similarity measure (in fact the dist/sim. is almost ⊥ to the algorithm)

## Distance Measures

14

- Given a space and a set of points on this space, a distance measure d(x,y) maps two points x and y to a real number, and satisfies three axioms

1. d(x,y) ≥ 0
2. d(x,y) = 0 if and only x=y
3. d(x,y) = d(y,x)
4. d(x,y) ≤ d(x,z) + d(z,y)

Prof. Pierluca Lanzi POLITECNICO DI MILANO

## What Similarity Measure?

15

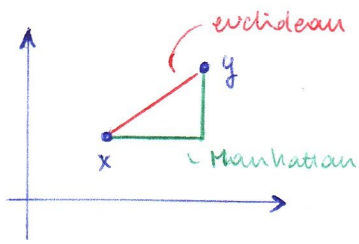
- Euclidian distance is the typical function used to compute the similarity between two examples

$$d([x_1, x_2, \ldots, x_n], [y_1, y_2, \ldots, y_n]) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

- Another popular metric is city-block (Manhattan) metric, distance is the sum of absolute differences

$$d([x_1, x_2, \ldots, x_n], [y_1, y_2, \ldots, y_n]) = \sum_{i=1}^{n} |x_i - y_i|$$

Prof. Pierluca Lanzi POLITECNICO DI MILANO



## Jaccard Distance

16

- Jaccard distance is a measure of how dissimilar two sets are. Examples are represented by binary variables and are viewed as representations of set
- Jaccard distance is defined as

$$d(x, y) = 1 - J(x, y)$$

- J is the Jaccard similarity which is computed as the number of .

$$J(x, y) = \frac{|x \cap y|}{|x \cup y|}$$

- Which can also be interpreted as the percentage of identical attributes

Prof. Pierluca Lanzi POLITECNICO DI MILANO

we're considering boolean vectors (or itemsets) and we're making intersection over union without considering multiple copies of elements (actually they don't have to be boolean)

Example:
person 1 buy: {2A, B, 3C}
person 2 buy: {A, 3C, D}
$|x \cap y| = |\{3C\}| = 1$
$|x \cup y| = |\{A|2A, B, 3C, D\}| = 4$
$J(x,y) = \frac{1}{4} \rightarrow d(x,y) = \frac{3}{4}$

However JACCARD IS TYPICALLY USED FOR BINARY DATA!

## Examples of Jaccard Distance

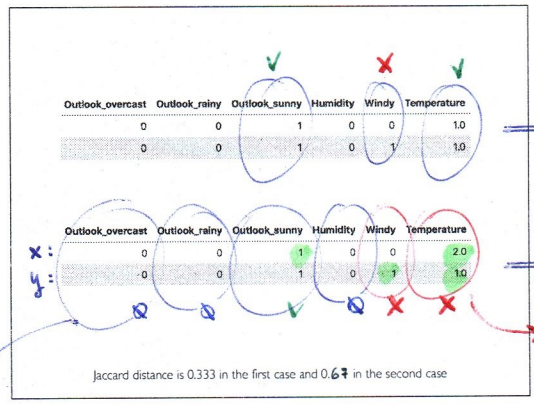
17

- Consider the following example consisting of two examples

| Outlook_overcast | Outlook_rainy | Outlook_sunny | Humidity | Windy |
|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |

- They are at Jaccard distance of 0.5 since the first example "contains" only Outlook_sunny whereas the second example "contains" both Outlook_sunny and Windy so intersection has size one while union has size two. The Jaccard distance is 0.5 so the similarity is 1-0.5 that is 0.5

Prof. Pierluca Lanzi POLITECNICO DI MILANO




$\Rightarrow d(x,y) = 1 - J(x,y) = 1 - \frac{2}{3} = \frac{1}{3}$

$\Rightarrow d(x,y) = 1 - J(x,y) = 1 - \frac{1}{3} = \frac{2}{3}$

these are 2 values counted as 1 (+1 when we count the union) (+1 when we have 0/1)

we need values $\neq 0$ to count it in the union and we need a MATCH to count it in the intersection

Unlike the Hamming distance, Jaccard considers 0 as a NON EXISTING VALUE

## Hamming Distance

19

- Hamming distance between two vectors is the number of components in which they differ
- Or equivalently, given the number of variables p, and the number m of matching components, we define

$$d(x, y) = \frac{p - m}{p}$$

Prof. Pierluca Lanzi POLITECNICO DI MILANO

## Examples of Hamming Distance

20

- Consider the same example

| | Outlook_overcast | Outlook_rainy | Outlook_sunny | Humidity | Windy |
|---|---|---|---|---|---|
| x: | 0 | 0 | 1 | 0 | 0 |
| y: | 0 | 0 | 1 | 0 | 1 |

- They are at ~~Jaccard~~ Hamming distance of 0.2 since all their attribute values are the same except for Windy so they differ for one value over 5, that is 0.2

Prof. Pierluca Lanzi POLITECNICO DI MILANO

$\Rightarrow d(x,y) = \frac{1}{5} = 0.2$

| Outlook_overcast | Outlook_rainy | Outlook_sunny | Humidity | Windy | Temperature |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 1.0 |
| 0 | 0 | 1 | 0 | 1 | 1.0 |

| Outlook_overcast | Outlook_rainy | Outlook_sunny | Humidity | Windy | Temperature |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 2.0 |
| 0 | 0 | 1 | 0 | 1 | 1.0 |

Hamming distance is 1/6 in the first case and 1/3 in the second case.

very useful in text mining

## Cosine Distance

22

- The cosine distance between x, y is the angle that the vectors to those points make

$$d(x,y) = \arccos \frac{\sum_1^n x_i y_i}{\sqrt{\sum_i^n x_i^2}\sqrt{\sum_i^n y_i^2}}$$

- This angle will be in the range 0 to 180 degrees, regardless of how many dimensions the space has.
- Example: given x = (1,2,-1) and y = (2,1,1) the angle between the two vectors is 60

Prof. Pierluca Lanzi POLITECNICO DI MILANO

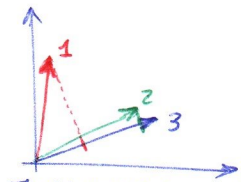
## Cosine Similarity

23

- The cosine similarity between x, y is simply computed as,

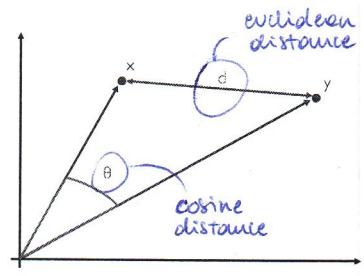
$$s(x,y) = \frac{\sum_1^n x_i y_i}{\sqrt{\sum_i^n x_i^2}\sqrt{\sum_i^n y_i^2}}$$

= normalized projection of one vector onto the other (we cannot do 1 − dissimilarity to obtain similarity in this case)



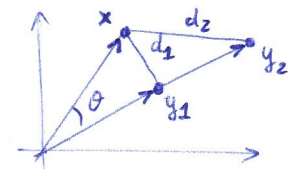

These are 3 (normalized) vectors: the projection of 2 on 3 will be higher than the projection of 1 on 3 ⇒ 2 and 3 are more similar than 1 and 3 (and the cosine distance confirms)

Prof. Pierluca Lanzi POLITECNICO DI MILANO




Cosine Distance θ compared to the Euclidean distance d

This distance is scale-ll :




$y_1$ and $y_2$ are the same point but scaled: $d_1 \neq d_2$, $\theta_1 = \theta_2 \equiv \theta$ (the euclidean distance between x and $y_1$ is different from the eucl. distance between x and $y_2$, meanwhile the cosine distance remains the same)

## Edit Distance

25

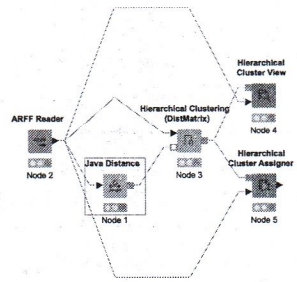
- The distance between a string x=x1x2…xn and y=y1y2…ym is the smallest number of insertions and deletions of single characters that will transform x into y
- Alternatively, the edit distance d(x, y) can be computed as the longest common subsequence (LCS) of x and y and then,

$$d(x,y) = |x| + |y| - 2|LCS|$$

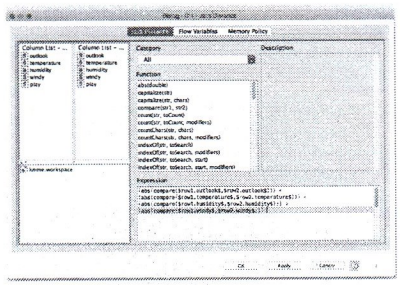
- Example
  - The edit distance between **x=abcde** and **y=acfdeg** is 3 (delete b, insert f, insert g), the LCS is acde which is coherent with the previous result

Prof. Pierluca Lanzi POLITECNICO DI MILANO

Some tools allow users to define their own distance function




KNIME workflow applying hierarchical clustering with a custom distance function implemented in Java



The definition of the custom distance function used in the previous KNIME workflow.

Other tools provide only the usual distance function (Euclidean, Manhattan, etc.)

When a custom function is not an option, we can try to transform the data

Normalization

## Normalization and Other Issues



- Different attributes are measured on different scales, thus we often need to normalize them, using for instance range normalization o standard score normalization

$$x'_i = \frac{x_i - \min_i x_i}{\max_i x_i - \min_i x_i} \qquad x'_i = \frac{x_i - \mu}{\sigma}$$

- For nominal attributes, the distance either 0 (the value is the same) or 1 (the value is different)
- Missing values are usually assumed to be maximally distant (given normalized attributes)



# Requirements on Clustering Algorithms

## Requisites for Clustering Algorithms



- Scalability
- Ability to deal with different types of attributes
- Ability to handle dynamic data
- Discovery of clusters with arbitrary shape
- Minimal requirements for domain knowledge to determine input parameters
- Able to deal with noise and outliers
- Insensitive to order of input records
- High dimensionality
- Incorporation of user-specified constraints
- Interpretability and usability

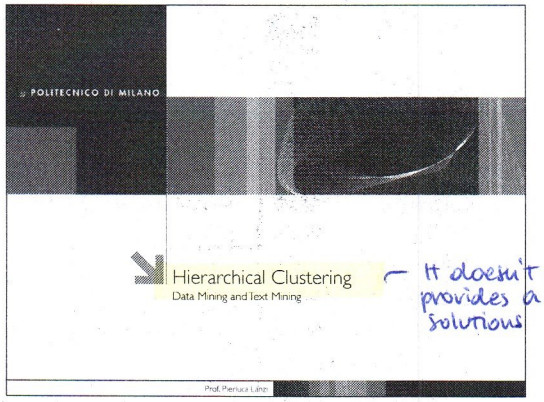


# Curse of Dimensionality

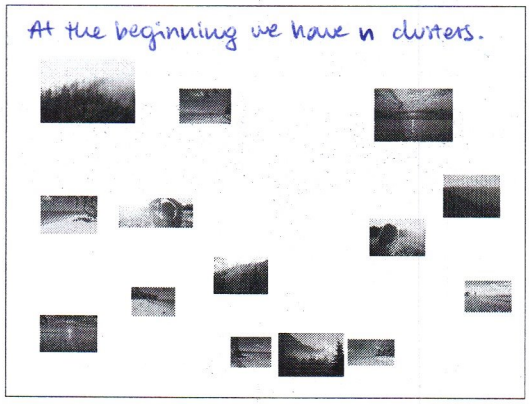
## Curse of Dimensionality

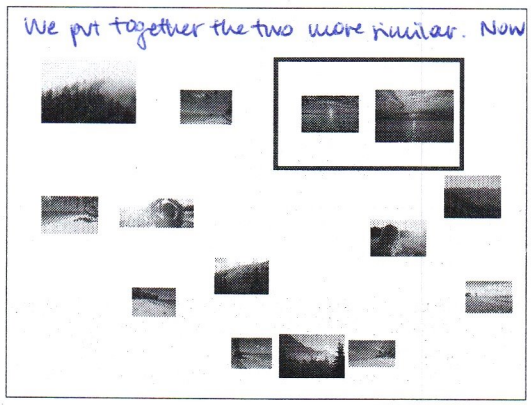
in high dimensions, almost all pairs of points are equally far away from one another

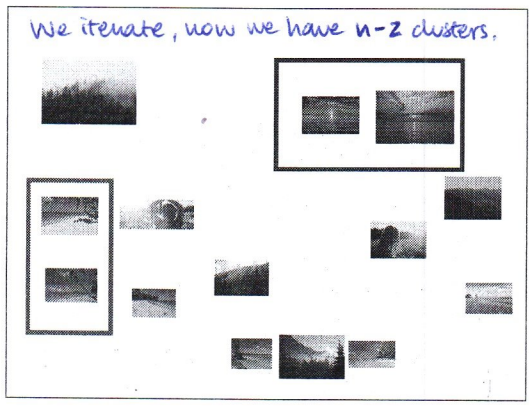
almost any two vectors are almost orthogonal

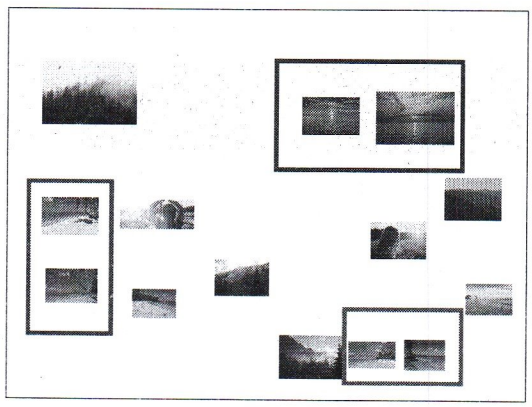



— It doesn't provide a solution, it provides a set (a hierarchy) of solutions

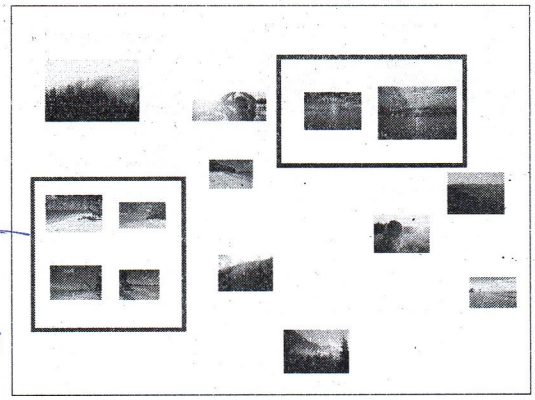


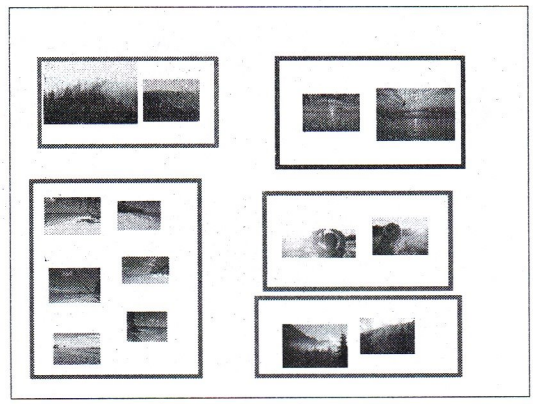



In this step we unified two already existing groups (before we used a distance for: one element vs. another; now we're using a distance between groups! How do we do it?)

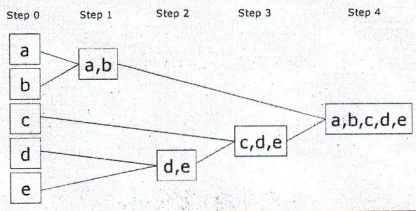




## Agglomerative Hierarchical Clustering



- Suppose we have five items, a, b, c, d, and e.
- Initially, we consider one cluster for each item
- Then, at each step we merge together the most similar clusters, until we generate one cluster






| Clustering | Partition |
|---|---|
| $C_1$ | {A} {B} {C} {D} {E} |
| $C_2$ | {A, B} {C} {D} {E} |
| $C_3$ | {A, B} {C} {D, E} |
| $C_4$ | {A, B} {C, D, E} |
| $C_5$ | {A, B, C, D, E} |

## What is Hierarchical Clustering?

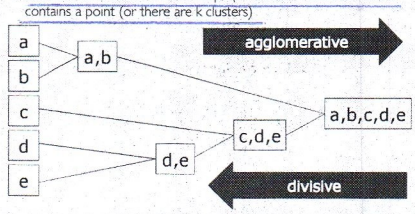
By far, one of the most common clustering techniques

Produces a hierarchy (dendrogram) of nested clusters that can be analyzed and visualized

## What Approaches?

11

- **Agglomerative**
  - Start individual clusters, at each step, merge the closest pair of clusters until only one cluster (or k clusters) left
- **Divisive**
  - Start with one cluster, at each step, split a cluster until each cluster contains a point (or there are k clusters)

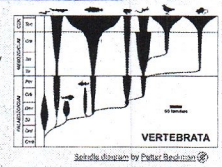





## Strengths of Hierarchical Clustering

12

- No need to assume any particular number of clusters
  - Any desired number of clusters can be obtained by 'cutting' the dendrogram at the proper level
  - They may correspond to meaningful taxonomies
- Example in biological sciences include animal kingdom, phylogeny reconstruction, etc.
- Traditional hierarchical algorithms use a similarity or distance matrix to merge or split one cluster at a time




Scientific diagram by Peter Beckman

by giving a set of solutions instead of just one we give more informations about the data:

"there are k macrogroups, one of these macrogroups could be partitioned in m subgroups and another of these macrogroups can be partitioned in l subgroups, ..."

we're saying more about the structure of the data



## Agglomerative Clustering Algorithm

13

1. Compute proximity matrix of pairwise distances between all points
2. Let each data point be a cluster
3. Repeat until a single cluster remains
   - Merge two closest clusters
   - Update proximity matrix

- Most popular hierarchical clustering technique
- Key operation is the computation of the proximity of two clusters
- Different approaches to defining the distance between clusters distinguish the different algorithms

!!



**ALGORITHM 14.1. Agglomerative Hierarchical Clustering Algorithm**

```
AGGLOMERATIVECLUSTERING(D, k):
C ← {C_i = {x_i} | x_i ∈ D} // Each point in separate cluster
Δ ← {δ(x_i, x_j): x_i, x_j ∈ D} // Compute distance matrix
repeat
    Find the closest pair of clusters C_i, C_j ∈ C
    C_ij ← C_i ∪ C_j // Merge the clusters
    C ← (C \ {C_i, C_j}) ∪ {C_ij} // Update the clustering
    Update distance matrix Δ to reflect new clustering
until |C| = k
```

"Data Mining and Analysis" by Zaki & Meira - Chapter 14
http://www.dataminingbook.info

## Hierarchical Clustering: Time and Space Requirements

15

- $O(N^2)$ space since it uses the proximity matrix.
  - N is the number of points.
- There are N steps and at each step the size, $N^2$, proximity matrix must be updated and searched. $O(N^3)$ time in many cases
- Complexity can be reduced to $O(N^2 \log(N))$ time for some approaches

computationally is a lot!

## Efficient Implementation

16

- Compute the distance between all pairs of points [$O(N^2)$]
- insert the pairs and their distances into a priority queue to fine the min in one step [$O(N^2)$]
- When two clusters are merged, we remove all entries in the priority queue involving one of these two clusters [O(N log N)]
- Compute all the distances between the new cluster and the re- maining clusters [O(NlogN)]
- Since the last two steps are executed at most N time, the complexity of the whole algorithms is $O(N^2 \log N)$



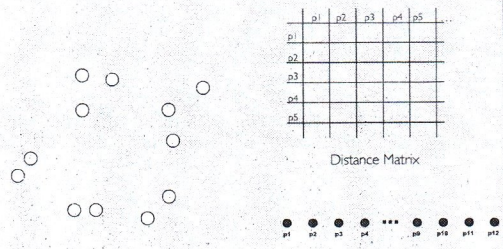
## Distance Between Clusters

It's simple to evaluate the distance between two points, but how about the distance between a point and a cluster? How about the distance between two clusters?

## Initial Configuration

18

- Start with clusters of individual points and the distance matrix

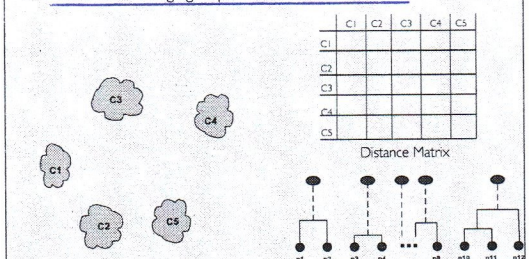





## Intermediate Situation

19

- After some merging steps, we have some clusters

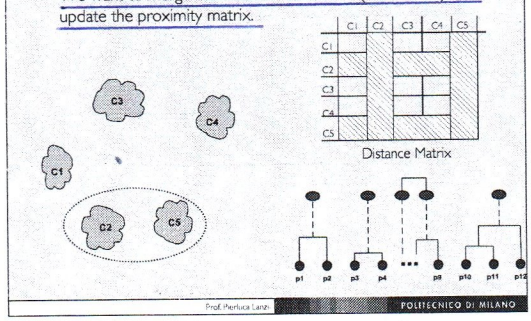


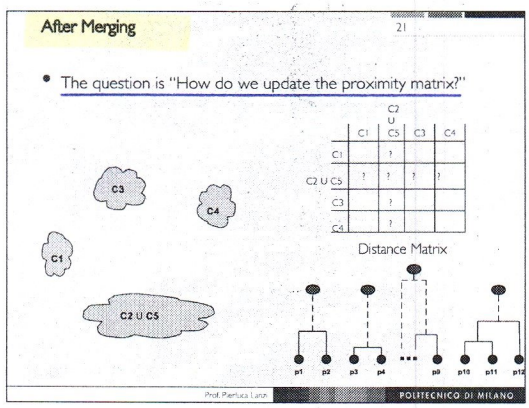


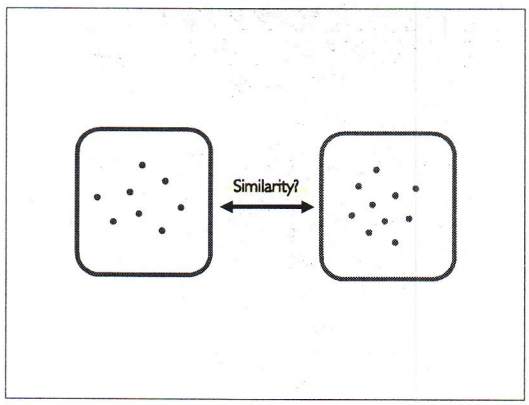

## Intermediate Situation

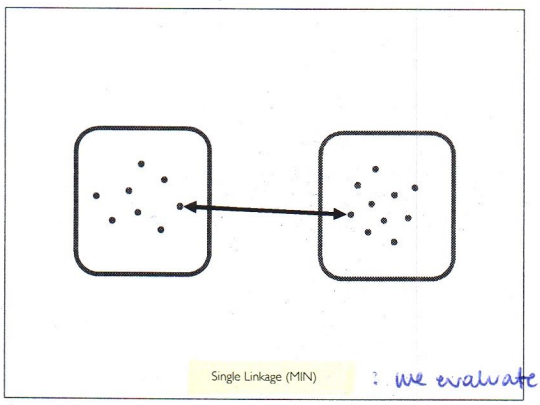
20

- We want to merge the two closest clusters (C2 and C5) and update the proximity matrix.



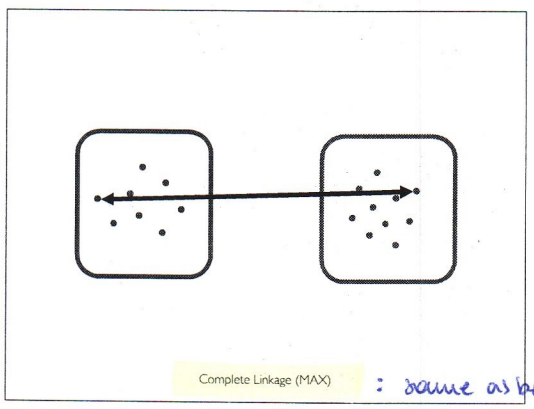


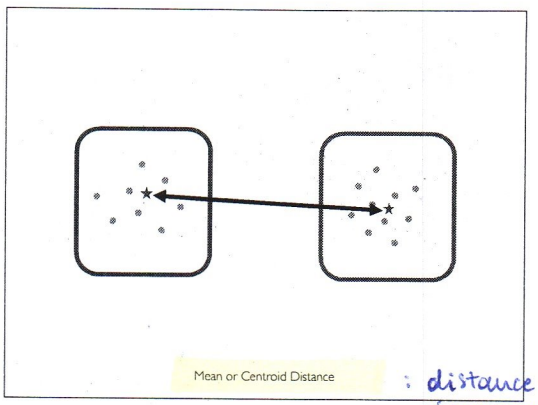



Single Linkage (MIN) : we evaluate all the distances and we consider as distance between the two groups the minimum distance

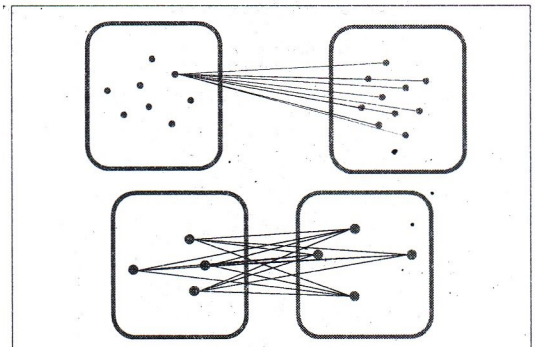
How are they different? For example: if the minimum distance is > 0 then the two groups are separable, if the maximum distance is > 0 the two groups can still be overlapped



Complete Linkage (MAX) : same as before but with the maximum distance



Mean or Centroid Distance : distance between the two averages (averages are each the average of one group)



Group Average : in this case we do the average of the distances between every point of one group and every point of the other

## Typical Alternatives to Calculate the Distance Between Clusters

27

- **Single link (or MIN)**
  - smallest distance between an element in one cluster and an element in the other, i.e., $d(C_i, C_j) = \min(t_{ip}, t_{jq})$
- **Complete link (or MAX)**
  - largest distance between an element in one cluster and an element in the other, i.e., $d(C_i, C_j) = \max(t_{ip}, t_{jq})$
- **Mean Distance**
  - distance between the centroids of two clusters, i.e., $d(C_i, C_j) = d(\mu_i, \mu_j)$ where $\mu_i$ and $\mu_j$ are the cluster means (or centroids)
- **Group average**
  - average distance between an element in one cluster and an element in the other, i.e., $d(C_i, C_j) = \text{avg}(d(t_{ip}, t_{jq}))$
- ...



## Example

28

- Suppose we have five items, a, b, c, d, and e.
- We wanto to perform hierarchical clustering on five instances following an agglomerative approach
- First: we compute the distance or similarity matrix
- Dij is the distance between instancce "i" and "j"

$$D = \begin{pmatrix} 0.0 & & & & \\ 2.0 & 0.0 & & & \\ 6.0 & 5.0 & 0.0 & & \\ 10.0 & 9.0 & 4.0 & 0.0 & \\ 9.0 & 8.0 & 5.0 & 3.0 & 0.0 \end{pmatrix} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix}$$

1 2 3 4 5

The first merge is between the elements 1 and 2 and it's performed at distance 2.0

⟹ the two rows and the two columns have to be merged in one row and one column

The new element is (1,2). What is the distance between (1,2) and the other elements?



## Example

29

- Group the two instances that are closer
- In this case, a and b are the closest items (D2,1=2)
- Compute again the distance matrix, and start again.
- Suppose we apply single-linkage (MIN), we need to compute the distance between the new cluster {1,2} and the others
  - d(12)3 = min[d13,d23] = d23 = 5.0
  - d(12)4 = min[d14,d24] = d24 = 9.0
  - d(12)5 = min[d15,d25] = d25 = 8.0



## Example

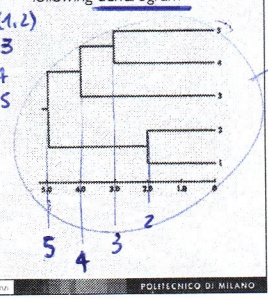
30

- The new distance matrix is,

$$D = \begin{pmatrix} 0.0 & & & \\ 5.0 & 0.0 & & \\ 9.0 & 4.0 & 0.0 & \\ 8.0 & 5.0 & 3.0 & 0.0 \end{pmatrix} \begin{matrix} (1,2) \\ 3 \\ 4 \\ 5 \end{matrix}$$

(1,2) 3 4 5

- At the end, we obtain the following dendrogram




5 4 3 2

the dendrogram says the merges + the distances at which the merges were done

# Determining the Number of Clusters

Hierarchical clustering generates a set of N possible partitions

Which one should I choose?

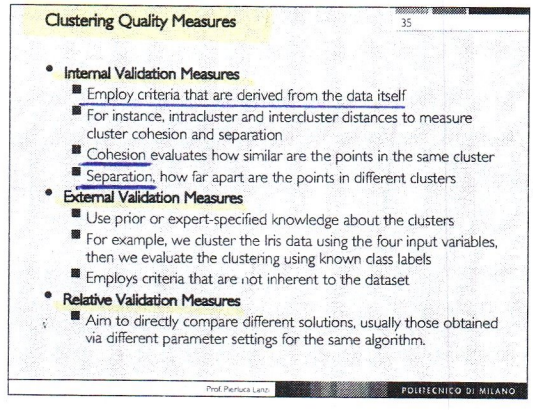
Ideally a good clustering should data partition points so that …

Data points in the same cluster should have a small distance from one another } measure of COHESION

Data points in different clusters should be at a large distance from one another. } measure of SEPARATION

How can we evaluate the quality of a clustering solution?

3 possible approaches



Clustering Quality Measures 35

- **Internal Validation Measures**
  - Employ criteria that are derived from the data itself
  - For instance, intracluster and intercluster distances to measure cluster cohesion and separation
  - Cohesion evaluates how similar are the points in the same cluster
  - Separation, how far apart are the points in different clusters
- **External Validation Measures**
  - Use prior or expert-specified knowledge about the clusters
  - For example, we cluster the Iris data using the four input variables, then we evaluate the clustering using known class labels
  - Employs criteria that are not inherent to the dataset
- **Relative Validation Measures**
  - Aim to directly compare different solutions, usually those obtained via different parameter settings for the same algorithm.

WSS → 0 as k → N

BSS = 0 if k = 1 and BSS starts increasing as k increases

## Internal Measures: Cohesion and Separation



- **Cohesion measures how closely related are objects in a cluster**
  - Within-cluster sum of squares

$$\mathrm{WSS}(C) = \sum_{i=1}^{k} \sum_{x_j \in C_i} d(x_j, \mu_i)^2$$

  - where $\mu_i$ is the centroid of cluster $C_i$ (in case of Euclidean spaces) ($\mu_i$ = mean of the points of group $i$)
- **Separation measures wells separated a cluster is from other clusters**
  - Between-cluster sum of squares

$$\mathrm{BSS}(C) = \sum_{i=1}^{k} |C_i| d(\mu, \mu_i)^2$$

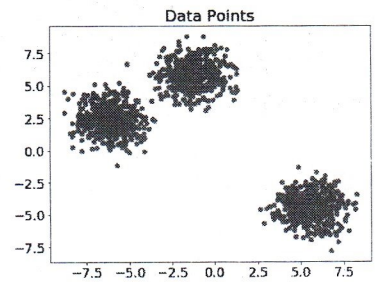
where $\mu$ is the centroid of the whole dataset (= mean of the whole dataset)



## Evaluation of Hierarchical Clustering using Knee/Elbow Analysis

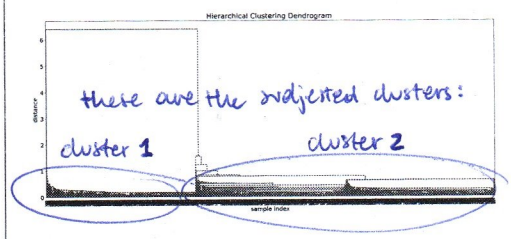
plot the WSS and BSS for every clustering and look for a knee in the plot that show a significant modification in the evaluation metrics

Suppose that we have:



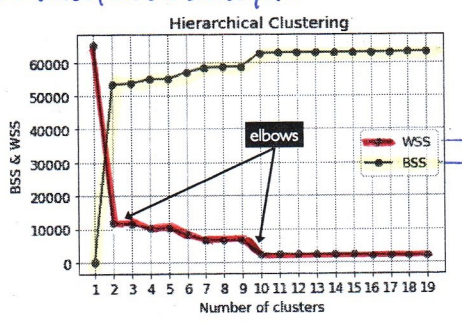

Example data generated using the make_blob function of Scikit-Learn

The dendrogram that comes out is:



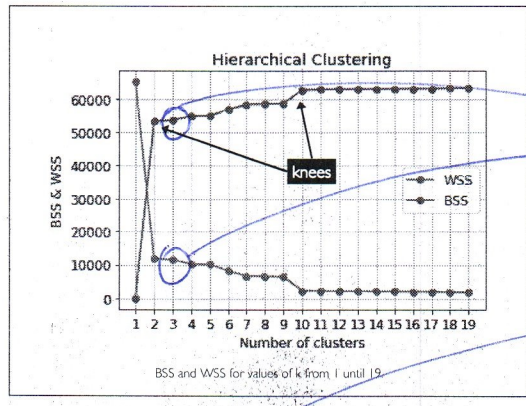

Dendrogram computed using single linkage.

The knee/elbow analysis:



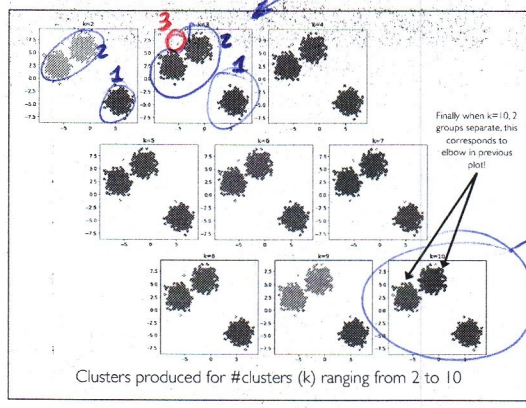

→ COHESION
→ SEPARATION

BSS and WSS for values of k from 1 until 19.




knowing the data we know that there are 3 clusters. That's why we're a little bit surprised to see only 2 groups. What's happening in groups when we choose 2 or 3 clusters?

It creates a one-point cluster (not really interesting)




(k=10)
When we step from 9 to 10 we finally creates 3 supergroups. The problem is the small clusters. We can either eliminate them or we can merge the one-point clusters to the major clusters.

How can we know that this is the scenario without analysing the plots? We can see the numerosity of the clusters. If from k=2 to k=3 the numerosity it [500, 500] → [500, 499, 1] we already know what's going on.

Knee/elbow plots might not tell the whole story!

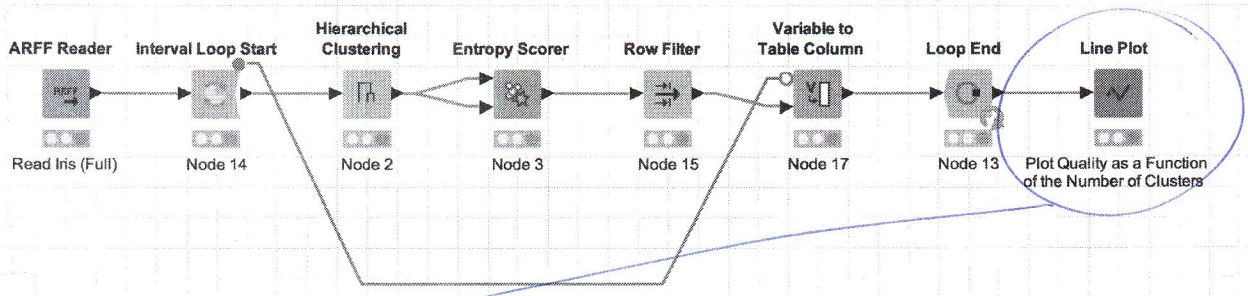
In the example, the plots suggested several choices of clustering

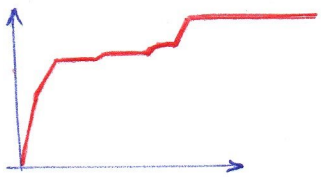
We should use the elbow as a guide but test multiple solutions around that value

We should analyze different solutions around the knees/elbow

We should also analyze the merges to eliminate the ones due to noise

Examples using KNIME



Computing cluster quality from one to 20 clusters using the entropy scorer

## How can we represent clusters?

### Euclidean vs Non-Euclidean Spaces

48

- **Euclidean Spaces**
  - We can identify a cluster using for instance its centroid (e.g. computed as the average among all its data points)
  - Alternatively, we can use its convex hull
- **Non-Euclidean Spaces**
  - We can define a distance (jaccard, cosine, edit)
  - If we cannot compute a centroid, we can introduce medoid
- **Medoid**
  - Existing data point that we take as a cluster representative
  - Point that minimizes the sum of distances to all other points in cluster
- **Alternatives to Medoid**
  - Choose a point that minimize the maximum distance to another point or the sum of the squares of the distances to the other points in the cluster

Prof. Pierluca Lanzi — POLITECNICO DI MILANO

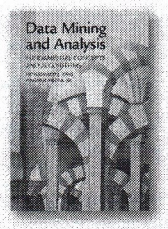
we consider a point and we evaluate the distance between this point and all the others (we sum). We repeat for the whole cluster: the point for which the sum of distances is minimum is the MEDOID.

## Summary

### Study Material

51

- "Data Mining and Analysis" by Zaki & Meira
  - Chapter 14
- http://www.dataminingbook.info



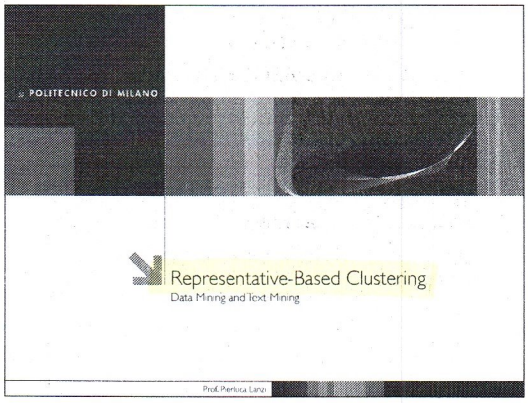

Prof. Pierluca Lanzi — POLITECNICO DI MILANO

### Hierarchical Clustering: Problems and Limitations

50

- Once a decision is made to combine two clusters, it cannot be undone
- No objective function is directly minimized
- Different schemes have problems with one or more of the following:
  - Sensitivity to noise and outliers
  - Difficulty handling different sized clusters and convex shapes
  - Breaking large clusters
- Major weakness of agglomerative clustering methods
  - They do not scale well: time complexity of at least $O(n^2)$, where n is the number of total objects
  - They can never undo what was done previously

Prof. Pierluca Lanzi — POLITECNICO DI MILANO




# Representative-Based Clustering

Data Mining and Text Mining



For each cluster there is a point (or a parameter vector) that summarizes it

## Representative-Based Algorithms



- Given a dataset of N instances, and a desired number of clusters k, this class of algorithms generates a partition C of N in k clusters $\{C_1, C_2, \ldots, C_k\}$
- For each cluster there is a point that summarizes the cluster
- The common choice being the mean of the points in the cluster

$$\mu_i = \frac{1}{n_i} \sum_{x_j \in C_i} x_j$$

where $n_i = |C_i|$ and $\mu_i$ is the centroid



## Brute-force Approach

Generate all the possible clustering $C = \{C_1, C_2, \ldots, C_k\}$

Then, select the best one

Unfortunately, there are $O(k^N/k!)$ possible partitions

## K-Means algorithm

## K-Means Algorithm



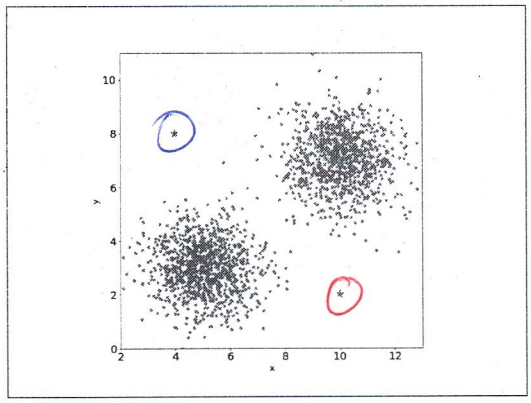
- Greedy iterative approach to find a clustering that minimizes the SSE objective:

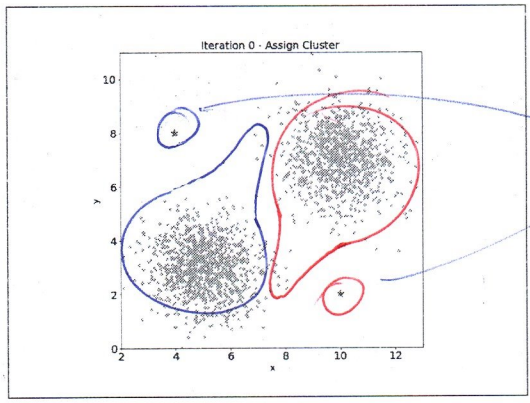
$$\text{SSE}(C) = \sum_{i=1}^{K} \sum_{x_j \in C_i} ||x_j - \mu_i||^2$$

- The goal of the clustering process is thus to find

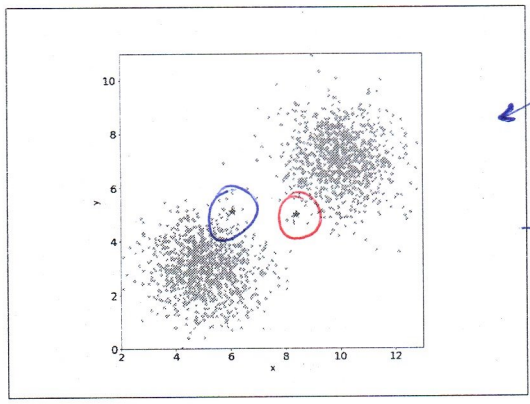
$$C^\star = \arg\min_C \text{SSE}(C)$$



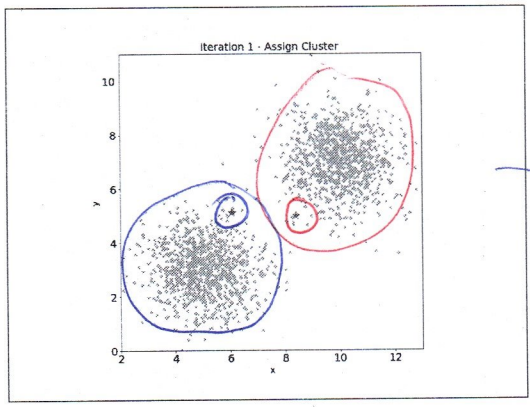


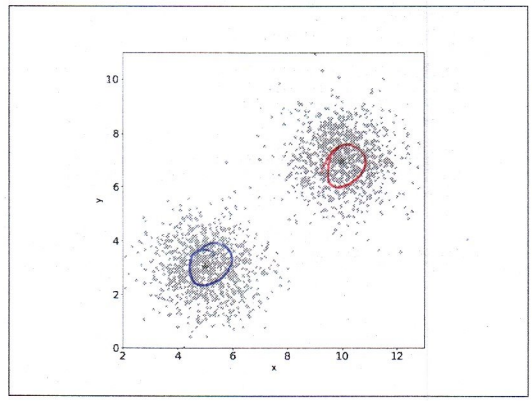

not good representatives and so, now that we have the clusters we take the clusters and update the centroids as means of the clusters



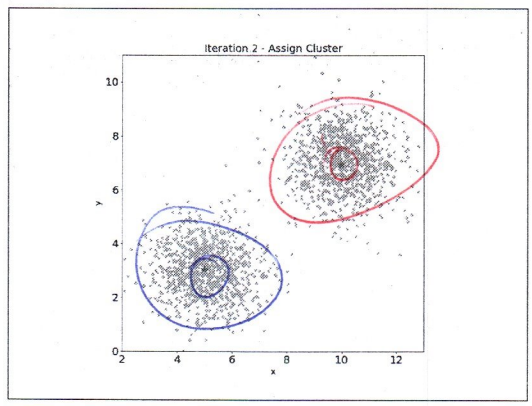

now that we changed (updated) centroids, also the clusters will change (update)

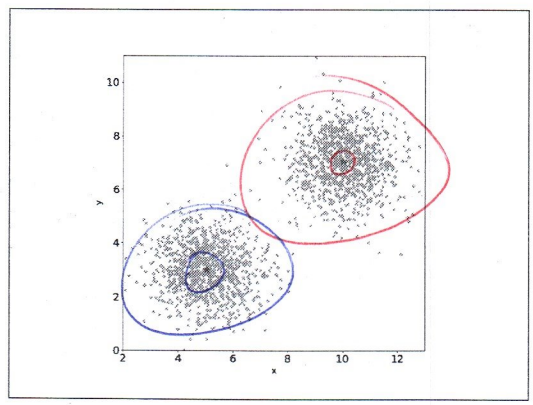



again we update the centroids




again we update the clusters




At some points we reach some convergence (even if we may have some oscillating points)

Notice that K-means is shape-biased! It's biased towards classes that are spheres in n-dimension

**ALGORITHM 13.1. K-means Algorithm**

```
K-MEANS (D, k, ε):
t = 0
Randomly initialize k centroids: μ_1^t, μ_2^t, ..., μ_k^t ∈ R^d
repeat
    t ← t + 1
    C_j ← ∅ for all j = 1, ..., k
    // Cluster Assignment Step
    foreach x_i ∈ D do
        j* ← argmin_j { ||x_i − μ_j^{t−1}||^2 }  // Assign x_i to closest centroid
        C_j* ← C_j* ∪ {x_i}
    // Centroid Update Step
    foreach j = 1 to k do
        μ_j^t ← (1/n_j) Σ_{x_i ∈ C_j} x_i
until Σ_{j=1}^{k} ||μ_j^t − μ_j^{t−1}||^2 ≤ ε
```

"Data Mining and Analysis" by Zaki & Meira - Chapter 13
http://www.dataminingbook.info

## k-Means Algorithm

15

- Most widely known representative-based algorithm
- Assumes an Euclidean space but sometimes it can be extended to the non-Euclidean case
- Employs a greedy iterative approaches that minimizes the SSE objective. Accordingly it can converge to a local optimal instead of a globally optimal clustering.

## Computational Complexity



- Cluster assignment takes O(nkd) time since, for each of the n points, it computes its distance to each of the k clusters, which takes d operations in d dimensions
- The centroid re-computation step takes O(nd) time because it adds at total of n d-dimensional points
- Assuming that there are t iterations, the total time for K-means is given as O(tnkd).
- In terms of the I/O cost it requires O(t) full database scans, because we have to read the entire database in each iteration.



## Centroid initialization

## Centroid Initialization



- **Solution 1**
  - Pick points that are as far away from one another as possible.
- **Solution 2**

```
Pick the first point at random;
WHILE there are fewer than k points DO
   Add the point whose minimum distance
   from the selected points is as large as
   possible;
END;
```

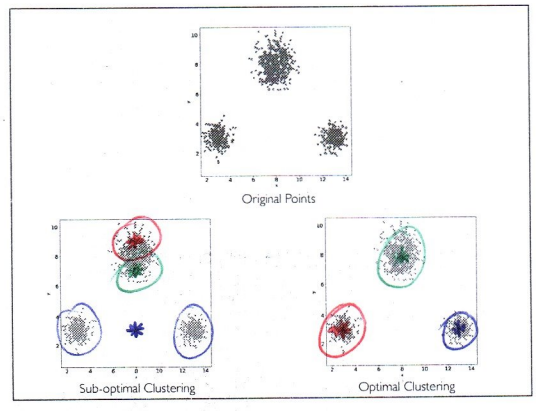
we pick randomly one point (1), then we pick the most-far point from (1), then we pick the most-far point from both, then ..

- **Solution 3**
  - Cluster a sample of the data, perhaps hierarchically, so there are k clusters. Pick a point from each cluster, perhaps that point closest to the centroid of the cluster.

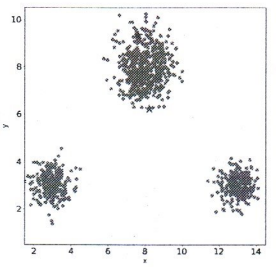
take a sample of the data, apply hierarchical clustering, cut at k (given by user) and consider the centroids of the clusters generated by hierarchical



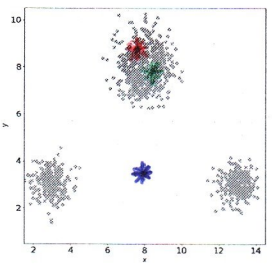
## Initial centroids matters!



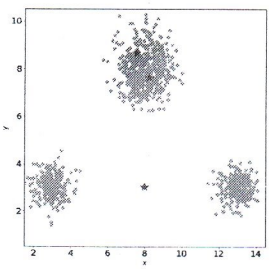
# Run #1



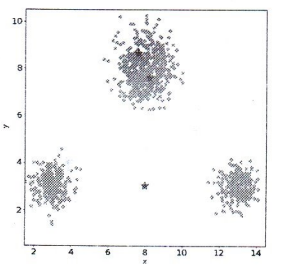

Run #1 – Start



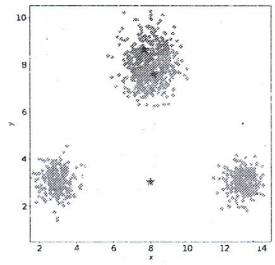

Run #1 – Step 1



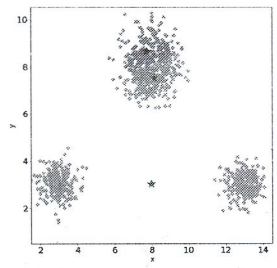

Run #1 – Step 2




Run #1 – Step 3

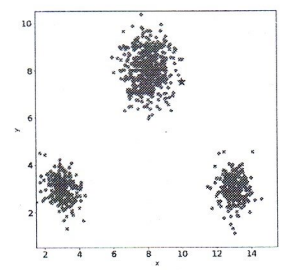



Run #1 – Step 4

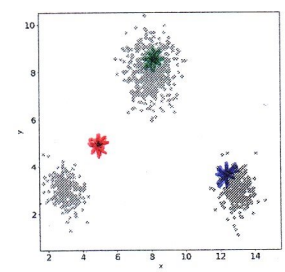



Run #1 – Step 5

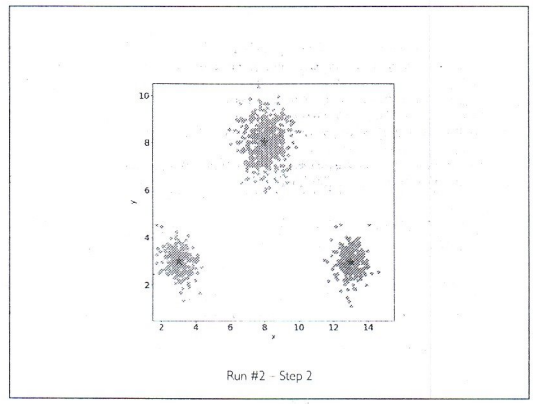
## Run #2



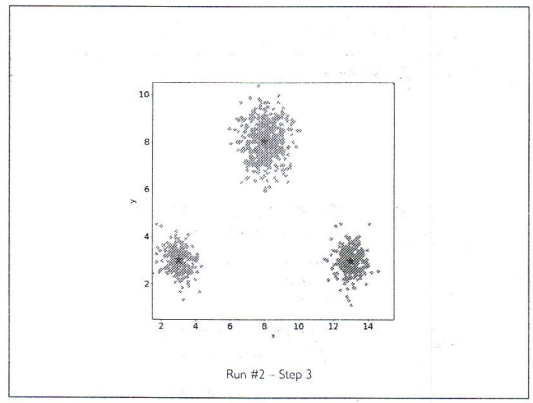

Run #2 – Start




Run #2 – Step 1



Run #2 - Step 2



Run #2 - Step 3

## Why Selecting the Best Initial Centroids is Difficult?

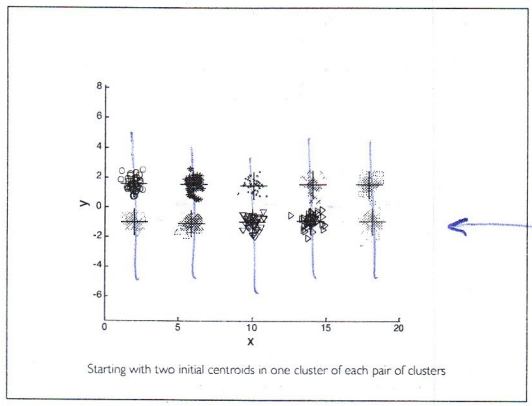
33

- If there are K 'real' clusters then the chance of selecting one centroid from each cluster is small.
- Chance is relatively small when K is large
- If clusters are the same size, n, then

$$P = \frac{\text{number of ways to select one centroid from each cluster}}{\text{number of ways to select } K \text{ centroids}} = \frac{K! n^K}{(Kn)^K} = \frac{K!}{K^K}$$

- For example, if K = 10, then probability = 10!/1010 = 0.00036
- Sometimes the initial centroids will readjust themselves in 'right' way, and sometimes they don't
- Consider an example of five pairs of clusters

Prof. Pier Luca Lanzi — POLITECNICO DI MILANO



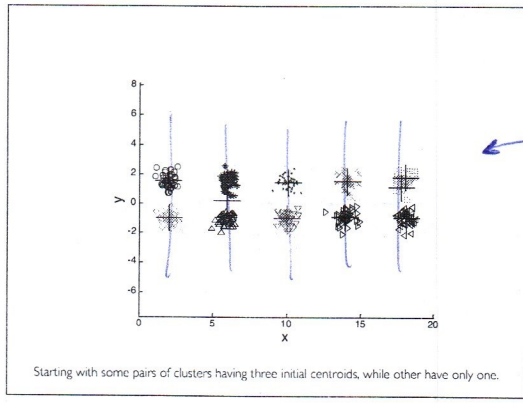
Starting with two initial centroids in one cluster of each pair of clusters

**BISECTING K-MEANS**

We start with 2 clusters. We take the worst cluster and we add a "seed". Then we continue.

Consider for example:

we put 2 centroids in each one ✓



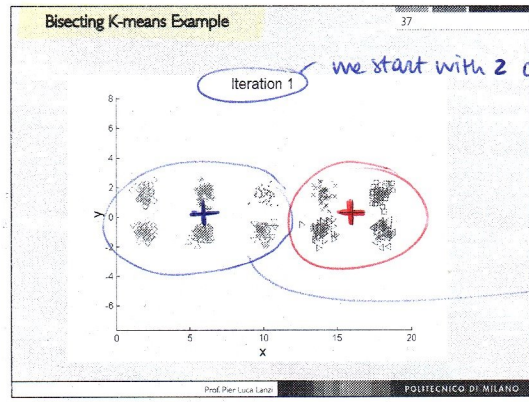
Starting with some pairs of clusters having three initial centroids, while other have only one.

we put 2 in some, 3 in others and 1 in others ✗

Since we don't know which case will be good for us we have to establish a rigorus method. Let's try the bisecting k-means!

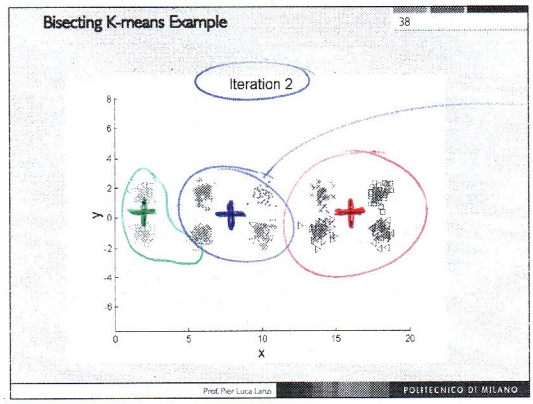
**Algorithm 3** Bisecting K-means Algorithm.

1: Initialize the list of clusters to contain the cluster containing all points.
2: **repeat**
3: Select a cluster from the list of clusters
4: **for** $i = 1$ to *number_of_iterations* **do**
5: Bisect the selected cluster using basic K-means
6: **end for**
7: Add the two clusters from the bisection with the lowest SSE to the list of clusters.
8: **until** Until the list of clusters contains $K$ clusters

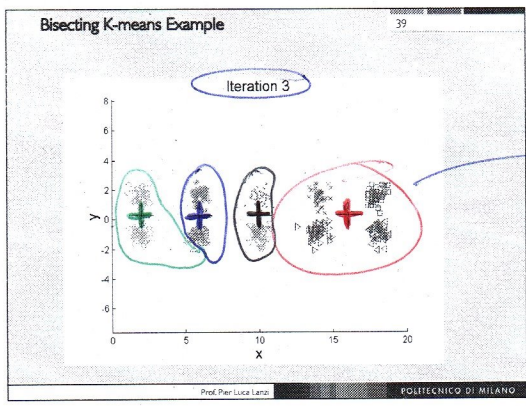



we start with 2 clusters

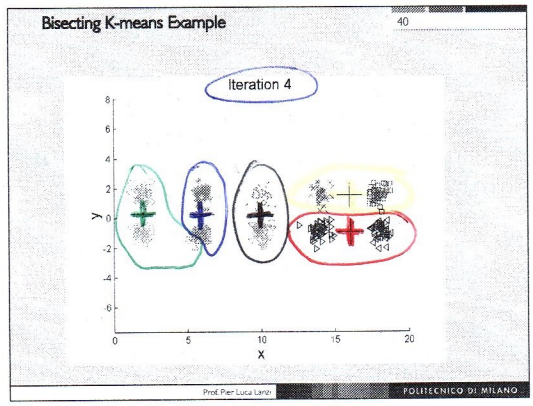
this cluster is worse in terms of error (it less cohese)
⇒ we add a seed to this group (the other remains unchanged)



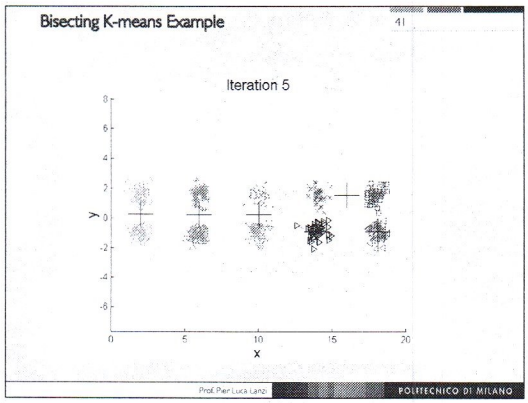

this is again the less cohese, we put another seed

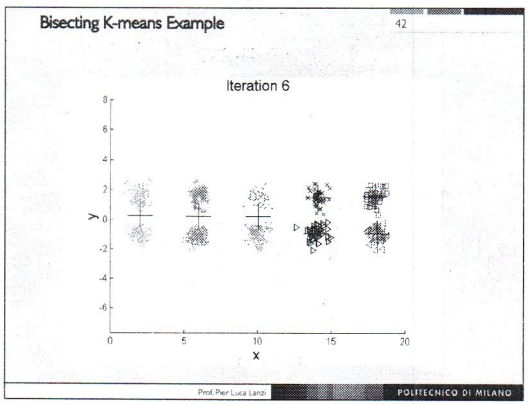


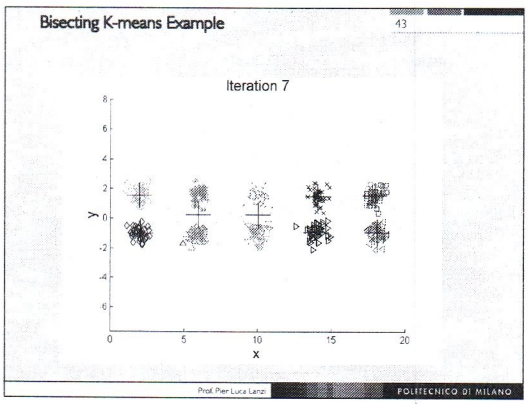

now this is the less cohese. we add a seed.

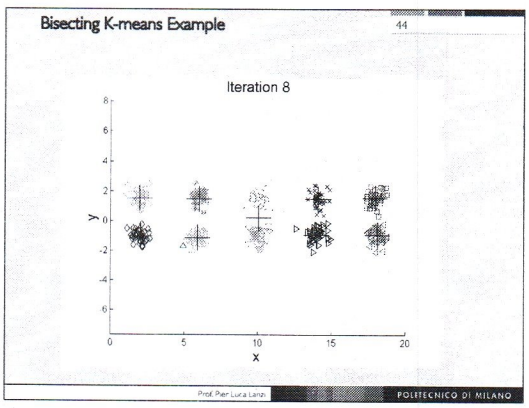


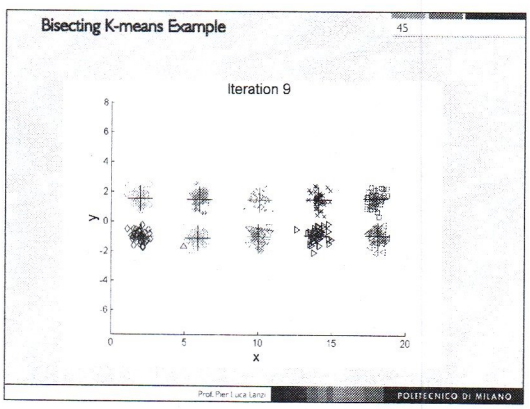

we go on.


Bisecting K-means Example
41
Iteration 5
Prof. Pier Luca Lanzi
POLITECNICO DI MILANO


Bisecting K-means Example
42
Iteration 6
Prof. Pier Luca Lanzi
POLITECNICO DI MILANO

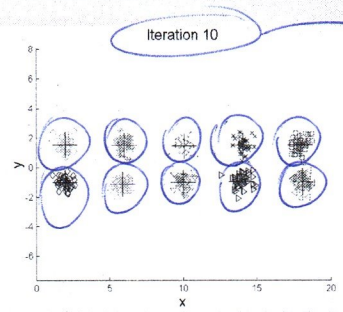

Bisecting K-means Example
43
Iteration 7
Prof. Pier Luca Lanzi
POLITECNICO DI MILANO


Bisecting K-means Example
44
Iteration 8
Prof. Pier Luca Lanzi
POLITECNICO DI MILANO


Bisecting K-means Example
45
Iteration 9
Prof. Pier Luca Lanzi
POLITECNICO DI MILANO

## Bisecting K-means Example








we stop here because if we add a seed the SSE does not improve
→ 10 clusters
(doesn't improve or it improves so little)

## Dealing with the Initial Centroids Issue



- Multiple runs, helps, but probability is not on your side
- Sample and use another clustering method (hierarchical?) to determine initial centroids
- Select more than k initial centroids and then select among these initial centroids
- Postprocessing
- Bisecting K-means, not as susceptible to initialization issues



## Extensions

## Updating Centers Incrementally



- In the basic K-means algorithm, centroids are updated after all points are assigned to a centroid
- An alternative is to update the centroids after each assignment (incremental approach)
  - Each assignment updates zero or two centroids
  - More expensive
  - Introduces an order dependency
  - Never get an empty cluster
  - Can use "weights" to change the impact



## Pre-processing and Post-processing



- Pre-processing
  - Normalize the data → ⊥ method, we do it because distances are involved
  - Eliminate outliers → this is based on the method
- Post-processing
  - Eliminate small clusters that may represent outliers
  - Split 'loose' clusters, i.e., clusters with relatively high SSE
  - Merge clusters that are 'close' and that have relatively low SSE
  - These steps can be used during the clustering process

## Other K-Means Variants



- A few variants of the k-means which differ in
  - Selection of the initial k means
  - Dissimilarity calculations
  - Strategies to calculate cluster means
- Handling categorical data: k-modes
  - Replacing means of clusters with modes
  - Using new dissimilarity measures to deal with categorical objects
  - Using a frequency-based method to update modes of clusters
  - A mixture of categorical and numerical data: k-prototype method



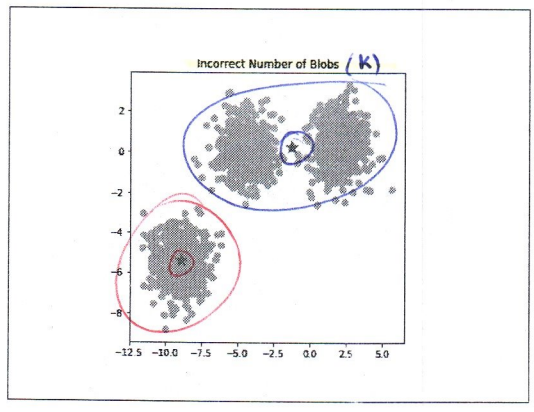
# Limitations

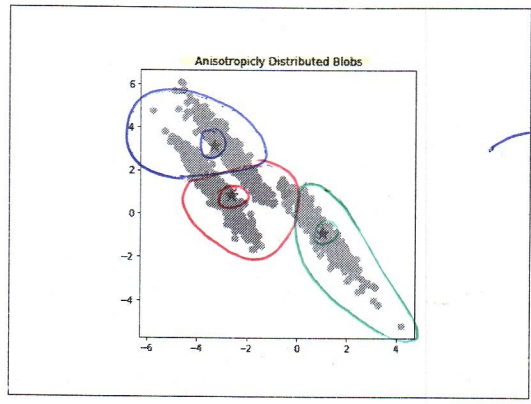
## Limitations of K-means



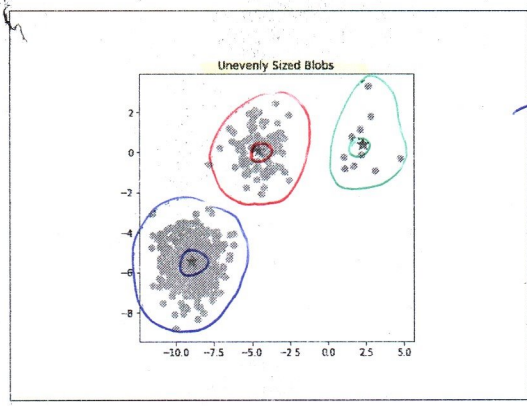
- K-means has problems when clusters are of differing
  - Sizes
  - Densities
  - Non-globular shapes
- K-means has also problems when the data contains outliers.
- we have to decide k apriori



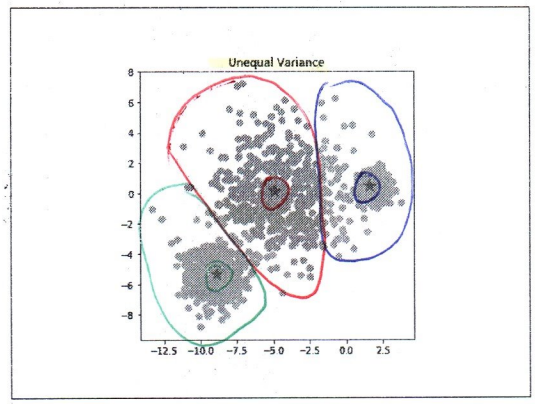


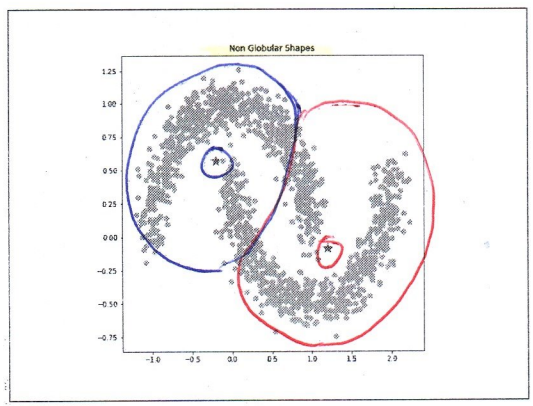



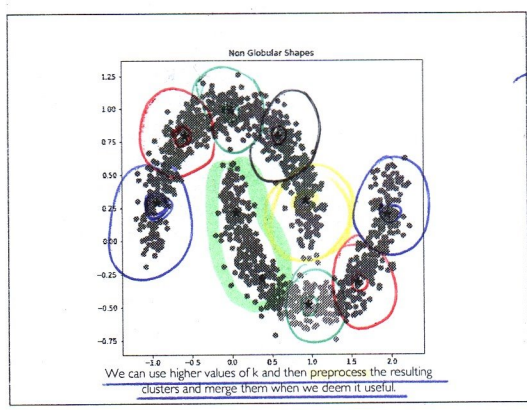

The clusters try to remain spherical




The final clusters are good but the SSE for the different groups vary a lot because of densities




(the double colors are correlated)

## k-Means Clustering Summary

60

- **Strength**
  - Relatively efficient
  - Often terminates at a local optimum
  - The global optimum may be found using techniques such as: deterministic annealing and genetic algorithms
- **Weakness**
  - Applicable only when mean is defined, then what about categorical data?
  - Need to specify k, the number of clusters, in advance
  - Unable to handle noisy data and outliers
  - Not suitable to discover clusters with non-convex shapes
- **Advantages**
  - Simple, understandable
  - Items automatically assigned to clusters
- **Disadvantages**
  - Must pick number of clusters before hand
  - All items forced into a cluster
  - Too sensitive to outliers

## Variants of the K-Means Method



- A few variants of the k-means which differ in
  - Selection of the initial k means
  - Dissimilarity calculations
  - Strategies to calculate cluster means
- Handling categorical data: k-modes
  - Replacing means of clusters with modes
  - Using new dissimilarity measures to deal with categorical objects
  - Using a frequency-based method to update modes of clusters
  - A mixture of categorical and numerical data: k-prototype method

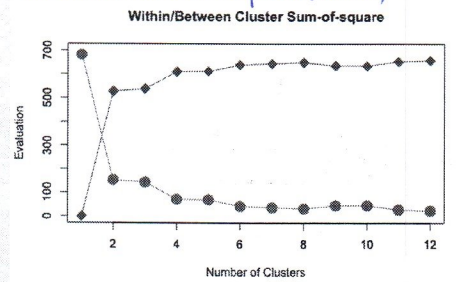


## How do we choose k?

## Elbow & Knee Analysis



(internal measure of performance)





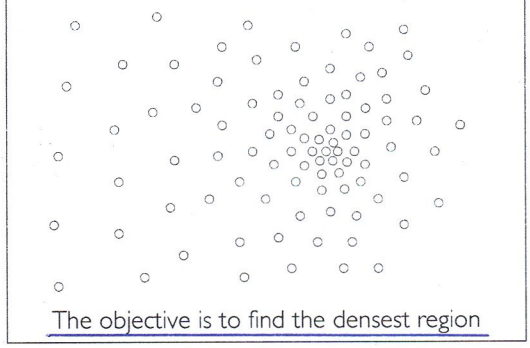

## Mean Shift Clustering

Mean shift is an iterative, nonparametric, and versatile algorithm

It searches for the mode (i.e., the point of highest density) of a data distribution

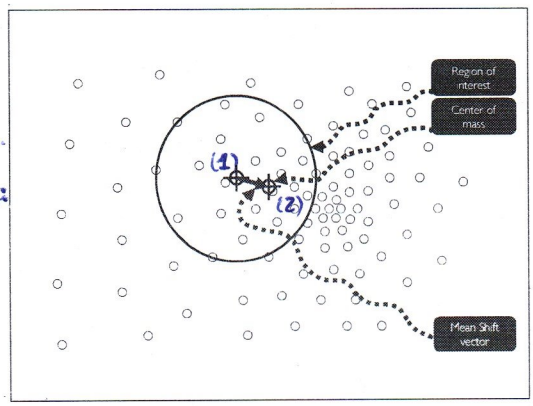
Not really a clustering procedure. This finds points with maximal density (clouds with maximal density)

Underlying intuition

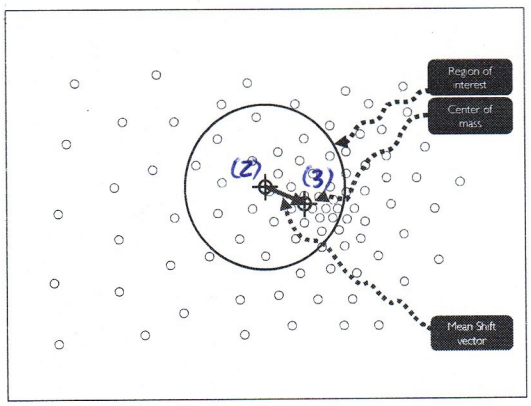


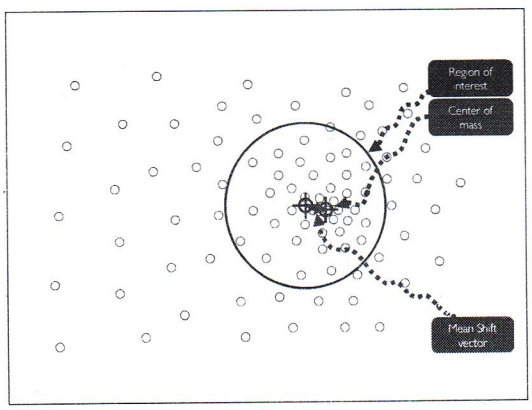
The objective is to find the densest region

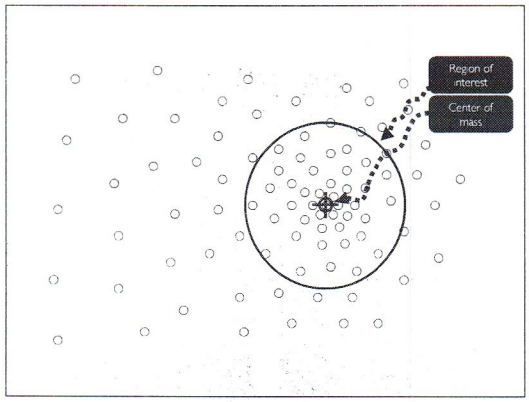
We take a random point (1) and we consider its surroundings. We move to the center of mass of the whole surrounding: (1) → (2)



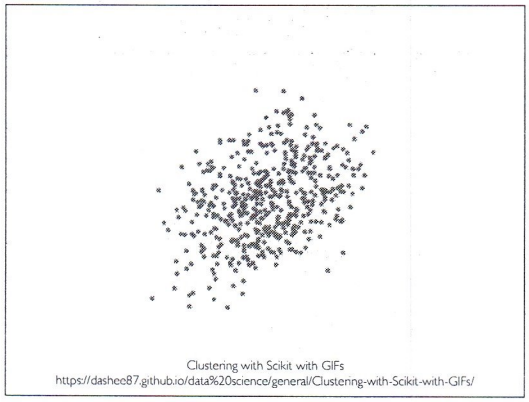

Again : (2) → (3)






At some point the mean shift vector will be extremely small
→ we stop (and we found the point with the highest density)



Clustering with Scikit with GIFs
https://dashee87.github.io/data%20science/general/Clustering-with-Scikit-with-GIFs/

## Mean Shift Algorithm Pseudocode

73

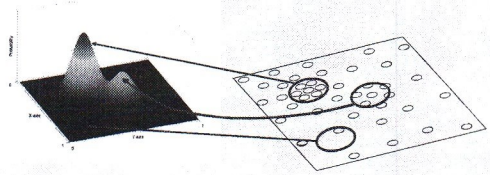
1. Choose a search window size (the bandwidth)
2. Choose the initial location of the search window
3. Compute the mean location in the search window
4. Center the search window at the location computed in Step 3
5. Repeat Steps 3 and 4 until convergence



## Non-Parametric Density Estimation

74

- We assume that the data points are sampled from an underlying probability density function (PDF)



Assumed Underlying PDF  Real Data Samples



## Kernel Density Estimation - Examples

75

- Epanechnikov Kernel

$$K_E(x) = \begin{cases} c(1 - ||x||^2) & ||x|| \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

- Uniform Kernel

$$K_U(x) = \begin{cases} c & ||x|| \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

- Normal/Gaussian Kernel

$$K_N(x) = c \cdot e^{-0.5||x||^2}$$

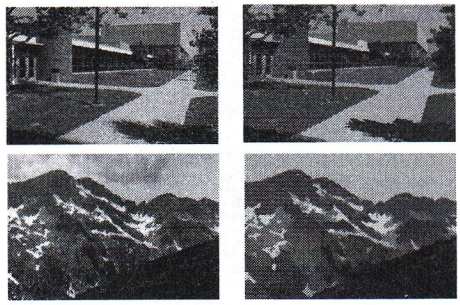
$$K_G(\frac{x - x_i}{h}) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-x_i)^2}{2h^2}}$$



When we have to evaluate the center of mass we can actually use other methods (we don't necessarily have to use the mean)

## Mean Shift Segmentation

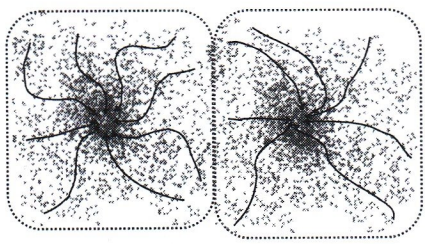
(kind of image-based clustering)

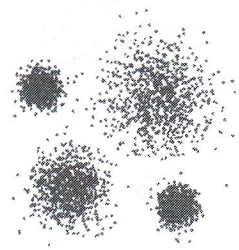


D. Comaniciu and P. Meer, Mean Shift: A Robust Approach toward Feature Space Analysis, PAMI 2002.

## Mean Shift Clustering

Trajectories that lead to the same mode (attraction basin) should be in the same cluster





Clustering with Scikit with GIFs
https://dashee87.github.io/data%20science/general/Clustering-with-Scikit-with-GIFs/

## Mean Shift Clustering - Pseudocode



1. Choose kernel and bandwidth
2. For each point
   a. Center a window on that point
   b. Compute the mean of the data in the search window
   c. Center the search window at the new mean location
   d. Repeat (b,c) until convergence
3. Assign points that lead to nearby modes to the same cluster



## K-Means & Mean Shift Clustering



- **Number of clusters and shape**
  - k-means assumes that the number of clusters is already known and the clusters are shaped spherically (or elliptically)
  - Mean shift does not assume anything about number of clusters. Instead, the number of modes give the number of clusters
  - Also, since mean shift is based on density estimation, it can handle arbitrarily shaped clusters.
- **Initialization and outliers**
  - k-means is sensitive to initialization
  - Mean shift is robust to initializations. Typically, mean shift is run for each point or sometimes points are selected uniformly from the feature space
  - Similarly, k-means is sensitive to outliers while Mean Shift is less sensitive.
- **Time complexity**
  - k-means is fast and has a time complexity $O(knT)$ where k is the number of clusters, n is the number of points and T is the number of iterations.
  - Classic mean shift is computationally expensive with a time complexity $O(Tn^2)$.



## Mean Shift Clustering



- **Strength**
  - Does not assume any prior shape on data clusters
  - Can handle arbitrary feature spaces
  - Only the window size h to choose
  - The window size h (or bandwidth) has a physical meaning
- **Weaknesses**
  - The window size h (or bandwidth) selection is not trivial
  - Inappropriate window size can cause modes to be merged, or generate additional "shallow" modes
  - Adaptive window size can help
  - Not suitable for high-dimensional features



## Computational Complexity



- MeanShift clustering runs at $O(Tn^2)$, k-Means at $O(knT)$ where T is the number of iterations and n is the number of data points
- The functions to estimate the bandwidth scale particularly badly (see sklearn estimate_bandwidth documentation)



# Expectation Maximization

## Expectation-Maximization (EM) Clustering



- k-means assigns each point to only one cluster (hard assignment)
- The approach can be extended to consider soft assignment of points to clusters, so that each point has a probability of belonging to each cluster
- We assume that each cluster $C_i$ is characterized by a multivariate normal distribution and thus identified by
  - The mean vector $\boldsymbol{\mu}_i$
  - The covariance matrix $\boldsymbol{\Sigma}_i$
- A clustering is identified by a vector of parameter $\boldsymbol{\theta}$ defined as

$$\boldsymbol{\theta} = \{\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i, P(C_i)\}$$

where $P(C_i)$ are the prior probability of all the clusters $C_i$ which sum up to one



## Expectation-Maximization (EM) Clustering

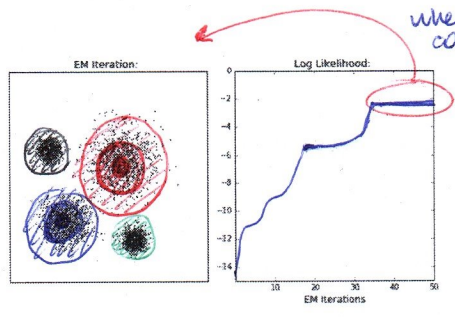


- The goal of maximum likelihood estimation (MLE) is to choose the parameters $\boldsymbol{\theta}$ that maximize the likelihood, that is

$$\theta^\star = \arg\max_\theta P(D|\theta)$$

- General idea
  - Starts with an initial estimate of the parameter vector
  - Iteratively rescores the patterns against the mixture density produced by the parameter vector
  - The rescored patterns are used to update the parameter updates
  - Patterns belonging to the same cluster, if they are placed by their scores in a particular component






when we arrive here the correspondent gaussians are those on the left

Clustering with Scikit with GIFs
https://dashee87.github.io/data%20science/general/Clustering-with-Scikit-with-GIFs/

## Example in One Dimension

## EM in One Dimension - Initialization



- Consider a dataset consisting of a single attribute $X = \{x_1, \ldots, x_n\}$
- Clusters will be represented using univariate normal,

$$f_i(x) = f(x|\mu_i, \sigma_i^2) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left\{-\frac{(x-\mu_i)^2}{2\sigma_i^2}\right\}$$

- Each cluster will be represented by the parameters $\{\boldsymbol{\mu}_i\ \boldsymbol{\sigma}_i^2\ P(C_i)\}$
- We initialize the parameters as follows, the mean $\boldsymbol{\mu}_i$ is selected uniformly at random, $\boldsymbol{\sigma}_i^2$ is initialized to 1, the cluster $P(C_i)$ probability with 1/k (where k is the number of clusters).

## EM in One Dimension – Expectation Step



- For each cluster, 1… k, we can use the current estimate of the parameters $\{\mu_i\ \sigma_i^2\ P(C_i)\}$ to compute the posterior probabilities,

$$P(C_i|x_j) = \frac{f(x_j|\mu_i, \sigma_i^2) \cdot P(C_i)}{\sum_{a=1}^{k} f(x_j|\mu_a, \sigma_a^2) \cdot P(C_a)}$$

- We denote with
  - $w_{ij}$ the values $P(C_i|x_j)$ = probability of point j to belong to cluster i
  - $w_i$ the weight vector for cluster $C_i$ over all the n points, that is, $(w_{i,1} \ldots w_{i,n})^T$



## EM in One Dimension – Maximization Step (1)



- Given all the posterior probability values $w_{ij}$ the maximization step computes the maximum likelihood estimates of the cluster parameters $\{\mu_i\ \sigma_i^2\ P(C_i)\}$
- The means are updated as,

$$\mu_i = \frac{\sum_{j=1}^{n} w_{ij} \cdot x_j}{\sum_{j=1}^{n} w_{ij}}$$

which can be rewritten as

$$\mu_i = \frac{\mathbf{w}_i^T X}{\mathbf{w}_i^T \mathbf{1}}$$



## EM in One Dimension – Maximization Step (2)



- Similarly, for $\sigma_i^2$

$$\sigma_i^2 = \frac{\sum_{j=1}^{n} w_{ij}(x_j - \mu_i)^2}{\sum_{j=1}^{n} w_{ij}} \quad \text{or} \quad \sigma_i^2 = \frac{\mathbf{w}_i^T Z_i^s}{\mathbf{w}_i^T \mathbf{1}}$$

where $Z_i = (x_1 - \mu_i, \ldots, x_n - \mu_i)^T$

- Finally,

$$P(C_i) = \frac{\sum_{j=1}^{n} w_{ij}}{\sum_{a=1}^{k} \sum_{j=1}^{n} w_{aj}} = \frac{\sum_{j=1}^{n} w_{ij}}{\sum_{j=1}^{n} 1} = \frac{\sum_{j=1}^{n} w_{ij}}{n}$$

- The expectation and maximization steps are repeated until convergence (e.g. until the means change a little between steps)



## Example with one variable (Example 13.4 from the textbook)

## Example 13.4 from the Textbook (1)

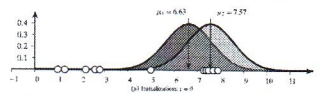


- Data Points

| | | | | | |
|---|---|---|---|---|---|
| $x_1 = 1.0$ | $x_2 = 1.3$ | $x_3 = 2.2$ | $x_4 = 2.6$ | $x_5 = 2.8$ | |
| $x_6 = 5.0$ | $x_7 = 7.3$ | $x_8 = 7.4$ | $x_9 = 7.5$ | $x_{10} = 7.7$ | $x_{11} = 7.9$ |

- Initialization of the parameters

| | | |
|---|---|---|
| $\mu_1 = 6.63$ | $\sigma_1^2 = 1$ | $P(C_1) = 0.5$ |
| $\mu_2 = 7.57$ | $\sigma_2^2 = 1$ | $P(C_2) = 0.5$ |

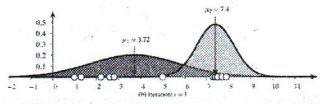


## Example 13.4 from the Textbook (2)



- After one expectation-maximitation iteraton the parameters are

| | | |
|---|---|---|
| $\mu_1 = 3.72$ | $\sigma_1^2 = 6.13$ | $P(C_1) = 0.71$ |
| $\mu_2 = 7.4$ | $\sigma_2^2 = 0.69$ | $P(C_2) = 0.29$ |



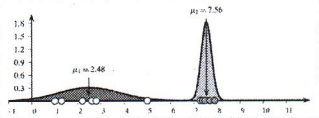


## Example 13.4 from the Textbook (3)



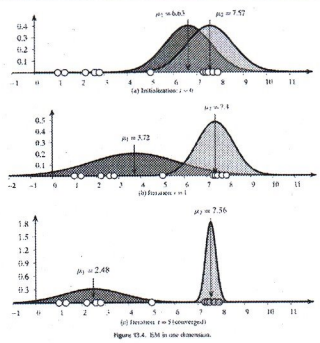
- The procedure converges after five iterations to

| | | |
|---|---|---|
| $\mu_1 = 2.48$ | $\sigma_1^2 = 1.69$ | $P(C_1) = 0.55$ |
| $\mu_2 = 7.56$ | $\sigma_2^2 = 0.05$ | $P(C_2) = 0.45$ |



- EM Clustering returns probabilities, it does not assign points to clusters.
- In this example, we assigned elements to the cluster with the highest posterior probability

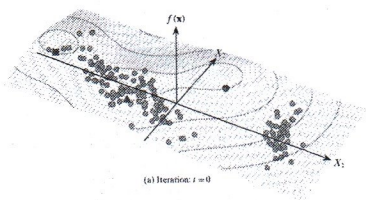




Example from Chapter 13 of the textbook

**Notice**: the number of clusters (k) is defined by the user

## Example with two variables




(a) Iteration: $t = 0$

Figure 13.5 from Chapter 13 of the textbook

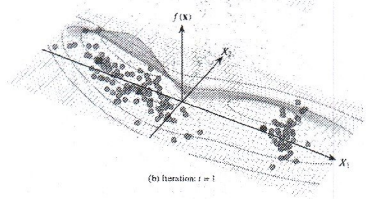



Figure 13.5 from Chapter 13 of the textbook (k=2)

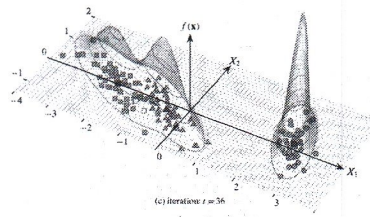



Figure 13.5 from Chapter 13 of the textbook (k=3)

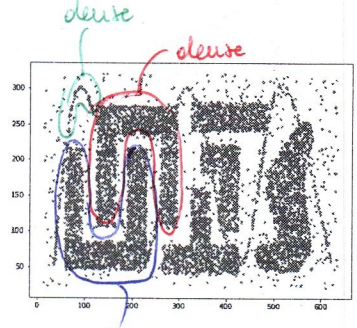
## Beyond convex clusters

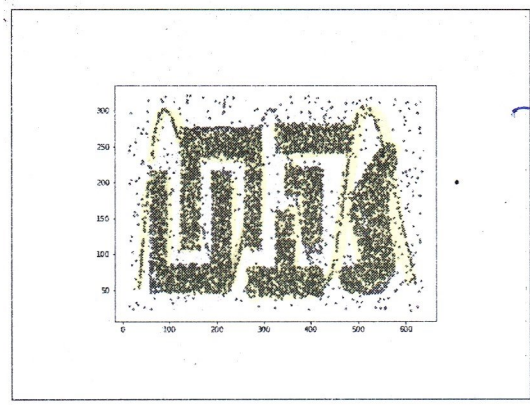
Representative-based clustering methods are suitable for finding ellipsoid-shaped clusters, or at best convex clusters

For nonconvex clusters, these methods have trouble finding the true clusters

Density-based methods can mine such nonconvex clusters

The idea is to find dense areas





We consider all the non yellow points as noise, so we look for clusters only within the yellows

## What is density-based clustering?



- Clustering based on density (local cluster criterion), such as density-connected points
- Major features:
  - Discover clusters of arbitrary shape
  - Handle noise
  - One scan
  - Need density parameters as termination condition

The user defines the definition of density (different meanings for densities lead to different results)



## DBSCAN: Basic Concepts



- The neighborhood within a radius ε of a given object is called the ε-neighborhood of the object

$$N_\epsilon(x) = \{y | \delta(x, y) \leq \epsilon\}$$

- **Core Point** (points in a dense area)
  - If the ε-neighborhood of an object contains at least **minpts** objects, then the object is a core object
- **Border Point** (points nearby a dense area)
  - If its ε-neighborhood does not contain at least **minpts,** but it is inside the neighborhood of a core point
- **Noise Point** (isolated points)
  - If its ε-neighborhood does not contain at least **minpts** and it does not belong to the neighborhood of a core point





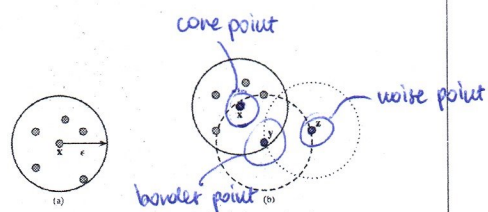



Figure 15.2. (a) Neighborhood of a point. (b) Core, border, and noise points.

Core, border and noise points when minpts is 6
"Data Mining and Analysis" by M.J. Zaki & W. Meira

Given these points and classifications, we want to connect dense areas

## DBSCAN: Basic Concepts

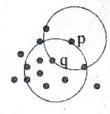


- **Directly density reachable**
  - An object x is directly density-reachable from object y if x is within the ε-neighborhood of y and y is a core object
- **Density Reachable**
  - An object x is density-reachable from object y if there is a chain of objects $x_1, \ldots, x_n$ where $x_1 = x$ and $x_n = y$ such that $x_{i+1}$ is directly density reachable from $x_i$
- **Density Connected**
  - An object p is density-connected to q with respect to ε and MinPts if there is an object o such that both p and q are density reachable from o
- **Density-Based Cluster**
  - A density-based cluster is defined as a maximal set of density connected points.

## Density-Reachable & Density-Connected

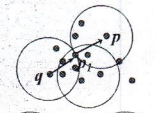


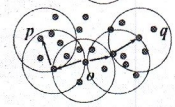
Directly density-reachable




MinPts = 5

Density-reachable




Density-connected



## DBSCAN: Basic Concepts



- Density corresponds to have at least minpts points within a specified radius ε
- A border point has fewer than minpt within ε, but is in the neighborhood of a core point
- A noise point is any point that is not a core point nor a border point



**ALGORITHM 15.1. Density-based Clustering Algorithm**

```
DBSCAN (D, ε, minpts):
Core ← ∅
foreach x_i ∈ D do // Find the core points
    Compute N_ε(x_i)
    id(x_i) ← ∅ // cluster id for x_i
    if N_ε(x_i) ≥ minpts then Core ← Core ∪ {x_i}
k ← 0 // cluster id
foreach x_i ∈ Core, such that id(x_i) = ∅ do
    k ← k + 1
    id(x_i) ← k // assign x_i to cluster id k
    DENSITYCONNECTED (x_i, k)
C ← {C_i}_{i=1}^{k}, where C_i ← {x ∈ D | id(x) = i}
Noise ← {x ∈ D | id(x) = ∅}
Border ← D \ {Core ∪ Noise}
return C, Core, Border, Noise
DENSITYCONNECTED (x, k):
foreach y ∈ N_ε(x) do
    id(y) ← k // assign y to cluster id k
    if y ∈ Core then DENSITYCONNECTED (y, k)
```

Density Based Scan

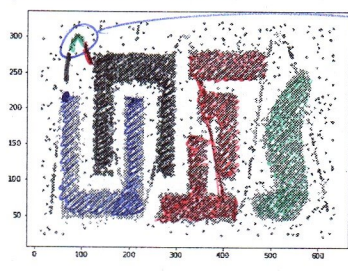
## Complexity of DBScan



- DBSCAN needs to compute the ε-neighborhood for each point
- If the dimensionality is not too high this can be done efficiently using a spatial index structure in O(n log n)
- If the dimensionality is high, it takes $O(n^2)$ to compute the neighborhood for each point.
- Once the ε-neighborhood has been computed the algorithm needs only a single pass over all the points to find the density connected clusters.
- The overall complexity of DBSCAN is between O(n log n) and in the worst-case.



Result of the DBScan




It can generate very small clusters

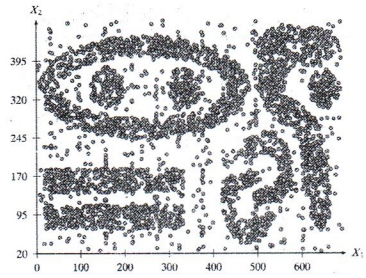



Figure 15.1 from Chapter 15 of the textbook "Data Mining and Analysis" by MJ. Zaki & W. Meira

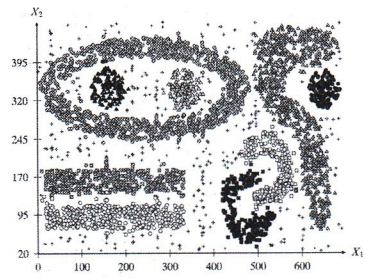



Figure 15.3 from Chapter 15 of the textbook "Data Mining and Analysis" by MJ. Zaki & W. Meira

Let's play with DBSCAN

https://www.naftaliharris.com/blog/visualizing-dbscan-clustering/

DBScan fails when applied to data of varying density !

HDBSCAN

Hierarchical clustering + Density based clustering

## HDBSCAN



- Extends DBSCAN by converting it into a hierarchical clustering algorithm, and then using a technique to extract a flat clustering based in the stability of clusters
- It defines the "core distance" of a point as the maxima distance of a point to its k-th nearest neighbor or $core_k(x)$
- Thus, points in high density areas will have a low core distance, while point in low density areas will have a high core distance
- This provides a local inexpensive estimate of density



## Mutual Reachability Distance



- This new distance metric is defined as,

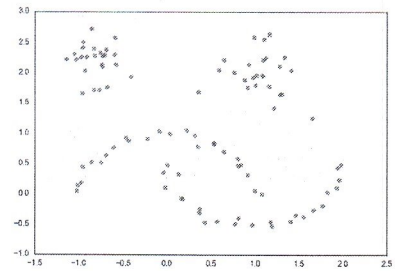
$$d_{mreach-k}(a,b) = \max\{core_k(a), core_k(b), d(a,b)\}$$

using the core distance and the original distance metrics d

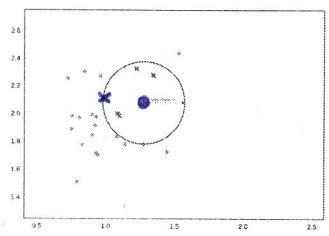
- This distance keeps dense points with lower core distance at the same distance while pushing away sparser points (at least at their core distance from any other point).
- Note that as with DBSCAN high values of k focus the algorithm toward very dense region



The goal is to remap the points in another space where dense points are kept at the same distance while sparse points are pushed away



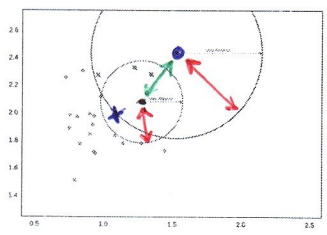
https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html



Let's consider a k equal to 5 and plot the core distance for one point.
https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html

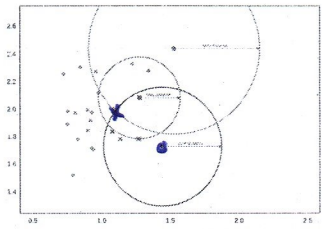
We consider the point and the 5 closest to it. The core distance of this point is the maximum of the 5 distances (the distance of the point with the far-est point of the 5)

dense area → small core dist.

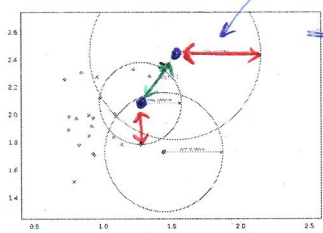


This point is in a lower density area and thus it has a higher core distance.
https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html

(MUTUAL REACHABILITY DISTANCE)
Now that we have two points.
The reachability is the maximum between three distances:
the two cores distances and the (original) distance between the two points

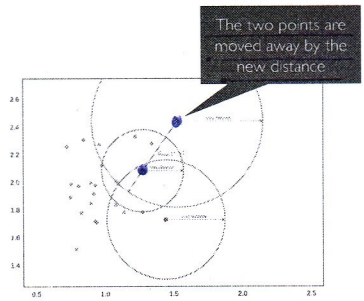


And let's add another point with its core distance
https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html

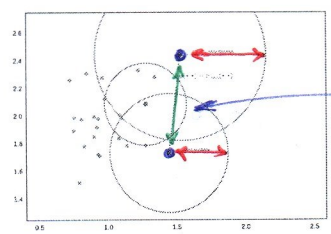


this it the largest among the 3 distances
→ this is the MUTUAL REACHABILITY DISTANCE between the two blue points
→ the two points will be pushed away

To compute the mutual reachability distance between blue and green points we draw d(,,) which crosses the blue circle but not the green one. So the green core distance is the largest value.




To compute the mutual reachability distance between blue and green points we draw d(,,) which crosses the blue circle but not the green one. So the green core distance is the largest value.



this is the longest distance among the 3 possible distances
→ the two blue points will be kept at the same distance (not pushed away)

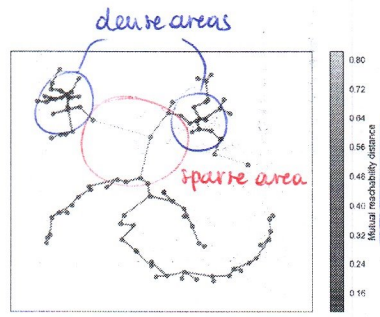
However, the mutal reachability distance between red and green is the original distance d(. , .) which crosses both the red and green circle.

Once we know which points to push and which not

## Next, Build the Minimum Spanning Tree using Prim's Algorithm

130

- Consider the data as a weighted graph
- Data points are vertices
- An edge between any two points with weight equal to the mutual reachability distance of those points
- Build the spanning tree one edge at a time by adding the lowest weight edge that connects the current tree to a vertex not yet in the tree

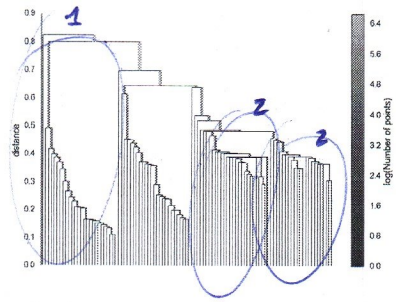



https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html

HDBSCAN converts the minimum spanning tree into a hierarchy of connected components

It sorts the edges of the tree by distance (in increasing order)

Then iterate creating a new merged cluster for each edge.




https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html

We have a hierarchy now
we want a set of flat clusters

**Condensing the Tree** 135

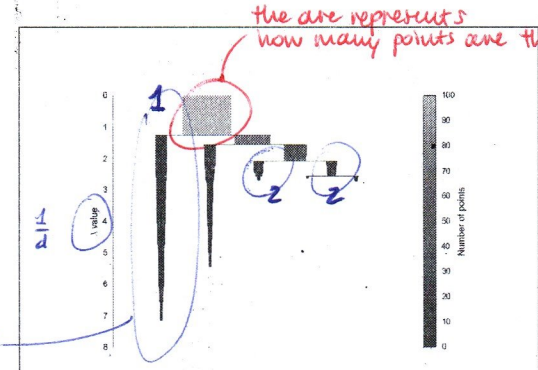
- We introduce the notion of "minimum cluster size" threshold
- We navigate the hierarchy and from the top and at each split we check the size of the merged clusters
- If a clusters has fewer points that the threshold, then the smaller clusters is eliminated (the points have "fallen out of the cluster")
- Otherwise both clusters are maintained.
- Note that, which a point p 'fell out of the cluster' and at what distance value that happened is stored as $\lambda_p = 1/\text{distance}$
- At the end, a much smaller tree with a small number of nodes remains with node has data about how the size of the cluster at that node decreases over varying distance

Prof. Pierluca Lanzi POLITECNICO DI MILANO

The idea: merges between clusters that are small are not so interesting. We want to find structures that last for a long time (structures that last during a lot of distances)
In the structure **1** we see a structure that lasts a lot (we keep adding points to an existing structure). In the structure **2** we see a "desperate attempt" of merging points (small clusters merged without a proper criterium) *

* This difference between the different mergings can be syntetized in the next (condensed) dendrogram

the are represents how many points are there




Condensed tree (λ = 1/distance)
https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html

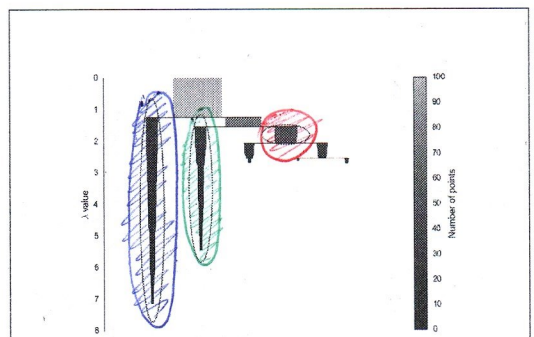
This is a clear representation of a structure that is present for a long time (unlike 2)

## Extracting the Cluster

137

- Given the simplified dendrogram we want to select clusters that persist and have a longer lifetime since short lived clusters probably represent artifacts
- We may say that we want to choose those clusters that have the greatest area of ink in the plot. Also, we cannot select any a descendant of a cluster we chose.

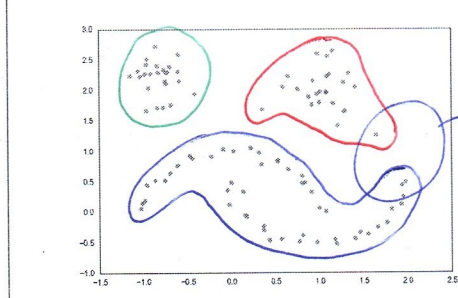





By applying the two principles (larger total ink area and no descendents) we extract three clusters
https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html

Finally, we can assign to each point a probability to belong to a cluster based on its $\lambda_p$

HDBSCAN complexity is comparable and lower than DBSCAN



Finally, we can assign to each point a probability to belong to a cluster based on its $\lambda_p$
https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html

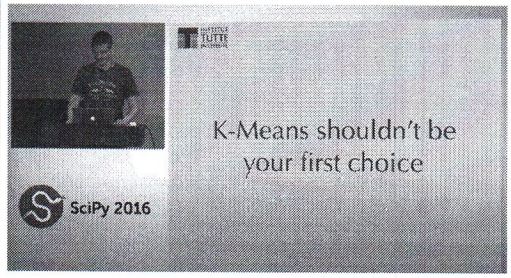
notice that these points are colored with a color which is in between red and blue since the color represents the probability of belonging to a class (it **not** a strict labeling)

## These are only the basic ones

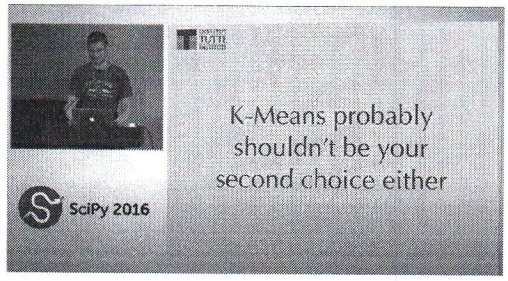


- Birch and Chameleon are other hierarchical clustering algorithms
- There are also probabilistic hierarchical clustering algorithms
- OPTICS, and DENCLU are interesting density-based algorithms
- STING and CLIQUE are grid-based methods, where CLIQUE is also a subspace clustering algorithm
- …






https://www.youtube.com/watch?v=AgPQ76RIi6A




https://www.youtube.com/watch?v=AgPQ76RIi6A


https://www.youtube.com/watch?v=AgPQ76RIi6A

## Study Material



- "Data Mining and Analysis" by Zaki & Meira
  - Chapter 13
  - Chapter 15


- http://www.dataminingbook.info
- https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html