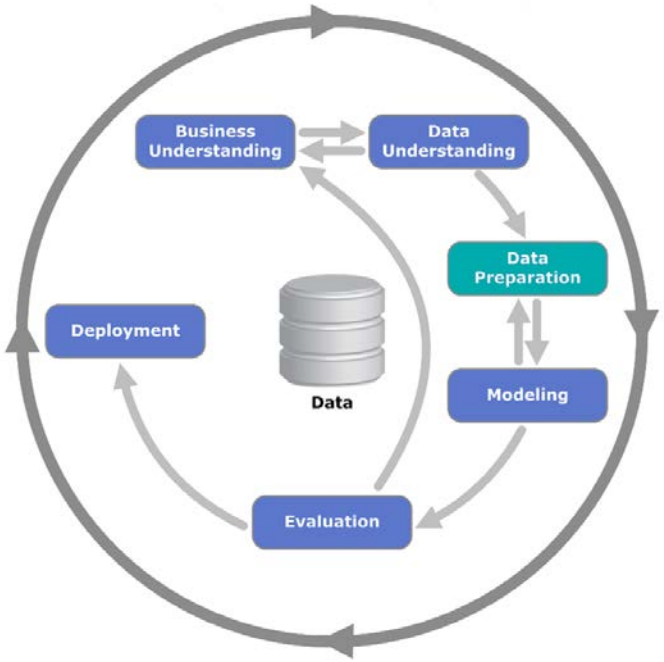
INF FAKULTÄT FÜR INFORMATIK

Fuzzy Data Analysis

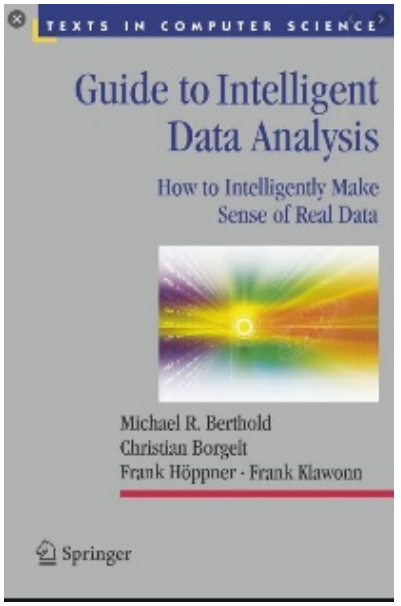
Clustering

Prof. Dr. Rudolf Kruse

Data Analysis



Cross Industry Standard Process for Data Analysis



Example: Bank Note Analysis

Clustering: Given 200 Swiss Banknotes. How to find groups of similar banknotes?
Use attributes of each banknote. Unsupervised Learning

Classification: Given a data set with 100 real and 100 counterfeit Swiss banknotes. How to separate real from counterfeit banknotes, especially for new banknotes?
Use attributes of each banknote. Supervised learning

Understanding Data is crucial:
- gathering (source,…?)
- description (structure,…?)
- exploration (outlier,…?)
- data quality (context,…?)

Understanding Data

- Noting patterns and trends Do the trends, patterns, and conclusions make sense?
- Clustering Grouping events, places and people together if they have similar characteristics
- Counting Helps to enable the researcher to see what is there by counting frequencies
- Making comparisons Establishing similarities between and within data sets
- Partitioning variables Splitting variables to find more coherent descriptions
- Subsuming particulars into the general. Linking specific data to general concepts
- Factoring Attempting to discover the factors underlying the process under investigation
- Noting relations between variables
- Finding intervening variables Establish the presence and effects of variables
- Building a logical chain of evidence Trying to understand trends and patterns through developing logical relationships.
- Making conceptual/theoretical coherence Moving from data to constructs to theories through analysis and categorisation

Many more other analyses are useful…

Two Interpretations of the Research on Fuzzy Data Analysis

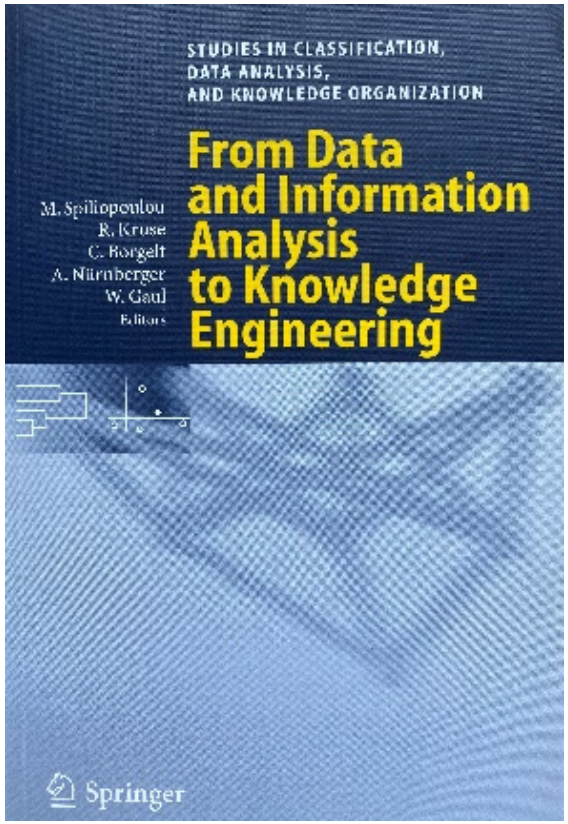
FUZZY data analysis = Fuzzy Techniques for the analysis of (crisp) data

In this course: Fuzzy Clustering

FUZZY DATA analysis = Analysis of Data that are described by Fuzzy Sets

In this course: Statistics with Fuzzy Data

Cluster Analysis



This is an **unsupervised** learning task.

The goal is to divide the dataset such that both constraints hold:

- objects belonging to same cluster: as similar as possible
- objects belonging to different clusters: as dissimilar as possible

The **similarity** is measured in terms of a **distance** function.

The smaller the distance, the more similar two data tuples.

Definition

$d : \mathbb{R}^p \times \mathbb{R}^p \rightarrow [0, \infty)$ is a distance function if $\forall \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \mathbb{R}^p$:

(i)	$d(\boldsymbol{x}, \boldsymbol{y}) = 0 \Leftrightarrow \boldsymbol{x} = \boldsymbol{y}$	(identity),
(ii)	$d(\boldsymbol{x}, \boldsymbol{y}) = d(\boldsymbol{y}, \boldsymbol{x})$	(symmetry),
(iii)	$d(\boldsymbol{x}, \boldsymbol{z}) \leq d(\boldsymbol{x}, \boldsymbol{y}) + d(\boldsymbol{y}, \boldsymbol{z})$	(triangle inequality).

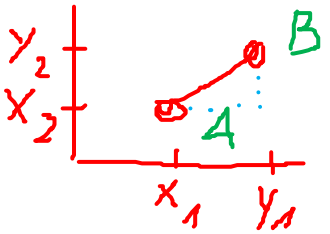
Illustration of Distance Functions

Minkowski family

$$d_k(\boldsymbol{x}, \boldsymbol{y}) = \left(\sum_{d=1}^{p} |x_d - y_d|^k \right)^{\frac{1}{k}}$$

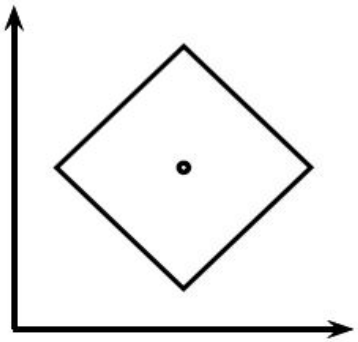
$(x_1, \ldots, x_p)$

$A(x_1, x_2)\ B(y_1, y_2)$

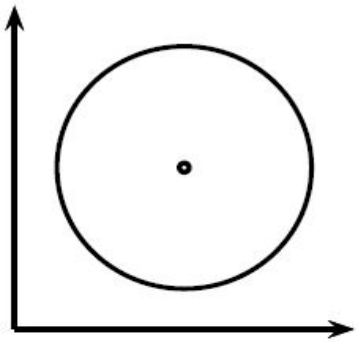


Well-known special cases from this family are

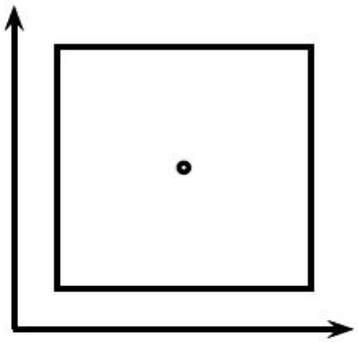
$k = 1$: Manhattan or city block distance,
$k = 2$: Euclidean distance,
$k \to \infty$: maximum distance, *i.e.* $d_\infty(\boldsymbol{x}, \boldsymbol{y}) = \max_{d=1}^{p} |x_d - y_d|$.



$k = 1$



$k = 2$



$k \to \infty$

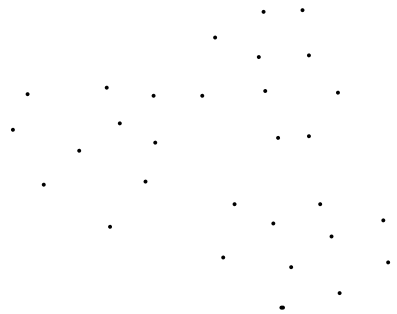
Partitioning Algorithms

Here, we focus only on partitioning algorithms,

- *i.e.* given $c \in \mathbb{N}$, find the best partition of data into c groups.

Usually the number of (true) clusters is unknown. However,

with partitioning methods we must specify a c-value.



$c = 2$

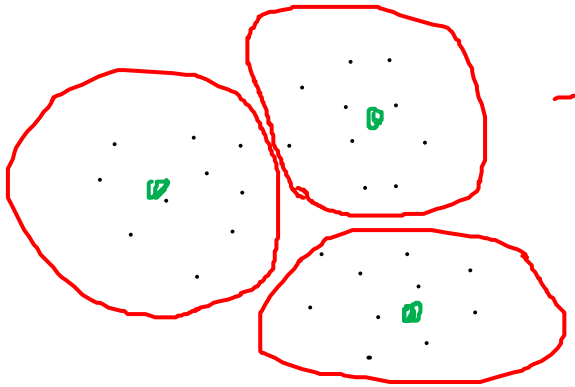
Partitioning Algorithms

Here, we focus only on partitioning algorithms,

- *i.e.* given $c \in \mathbb{N}$, find the best partition of data into c groups.

Usually the number of (true) clusters is unknown.

However, in partitioning methods we must specify a c-value.



→ c = 3

→ 3 point in the middle, cluster centers

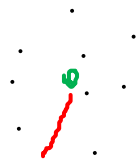
Another restriction of prototype-based clustering algorithms:

- Clusters are represented by cluster prototypes $C_i, i = 1, \ldots, c$.

Prototypes capture the structure (distribution) of data in each cluster.

The set of prototypes is $C = \{C_1, \ldots, C_c\}$.

example



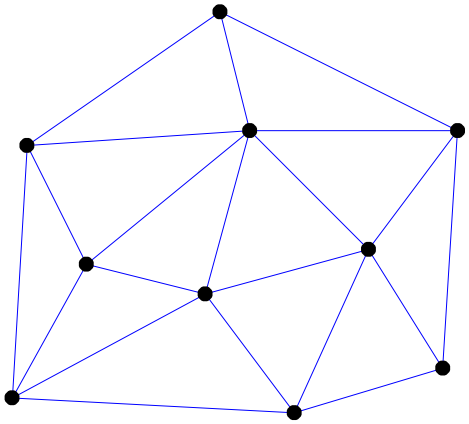
Every prototype C_i is an n-tuple with

- the cluster center $\boldsymbol{c}_i$ and
- additional parameters about its size and shape.

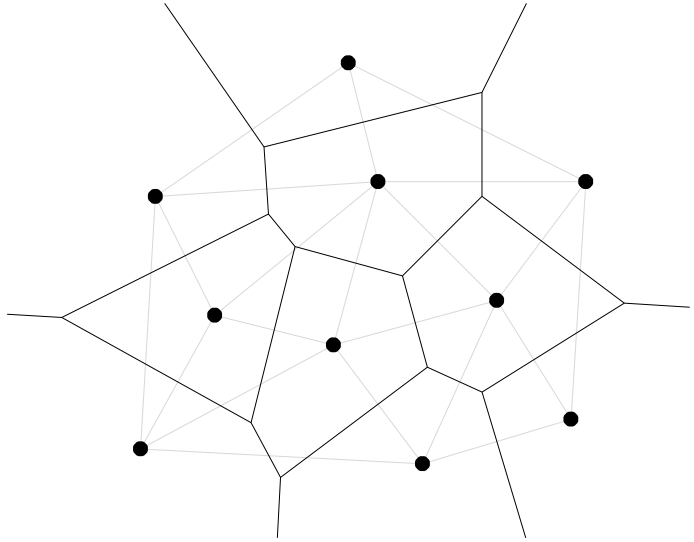
(center, radius)

Prototypes are constructed by clustering algorithms.

Dots represent cluster centers.



Delaunay Triangulation
The circle through the corners of a triangle does not contain another point.



Voronoi Diagram Midperpendiculars of the Delaunay triangulation: boundaries of the regions of points that are closest to the enclosed cluster center (Voronoi cells).

(Hard) *c*-Means Clustering

1) Choose a number *c* of clusters to be found (user input).

2) Initialize the cluster centers randomly
(for instance, by randomly selecting *c* data points).

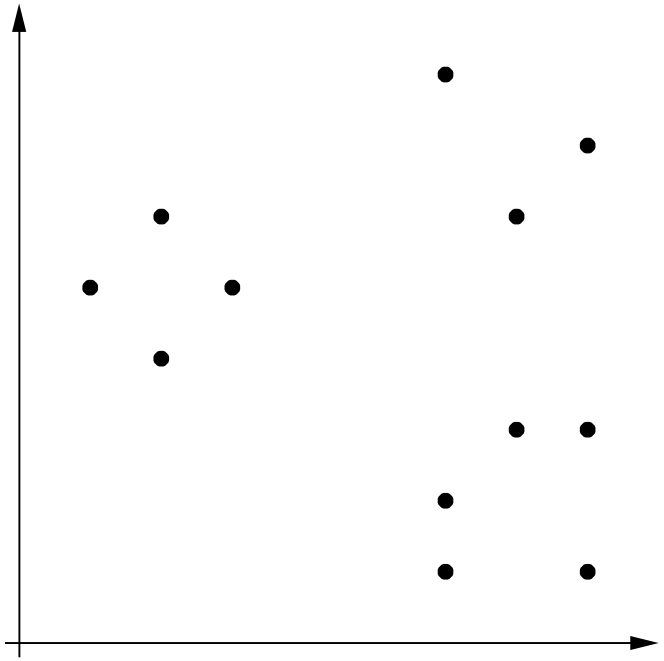
3) Data point assignment:
Assign each data point to the cluster center that is closest to it
(i.e. closer than any other cluster center).

4) Cluster center update:
Compute new cluster centers as mean of the assigned data points.
(Intuitively: center of gravity)

5) Repeat steps 3 and 4 until cluster centers remain constant.

It can be shown that this scheme must converge

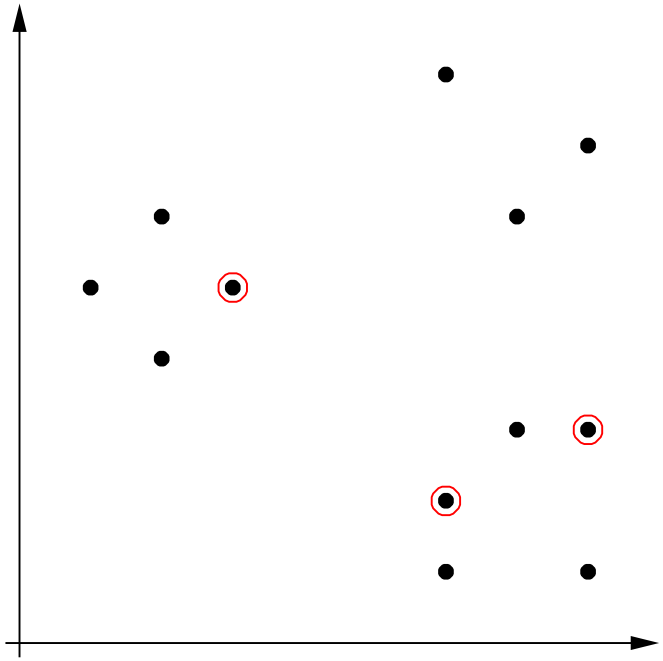
c-Means Clustering: Example



Data set to cluster.

Choose $c = 3$ clusters.

(From visual inspection, can be difficult to determine in general.)

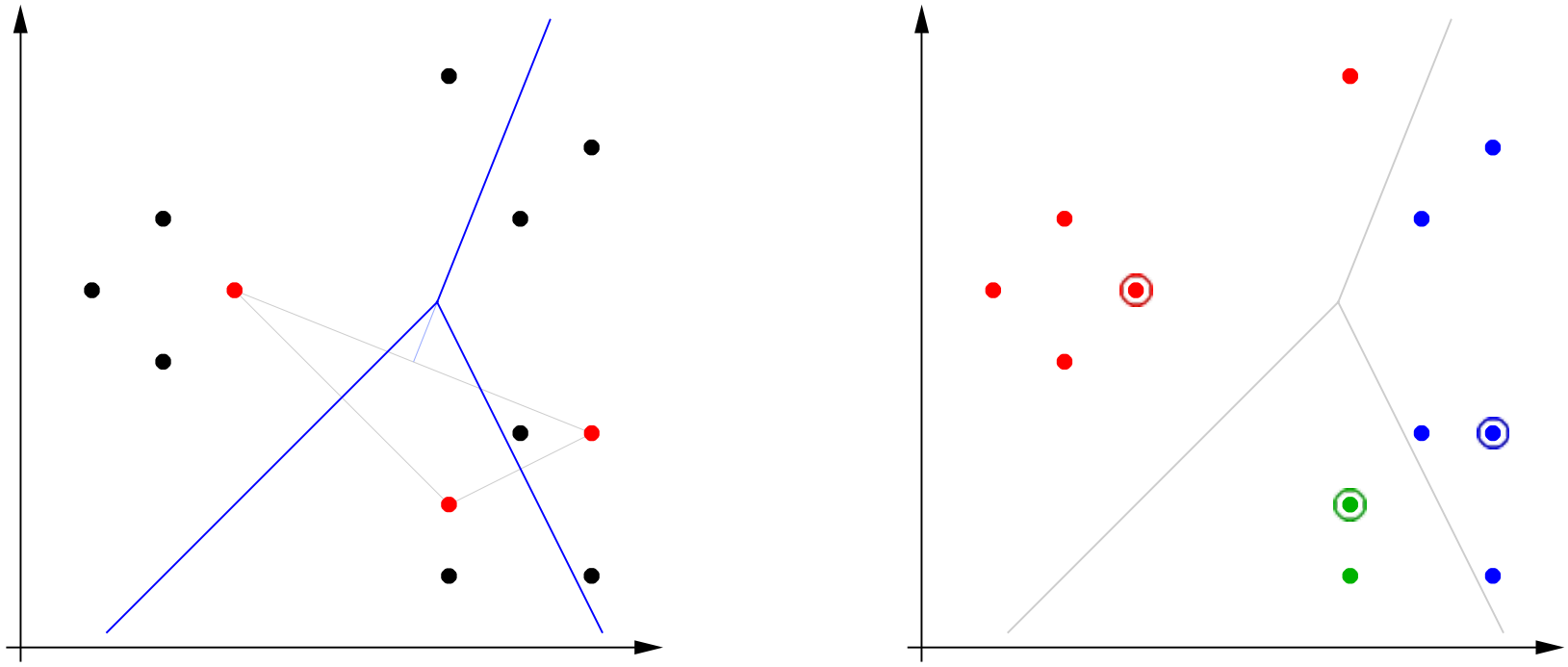


Initial position of cluster centers.

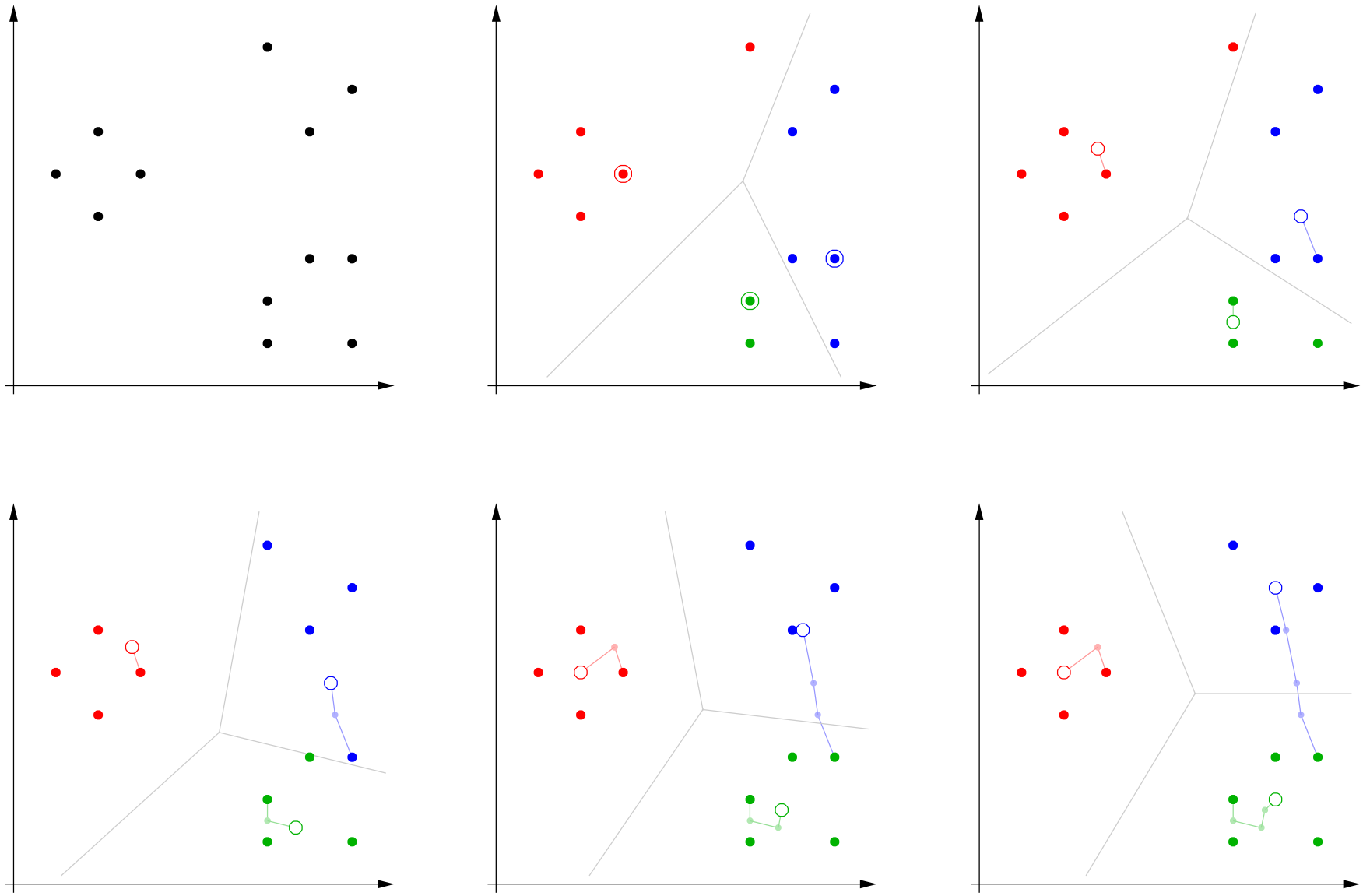
Randomly selected data points.
(Alternative methods include e.g. latin hypercube sampling)

Delaunay Triangulation: simple triangle (shown in grey on the left)

Voronoi Diagram: midperpendiculars of the triangle's edges (shown in blue on the left, in grey on the right)



c-Means Clustering: Example



c-Means Clustering: Local Minima

In the example Clustering was successful and formed intuitive clusters

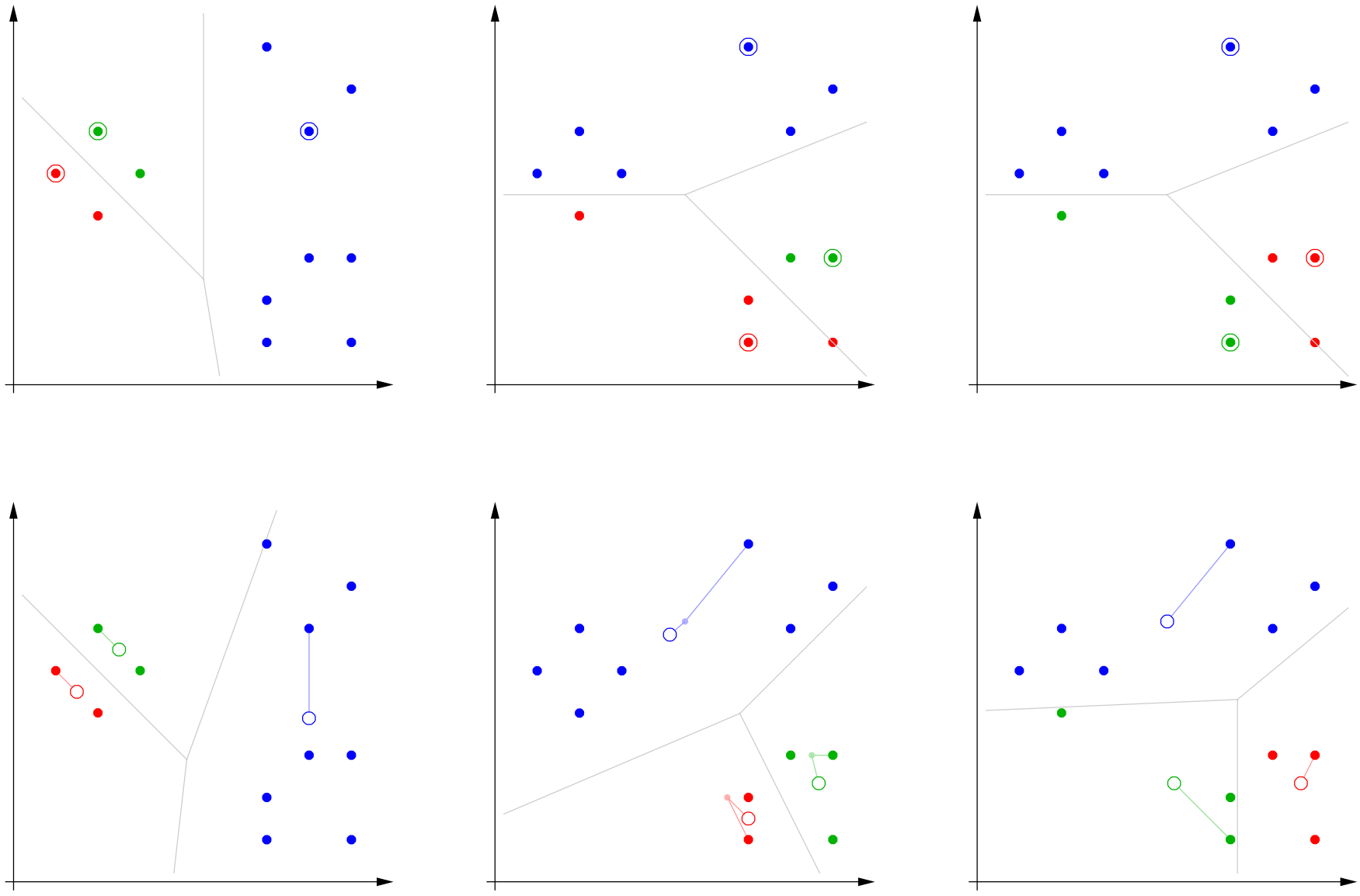
Convergence achieved after only 5 steps.
(This is typical: convergence is usually very fast.)

Nevertheless: Result is **sensitive to the initial positions** of cluster centers.

With a bad initialization clustering may fail (the alternating update process gets stuck in a local minimum).

The number of clusters has to be known. Some approaches for determining a reasonable number of clusters exists.

c-Means Clustering: Local Minima



Center Vectors and Objective Functions

Consider the simplest cluster prototypes, *i.e.* center vectors $C_i = (\boldsymbol{c}_i)$.

The distance d is based on the Euclidean distance.

All algorithms are based on an objective functions J which

- quantifies the goodness of the cluster models and
- must be minimized to obtain optimal clusters.

Cluster algorithms determine the best decomposition by minimizing J.

Hard *c*-means

Each point x_j in the dataset $X = \{x_1, \ldots, x_n\}$, $X \subseteq \mathbb{R}^p$ is assigned to exactly one cluster.

That is, each cluster $\Gamma_i \subset X$.

The set of clusters $\Gamma = \{\Gamma_1, \ldots, \Gamma_c\}$ must be an exhaustive partition of X into c non-empty and pairwise disjoint subsets Γ_i, $1 < c < n$.

The data partition is optimal when the sum of squared distances between cluster centers and data points assigned to them is minimal.

Clusters should be as homogeneous as possible.

The objective function of the hard c-means is

$$J_h(X, U\ , C) = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij} d_{ij}^2$$

where d_{ij} is the distance between $\boldsymbol{c}_i$ and $\boldsymbol{x}_j$, *i.e.* $d_{ij} = d(\boldsymbol{c}_i, \boldsymbol{x}_j)$, and $U = (u_{ij})_{c \times n}$ is the the **partition matrix** with

$$u_{ij} = \begin{cases} 1, & \text{if } \boldsymbol{x}_j \in \Gamma_i \\ 0, & \text{otherwise.} \end{cases}$$

Each data point is assigned exactly to one cluster and every cluster must contain at least one data point:

$$\forall j \in \{1, \ldots, n\} : \quad \sum_{i=1}^{c} u_{ij} = 1 \quad \text{and} \quad \forall i \in \{1, \ldots, c\} : \quad \sum_{j=1}^{n} u_{ij} > 0.$$

AO Scheme for Hard c-means

(i) Chose an initial $\boldsymbol{c}_i$, *e.g.* randomly picking c data points $\in X$.

(ii) Hold C fixed and determine U that minimize J_h:
Each data point is assigned to its closest cluster center:

$$u_{ij} = \begin{cases} 1, & \text{if } i = \arg\min_{k=1}^{c} d_{kj} \\ 0, & \text{otherwise.} \end{cases}$$

(iii) Hold U fixed, update $\boldsymbol{c}_i$ as mean of all x_j assigned to them:
The mean minimizes the sum of square distances in J_h:

$$\boldsymbol{c}_i = \frac{\sum_{j=1}^{n} u_{ij} \boldsymbol{x}_j}{\sum_{j=1}^{n} u_{ij}}.$$

(iv) Repeat steps (ii)+(iii) until no changes in C or U are observable.

Discussion: Hard *c*-means

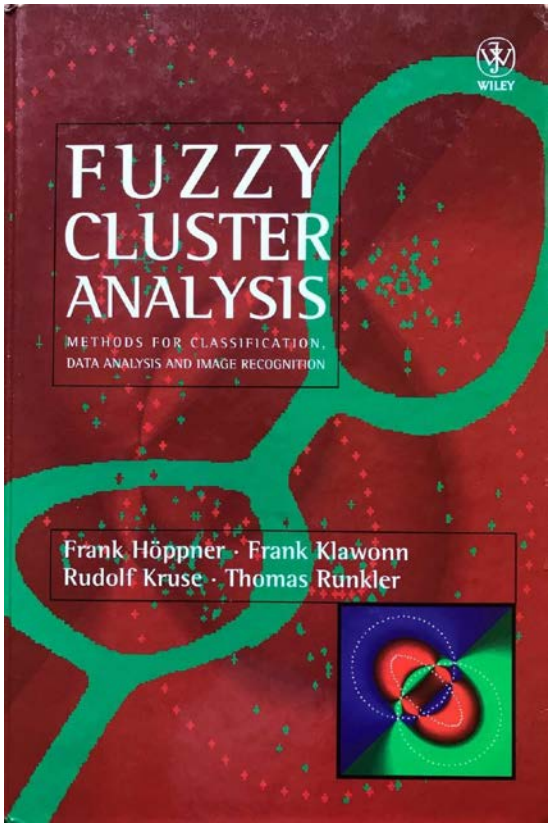
It tends to get stuck in a local minimum.

Thus necessary are several runs with different initializations.There are sophisticated initialization methods available, *e.g.* Latin hypercube sampling. The best result of many clusterings is chosen based on J_h.

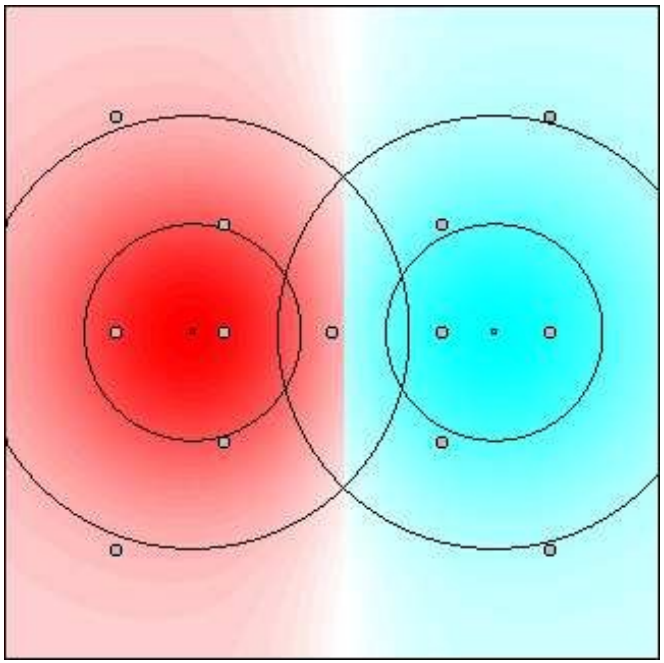
Crisp memberships {0, 1} prohibit ambiguous assignments.

For badly delineated or overlapping clusters, one should relax the requirement $u_{ij} \in \{0, 1\}$.

Fuzzy Cluster Analysis



Example



Given a symmetric dataset with two clusters.

Hard *c*-means assigns a crisp label to the data point in the middle.

That‘s not very intuitive.

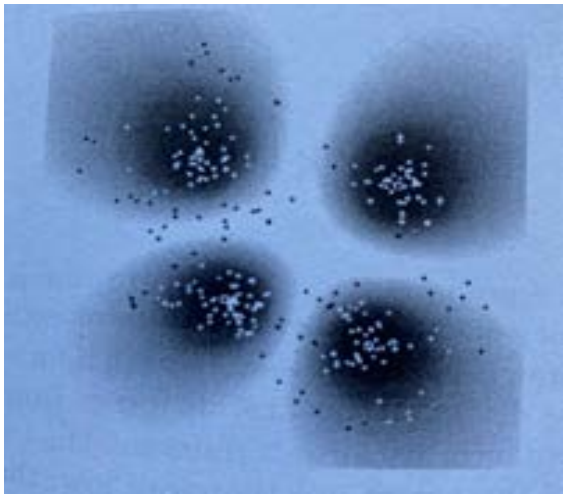
In lots of cases the concept of a „fuzzy cluster“ is more intuitive.

Fuzzy Clustering

Fuzzy clustering allows gradual memberships of data points to clusters.

It offers the flexibility to express whether a data point belongs to more than one cluster.

Thus, membership degrees
- offer a finer degree of detail of the data model, and
- express how ambiguously/definitely a data point belong to a cluster.



Note that in the picture there are 4 clusters, the element are represented by dots (white/black is used for visibility only) , the greyness indicates the (maximum) membership to a cluster

The clusters Γ_i have been classical subsets so far.

Now, they are represented by fuzzy sets μ_{Γ_i} of X.

Thus, u_{ij} is a membership degree of $\boldsymbol{x}_j$ to Γ_i such that $u_{ij} = \mu_{\Gamma_i}(\boldsymbol{x}_j) \in [0, 1]$.

The fuzzy label vector $\boldsymbol{u} = (u_{1j}, \ldots, u_{cj})^T$ is linked to each $\boldsymbol{x}_j$.

$U = (u_{ij}) = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n)$ is called **fuzzy partition matrix**.

There are several types of fuzzy cluster partitions.

Definition

Let $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ be the set of given examples and let c be the number of clusters ($1 < c < n$) represented by the fuzzy sets $\mu_{\Gamma_i}, (i = 1, \ldots, c)$. Then we call $U = (u_{ij}) = (\mu_{\Gamma_i}(\boldsymbol{x}_j))$ a *fuzzy cluster partition* of X if

$$\sum_{i=1}^{n} u_{ij} > 0, \quad \forall i \in \{1, \ldots, c\}, \quad \text{and}$$

$$\sum_{i=1}^{c} u_{ij} = 1, \quad \forall j \in \{1, \ldots, n\}$$

hold. The $u_{ij} \in [0, 1]$ are interpreted as the membership degree of datum $\boldsymbol{x}_j$ to cluster Γ_i relative to all other clusters.

Fuzzy Cluster Partition

The 1st constraint guarantees that there aren't any empty clusters.

- This is a requirement in classical cluster analysis.
- Thus, no cluster, represented as (classical) subset of X , is empty.

The 2nd condition says that sum of membership degrees must be 1 for each $\boldsymbol{x}_j$.

- Each datum gets the same weight compared to other data points.
- So, all data are (equally) included into the cluster partition.
- This relates to classical clustering where partitions are exhaustive.

The consequence of both constraints are as follows:

- No cluster can contain the full membership of all data points.
- The membership degrees *resemble* probabilities of being member of corresponding cluster.

Fuzzy c-means Clustering

Minimize the objective function

$$J_f(X, U_h, C) = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^m d_{ij}^2$$

subject to

$$\sum_{i=1}^{c} u_{ij} = 1, \quad \forall j \in \{1, \ldots, n\}$$

and

$$\sum_{j=1}^{n} u_{ij} > 0, \quad \forall i \in \{1, \ldots, c\}$$

where d_{ij} is a distance between cluster i and data point j,

and $m \in \mathbb{R}$ with $m > 1$ is the so called fuzzifier

Fuzzy c-means Clustering

Objective Function The objects are assigned to the clusters in such a way that the sum of the (weighted) squared distances becomes minimal. The optimization takes place under the constraints required from a fuzzy partition.

Distance There are several options for defining the distance between a data point and a cluster. In most cases the Euclidean distance between the data point and the cluster center is used.

Fuzzifier The actual value of m determines the “fuzziness” of the classification. Clusters become softer/harder with a higher/lower value of m. Often m = 2 is chosen.

How to minimize the objective function?

Some mathematical technics and tricks are necessary!

Reminder: Function Optimization

Task: Find $\boldsymbol{x} = (x_1, \ldots, x_m)$ such that $f(\boldsymbol{x}) = f(x_1, \ldots, x_m)$ is optimal.

Often a feasible approach is to

- define the necessary condition for (local) optimum (max./min.): partial derivatives *w.r.t.* parameters vanish.
- Thus we (try to) solve an equation system coming from setting all partial derivatives *w.r.t.* the parameters equal to zero.

Example task: minimize $f(x, y) = x^2 + y^2 + xy - 4x - 5y$

Solution procedure:

- Take the partial derivatives of f and set them to zero:

$$\frac{\partial f}{\partial x} = 2x + y - 4 = 0, \qquad \frac{\partial f}{\partial y} = 2y + x - 5 = 0.$$

- Solve the resulting (here linear) equation system: $x = 1,\ y = 2$.

We can use the **Method of Lagrange Multipliers:**

Given: $f(\boldsymbol{x})$ to be optimized, k equality constraints

$C_j(\boldsymbol{x}) = 0,\ 1 \le j \le k$

procedure:

1. Construct the so-called **Lagrange function** by incorporating $C_i,\ i = 1, \ldots, k,$ with (unknown) **Lagrange multipliers** λ_i:

$$L(\boldsymbol{x}, \lambda_1, \ldots, \lambda_k) = f(\boldsymbol{x}) + \sum_{i=1}^{k} \lambda_i C_i(\boldsymbol{x}).$$

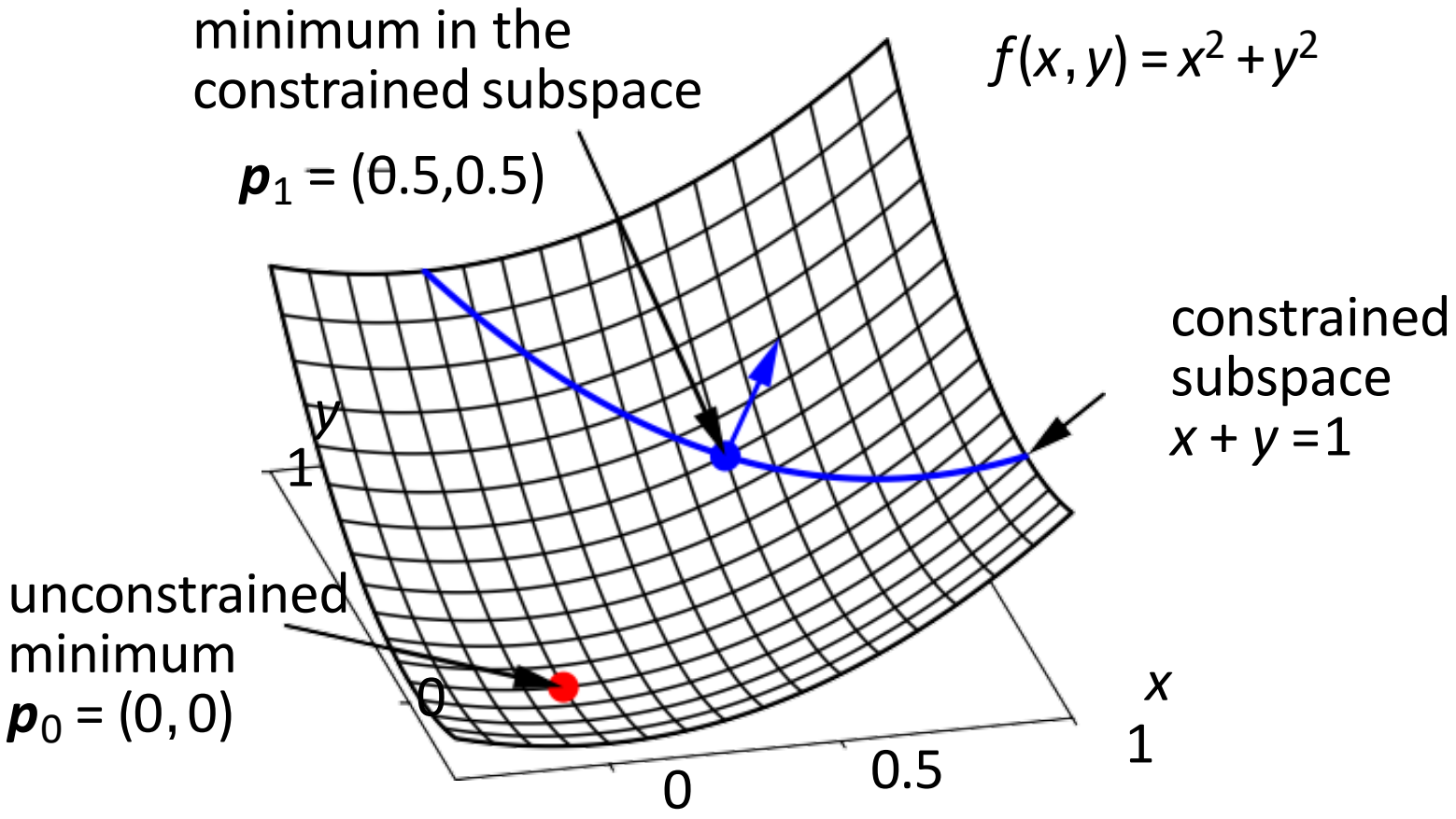
2. Set the partial derivatives of Lagrange function equal to zero:

$$\frac{\partial L}{\partial x_1} = 0, \quad \ldots, \quad \frac{\partial L}{\partial x_m} = 0, \quad \frac{\partial L}{\partial \lambda_1} = 0, \quad \ldots, \quad \frac{\partial L}{\partial \lambda_k} = 0.$$

3. (Try to) solve the resulting equation system.

Lagrange Theory: Revisited Example 1

Example task: Minimize $f(x, y) = x^2 + y^2$ subject to $x + y = 1$.



The unconstrained minimum is not in the constrained subspace. At the minimum in the constrained subspace the gradient does not vanish.

Lagrange Theory: Example 2

Example task: find the side lengths x, y, z of a box with maximum volume for a given area S of the surface.

Formally: max. $f(x, y, z) = xyz$ such that $2xy + 2xz + 2yz = S$

Solution procedure:

- The constraint is $C(x, y, z) = 2xy + 2xz + 2yz - S = 0$.
- The Lagrange function is

$$L(x, y, z, \lambda) = xyz + \lambda(2xy + 2xz + 2yz - S).$$

- Taking the partial derivatives yields (in addition to constraint):

$$\frac{\partial L}{\partial x} = yz + 2\lambda(y+z) = 0, \quad \frac{\partial L}{\partial y} = xz + 2\lambda(x+z) = 0, \quad \frac{\partial L}{\partial z} = xy + 2\lambda(x+y) = 0.$$

- The solution is $\lambda = -\frac{1}{4}\sqrt{\frac{S}{6}}, \quad x = y = z = \sqrt{\frac{S}{6}}$ (*i.e.* box is a cube).

Alternating Optimization Scheme

Finding the parameters that minimize J_h is NP-hard.

Fuzzy *c*-means minimizes J_f by **alternating optimization (AO)**:

- The parameters to optimize are split into 2 groups.
- One group is optimized holding other one fixed (and vice versa).
- This is an iterative update scheme: repeated until convergence.

There is no guarantee that the global optimum will be reached.

AO may get stuck in a local minimum.

Optimizing the Objective Function

J_f is alternately optimized, *i.e.*

- optimize U for a fixed cluster parameters $U_\tau = j_U(C_{\tau-1})$,
- optimize C for a fixed membership degrees $C_\tau = j_C(U_\tau)$.

The update formulas are obtained by setting the derivative J_f *w.r.t.* parameters U, C to zero. .

The resulting equations form the fuzzy c-means (FCM) algorithm

Introduce the Lagrange multipliers λ_j, $0 \leq j \leq n$, to incorporate the constraints $\forall j; 1 \leq j \leq n : \sum_{i=1}^{c} u_{ij} = 1$.

Then, the Lagrange function (to be minimized) is

$$L(X, U_f, C, \Lambda) = \underbrace{\sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^m d_{ij}^2}_{=J(X,U_f,C)} + \sum_{j=1}^{n} \lambda_j \left(1 - \sum_{i=1}^{c} u_{ij}\right).$$

The necessary condition for a minimum is that the partial derivatives of the Lagrange function *w.r.t.* membership degrees vanish, *i.e.*

$$\frac{\partial}{\partial u_{kl}} L(X, U_f, C, \Lambda) = m u_{kl}^{m-1} d_{kl}^2 - \lambda_l \stackrel{!}{=} 0$$

which leads to

$$\forall i; 1 \leq i \leq c : \forall j; 1 \leq j \leq n : \qquad u_{ij} = \left(\frac{\lambda_j}{m d_{ij}^2}\right)^{\frac{1}{m-1}}.$$

Optimizing the Membership Degrees

Summing these equations over clusters leads

$$1 = \sum_{i=1}^{c} u_{ij} = \sum_{i=1}^{c} \left(\frac{\lambda_j}{m d_{ij}^2} \right)^{\frac{1}{m-1}}.$$

Thus the λ_j, $1 \leq j \leq n$ are

$$\lambda_j = \left(\sum_{i=1}^{c} \left(m d_{ij}^2 \right)^{\frac{1}{1-m}} \right)^{1-m}.$$

Inserting this into the equation for the membership degrees yields

$$\forall i; 1 \leq i \leq c : \forall j; 1 \leq j \leq n : \qquad u_{ij} = \frac{d_{ij}^{\frac{2}{1-m}}}{\sum_{k=1}^{c} d_{kj}^{\frac{2}{1-m}}}.$$

This update formula is independent of any distance measure.

Second step: Fix the Membership Degrees

The optimization of the Cluster Prototypes depends on the chosen parameters (location, shape, size) and the distance measure. So a general update formula cannot be given.

If the cluster center serve as prototypes and the Euclidean distance between cluster centers and data point are used, then the second step yields

$$c_i = \frac{\sum_{j=1}^{n} u_{ij}^m x_j}{\sum_{j=1}^{n} u_{ij}^m}.$$

The two steps are performed iteratively until convergence. The algorithm finds a local minimum. Therefore the choice of different initializations are recommemded.

Example

Given 200 Swiss Banknotes.
How to find groups of similar Banknotes? (unsupervised learning)

Six variables (attributes) were recorded for each banknote
- The width of the banknote (WIDTH),
- the height on the banknote on the left (LEFT),
- the height on the banknote on the right (RIGHT),
- the distance between the colored print and the upper edge of the banknote (UPPER),
- the distance between the colored print and the lower edge of the banknote (LOWER) and
- the diagonal (bottom left to top right) of the colored print on the banknote (DIAGONAL).

Each banknote is characterized by a point in a 6-dimensional space.

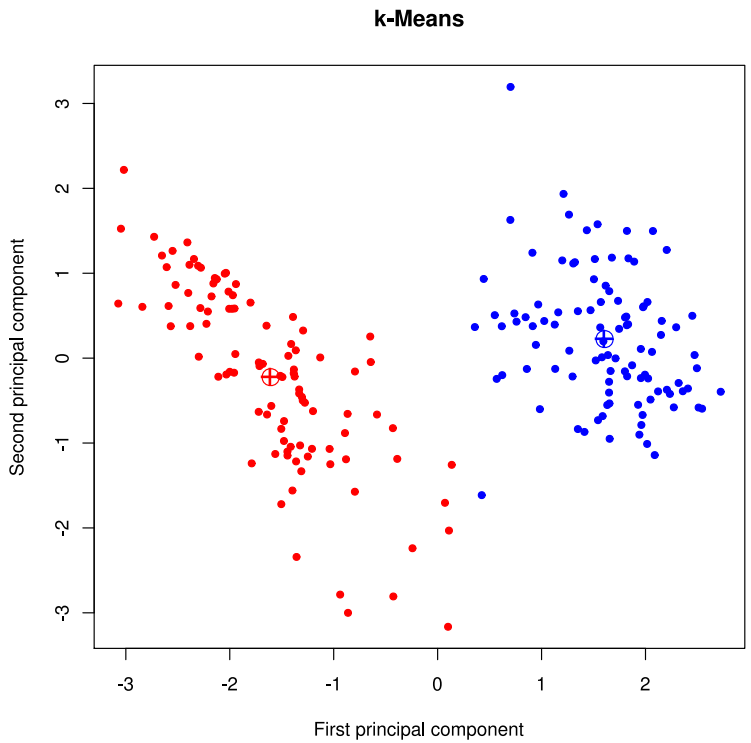
Example

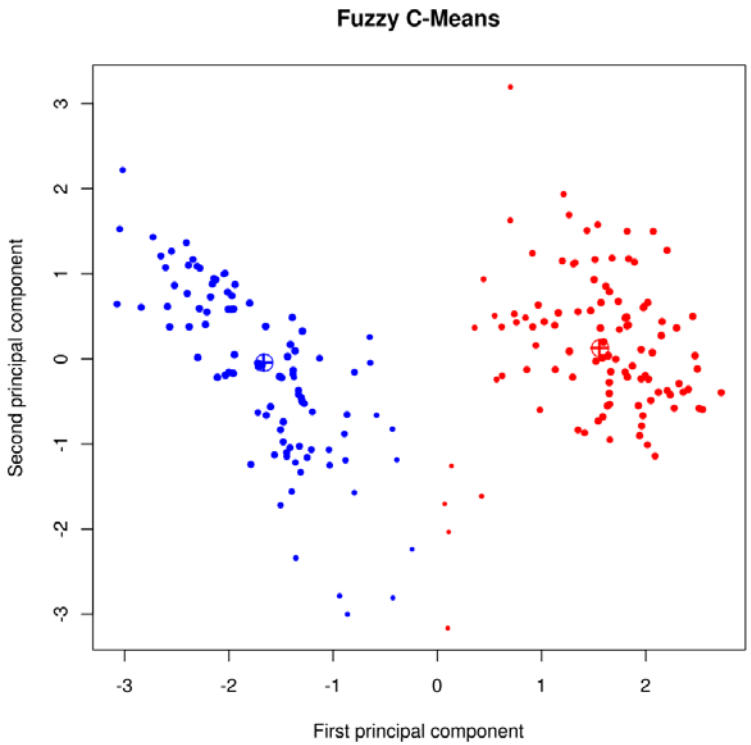
c-means clustering and Fuzzy c-means clustering is performed with 6-dim data

The results are presented only for the two principal components. In the fuzzy case the thickness of dots describes the membership to a respective cluster.

The clustering result of c-means (red/blue) in comparison with true classification (points below/above the diagonal) is already very good (only one misclassification).

The high uncertainty of the fuzzy algorithm near the diagonal is expressed by very small dots. In fact, the real banknotes were printed from one printing plate, while the counterfeit banknotes came from different sources and thus probably also from different forged printing plates.





Discussion: Fuzzy *c*-means clustering

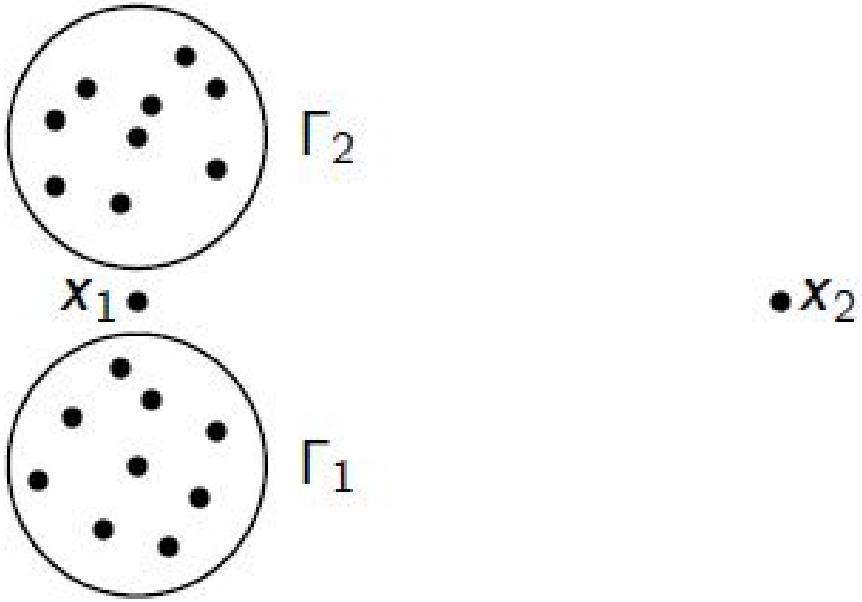
FCM is stable and robust.

Compared to hard *c*-means clustering, it's

- less insensitive to initialization and
- less likely to get a bad optimization result

There are lots of real applications, case studies, implementations , theoretic results and variants of the classical fuzzy c-means algorithm.

Variants of fuzzy *c*-means clustering



x_1 has the same distance to Γ_1 and $\Gamma_2 \Rightarrow \mu_{\Gamma_1}(x_1) = \mu_{\Gamma_2}(x_1) = 0.5$.

The same degrees of membership are assigned to x_2.

Variants of Fuzzy *c*-means clustering

The normalization of memberships is a problem for noise and outliers.

A fixed data point weight causes a high membership of noisy data, although there is a large distance from the bulk of the data.

This has a bad effect on the clustering result.

Dropping the normalization constraint

$$\sum_{i=1}^{c} u_{ij} = 1, \quad \forall j \in \{1, \ldots, n\},$$

we obtain sometimes more intuitive membership assignments.

Definition

Let $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ be the set of given examples and let c be the number of clusters $(1 < c < n)$ represented by the fuzzy sets $\mu_{\Gamma_i}, (i = 1, \ldots, c)$. Then we call $U_p = (u_{ij}) = (\mu_{\Gamma_i}(\boldsymbol{x}_j))$ a *possibilistic cluster partition* of X if

$$\sum_{j=1}^{n} u_{ij} > 0, \quad \forall i \in \{1, \ldots, c\}$$

holds. The $u_{ij} \in [0, 1]$ are interpreted as degree of representativity or typicality of the datum $\boldsymbol{x}_j$ to cluster Γ_i.

now, u_{ij} for $\boldsymbol{x}_j$ resemble possibility of being member of corresponding cluster

The objective function for fuzzy clustering J_f is not appropriate for possibilistic clustering: Dropping the normalization constraint leads to a minimum for all u_{ij} =0.

Hence a penalty term is introduced which forces all u_{ij} away from zero.

The objective function J_f is modified to

$$J_p(X, U_p, C) = \sum_{i=1}^{c}\sum_{j=1}^{n} u_{ij}^m d_{ij}^2 + \sum_{i=1}^{c} \eta_i \sum_{j=1}^{n} (1 - u_{ij})^m$$

where $\eta_i > 0 (1 \leq i \leq c)$.

The values η_i balance the contrary objectives expressed in J_p.

Optimizing the Membership Degrees

The update formula for membership degrees is

$$u_{ij} = \frac{1}{1 + \left(\frac{d_{ij}^2}{\eta_i}\right)^{\frac{1}{m-1}}}$$

The membership of $\boldsymbol{x}_j$ to cluster i depends only on d_{ij} to this cluster.

A small distance corresponds to a high degree of membership.

Larger distances result in low membership degrees.

So, u_{ij}'s share a typicality interpretation.

Interpretation of η_i

The update equation helps to explain the parameters η_i.

Consider $m = 2$ and substitute η_i for d_{ij}^2 yields $u_{ij} = 0.5$.

Thus η_i determines the distance to Γ_i at which u_{ij} should be 0.5.

η_i can have a different geometrical interpretation:

- the hyperspherical clusters (*e.g.* PCM), thus $\sqrt{\eta_i}$ is the mean diameter.

Estimating η_i

If such properties are known, η_i can be set a priori.

If all clusters have the same properties, the same value for all clusters should be used.

However, information on the actual shape is often unknown a priori.

- So, the parameters must be estimated, *e.g.* by FCM.
- One can use the fuzzy intra-cluster distance, *i.e.* for all Γ_i, $1 \leq i \leq n$

$$\eta_i = \frac{\sum_{j=1}^{n} u_{ij}^m d_{ij}^2}{\sum_{j=1}^{n} u_{ij}^m}.$$

Optimizing the Cluster Centers

The update equations j_C are derived by setting the derivative of J_p *w.r.t.* the prototype parameters to zero (holding U_p fixed).

The update equations for the cluster prototypes are identical.

Then the cluster centers in the PCM algorithm are re-estimated as

$$\boldsymbol{c}_i = \frac{\sum_{j=1}^{n} u_{ij}^m \boldsymbol{x}_j}{\sum_{j=1}^{n} u_{ij}^m}.$$

Comparison of Clustering Results

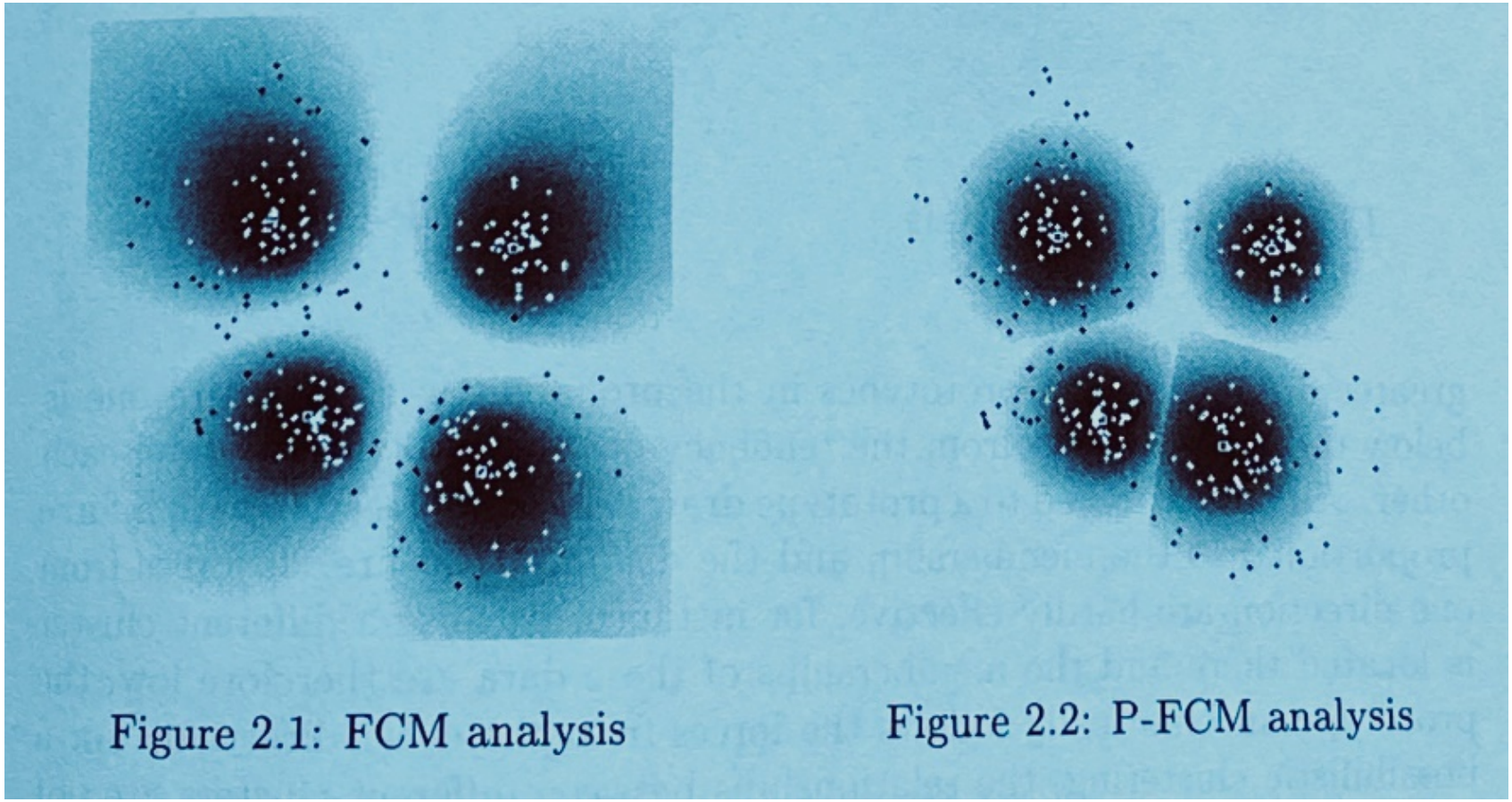


Figure 2.1: FCM analysis

Figure 2.2: P-FCM analysis

Note that in the picture there are 4 clusters, the element are represented by dots (white/black is used for visibility only) , the greyness indicates the (maximum) membership to a cluster

Distance Function Variants

So far, only Euclidean distance leading to standard FCM and PCM

Euclidean distance only allows spherical clusters

Several variants have been proposed to relax this constraint

- fuzzy Gustafson-Kessel algorithm
- fuzzy shell clustering algorithms

Can be applied to FCM and PCM

Gustafson-Kessel Algorithm

Replacement of the Euclidean distance by cluster-specific Mahalanobis distance

For cluster Γ_i, its associated Mahalanobis distance is defined as

$$d^2(\boldsymbol{x}_j, C_j) = (\boldsymbol{x}_j - \boldsymbol{c}_i)^T \Sigma_i^{-1} (\boldsymbol{x}_j - \boldsymbol{c}_i)$$

where Σ_i is covariance matrix of cluster

Euclidean distance leads to $\forall i : \Sigma_i = I$, *i.e.* identity matrix

Gustafson-Kessel (GK) algorithm leads to prototypes $C_i = (\boldsymbol{c}_i, \Sigma_i)$

Iris Data

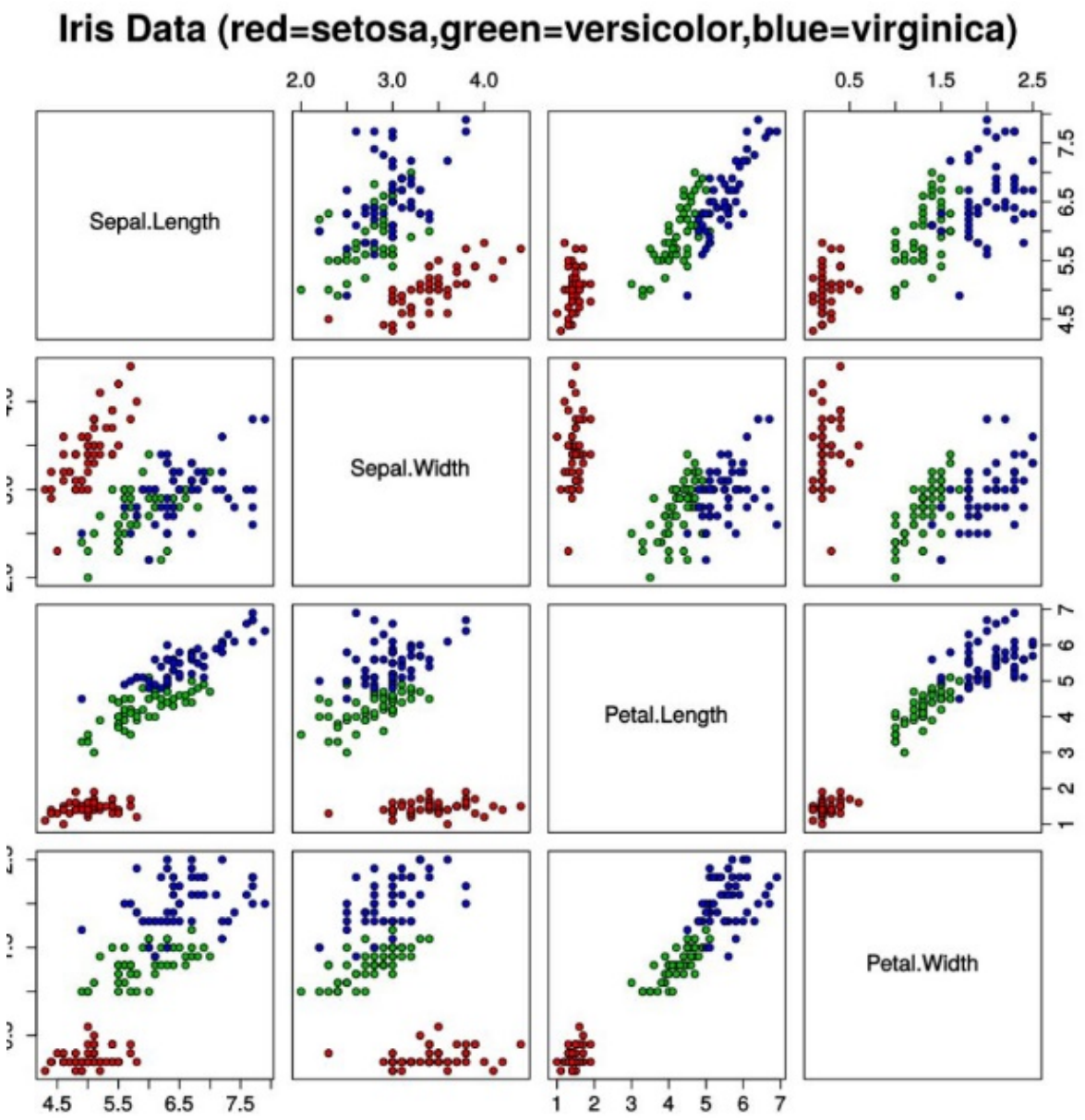
Iris setosa

Iris versicolor

Iris virginica

Collected by Ronald Aylmer Fischer (famous statistician).
150 cases in total, 50 cases per Iris flower type.
Measurements: sepal length/width, petal length/width (in cm).
Most famous dataset in pattern recognition and data analysis.

Classification of flowers

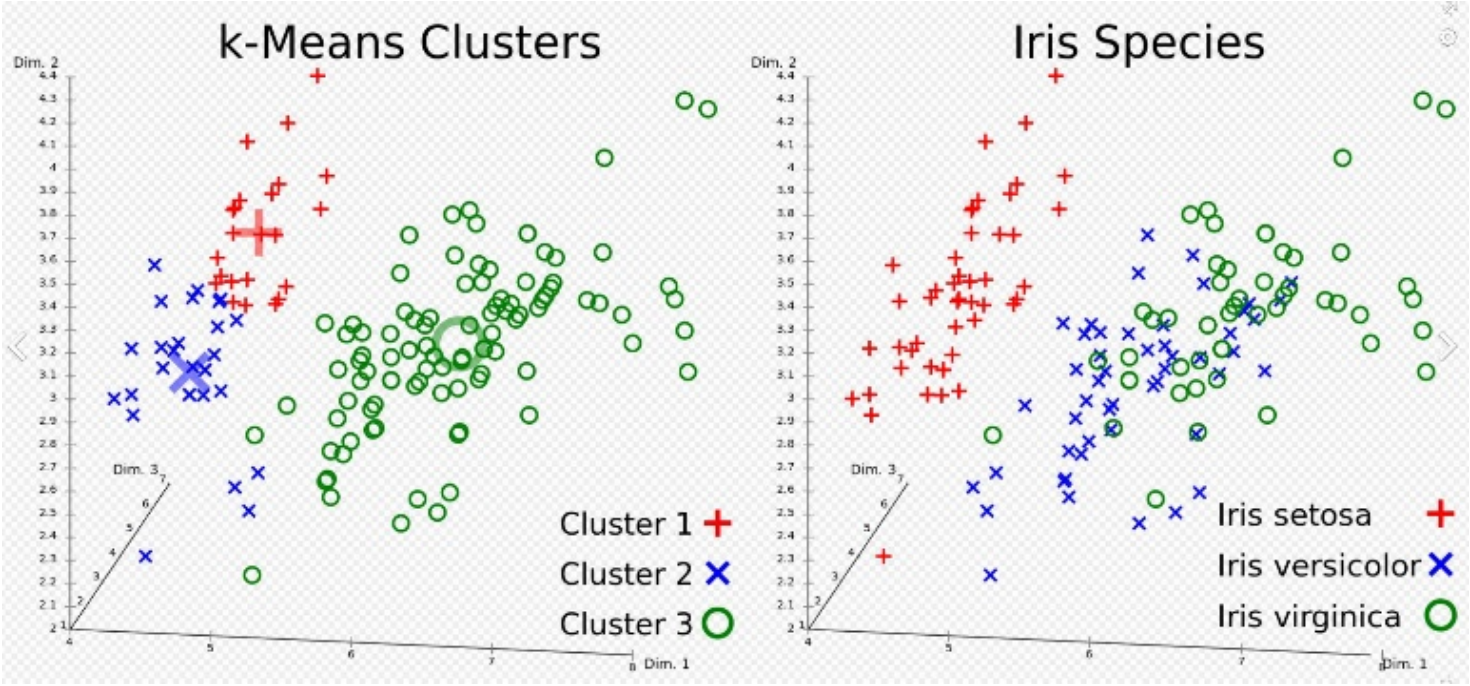


- Classification is supervised learning
- Visual Analysis with a Scatter plot: 4-dim Data represented in by 12 2-dim projections of **labelled** data
- Automatic classification is possible, e.g. by support vector machines
- There are also fuzzy versions of SVM

Clustering of flowers

Clustering is unsupervised learning based on unlabelled data

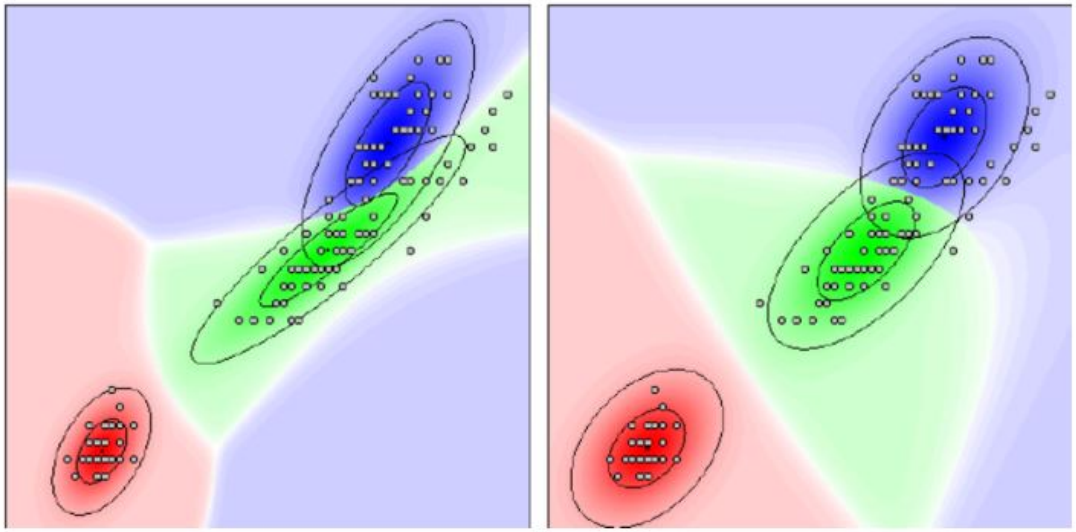
Testing clustering methods: Use labelled data, remove the labels of the data ,
cluster the unlabelled data, compare the clustering result with the original classification



Result of c-means clustering
on **Unlabelled** data

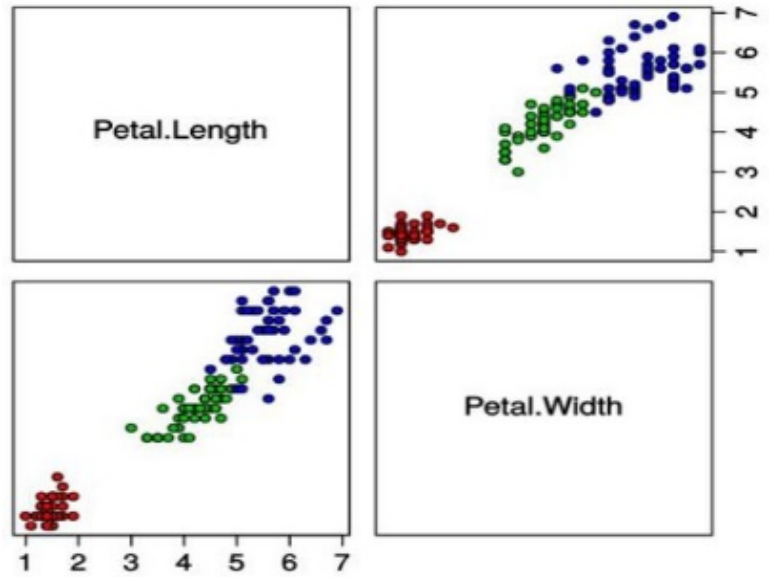
True Classification
Labelled data

Gustafson Kessel Clustering of the (unlabelled) Iris Data



Result of Gustafson-Kessel algorithm on the iris data with fixed cluster size without (left) and with shape regularization (right, method 2 with r = 4). Both images show the petal length (horizontal) and the petal width (vertical). Clustering was done on all four attributes (sepal length and sepal width in addition to the above). .

Good clustering result with only few errors



Projected original labelled data

Fuzzy Shell Clustering

Up to now: searched for convex “cloud-like” clusters

Corresponding algorithms = **solid clustering** algorithms

Especially useful in data analysis

For image recognition and analysis:
variants of FCM and PCM to detect lines, circles or ellipses

shell clustering algorithms

Replace Euclidean by other distances

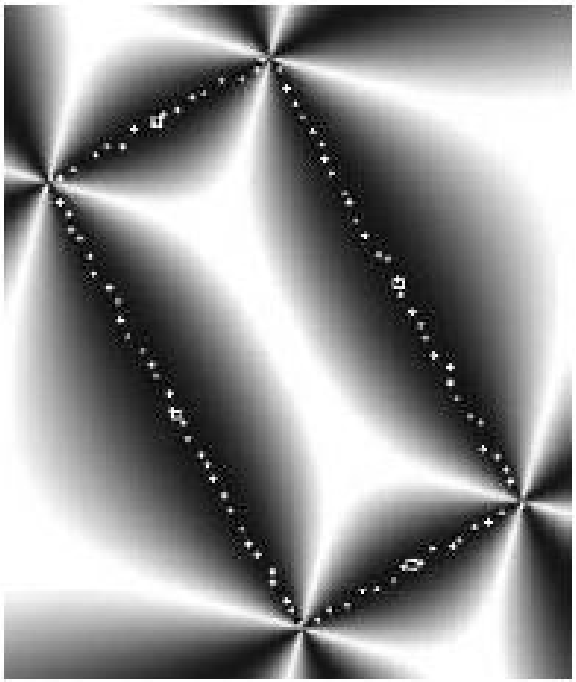
Fuzzy c-varieties Algorithm

Fuzzy c-varieties (FCV) algorithm recognizes lines, planes, or hyperplanes

Each cluster is affine subspace characterized by point and set of orthogonal unit vectors, $C_i = (\boldsymbol{c}_i, \boldsymbol{e}_{i1}, \ldots, \boldsymbol{e}_{iq})$ where q is dimension of affine subspace

Distance between data point $\boldsymbol{x}_j$ and cluster i

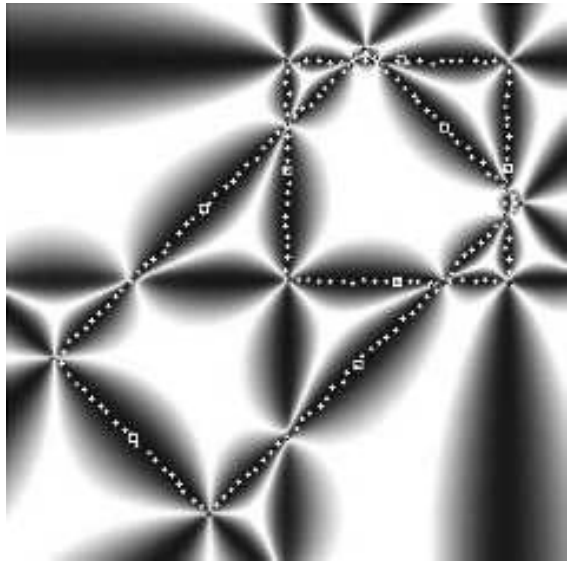
$$d^2(\boldsymbol{x}_j, \boldsymbol{c}_i) = \|\boldsymbol{x}_j - \boldsymbol{c}_i\|^2 - \sum_{l=1}^{q} (\boldsymbol{x}_j - \boldsymbol{c}_i)^T \boldsymbol{e}_{il}$$



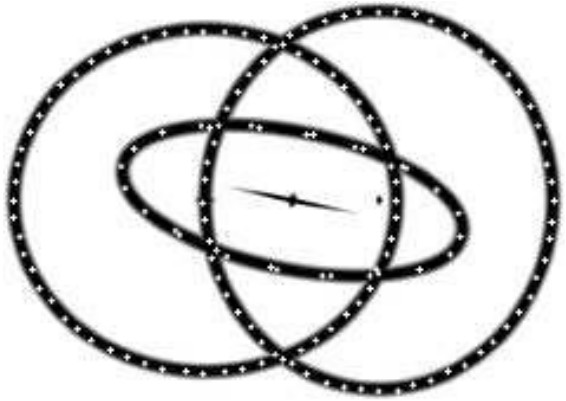
Also used for locally linear models of data with underlying functional interrelations

Other Shell Clustering Algorithms

Name	Prototypes
adaptive fuzzy *c*-elliptotypes (AFCE)	line segments
fuzzy *c*-shells	circles
fuzzy *c*-ellipsoidal shells	ellipses
fuzzy *c*-quadric shells (FCQS)	hyperbolas, parabolas
fuzzy *c*-rectangular shells (FCRS)	rectangles



AFCE



FCQS

FCRS