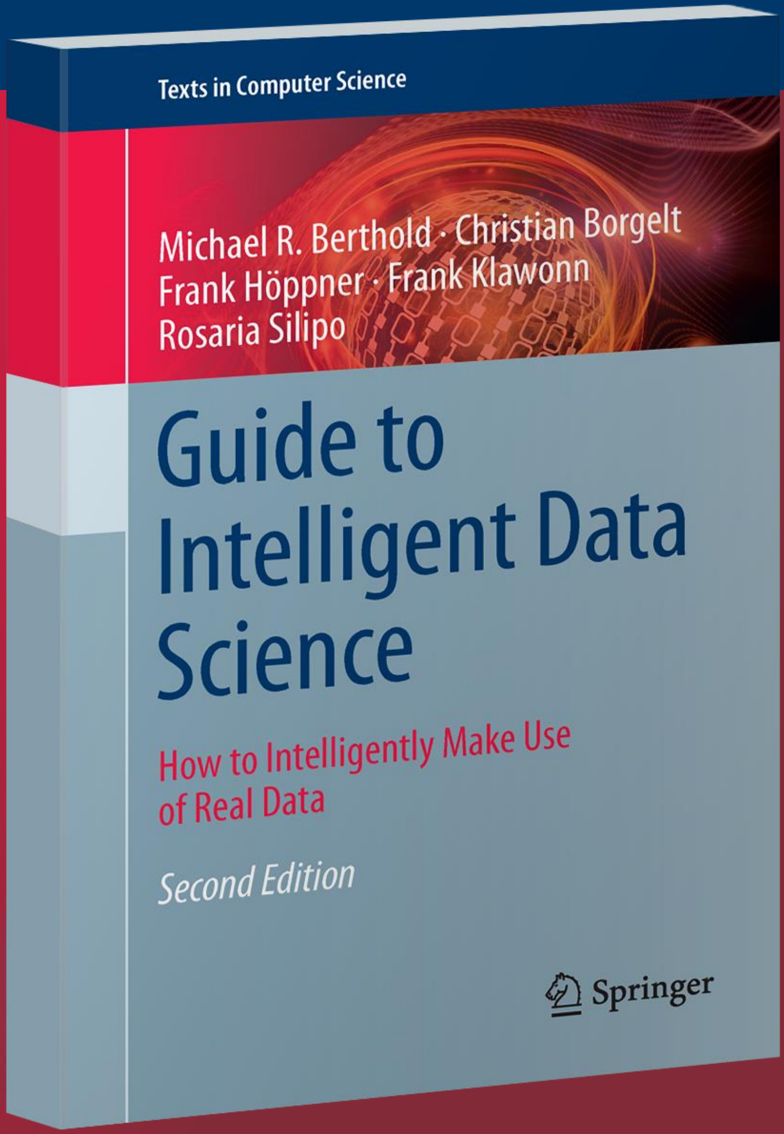
Clustering


Texts in Computer Science
Michael R. Berthold · Christian Borgelt
Frank Höppner · Frank Klawonn
Rosaria Silipo
Guide to Intelligent Data Science
How to Intelligently Make Use of Real Data
Second Edition
Springer

Summary of this lesson

„Data are becoming the new raw material of business”
-Craig Mundie

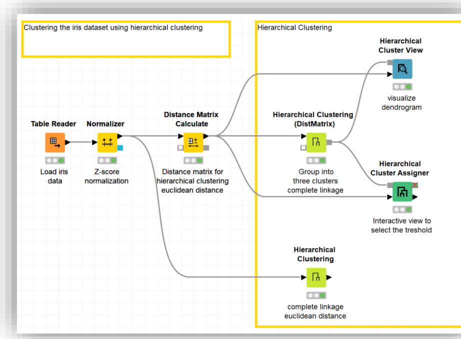
How do we find clusters of similar data objects?

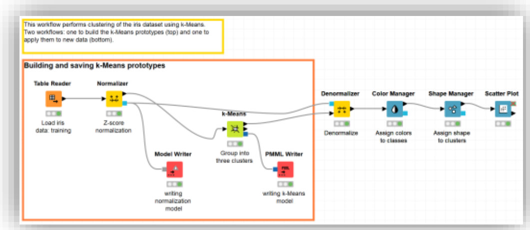
**This lesson refers to chapter 7 of the GIDS book*

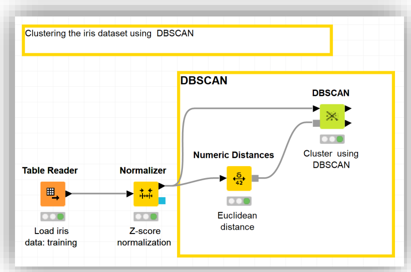
Content of this lesson

- Clustering
- Similarity Measures
- Hierarchical Clustering
  - DIANE
  - AGNES
- Prototype or Model –based Clustering
  - K-Means
- Quality Measures
- Gaussian Mixture Model
- Density based Clustering
  - DBScan

- Dataset used : iris dataset
- Example Workflows:
  - „Hiearchical Clustering“ https://kni.me/w/rIXFxYxQmbgNgSsM
    - Normalization
    - Distance calculation
    - Hiearachical clustering
  - „K-means clustering“ https://kni.me/w/t8UVEQH1sTTkus_w
    - Partitioning
    - Numeric error measures
  - „DBSCAN“ https://kni.me/w/WgQVjVJugr9Nu24W
    - Normalization
    - Distance caluclation
    - Clustering







Unsupervised Learning

What is a Learning Algorithm?

- Input features
- Input attributes
- Independent variables

$\boldsymbol{X} = (x_1, x_2, \ldots, xn)$

Model

y

- Class
- Label
- Target
- Output feature/attribute
- Dependent variable

Model parameters

$$y = f(\boldsymbol{\beta}, \boldsymbol{X}) \quad \text{with } \boldsymbol{\beta} = [\beta_1, \beta_2, \ldots, \beta_m]$$

A learning algorithm adjusts (learns) the model parameters $\boldsymbol{\beta}$ throughout a number of iterations to maximize/minimize a likelihood/error function on y.

Algorithm Training / Learning

- The model ***learns*** / ***is trained*** during the ***learning / training*** phase to produce the right answer *y* (a.k.a., label)

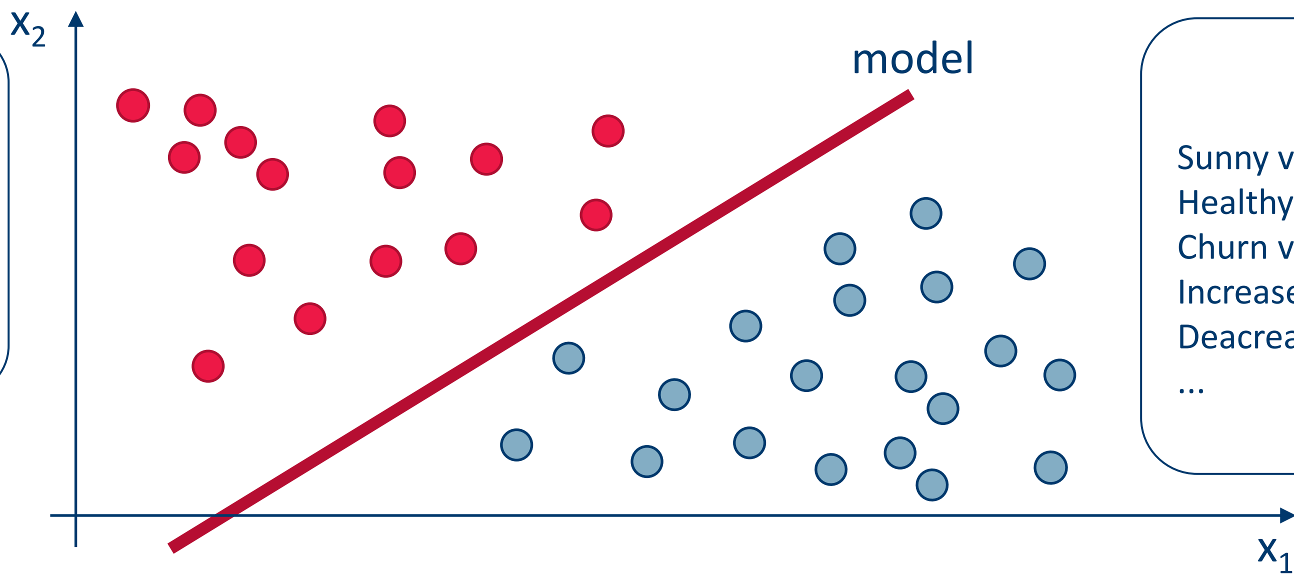
- That is why ***machine learning*** ☺

- Many different algorithms for three ways of learning:
  - Supervised
  - Unsupervised
  - Semi-supervised

- $\boldsymbol{X} = (x_1, x_2)$ and $y = \{yellow, gray\}$
- A training set with many examples of $(\boldsymbol{X}, y)$
- The model learns on the examples of the training set to produce the right value of y for an input vector $\boldsymbol{X}$

$\boldsymbol{X}$

Age
Money
Temperature
Speed
Number of taxi
…

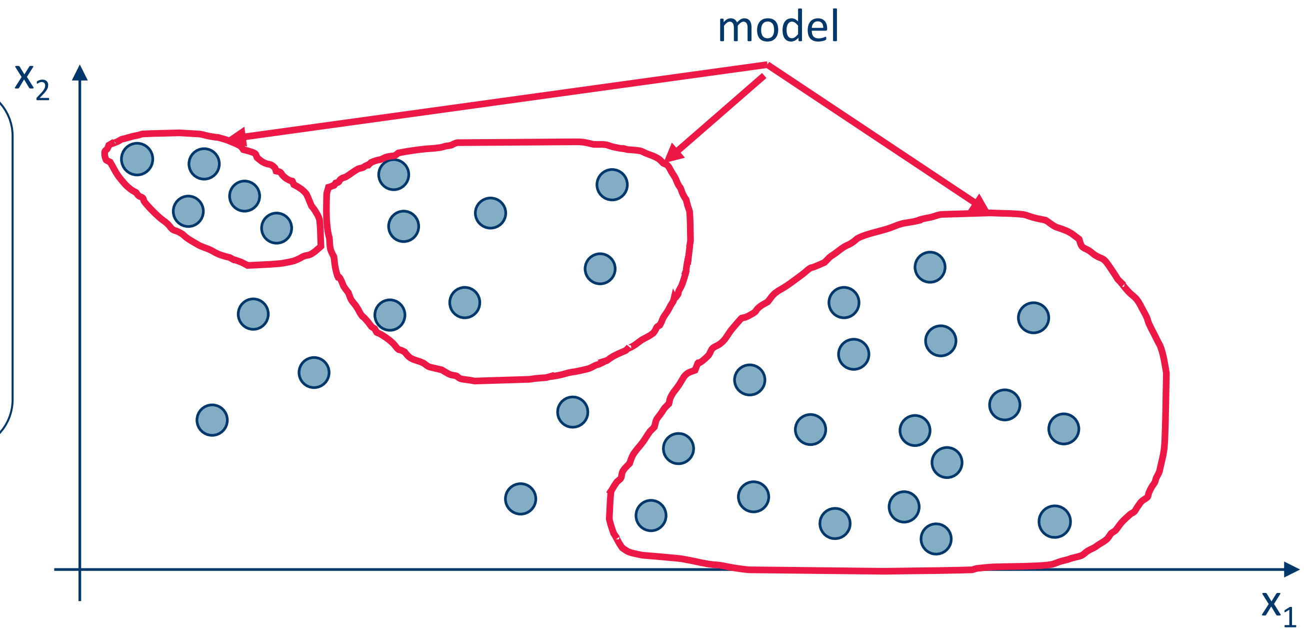


y

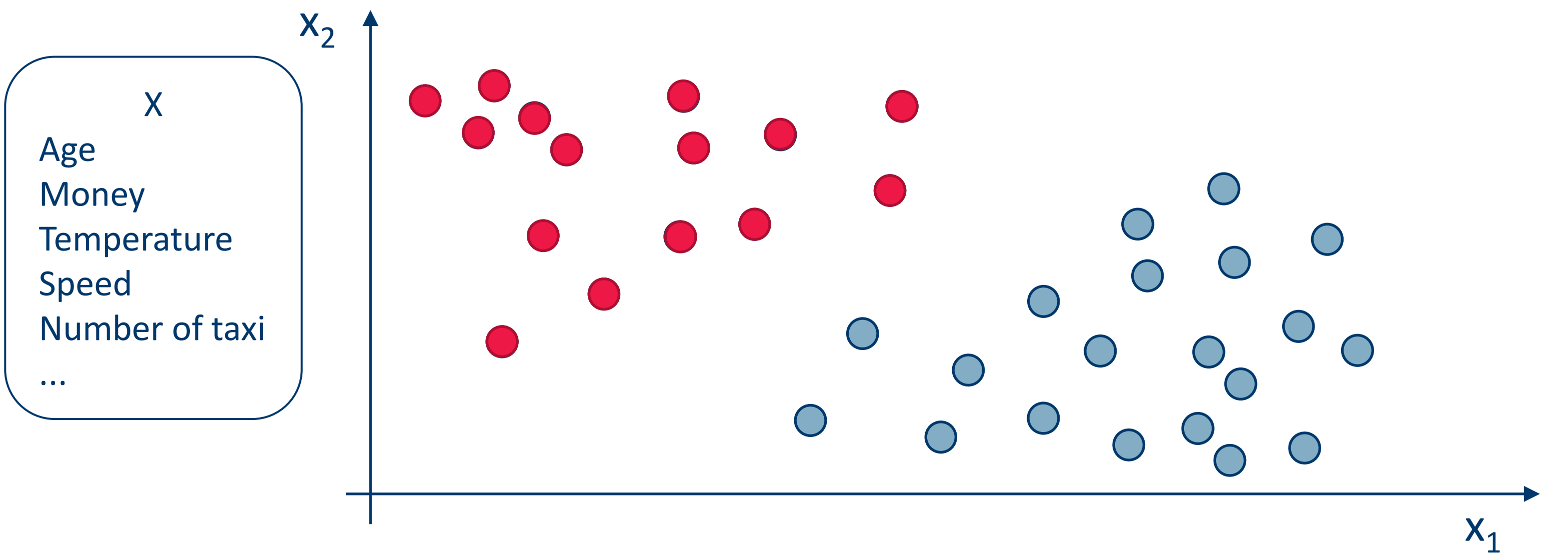
Sunny vs. Cloudy
Healthy vs. Sick
Churn vs. Remain
Increase vs. Deacrease
…

- $\boldsymbol{X} = (x_1, x_2)$ ~~and $y = \{yellow, gray\}$~~
- A training set with many examples of $(\boldsymbol{X}, \cancel{y})$
- The model learns to group the examples $\boldsymbol{X}$ of the training set based on similarity (closeness) or probability

$\boldsymbol{X}$
Age
Money
Temperature
Speed
Number of taxi
...



- $\boldsymbol{X} = (x_1, x_2)$ and $y = \{yellow, gray\}$
- A training set with many examples of $(\boldsymbol{X}, \cancel{y})$ and some samples $(\boldsymbol{X}, y)$
- The model labels the data in the training set using a modified unsupervised learning procedure



- $\boldsymbol{X} = (x_1, x_2)$ and $y = \boxed{\{label\ 1, \ldots, label\ n\}}$ or $y \in \mathbb{R}$
- A training set with many examples of $(\boldsymbol{X}, y)$
- The model learns on the examples of the training set to produce the right value of y for an input vector $\boldsymbol{X}$

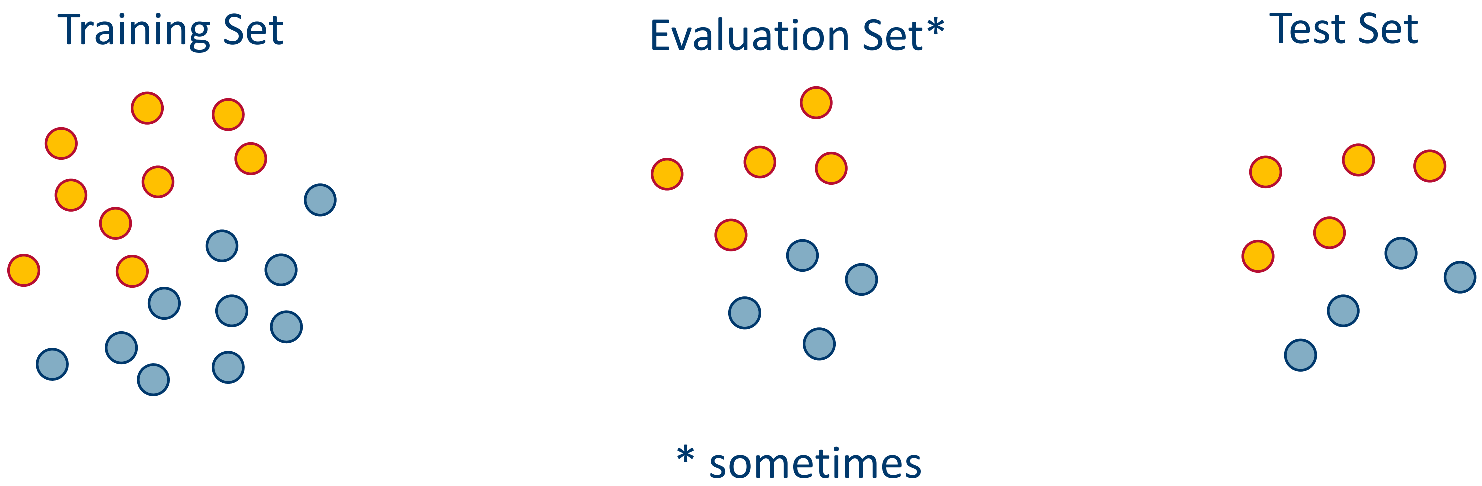
Classification

y = {yellow, gray}
y = {churn, no churn}
y = {increase, unchanged, decrease}
y = {blonde, gray, brown, red, black}
y = {job 1, job 2, ... , job n}

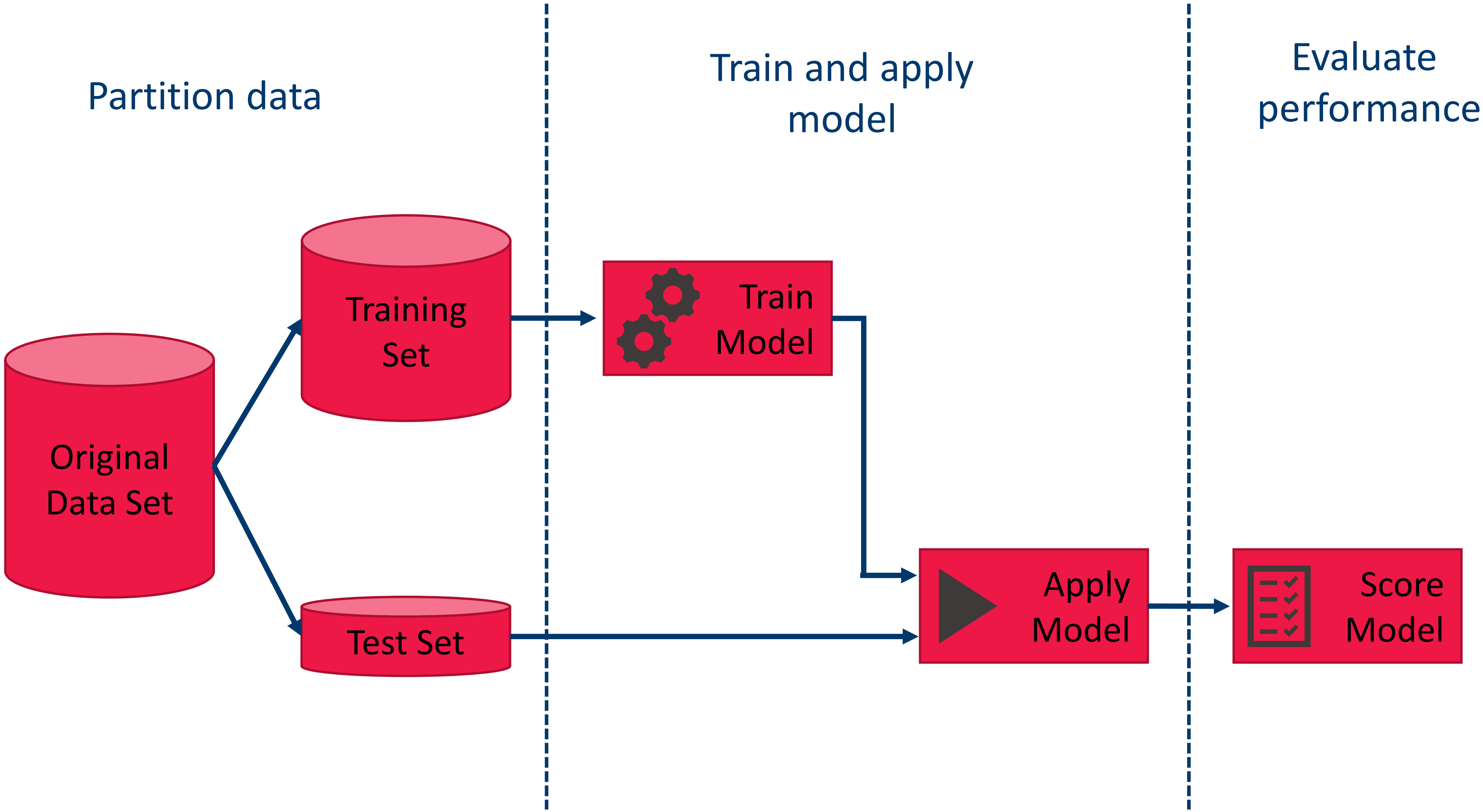
Numerical Predictions

y = temperature
y = number of visitors
y = number of kW
y = price
y = number of hours

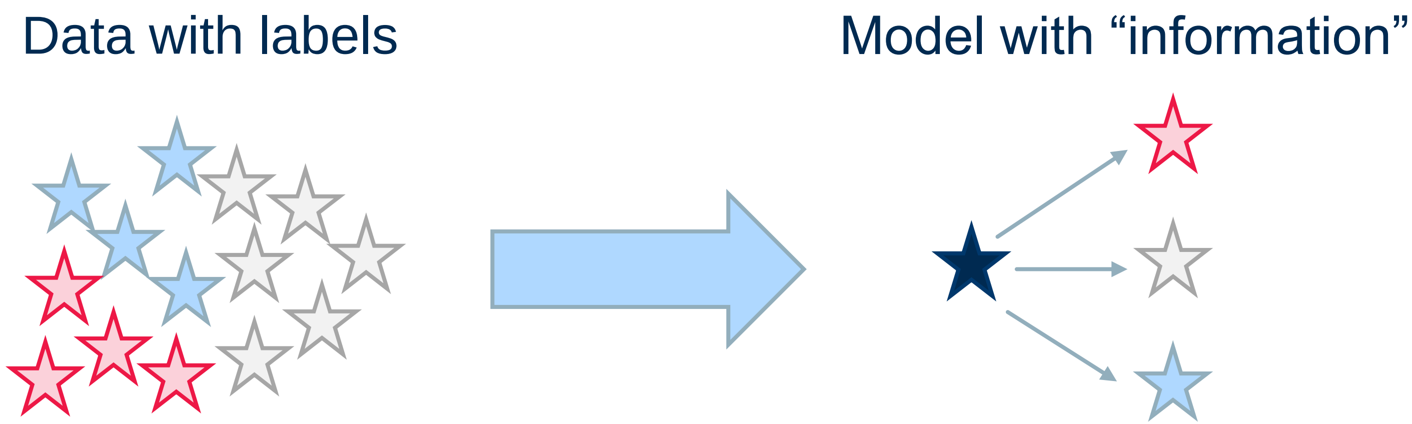
- ***Training phase***: the algorithm trains a model using the data in the training set
- ***Testing phase***: a metric measures how well the model is performing on data in a new dataset (the test set)



Data Science: Process Overview



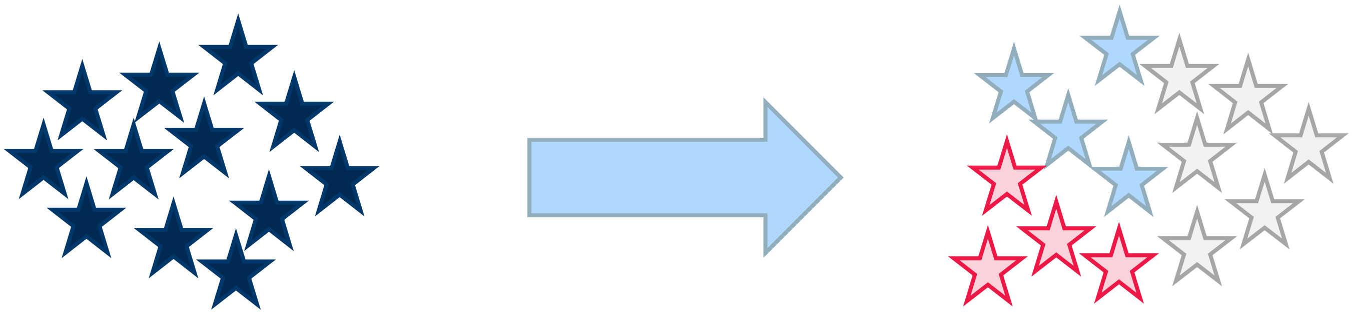
– Supervised Learning



– Unsupervised Learning

Data without labels

Data with “information”



Finding Patterns

Finding Patterns

- Patterns **describe or summarize** the data set or parts of it.
- Finding patterns is an exploratory data analysis task. There is **no specific target attribute** whose values should be predicted as in supervised learning.

- Methods that have already been discussed in the context of data understanding:
  - **Visualisation** techniques like scatter plots or MDS are methods for finding patterns by visual inspection.
  - **Correlation analysis** can find patterns in the form of dependencies of pairs of variables.

- **Cluster analysis.** Identifying groups of "similar" data objects.

- **Association rules.** Finding associations between attributes or typical combinations of values like:

IF *Demand=high* and *Supply=low* THEN *Price=high*.

Clustering

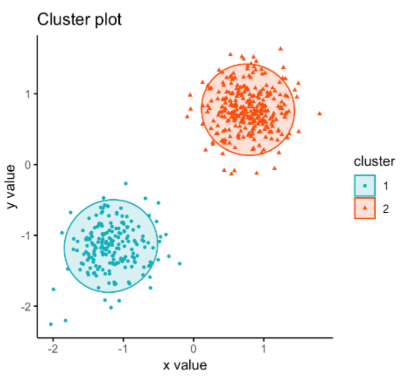
Goal of Cluster Analysis

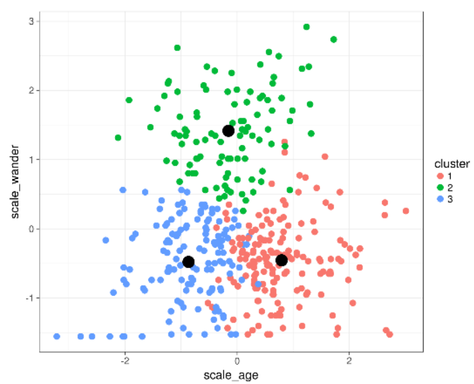
Discover hidden structures in unlabeled data (unsupervised)

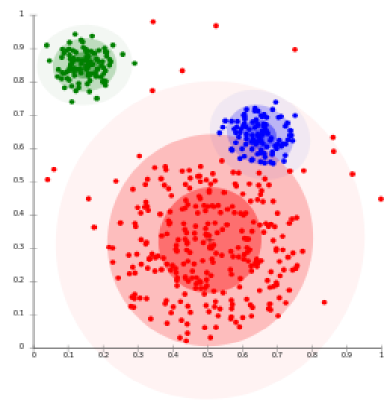
Clustering identifies a finite set of groups (*clusters*) $C_1, C_2 \cdots, C_k$ in the dataset such that:

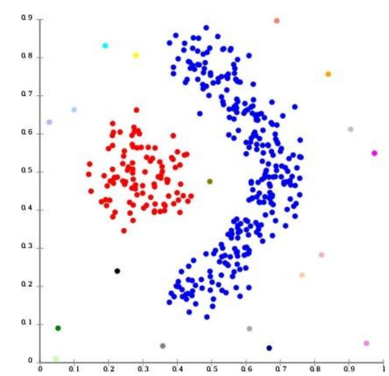
- Objects within the *same* cluster C_i shall be as similar as possible
- Objects of *different* clusters C_i, C_j $(i \neq j)$ shall be as dissimilar as possible

- Clusters may have different sizes, shapes, densities
- Clusters may form a hierarchy
- Clusters may be overlapping or disjoint









- Find “natural” clusters and desc
  → **Data understanding**
- Find useful and suitable groups
  → **Data Class Identification**
- Find representatives for homogenous groups
  → **Data Reduction**
- Find unusual data objects
  → **Outlier Detection**
- Find random perturbations of the data
  → **Noise Detection**

Definition:

Given a data set $D, |D| = n.$ Determine a *clustering C of D* with:

$$C = \{C_1, C_2, \cdots, C_\mathrm{k}\} \quad \text{where} \quad C_i \subseteq D \quad \text{and} \bigcup_{1 \le i \le k} C_i = D$$

that best fits the given data set D.

Inside the space

Cover the whole space

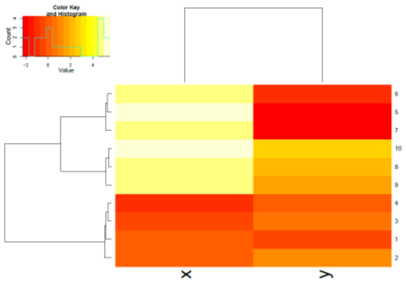
Clustering Methods:

1. partitioning
2. hierarchical (linkage based)
3. density-based

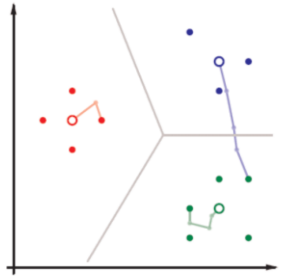
Application Examples

- Customer segmentation
  - Find groups of customers with similar behaviour; find customers with unusual behavior
- Molecule search
  - Find molecules with similar structure to already working ones
- Anomaly detection
  - Find unusual patterns in data from sensors monitoring mechanical engines
- Determining user groups on the WWW
  - *Clustering of activities in web-logs*
  - *Find groups of social media users with similar attitude.*
- Structuring large sets of text documents
  - *hierarchical clustering of the text documents*
- Generating thematic maps from satellite images
  - *clustering sets of raster images of the same area (feature vectors)*

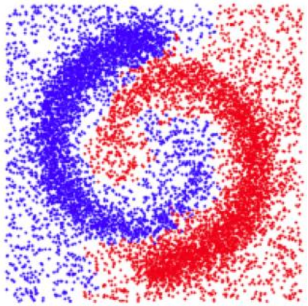
Linkage Based
e.g. Hierarchical Clustering



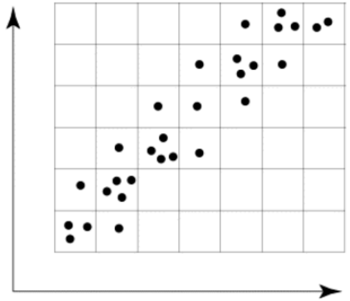
Clustering by Partitioning
e.g. k-Means



Density based Clustering
e.g. DBSCAN



Grid based Clustering



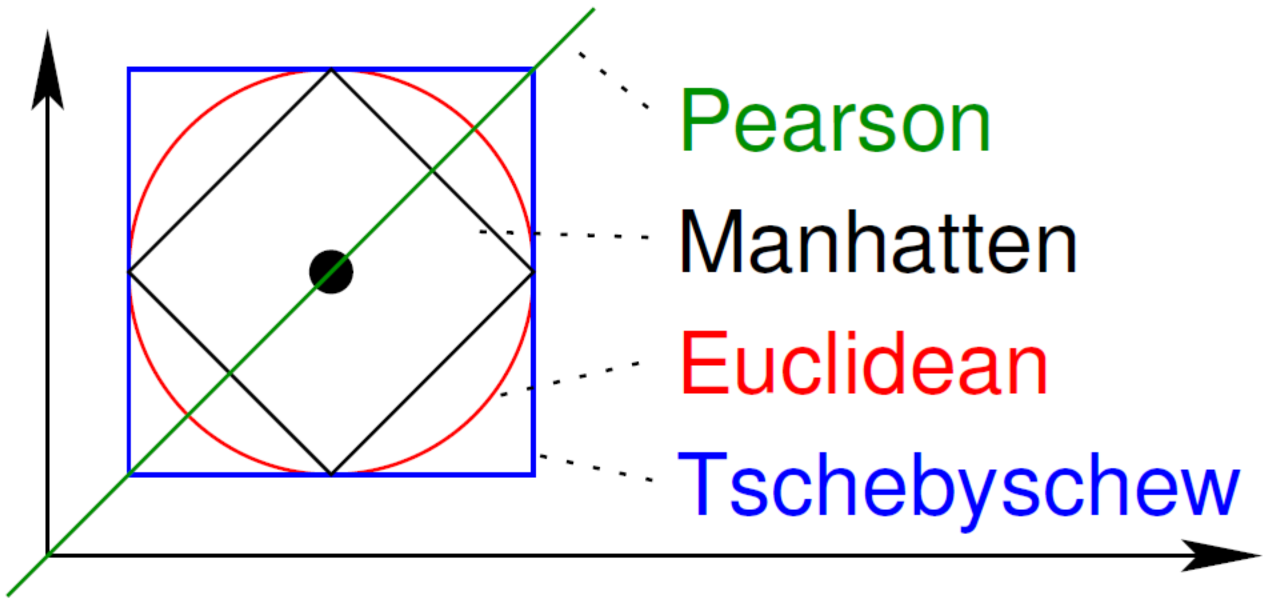
Similarity Measures

(Dis-)similarity Functions for Numeric Attributes

For two objects $\boldsymbol{x} = (x_1, x_2, \cdots, x_d)$ and $\boldsymbol{y} = (y_1, y_2, \cdots, y_d)$:

- *Minkowski-Distance (L_p-Metric)* $$d_p(\boldsymbol{x}, \boldsymbol{y}) = \sqrt[p]{\sum_{i=1}^{d} |x_i - y_i|^p}$$
- *Euclidean Distance (L_2 $- p = 2$)* $$d_E(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$$
- *Manhattan-Distance (L_1 $- p = 1$)* $$d_M(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{d} |x_i - y_i|$$
- *Tschebyschew-Distance (L_∞ $- p = \infty$)* $$d_\infty(\boldsymbol{x}, \boldsymbol{y}) = \max_{1 \le i \le d} \{|x_i - y_i|\}$$
- *Cosine Distance* $$d_C(\boldsymbol{x}, \boldsymbol{y}) = 1 - \frac{\boldsymbol{x}^T \boldsymbol{y}}{\|\boldsymbol{x}\| \|\boldsymbol{y}\|}$$
- *Tanimoto Distance* $$d_T(\boldsymbol{x}, \boldsymbol{y}) = 1 - \frac{\boldsymbol{x}^T \boldsymbol{y}}{\|\boldsymbol{x}\|^2 + \|\boldsymbol{y}\|^2 - \boldsymbol{x}^T \boldsymbol{y}}$$
- *Pearson Distance* Euclidean distance of z-score transformed $\boldsymbol{x}, \boldsymbol{y}$

– Various choice of dis-similarity between two numerical vectors



(Dis-)similarity Functions for Binary Attributes

- The two values (e.g. 0 and 1) of a binary attribute can be interpreted as some property being absent (0) or present (1).

- In this sense, a vector of binary attribute can be interpreted as a set of properties that the corresponding object has.

- Example:
  - The binary vector (0, 1, 1, 0, 1) corresponds to the set of properties $\{a_2, a_3, a_5\}$.
  - The binary vector (0, 0, 0, 0, 0) corresponds to the empty property set {}.
  - The binary vector (1, 1, 1, 1, 1) corresponds to the property set $\{a_1, a_2, a_3, a_4, a_5\}$.

(Dis-)similarity Functions for Binary Attributes

- Dissimilarity measures for two vectors of binary attributes.
- Each data object is represent by the corresponding set of properties.

- Simple match $$d_S = 1 - \frac{b+n}{b+n+x}$$
- Russel & Rao $$d_R = 1 - \frac{b}{b+n+x}$$
- Jaccard $$d_J = 1 - \frac{b}{b+x}$$
- Dice $$d_D = 1 - \frac{2b}{2b+x}$$

where b, n, and x are the number of predicates that:

- b = … hold in both records
- n = … do not hold in both records
- x = … holds in only one of the two records

Defining Distance – Text Data & Time Series

- For text data, convert to bag-of-words
- Word frequency vectors are normalized according to the text length
- Cosine measures to assess similarity between documents

- For time series or text data, apply z-score transformation to the data
- Euclidean distance ➔ Pearson metric
- Invariant to shifting – useful for embedded time series with different offsets

– Example:

x	y	Set X	Set Y	b n x	d_S d_R d_J d_D
101000	111000	$\{a_1, a_3\}$	$\{a_1, a_2, a_3\}$	2 3 1	0.16 0.66 0.33 0.20

(Dis-)similarity Functions for Ordinal Attributes

- Convert to a binary vector – similar in closer categories
- Convert to a numerical rank
- Example:

Number of previous owners	Binary Vector
1 previous owner	100
2 previous owners	110
More than 2 previous owners	111

Number of previous owners	Rank
1 previous owner	1
2 previous owners	2
More than 2 previous owners	3

- Nominal attributes may be transformed into a set of binary attributes, each of them indicating one particular feature of the attribute (1-of-*n* encoding).
- Example:
  - Attribute "*Manufacturer*" with 6 possible values *BMW, Chrysler, Dacia, ...* .

Manufacturer	Binary Vector
Volkswagen	000001
Dacia	000010
Ford	100000

  - Then one of the dissimilarity measures for binary attribute can be applied.

- Clustering vehicles:
  - red Ferrari
  - green Porsche
  - red Bobby car



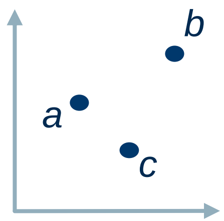
- Distance Function based on maximum speed (numeric distance function):
  - Cluster 1: Ferrari & Porsche
  - Cluster 2: Bobby car

The distance function affects the shape of the clusters

- Distance Function based on color (nominal attributes):
  - Cluster 1: Ferrari and Bobby car
  - Cluster 2: Porsche

Hierarchical Clustering

id	X	Y
a	1	2
b	3	3
c	2	1



- Given data points, how can we summarize how different they are?
  - Rather than summarizing the similarity
- Summarized by a single dissimilarity metric – ***distance***
- Distance matrix – pairwise differences of all data points

$[d_{i,j}]$	a	b	c
a	0.00	2.23	1.41
b	2.23	0.00	2.23
c	1.41	2.23	0.00

- Distance $d_{i,j}$ (between i and j) calculated as the Euclidean distance

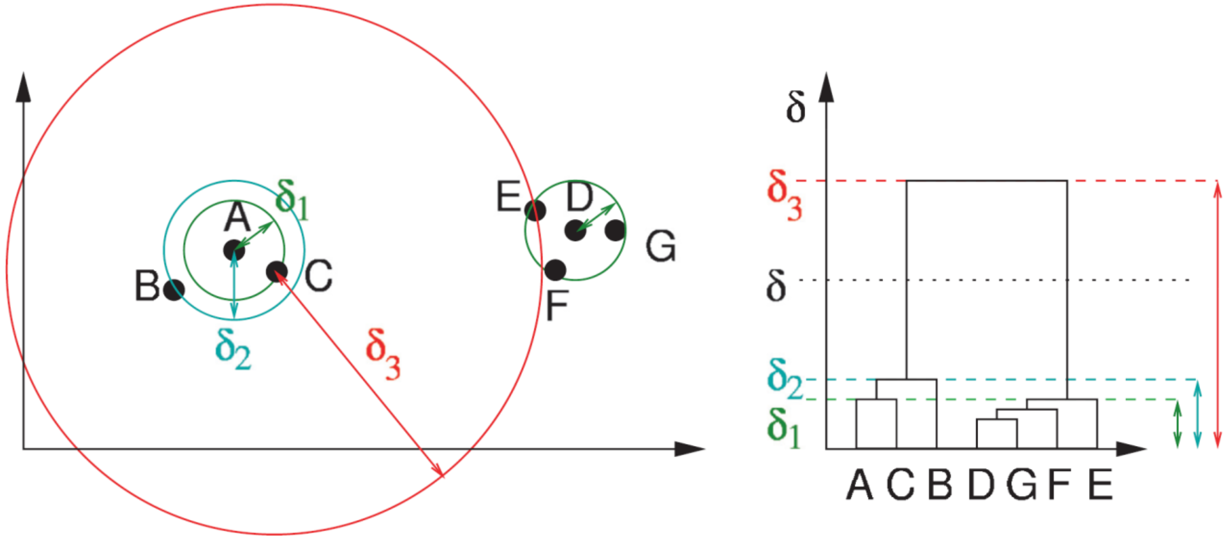
- Partition P – a set of clusters $\{C_1, C_2, \cdots, C_C\}$
- Cluster C_i – a non-empty subset of data
- Each data point belongs to a single cluster
  - No overlap between clusters
- Union of clusters ➔ entire data set

- Data points with $distance < \delta$ ➔ belong to the same cluster
- Known as ***agglomerative*** clustering
- Different values of δ ➔ different partitions

- What should be the ideal δ?
- ***Stable*** and ***robust*** clusters – small alterations should not produce completely different clusters

- Clusters found at δ_1 are contained in clusters found at δ_2 (for $\delta_1 < \delta_2$)
- Increasing δ results in a hierarchy of clusters

– Described by a ***dendrogram***



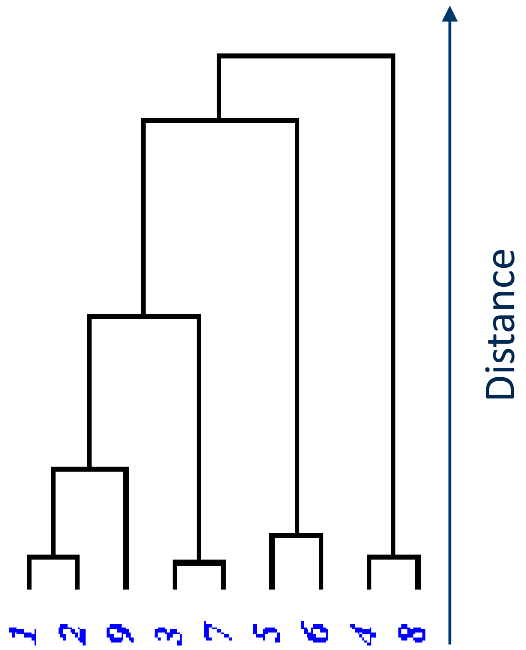
- At $\delta = 0$ ➔ 7 clusters of singletons
- At $\delta = \mathrm{d}(\mathrm{D}, \mathrm{G})$ ➔ D & G form a cluster, all others are singleton clusters
- At $\delta = \mathrm{d}(\mathrm{A}, \mathrm{C})$ ➔ A & C form a cluster, B remains a singleton, D, E, F, & G is another cluster
- And so on

Goal

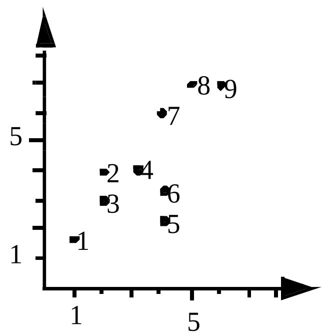
- Construction of a hierarchy of clusters (*dendrogram*) by merging/separating clusters with minimum/maximum distance

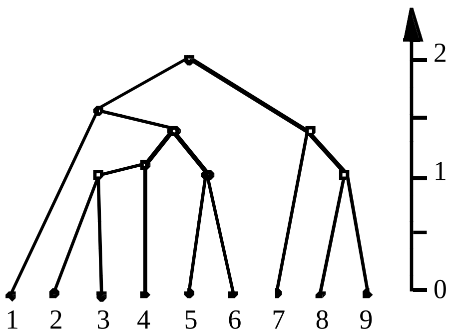
Dendrogram:

- A tree representing a hierarchy of clusters, with the following properties:
  - Root: single cluster with the whole data set.
  - Leaves: clusters containing a single object.
  - Branches: merges / separations between larger clusters and smaller clusters / objects



- Example dendrogram

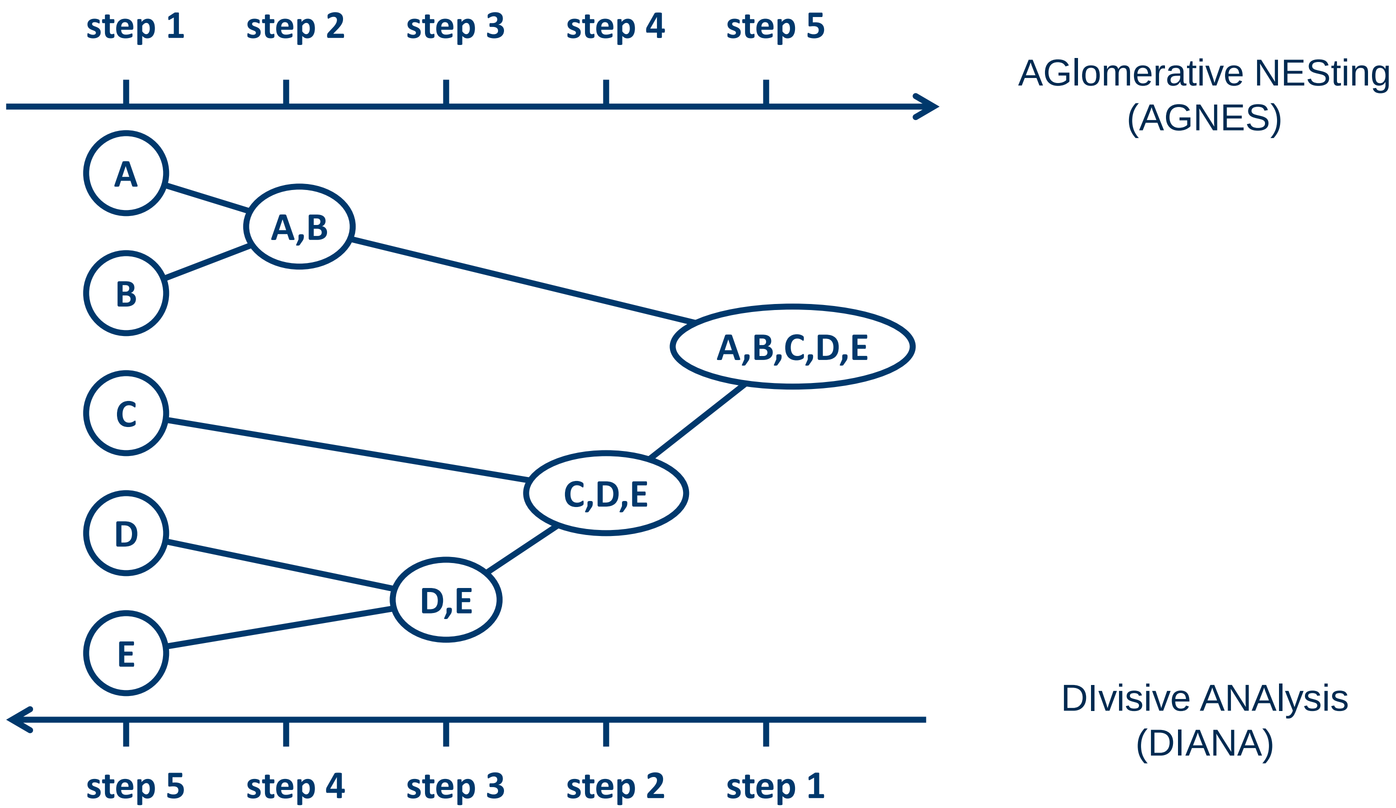




distance between clusters

- Types of hierarchical methods
  - Bottom-up construction of dendrogram (*agglomerative*)
  - Top-down construction of dendrogram (*divisive*)

Agglomerative vs. Divisive Hierarchical Clustering



Clustering algorithm

- Start at $\delta = 0$, with each data point as a cluster
- Calculate the distance matrix between all clusters
- Merge the two clusters with smallest distance
- Go back to re-calculate the distance matrix
- Repeat until there is only a single cluster

Algorithm HC($\mathcal{D}$) : $(\mathcal{P}_t)_{t=0..n-1}$, $(\delta_t)_{t=0..n-1}$

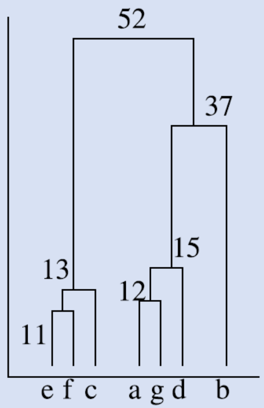
input: data set $\mathcal{D}$, $|\mathcal{D}| = n$
output: series of hierarchically nested partitions $(\mathcal{P}_t)_{t=0..n-1}$
series of hierarchy levels $(\delta_t)_{t=0..n-1}$

$\mathcal{P}_0 = \{\{\mathbf{x}\} \mid \mathbf{x} \in \mathcal{D}\}$
$t = 0, \delta_t = 0$
while current partition $\mathcal{P}_t$ has more than one cluster
  find pair of clusters $(\mathcal{C}_1, \mathcal{C}_2)$ with minimal distance $d'(\mathcal{C}_1, \mathcal{C}_2)$
  $\delta_{t+1} = d'(\mathcal{C}_1, \mathcal{C}_2)$
  construct $\mathcal{P}_{t+1}$ from $\mathcal{P}_t$ by removing $\mathcal{C}_1$ and $\mathcal{C}_2$ and inserting $\mathcal{C}_1 \cup \mathcal{C}_2$
  $t = t + 1$
end while

– Example: Distance matrix at each iteration of cluster forming

	a	b	c	d	e	f	g
a	0	50	63	17	72	81	12
b		0	49	41	42	54	37
c			0	52	13	16	61
d				0	56	66	15
e					0	**11**	64
f						0	74
g							0

	a	b	c	d	ef	g
a	0	50	63	17	72	**12**
b		0	49	41	42	37
c			0	52	13	61
d				0	56	15
ef					0	74
g						0



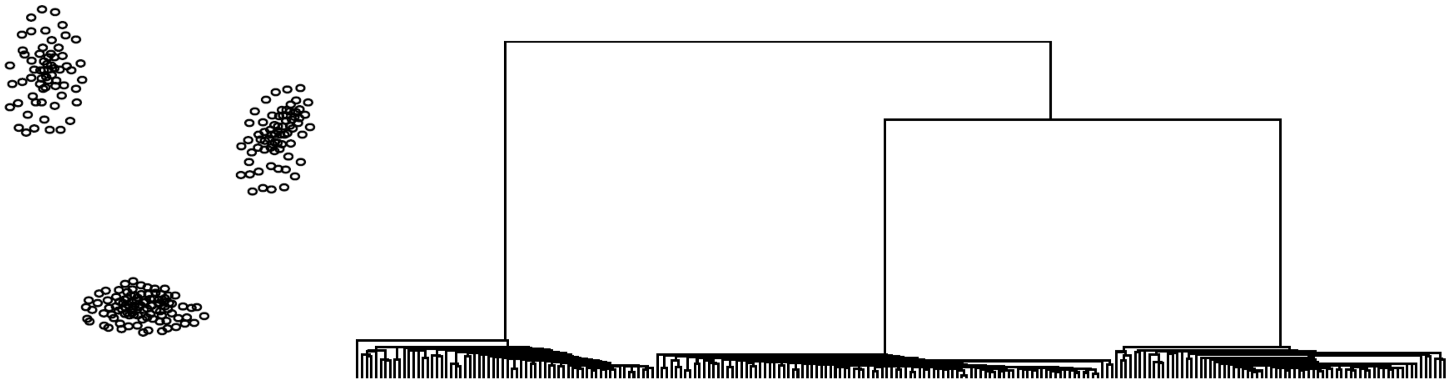
	ag	b	c	d	ef
ag	0	37	61	15	72
b		0	49	41	42
c			0	52	**13**
d				0	56
ef					0

	ag	b	d	efc
ag	0	37	**15**	61
b		0	41	42
d			0	52
efc				0

	agd	b	efc
agd	0	**37**	52
b		0	54
efc			0

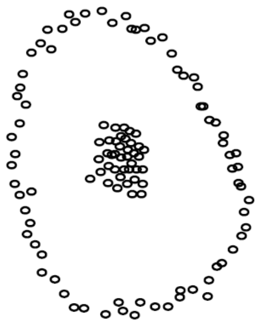
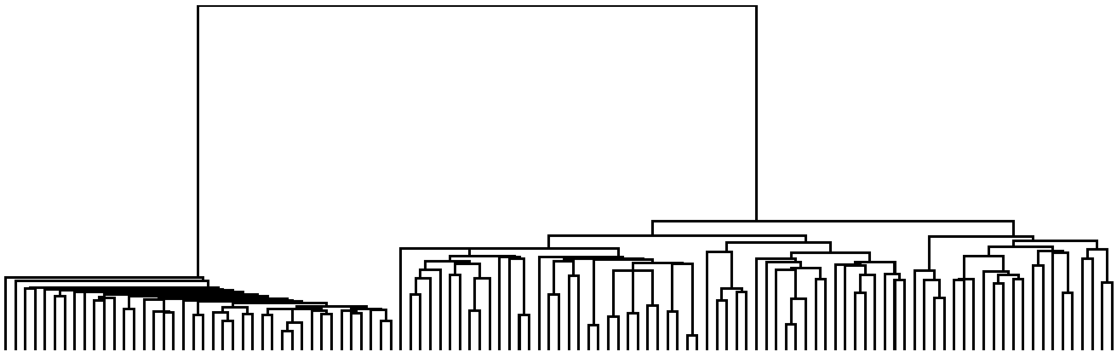
Hierarchy of Clusters (Examples)

- Three well-separated clusters
- Formed with small δ
- Remains stable for a wide range of δ (until large δ)



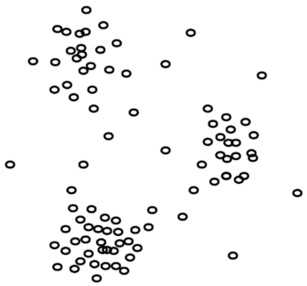
Hierarchy of Clusters (Examples)

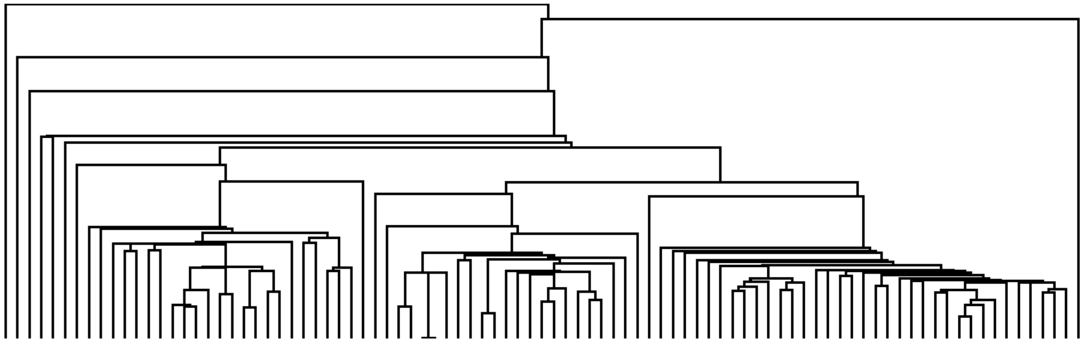
- Two well-separated clusters
- Clusters differ in sizes and shapes

Hierarchy of Clusters (Examples)

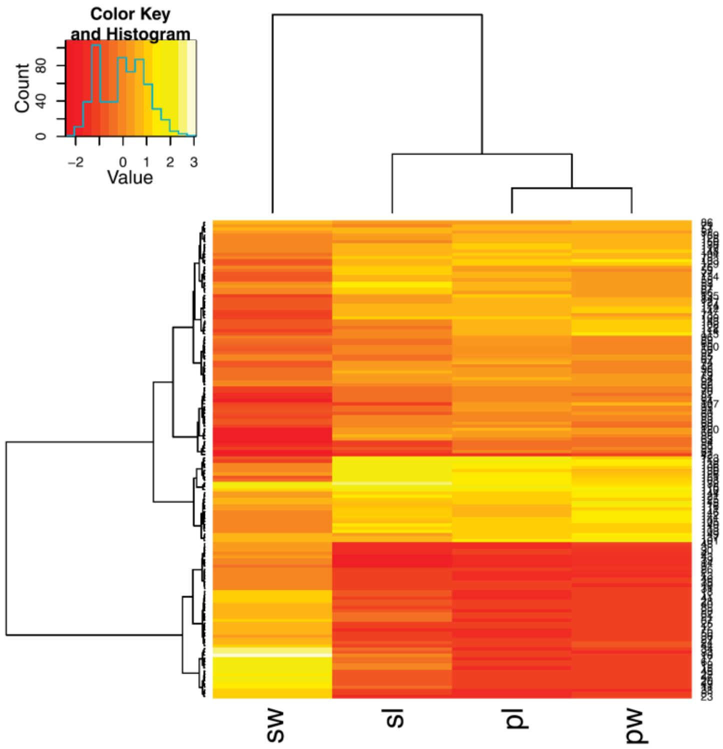
- Small noises to clusters
- Isolated noise points are ***chained*** – forming clusters with small δ
- No clearly defined robust clusters
- Hierarchical clustering (esp. single-linkage) is susceptible to noises





Heatmap

- Clustering on observations (rows)
- As well as clustering on columns (features)
- Color scale to represent numerical values
  - Show similarities in features within clusters



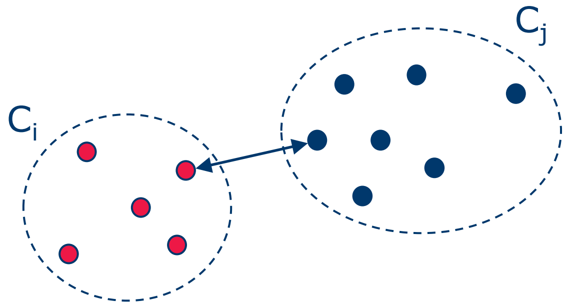
Single Linkage

- Distance between clusters ≡ distance between two closest points

$$d\left(C_i, C_j\right) = \min_{x,y}\left\{d(x, y) \mid x \in C_i, y \in C_j\right\}$$

Distance of the closest two points, one from each cluster

- Merge Step: Union of two subsets of data points



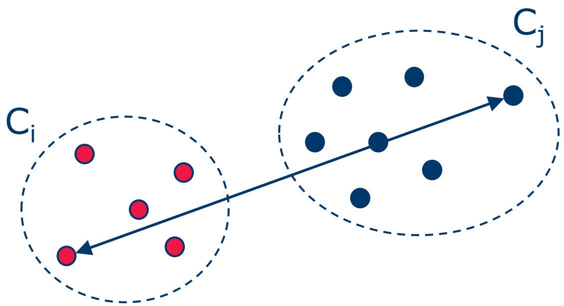
Complete Linkage

- Distance between clusters ≡ distance between two farthest points

$$d\left(C_i, C_j\right) = \max_{x,y}\left\{d(x, y) \mid x \in C_i, y \in C_j\right\}$$

Distance of the farthest two points, one from each cluster

- Merge Step: Union of two subsets of data points



- Distance between clusters (nodes):

$$Dist_{avg}(C_1, C_2) = \frac{1}{|C_1| \cdot |C_2|} \sum_{p \in C_1} \sum_{p \in C_2} dist(p, q)$$

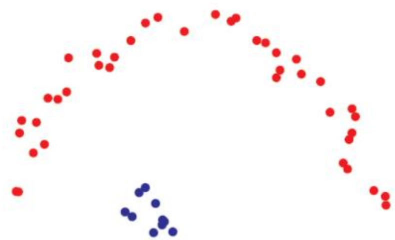
Average distance of all possible pairs of points between C_1 and C_2

$$Dist_{mean}(C_1, C_2) = dist\big(mean(C_1), mean(C_2)\big)$$

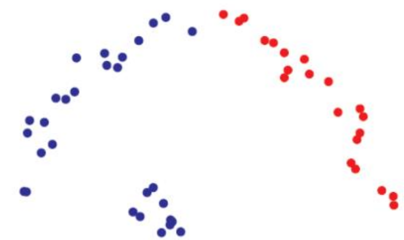
Distance between two centroids

- Merge Step:
  - union of two subsets of data points
  - construct the mean point of the two clusters

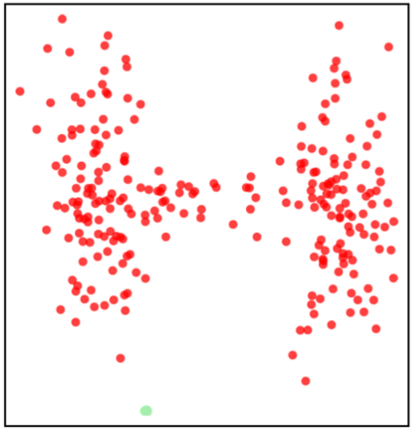
Single Linkage vs. Complete Linkage



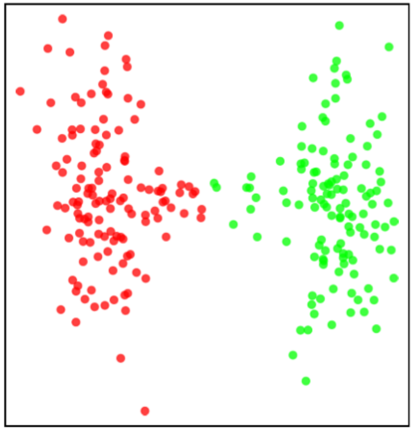
Single linkage



Complete linkage

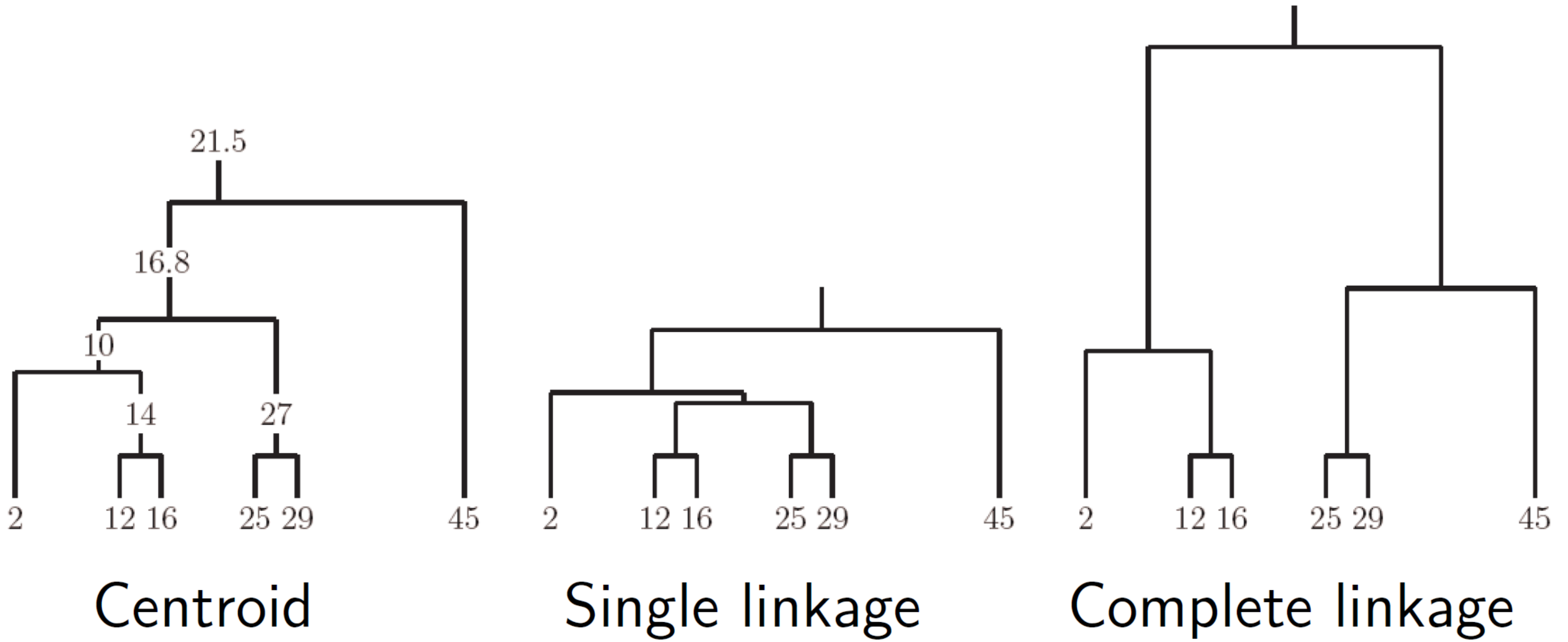


Single linkage



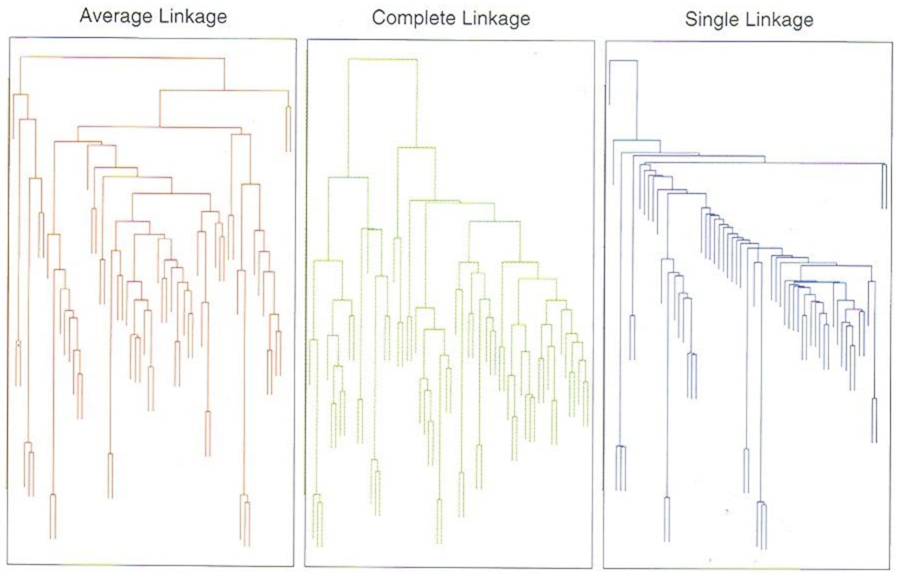
Complete linkage

- Clustering of the 1-dimensional data set {2, 12, 16, 25, 29, 45}.
- All three approaches to measure the distance between clusters lead to different dendrograms.

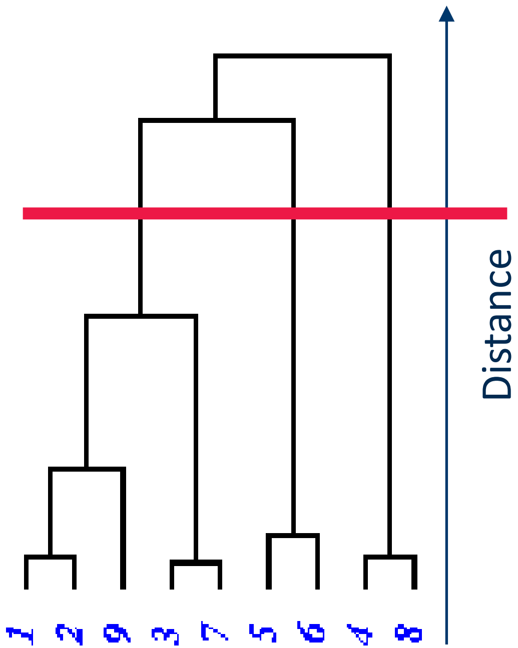


Linkage Based Clustering

- Single Linkage:
  - Prefers well-separated clusters
- Complete Linkage:
  - Prefers small, compact clusters
- Average Linkage:
  - Prefers small, well-separated clusters…



- **Advantages**:
  - Finds not only a „flat“ clustering, but a hierarchy of clusters (dendrogram)
  - A single clustering can be obtained from the dendrogram (e.g., by performing a horizontal cut)



- **Weaknesses**:
  - Decisions (merges/splits) cannot be undone
  - Sensitive to noise (Single-Link) (a „line“ of objects can connect two clusters)
  - Inefficient
    ➔ Runtime complexity at least $O(n^2)$ for n objects

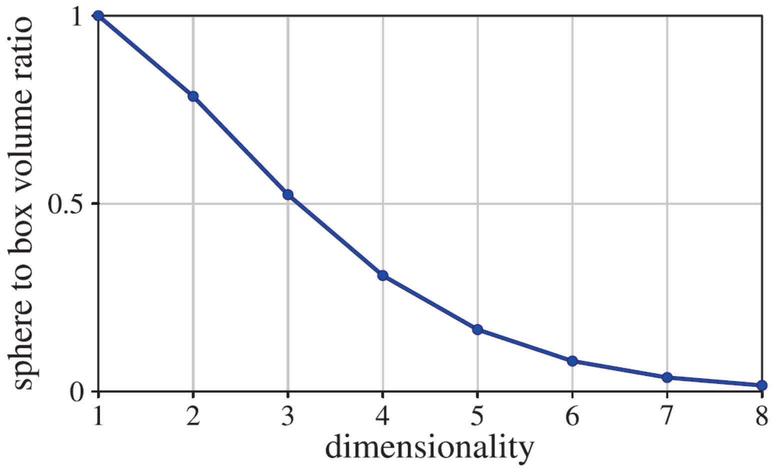
Attribute Weighting

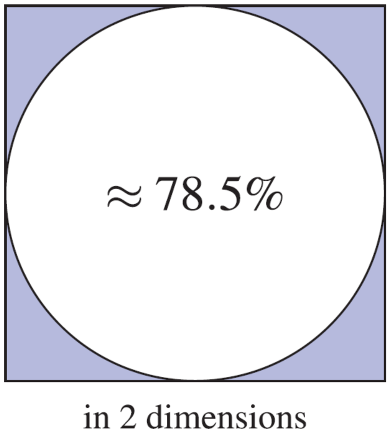
- Magnitude difference
  - Some attributes are larger in magnitude, thus dominates in calculation of dissimilarity
  - E.g, Horsepower (50-300) compared to mileage (5-25)

- Normalize the data – all numerical attributes have a unit variance

- Which features should weigh more?
  - E.g., cup holders or air conditioning? How do they affect car's value
  - Additional weighting of attributes may be needed to reflect this

- A unit sphere centered in a unit cube $[-1, 1]^d$ in d-dimensional space
- ➔ The higher the dimension, the smaller the coverage by the sphere



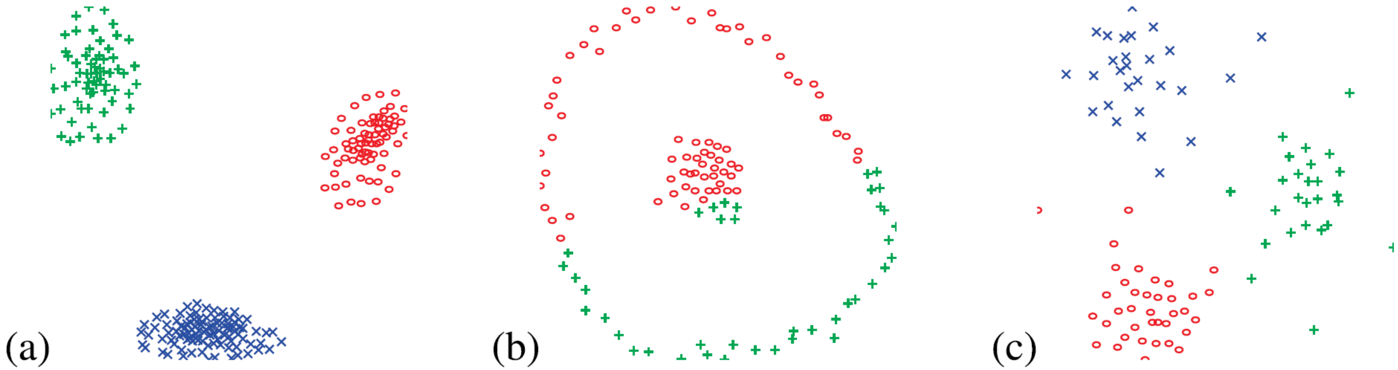


- High dimensional space → likelihood of finding a data point
- Known as the ***curse of dimensionality***

Prototype- and Model-Based Clustering

Overview

– Each cluster is represented by a ***model*** or ***prototype*** (representative data point)

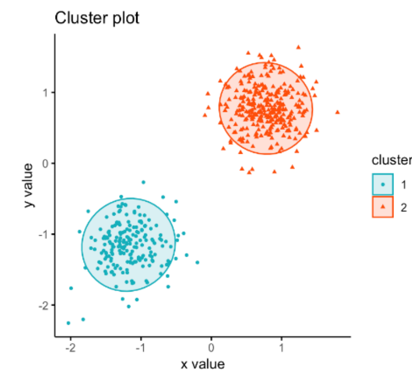


– Works well when the assumption is true (i.e. closest prototype represents a cluster)

– But does not work all cases – e.g., circular clusters

Goal:

A (disjoint) partitioning into k clusters with minimal costs



- Local optimization method:
  - choose k initial cluster representatives
  - optimize these representatives iteratively
  - assign each object to its **most similar cluster representative**

- Types of cluster representatives:
  - Mean of a cluster (*construction of central points*)
  - Median of a cluster (*selection of representative points*)
  - Probability density function of a cluster (*expectation maximization*)

Construction

- Cluster partition is described by a ***membership matrix*** $[p_{i|j}]$
  - $p_{i|j}$: membership belongingness of data point *j* to cluster *i.*
  - $p_{i|j}$ can be binary or real-valued

- The number of clusters k must be known beforehand

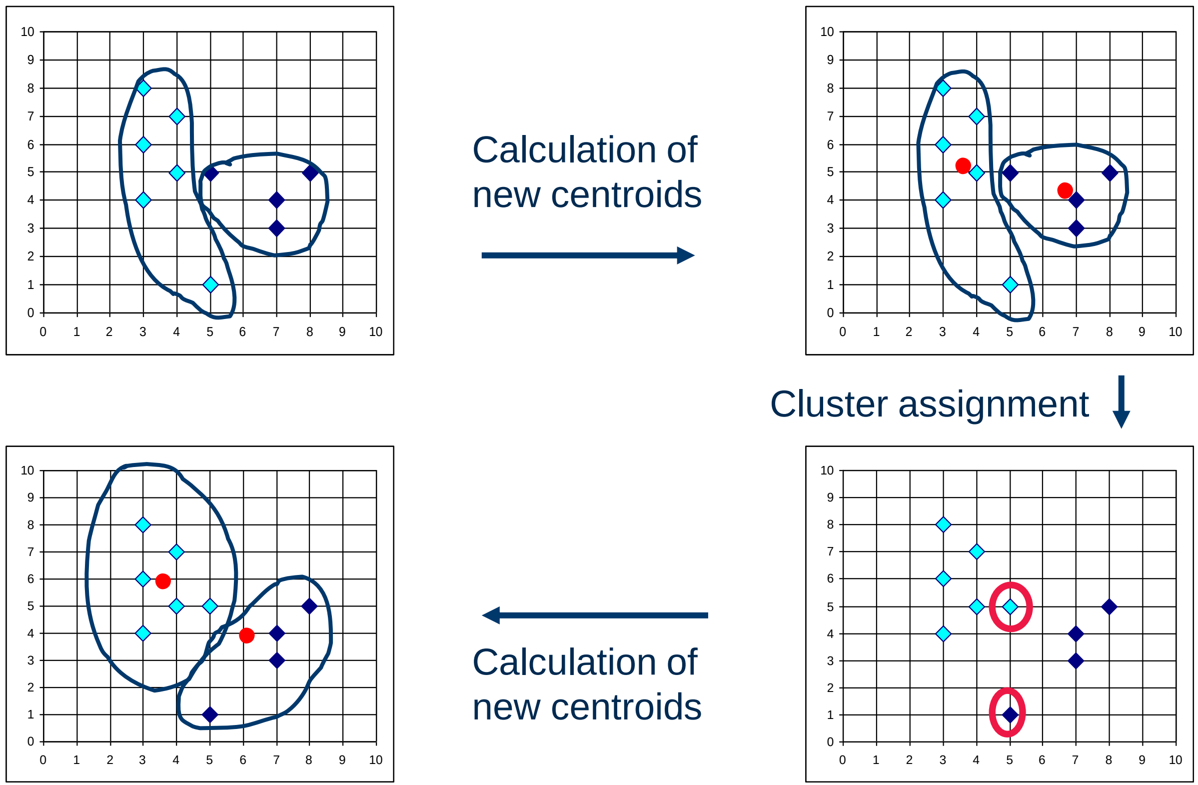
- Data points are assigned to the closest model or prototype
- Update the prototype based on the new cluster partition
- Repeat until the model / prototype stops moving

Given k, the k-Means algorithm is implemented in four steps:

1. Partition objects into k non-empty subsets, calculate their **centroids** (i.e., **mean point**, of the cluster)
2. Assign each object to the cluster with the **nearest** centroid (Euclidean distance)
3. Compute the centroids from the current partition as $p_i = \frac{\sum_{j=1}^{n} p_{i|j}\ x_j}{\sum_{j=1}^{n} p_{i|j}}$
4. Go back to Step 2, repeat until the updated centroids stop moving significantly

– Note: Each data point can only belong to a single cluster
  – $p_{i|j} = 1$ for the cluster with closest prototype, 0 otherwise

k-Means Algorithm

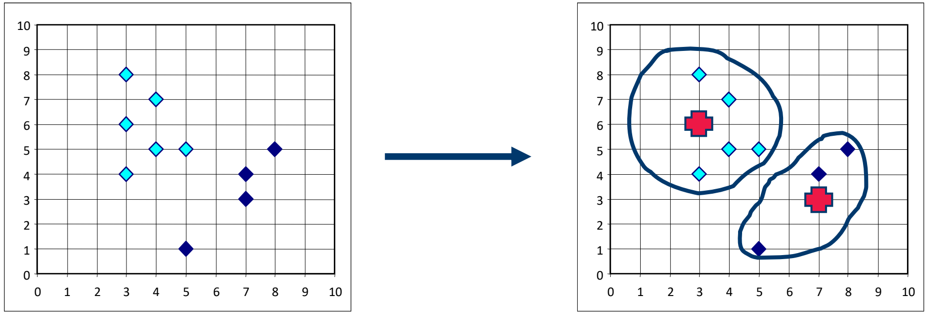


Comments of the k-Means Method

- **Advantages:**
  - Relatively efficient: $O(tkn)$ where n is #objects, k is #clusters, and t is #iterations; usually, $\mathrm{k}, t \ll n$ (t typically $5 - 10$)
  - Simple implementation
  - **k-means is the most popular partitioning clustering method!**

- **Weaknesses:**
  - Often terminates at a local optimum. A better local optimum may be found using techniques such as: deterministic annealing and genetic algorithms.
  - Applicable only when mean is defined (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

- **Problem with K-Means**
  - An object with an extremely large value can substantially distort the distribution of the data.
- **One solution: K-Medoids**
  - Instead of taking the **mean** value of the objects in a cluster as a reference point, **medoids** can be used, which are the most centrally located objects in a cluster.



Fuzzy c-Means Model

- Cluster belongingness can be a real value [0,1]

$$p_{i|j} = \frac{1}{\sum_{k=1}^{C} \frac{\left\|x_j - p_i\right\|^2}{\left\|x_j - p_k\right\|^2}}$$

- Prototype is calculated as

$$p_i = \frac{\sum_{j=1}^{n} p_{i|j}^2 x_j}{\sum_{j=1}^{n} p_{i|j}^2}$$

where the power 2 of $p_{i|j}^2$ is referred as ***fuzzifier***.

Gaussian Mixture Decomposition

- Each cluster follows a Gaussian distribution centered at the prototype

- Data follows a Gaussian mixture distribution of multiple clusters

- Cluster membership – given by a Gaussian probability

- Iterative updating of:
  - The mean and the variance of Gaussian for each cluster
  - Prototypes

Clustering: Quality Measures

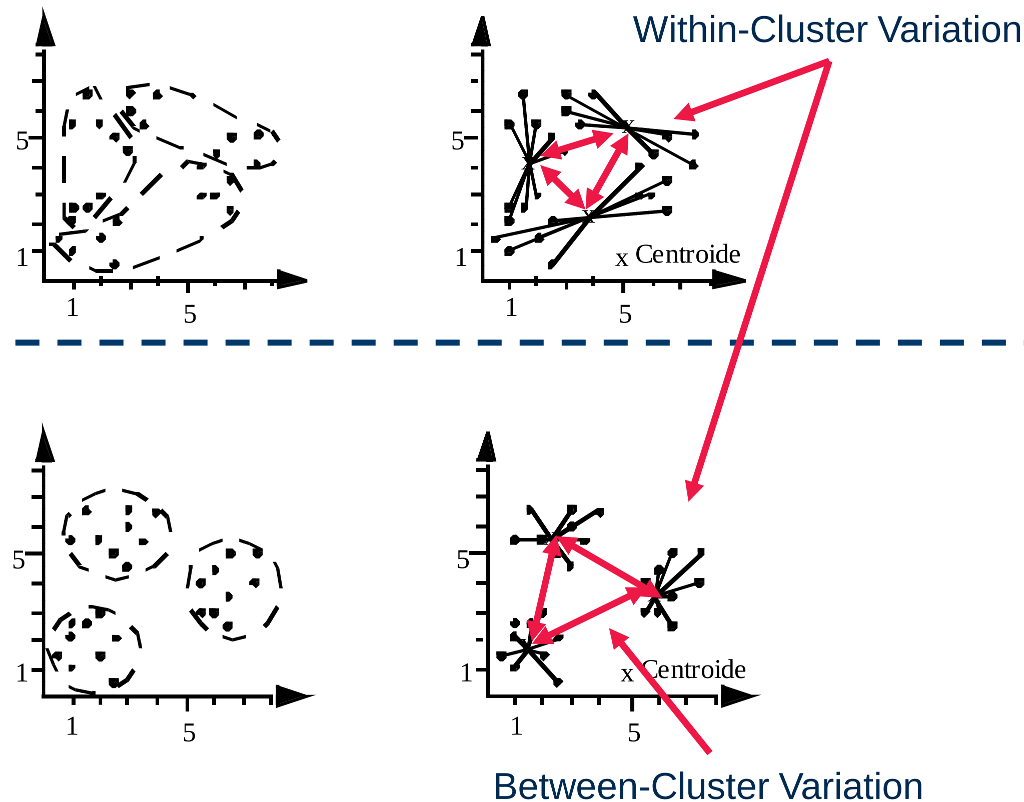
How Many Clusters

- You need to know the number of clusters k beforehand
- But you may not know – estimate from the data
- Run clustering algorithm multiple times with different k
- Plot the validity measures against the number of clusters
  - Silhouette coefficient
  - Separation index
- Find the k that produces the optimum result

Optimal Clustering: Example

Bad Clustering

Good Clustering



Centroid μ_C: mean vector of all objects in clustering C

- Within-Cluster Variation:

$$TD^2 = \sum_{i=1}^{k} \sum_{p \in \boldsymbol{C_i}} dist(p, \mu_{C_i})^2$$

- Between-Cluster Variation:

$$BC^2 = \sum_{j=1}^{k} \sum_{i=1}^{k} dist(\mu_{C_j}, \mu_{C_i})^2$$

- Clustering Quality (one possible measure):

$$CQ = \frac{BC^2}{TD^2}$$

Silhouette-Coefficient [Kaufman & Rousseeuw 1990] measures the quality of clustering

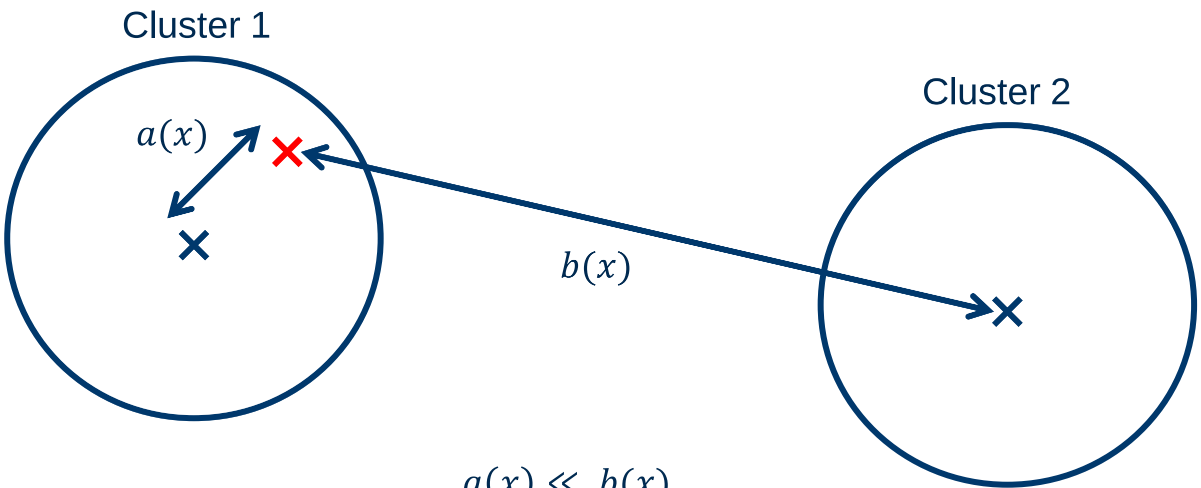
- $a(x)$: distance of object x to its cluster representative
- $b(x)$: distance of object x to the representative of the „second-best“ cluster
- **Silhouette** $s(x)$ of x

$$s(x) = \frac{b(x) - a(x)}{\max\{a(x), b(x)\}} \in [-1, 1]$$

- Average of $s(x)$ is calculated for each cluster
- Good clusters → high silhouette coefficient

Silhouette-Coefficient

Good clustering…

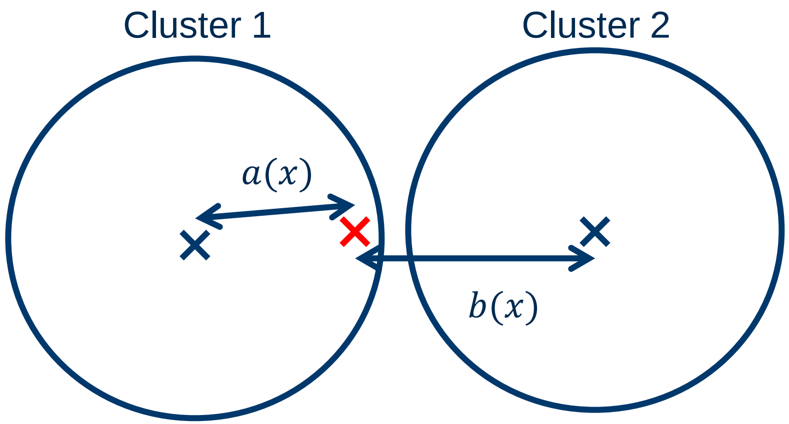


$$a(x) \ll b(x)$$

$$s(x) = \frac{b(x) - a(x)}{\max\{a(x), b(x)\}} \approx \frac{b(x)}{b(x)} = 1$$

Silhouette-Coefficient

…not so good…

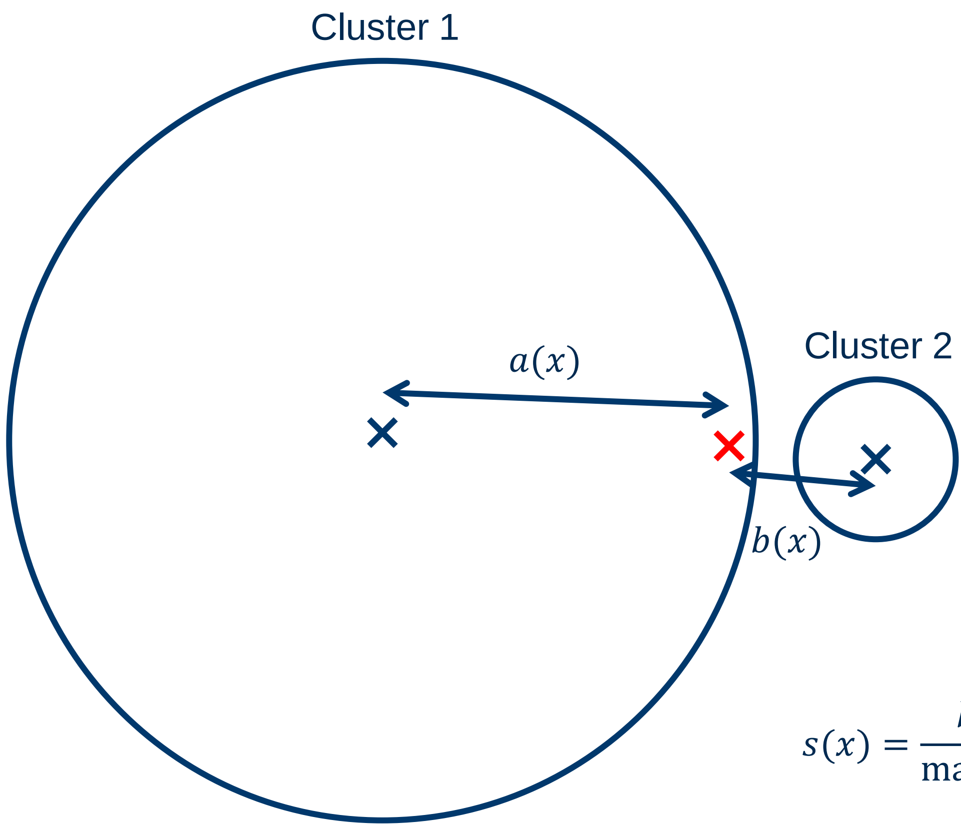


$$a(x) \approx b(x)$$

$$s(x) = \frac{b(x) - a(x)}{\max\{a(x), b(x)\}} \approx \frac{0}{b(x)} = 0$$

Silhouette-Coefficient

…bad clustering.



$$a(x) \gg b(x)$$

$$s(x) = \frac{b(x) - a(x)}{\max\{\, a(x), b(x)\}} \approx \frac{-a(x)}{a(x)} = -1$$

Silhouette-Coefficient for Clustering C

- Silhouette coefficient s_c for clustering C is the average silhouette over all objects $x \in C$

$$s_c = \frac{1}{n} \sum_{x \in C} s(x)$$

- Interpretation of silhouette coefficient:
  - $s_c > 0.7$: strong cluster structure,
  - $s_c > 0.5$: reasonable cluster structure,
  - . . .

Method

- For $k = 2, 3, \ldots, n-1$, determine one clustering each
- Choose k resulting in the highest clustering quality

Measure of clustering quality

- Uncorrelated with k
- for k-means and k-medoid:

  TD^2 and TD decrease monotonically with increasing k

Summary: Clustering by Partitioning

- Scheme always similar:
  - Find (random) starting clusters
  - Iteratively improve cluster positions
    (compute new mean, swap medoids, compute new distribution parameters,…)

- Important:
  - Number of clusters k
  - Initial cluster position influences (heavily):
    - quality of results
    - speed of convergence

- Problems for iterative clustering methods:
  - Clusters of varied size, density and shape

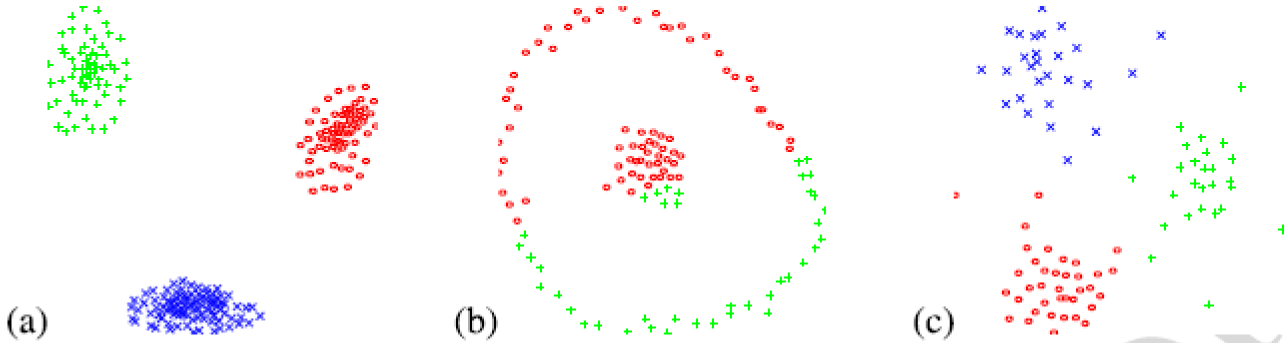
- Separation index
- Designed to identify compact and well-separated clusters

$$D = \min_{i=1,\cdots,C} \min_{j=i+1,\cdots,C} \frac{\min_{x \in C_i, y \in C_j} d(x_i, x_j)}{\min_{k=1,\cdots,C} diam_k}$$

where $diam_k$ expresses the extent of a cluster

Cluster Shape

- Clusters found with prototype-based clustering tend to be spherical
- Unrealistic in many situations

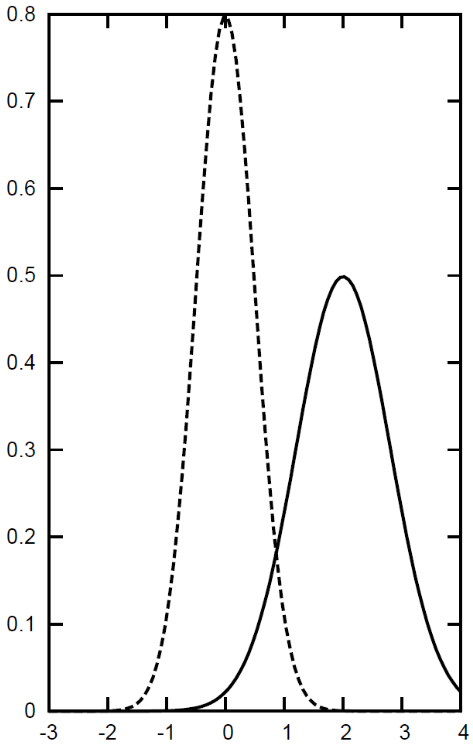


Solutions:

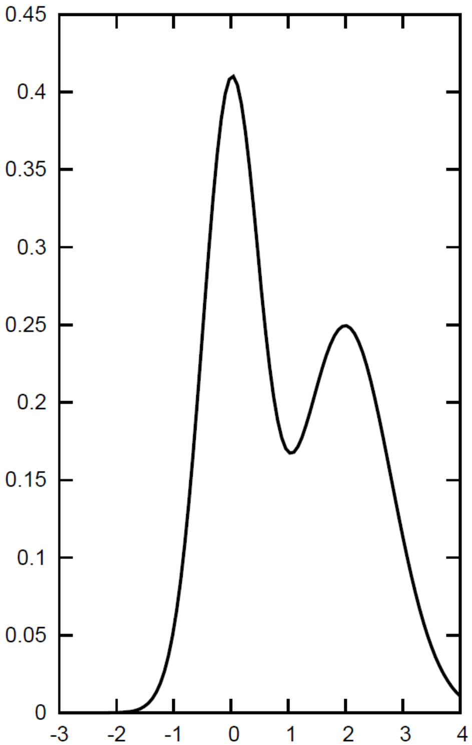
- Use of non-Euclidean distance
- Subdivide the data into many small clusters first, then cluster the resulting prototypes

Gaussian Mixture Models

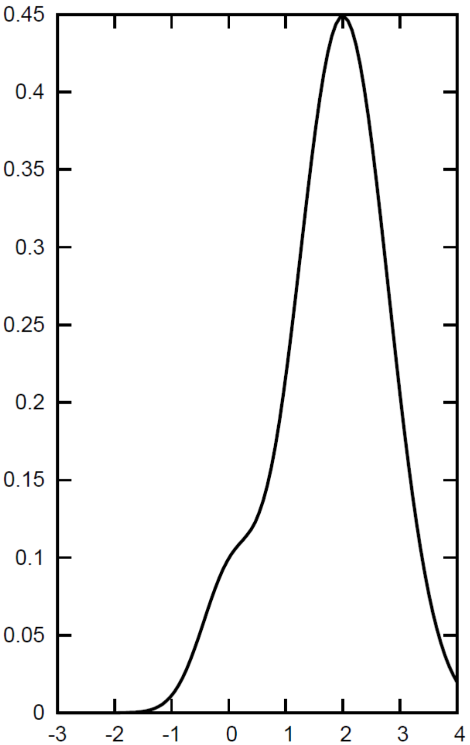
Gaussian Mixture Models



Two normal distributions

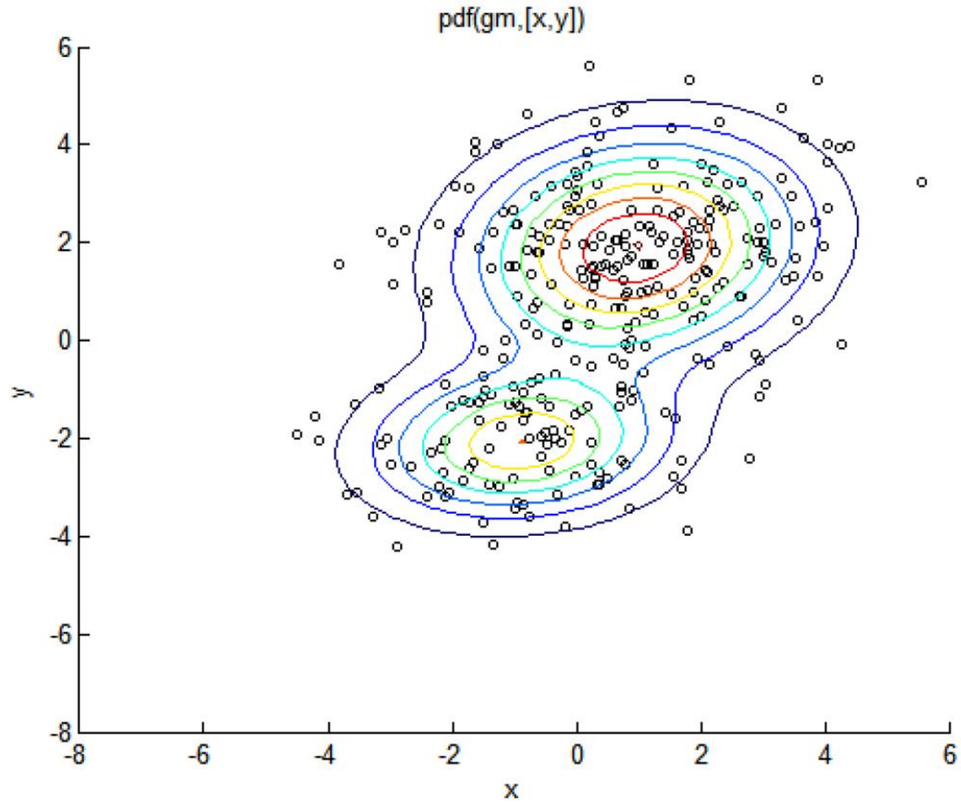


Mixture Model: both normal distributions contribute 50%



Mixture Model: one normal distribution contributes 10%, the other 90%

Gaussian Mixture Models



- First, draw a random integer i between 1 and k with probability p_i of drawing i. The data generated next will belong to cluster $\#i$.
- Second, draw a random sample from the Gaussian distribution $g(\boldsymbol{x}; \mu_i, \sigma_i)$ with the parameters μ_i, σ_i taken from model $\#i$.
- Overall density function is:

$$f(\boldsymbol{x};\ \theta) = \sum_{i=1}^{k} p_i\ g(\boldsymbol{x}; \mu_i, \sigma_i)$$

$$\theta = (\theta_1, \theta_2, \dots, \theta_k) \quad and \quad \theta_i = (p_i, \mu_i, \sigma_i)$$

- Given a mixture θ, the likelihood of the full data set being generated by this model is:

$$L = \prod_{j=1}^{n} f(\boldsymbol{x}_j;\ \theta)$$

- **Assumption:** Data were generated by sampling a set of normal distributions. (The probability density is a mixture of normal distributions.)
- **Aim:** Find the parameters for the normal distributions and how much each normal distribution contributes to the data.
- **Algorithm:** EM clustering (expectation maximisation).
- Alternating scheme in which the parameters of the normal distributions and the likelihoods of the data points to be generated by the corresponding normal distributions are estimated.

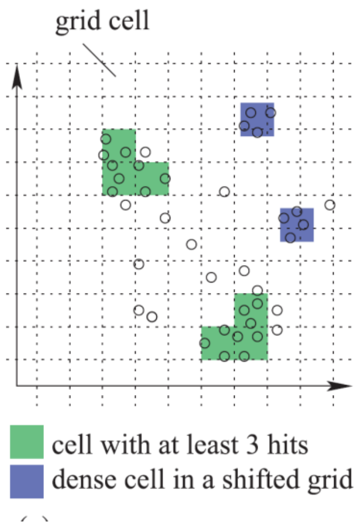
Density-Based Clustering

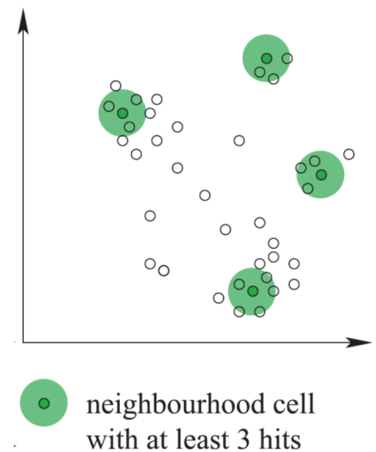
- For numerical data, density-based clustering algorithm often yield the best results.
- **Principle**: A connected region with high data density corresponds to one cluster.
- **DBScan** is one of the most popular density-based clustering algorithms.

- However...
- There are no representatives. The clusters can mostly not be investigated, only assigned (or, in few-D, visualized).

Overview

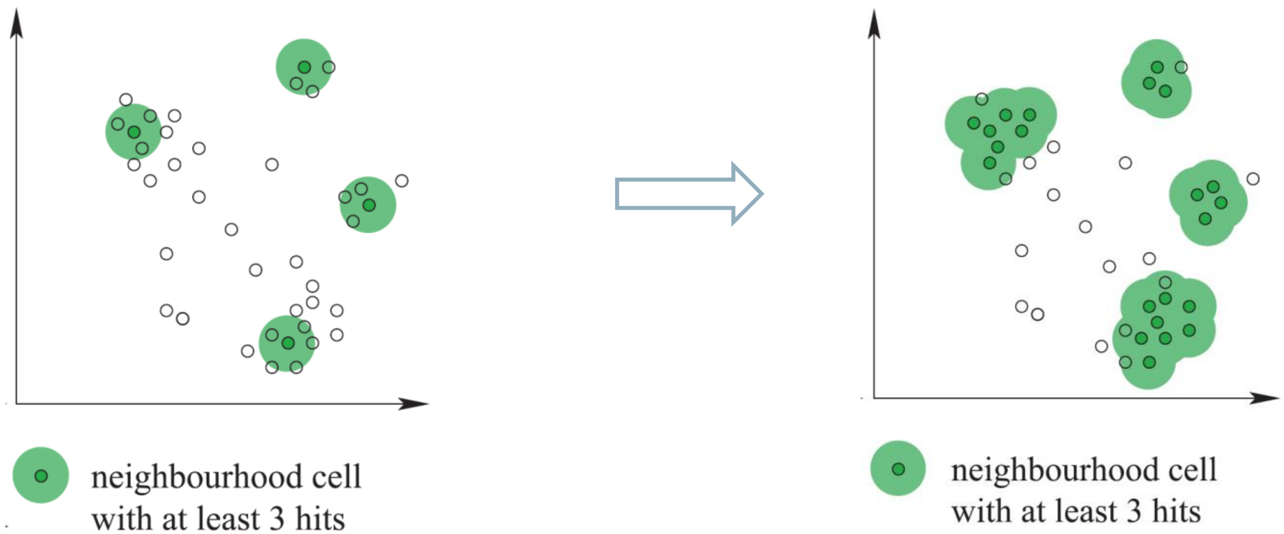
- In a grid, identify a square with more than 3 points (green)





- Problem: the grid is shifted slightly, then no longer 3+ points in a square (blue)
- Solution: a circle centered at each point (neighborhood). Identify a point with 3+ neighbors

– Continue expanding the cluster as long as dense cells are within a neighborhood



➔ ***DBScan*** algorithm

Construction

- At some location $\boldsymbol{x}$, the ε-***neighborhood*** is defined as

$$N_\varepsilon(\boldsymbol{x}) = \{\boldsymbol{y} : \ ||\boldsymbol{x} - \boldsymbol{y}|| \leq \varepsilon\}$$

- If the neighborhood contains at least *MinPts* elements
  → $\boldsymbol{x}$ is located withing a cluster
  → $\boldsymbol{x}$ is referred to as a ***core object***

- There may be other core objects withing $N_\varepsilon(\boldsymbol{x})$
- These points are ***density reachable*** from $\boldsymbol{x}$
- Clusters are expanded until all density reachable points are included

DBSCAN - a density-based clustering algorithm - defines five types of points in a dataset.

- **Core Points** are points that have at least a minimum number of neighbors (*MinPts*) within a specified distance (ε).
- **Noise Points** are neither core points nor border points.
- **Border Points** are points that are within ε of a core point, but have less than *MinPts* neighbors.
- **Directly Density Reachable Points** are within ε of a core point.
- **Density Reachable Points** are reachable with a chain of Directly Density Reachable points.

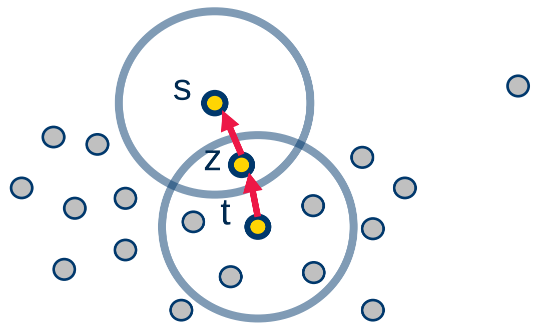
Clusters are built by joining core and density-reachable points to one another.

Example with MinPts = 3

Core Point
vs. Border Point
vs. Noise

- t = Core point
- s = Boarder point
- n = Noise point

Directly Density Reachable
vs. Density Reachable

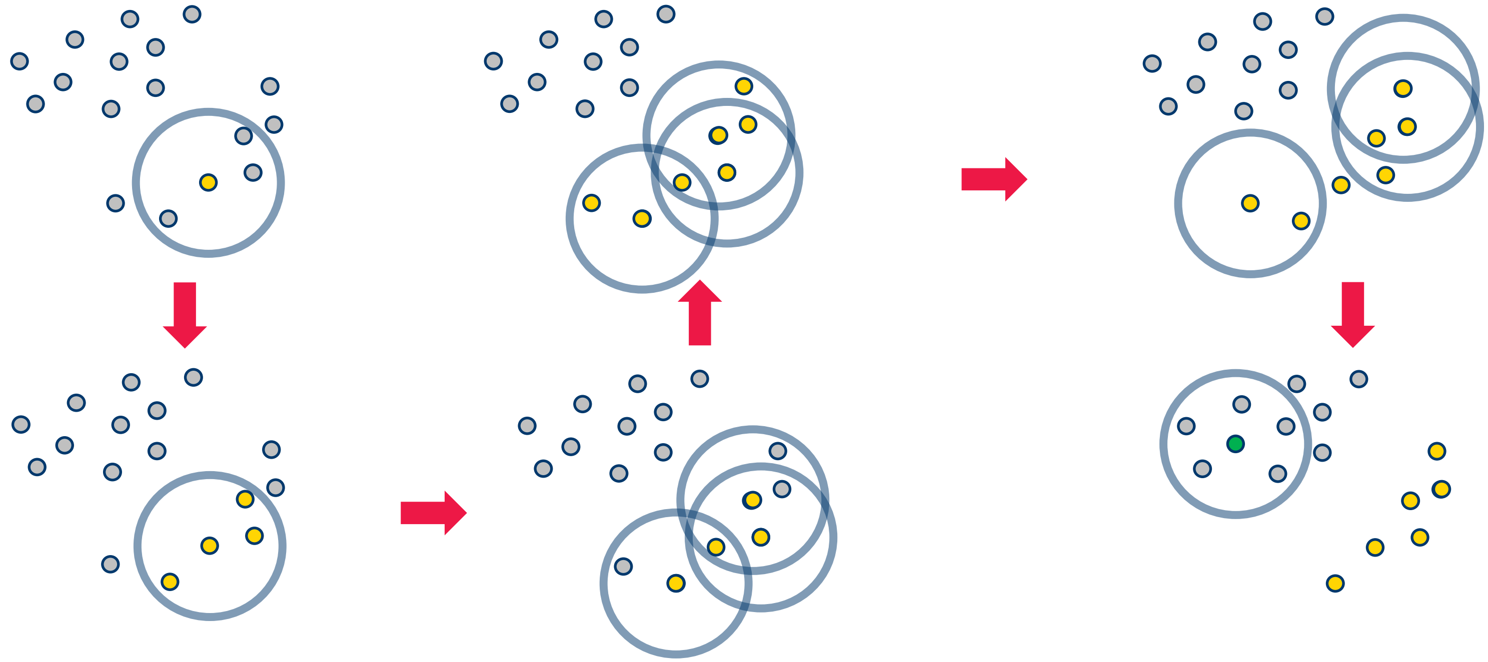


- z is directly density reachable from t
- s is not directly density reachable from t, but density reachable via z

Note: t is not density reachable from s, because s is not a Core point

DBSCAN - Density Based Spatial Clustering of Applications with Noise

- For each point, DBSCAN determines the ε-environment and checks whether it contains more than *MinPts* data points → **core** point
- Iteratively increases the cluster by adding density-reachable points



Clustering:

- A density-based clustering C of a dataset D w.r.t. ε and MinPts is the set of all density-based clusters C_i w.r.t. ε and MinPts in D.
- The set $NoiseCL$ („noise“) is defined as the set of all objects in D which do not belong to any of the clusters.

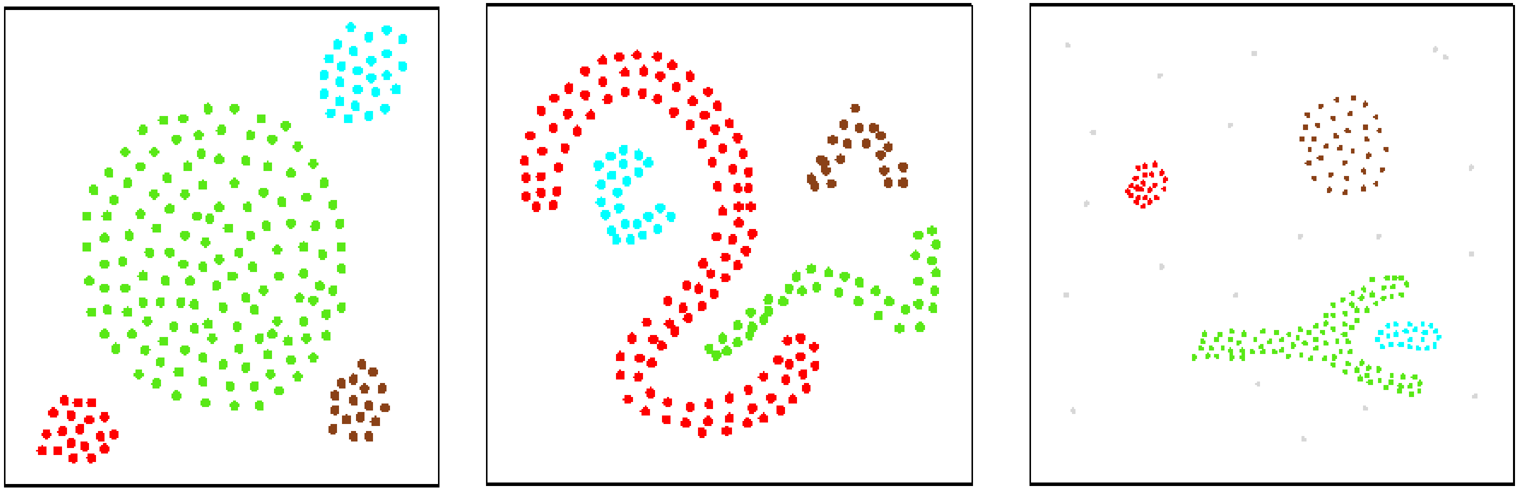
Property:

- Let C_i be a density-based cluster and $p \in C_i$ be a core object.

$$C_i = \{o \in D \mid o \text{ density-reachable from } p \text{ w.r.t. } \varepsilon \text{ and } MinPts\}.$$

DBSCAN - Density Based Spatial Clustering of Applications with Noise

- DBSCAN uses (spatial) index structures for determining the ε-environment:
  ➔ computational complexity $O(n \log n)$ instead of $O(n^2)$
- Arbitrary shape clusters found by DBSCAN
- Parameters: ε and $MinPts$

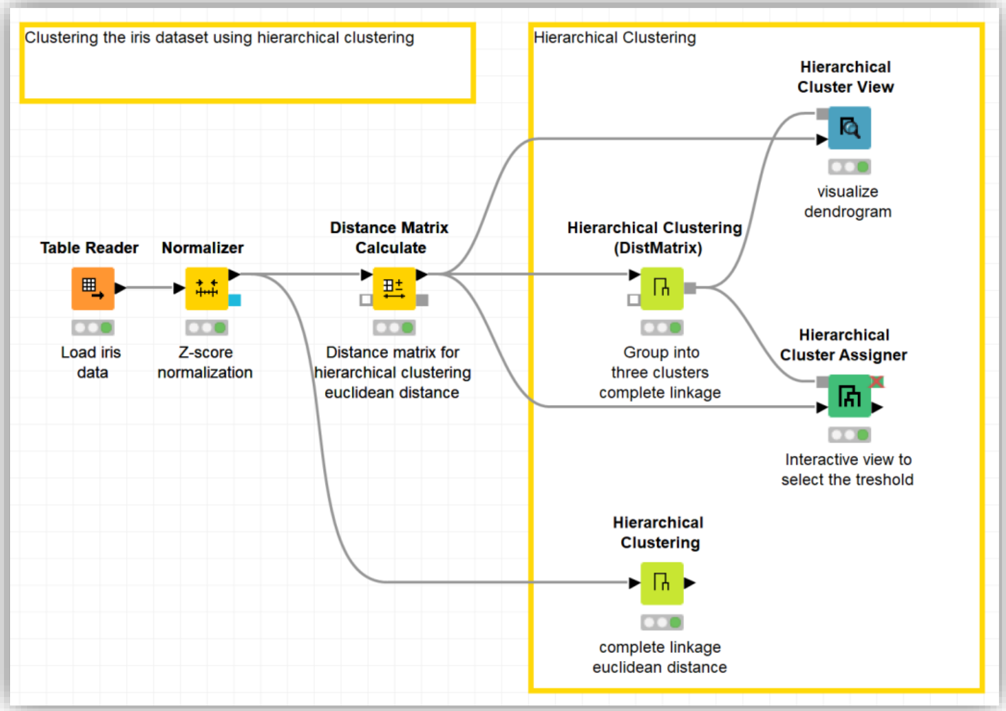


Summary

- Clustering
- Similarity Measures
- Hierarchical Clustering
  - DIANE
  - AGNES
- Prototype or Model –based Clustering
  - K-Means
- Quality Measures
- Gaussian Mixture Model
- Density based Clustering
  - DBScan

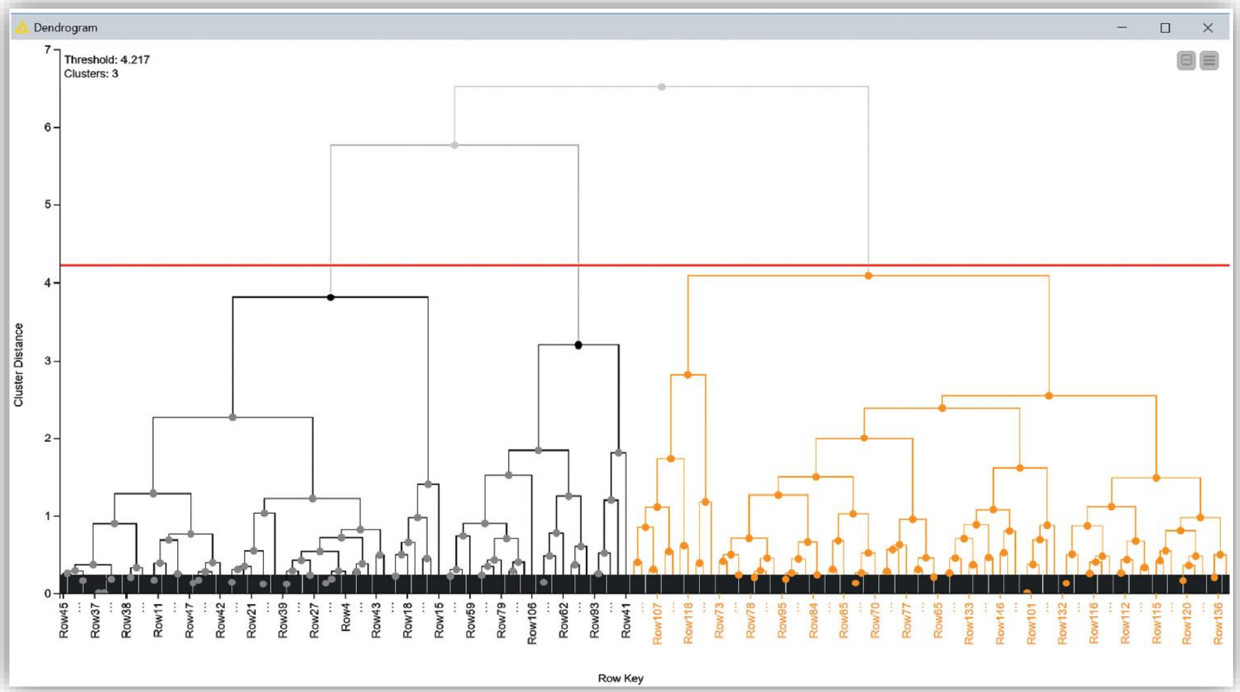
Practical Examples with KNIME Analytics Platform

Hierarchical Clustering

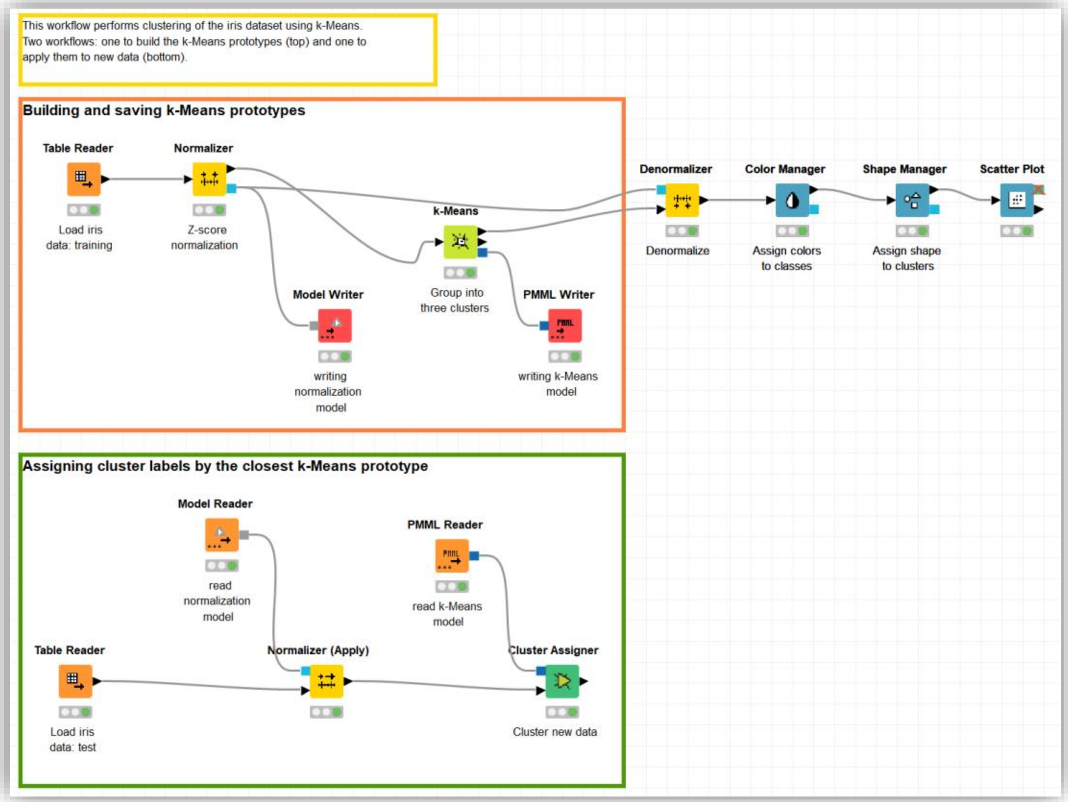


Workflow implementing hierarchical clustering with the simple Hierarchical Clustering node and with the more complex sequence of nodes, including the Distance Matrix Calculate node

Hierarchical Clustering

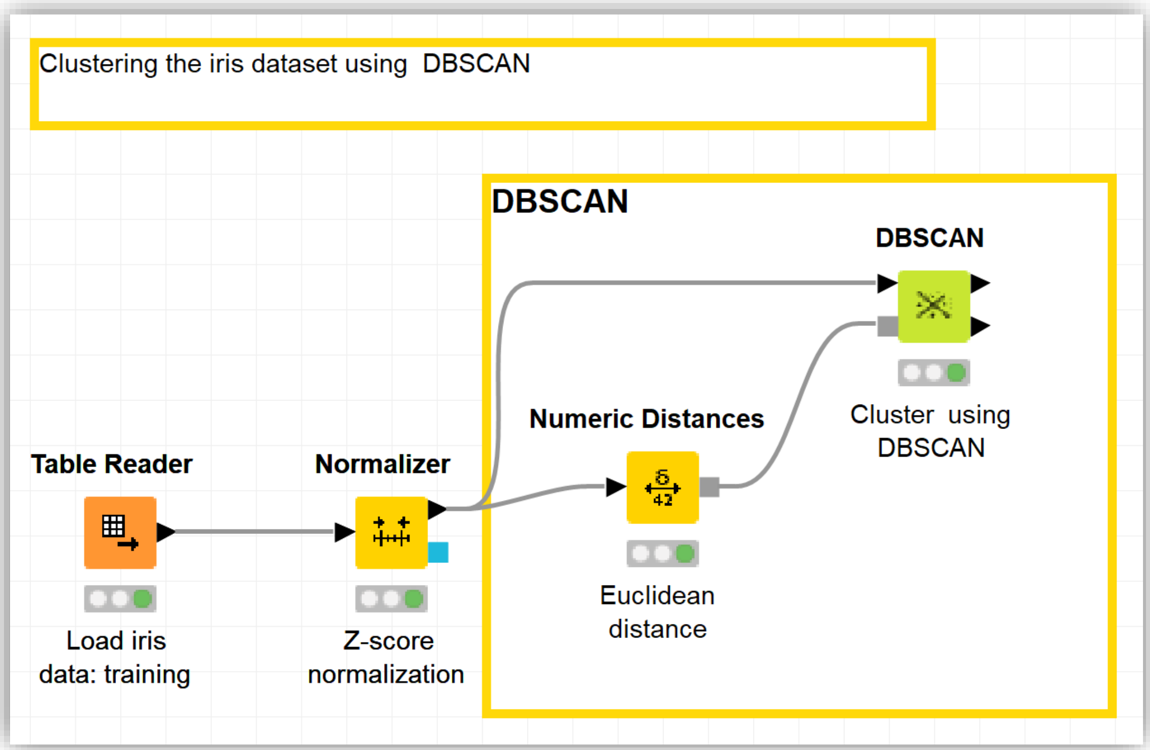


A dendrogram for the iris data obtained with Euclidean distance and complete linkage. Moving the threshold line changes the number of clusters and the assignment for the input rows.



Building of k-Means prototypes (top) and cluster assignment (bottom)

DBSCAN



This workflow implements a DBSCAN clustering on the iris data set. Notice the possibility to choose the distance metric in the Numeric Distances node

Thank you