OTTO VON GUERICKE
UNIVERSITÄT
MAGDEBURG

INF

FAKULTÄT FÜR
INFORMATIK

Fuzzy Systems

Fuzzy Clustering 1

Prof. Dr. Rudolf Kruse **Christoph Doell**

{kruse,doell}@iws.cs.uni-magdeburg.de
Otto-von-Guericke University of Magdeburg
Faculty of Computer Science
Department of Knowledge Processing and Language Engineering

Outline

Example: Automatic Gear Box

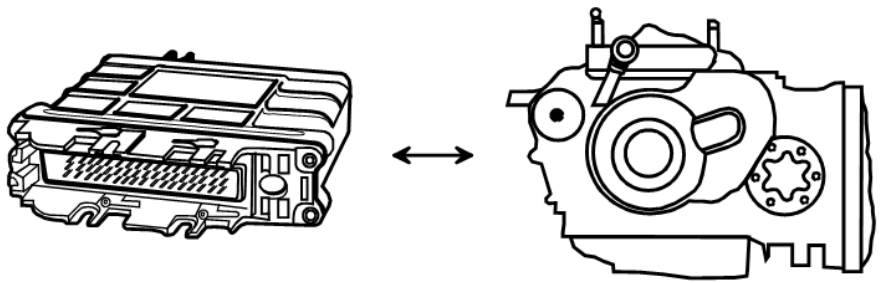
7 Mamdani fuzzy rules

Optimized program

- 24 byte RAM
- 702 byte ROM

Runtime 80 ms

- 12 times per second new sport factor is assigned



How to find these fuzzy rules?

Learning from Examples (Observations)

Statistics:

- Parameter fitting, structure identification, inference method, model selection

Machine learning:

- Computational learning (PAC learning), inductive learning, decision tree learning, concept learning

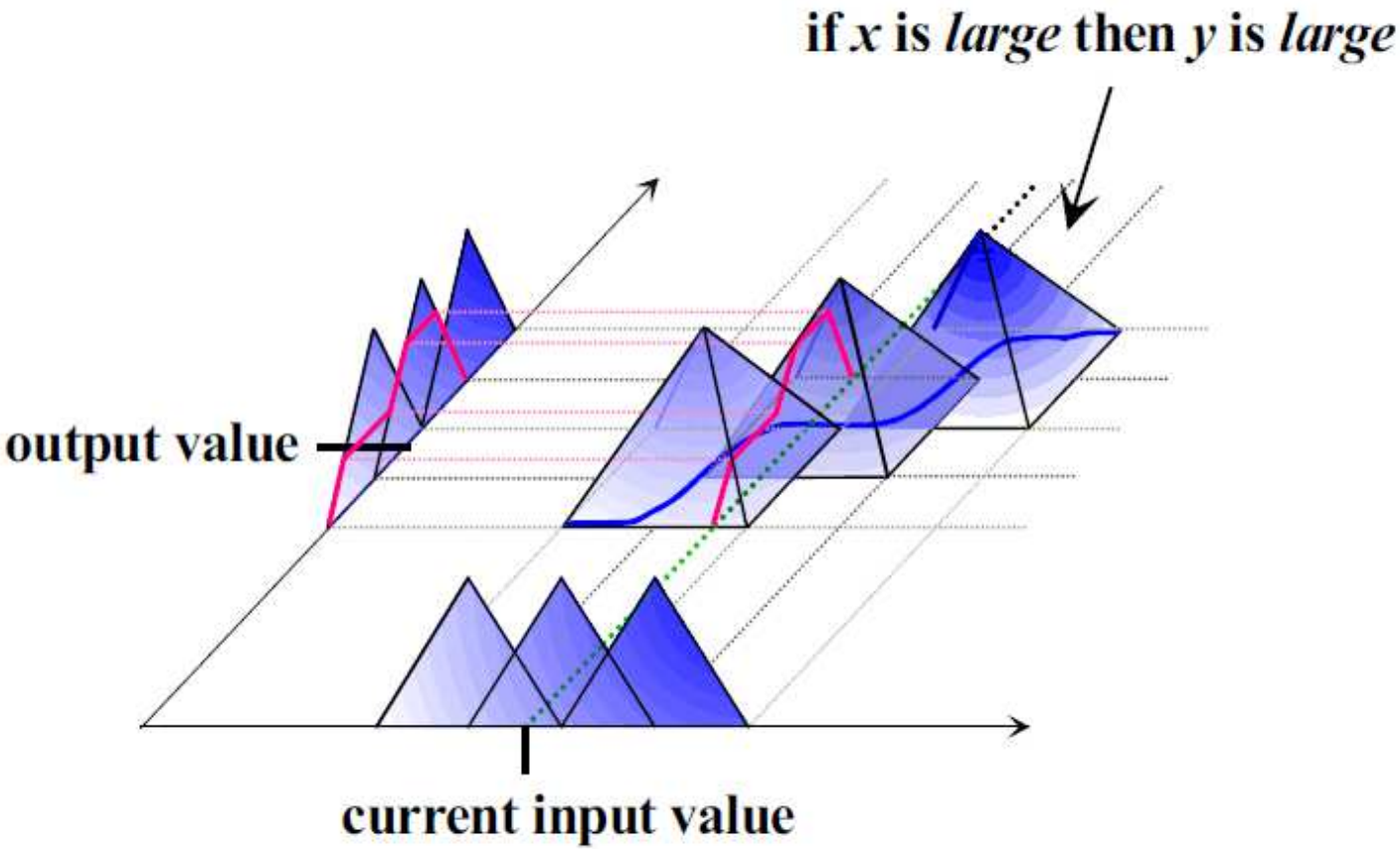
Neural networks: learning from data

Cluster analysis: unsupervised learning

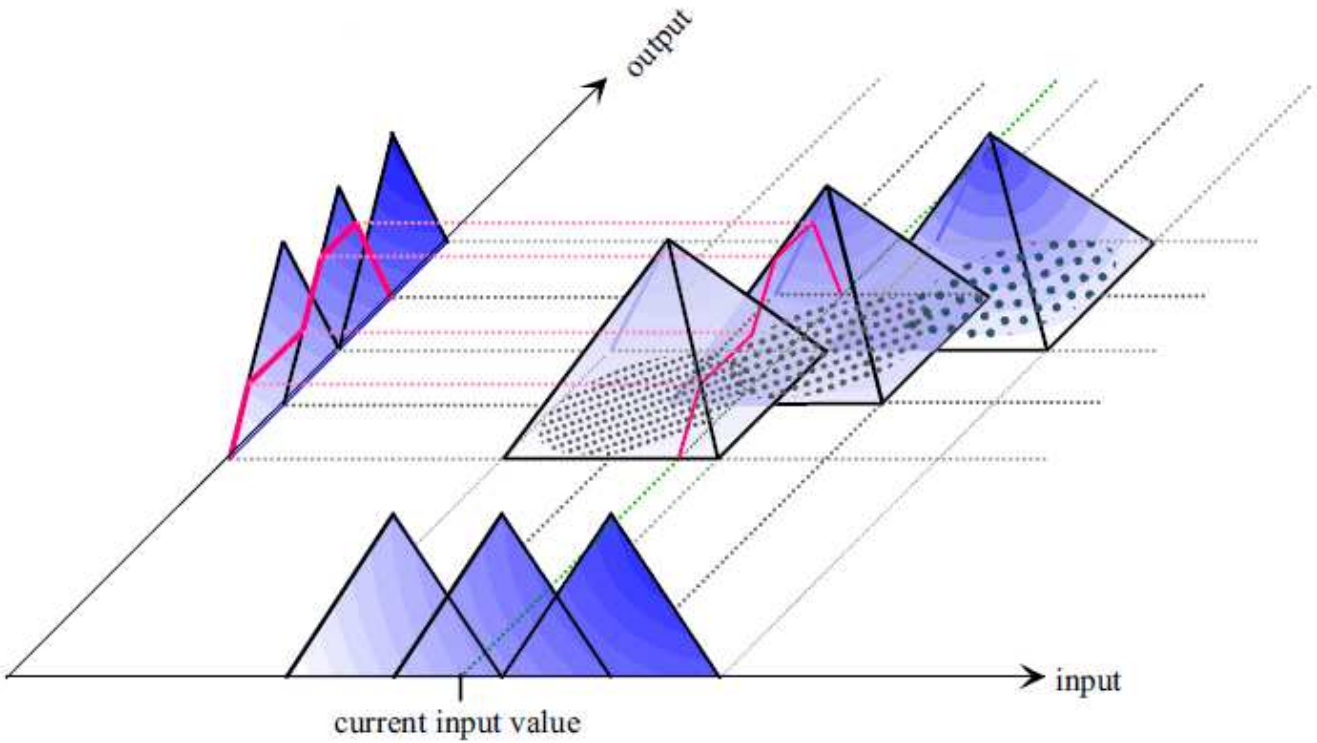
The learning problem becomes an optimization problem.

How to use these methods in fuzzy systems?

Function Approximation with Fuzzy Rules



Learning Fuzzy Rules from Data



Perform a fuzzy cluster analysis of the input-output data.

Then project the clusters.

Finally, obtain the fuzzy rules, *e.g.* "if x is small, then y is medium".

Outline

Clustering

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

$d : \mathbb{R}^p \times \mathbb{R}^p \to [0, \infty)$ 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).

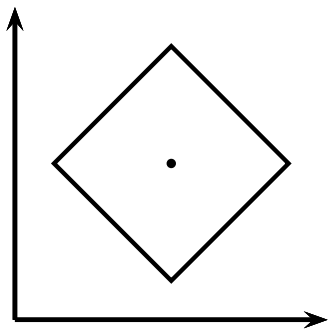
Illustration of Distance Functions

Minkowski family

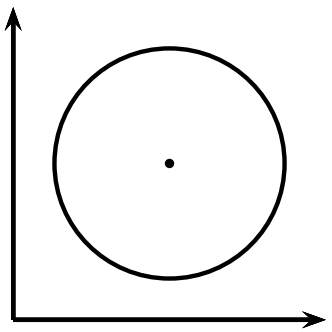
$$d_k(\boldsymbol{x}, \boldsymbol{y}) = \left(\sum_{d=1}^{p} |x_d - y_d|^k \right)^{\frac{1}{k}}$$

Well-known special cases from this family are

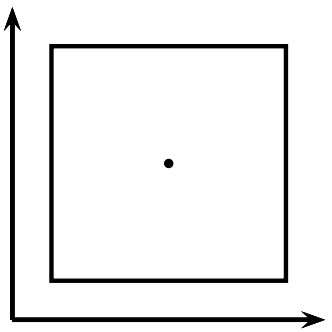
$k = 1$: Manhattan or city block distance,
$k = 2$: Euclidean distance,
$k \to \infty$: maximum distance, *i.e.* $d_\infty(\boldsymbol{x}, \boldsymbol{y}) = \max_{d=1}^{p} |x_d - y_d|$.



$k = 1$



$k = 2$



$k \to \infty$

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.
- That is different from hierarchical techniques,
  i.e. organize data in a nested sequence of groups.

Usually the number of (true) clusters is unknown.

However, partitioning methods must specify a c-value.

Prototype-based Clustering

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 $C = \{C_1, \ldots, C_c\}$.

Every prototype C_i is an n-tuple with

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

Prototypes are constructed by clustering algorithms.

(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).

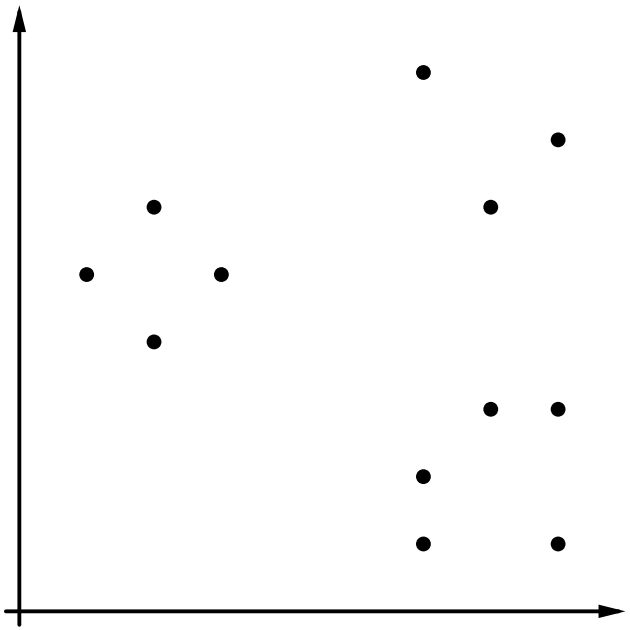
3) **Data point assignment**:
Assign each data point to the cluster center that is closest to it
(i.e. closer than any other cluster center).

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.

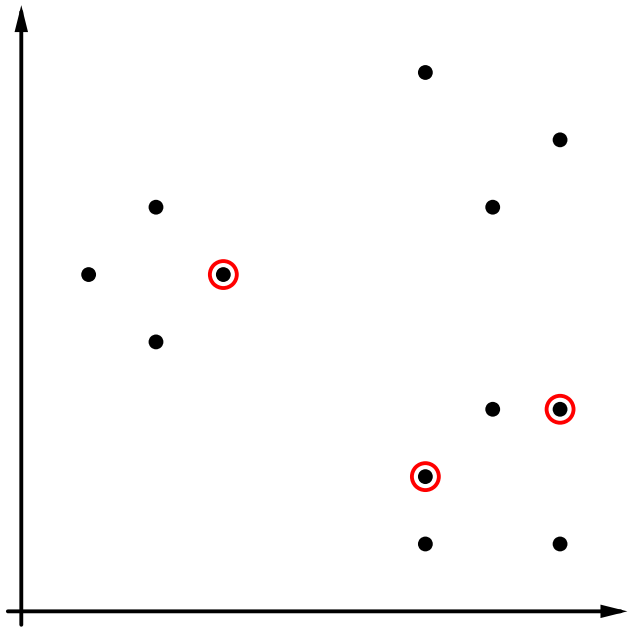
It can be shown that this scheme must converge

c-Means Clustering: Example



Data set to cluster.

Choose $c = 3$ clusters.
(From visual inspection, can be difficult to determine in general.)

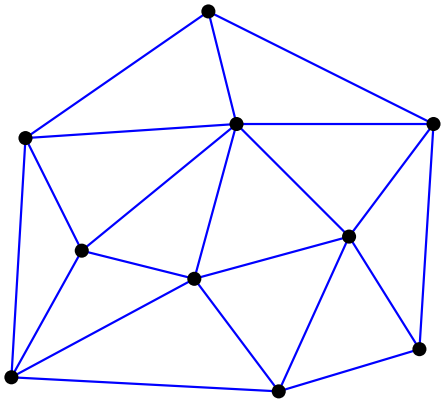


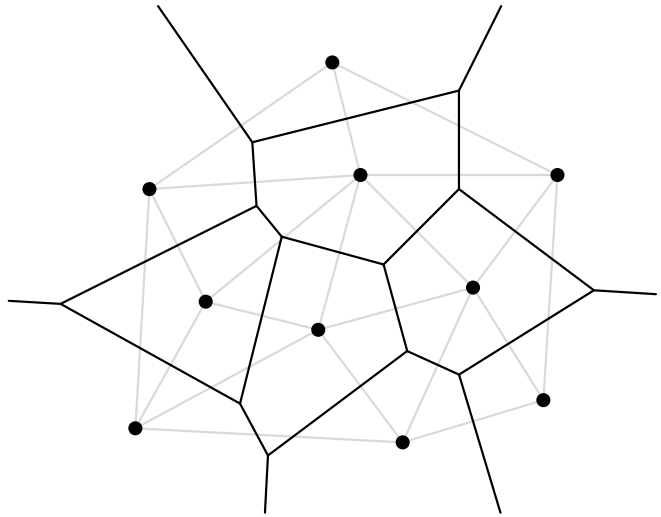
Initial position of cluster centers.

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

Delaunay Triangulations and Voronoi Diagrams

Dots represent cluster centers (quantization vectors).





Delaunay Triangulation

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

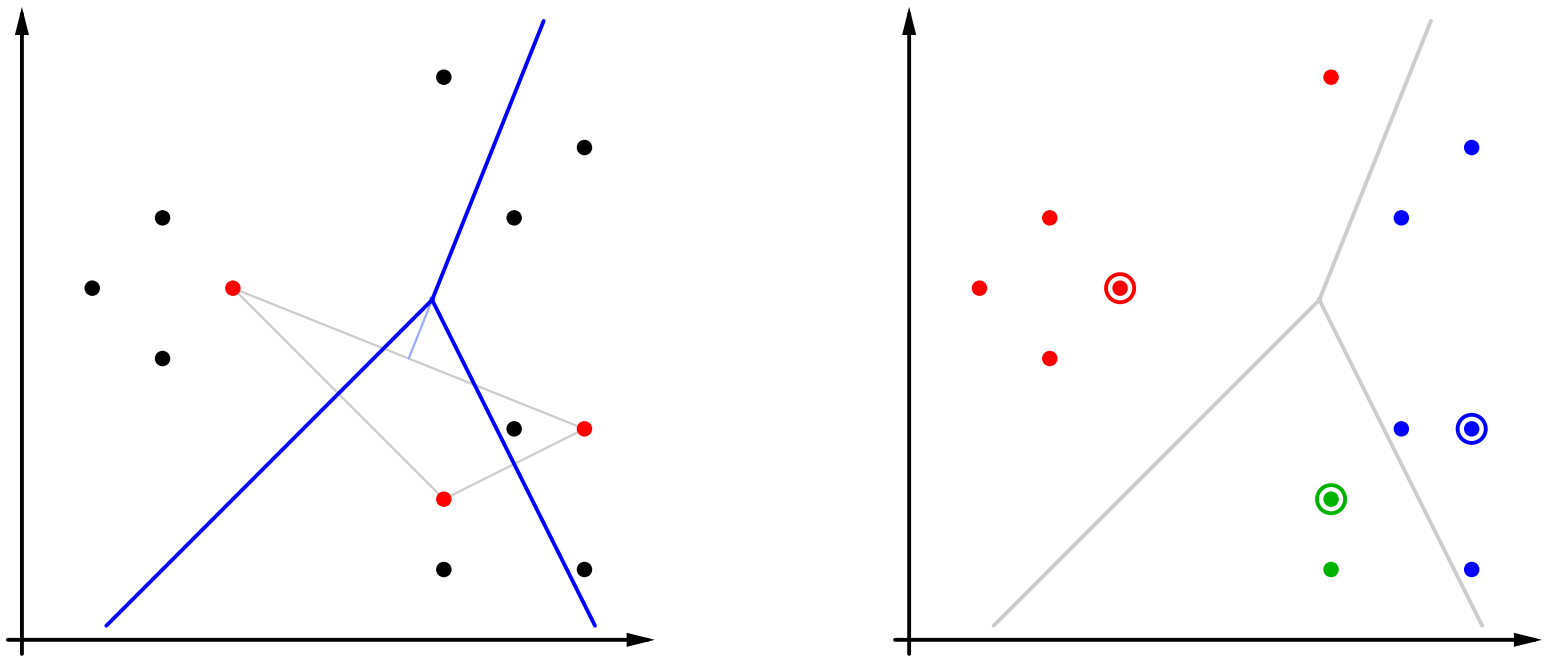
Voronoi Diagram

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

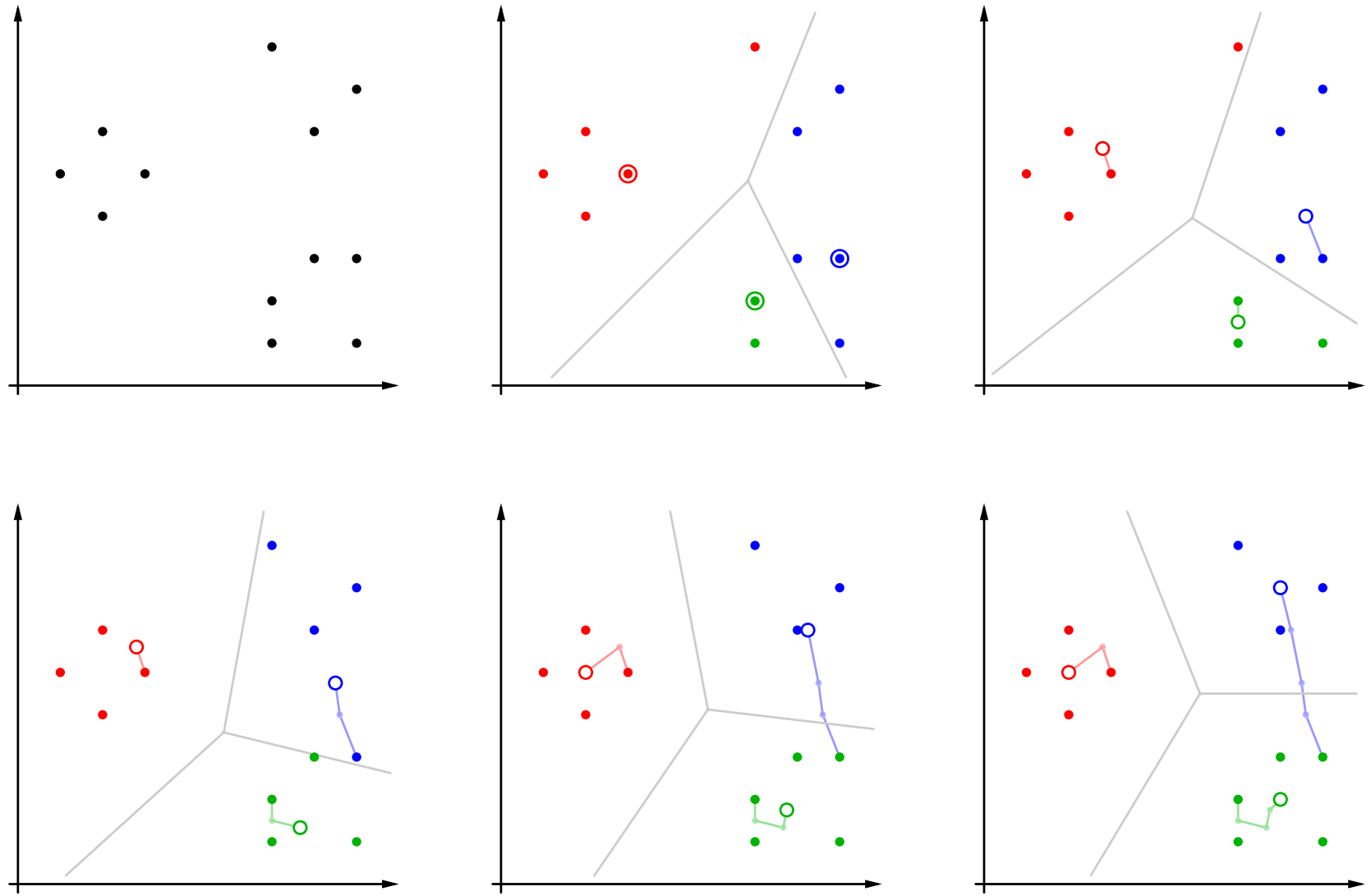
Delaunay Triangulations and Voronoi Diagrams

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.)

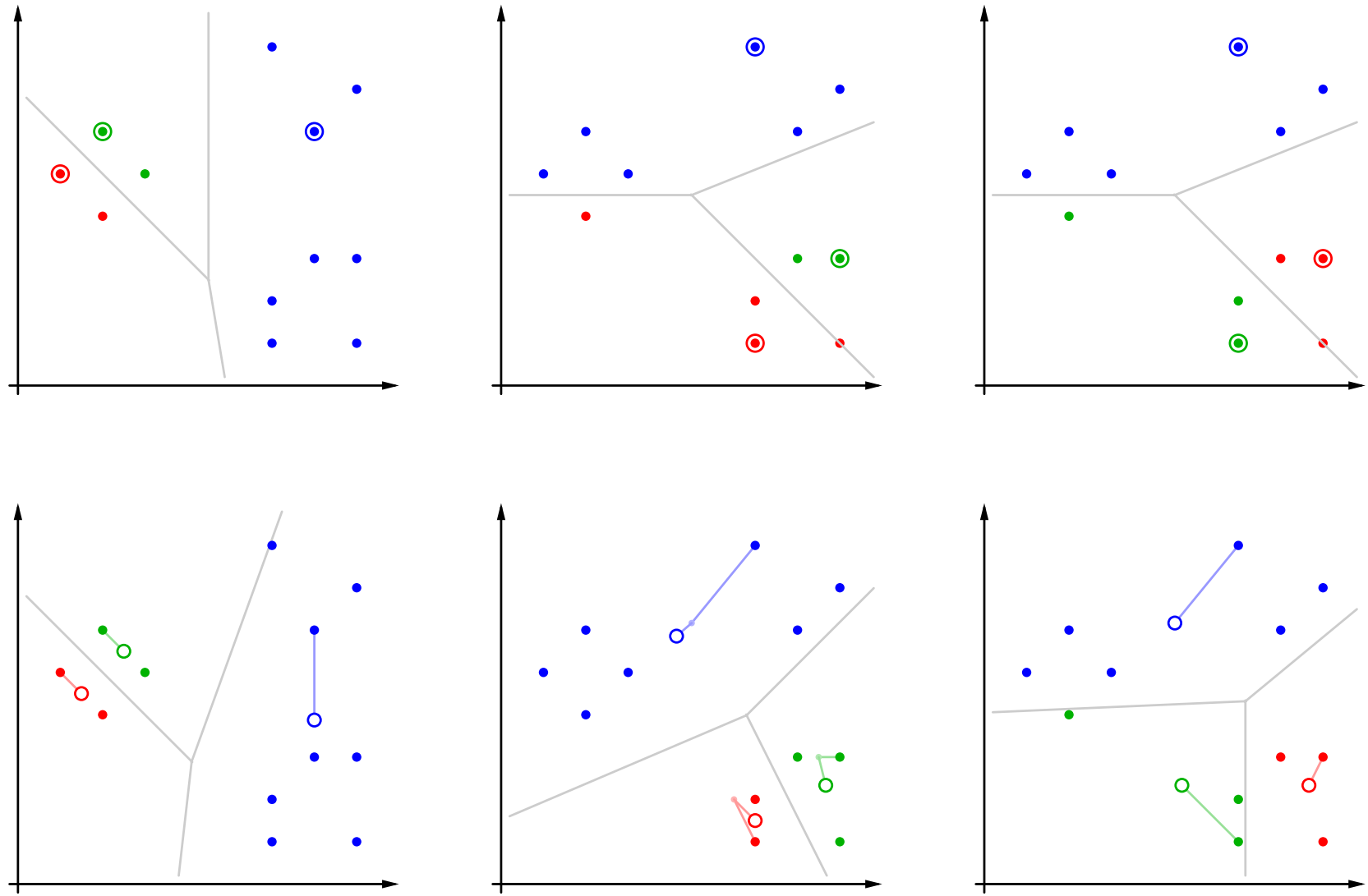
Nevertheless: Result is **sensitive to the initial positions** of cluster centers.

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

Fuzzy c-means clustering and the estimation of a mixture of Gaussians are much more robust (to be discussed later).

Research issue: How to determine the number of clusters automatically?
(Some approaches exists, but none of them is too successful.)

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 an inner product, *e.g.* 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 1 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 $\Gamma_i, 1 < c < n$.

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.

Hard c-means

The objective function of the hard c-means is

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

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:

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

Alternating Optimization Scheme

J_h depends on c, and U on the data points to the clusters.

Finding the parameters that minimize J_h is NP-hard.

Hard c-means minimizes J_h 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.

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:

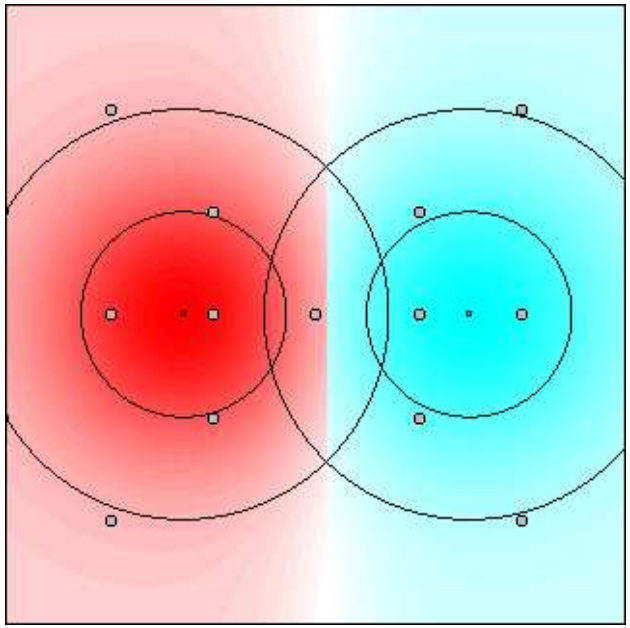
$$u_{ij} = \begin{cases} 1, & \text{if } i = \arg\min_{k=1}^{c} d_{kj} \\ 0, & \text{otherwise.} \end{cases}$$

(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.

Example



Given a symmetric dataset with two clusters.

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

Is that very intuitive?

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

Outline

Fuzzy Clustering

It allows gradual memberships of data points to clusters in $[0, 1]$.

It offers the flexibility to express whether a data point belongs to more than 1 cluster.

Thus, membership degrees

- offer a finer degree of detail of the data model,
- express how ambiguously/definitely $\boldsymbol{x}_j$ should belong to Γ_i.

The solution spaces equal fuzzy partitions of $X = \{x_1, \ldots, x_n\}$.

Fuzzy Clustering

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 2 types of fuzzy cluster partitions:

- **probabilistic and possibilistic**
- They differ in constraints they place on the membership degrees.

Probabilistic Cluster Partition

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 $U_f = (u_{ij}) = (\mu_{\Gamma_i}(\boldsymbol{x}_j))$ a *probabilistic cluster partition* of X if

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

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

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.

Probabilistic 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.

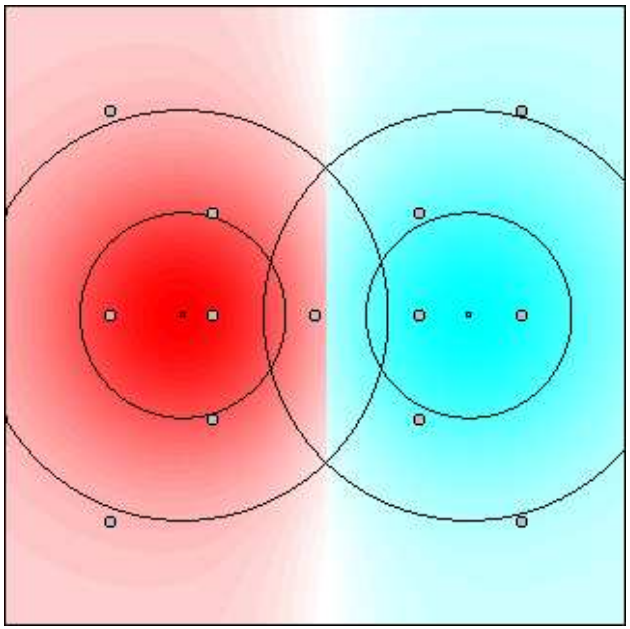
The 2nd condition says that sum of membership degrees must be 1 for each $\boldsymbol{x}_j$.

- 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.

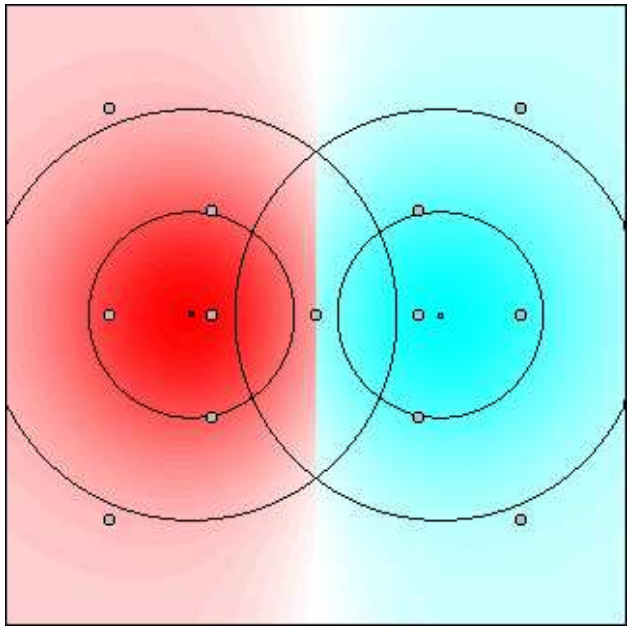
The consequence of both constraints are as follows:

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

Example



hard c-means



fuzzy c-means

There is no arbitrary assignment for the equidistant data point in middle anymore.

In the fuzzy partition it is associated with the membership vector $(0.5, 0.5)^T$ (which expresses the ambiguity of the assignment).

Objective Function

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 parameter $m \in \mathbb{R}$ with $m > 1$ is called the **fuzzifier** and $d_{ij} = d(\boldsymbol{c}_i, \boldsymbol{x}_j)$.

Fuzzifier

The actual value of m determines the "fuzziness" of the classification.

For $m = 1$ (*i.e.* $J_h = J_f$), the assignments remain hard.

Fuzzifiers of $m > 1$ lead to fuzzy memberships

Clusters become softer/harder with a higher/lower value of m.

Usually $m = 2$.

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:

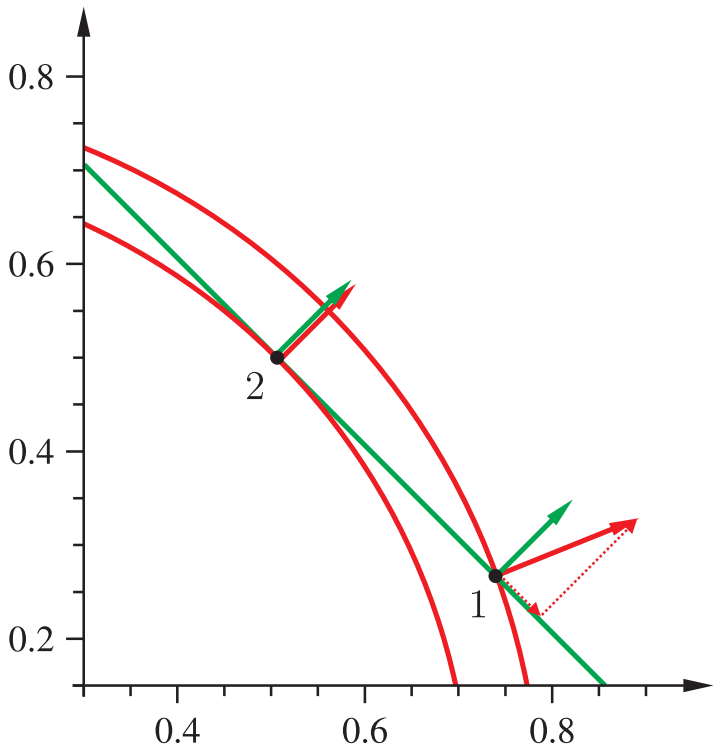
- Take the partial derivatives of f and set them to zero:

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

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

Example

Task: Minimize $f(x_1, x_2) = x_1^2 + x_2^2$ subject to $g : x_1 + x_2 = 1$.



Crossing a contour line: Point 1 cannot be a constrained minimum because ∇f has a non-zero component in the constrained space. Walking in opposite direction to this component can further decrease f.

Touching a contour line: Point 2 is a constrained minimum: both gradients are parallel, hence there is no component of ∇f in the constrained space that might lead us to a lower value of f.

Function Optimization: Lagrange Theory

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

Given: $f(\boldsymbol{x})$ to be optimized, k equality constraints $C_j(\boldsymbol{x}) = 0,\ 1 \leq j \leq k$

procedure:

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

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

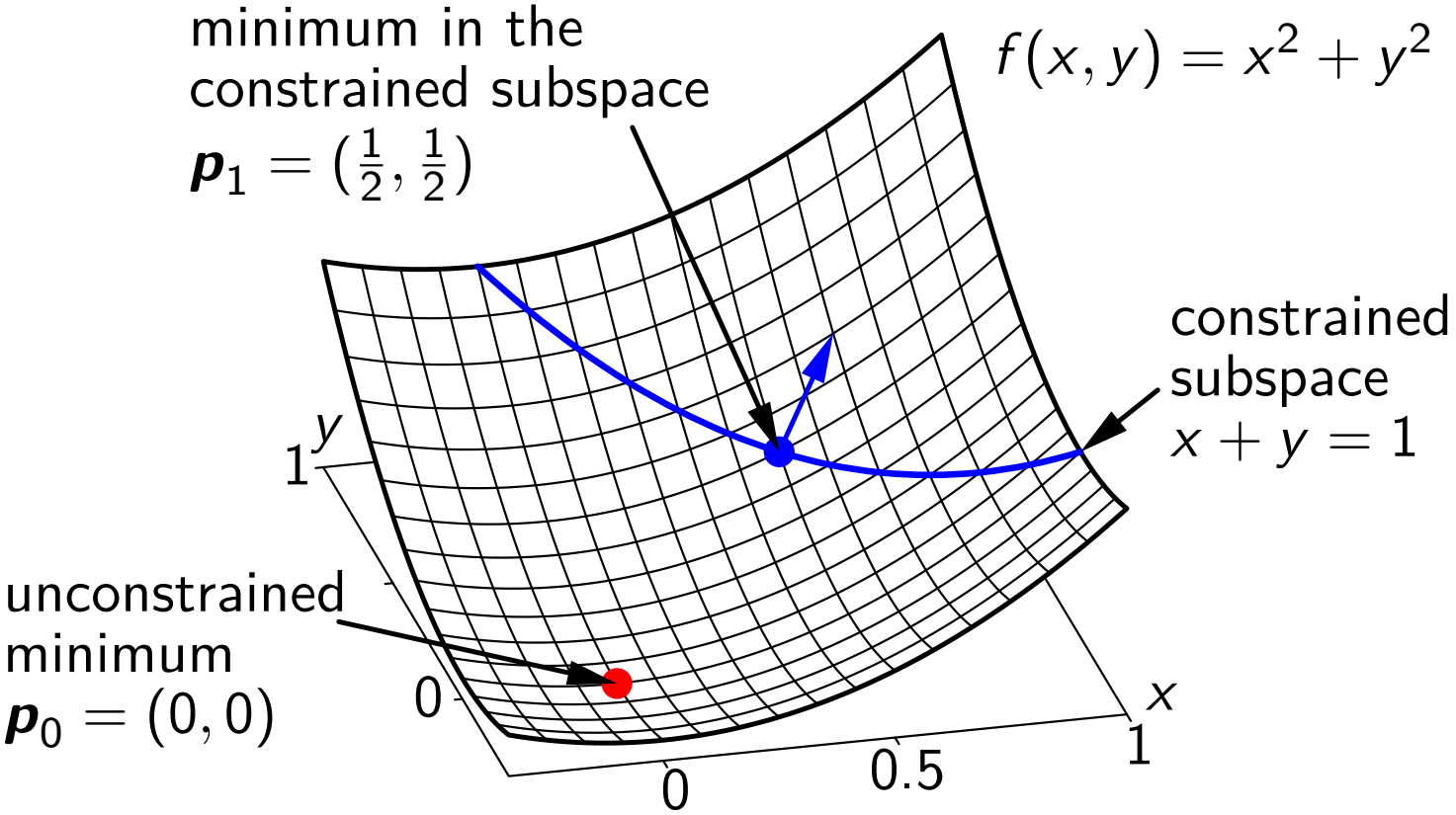
2. Set the partial derivatives of Lagrange function equal to zero:

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

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 $x + y = 1$.

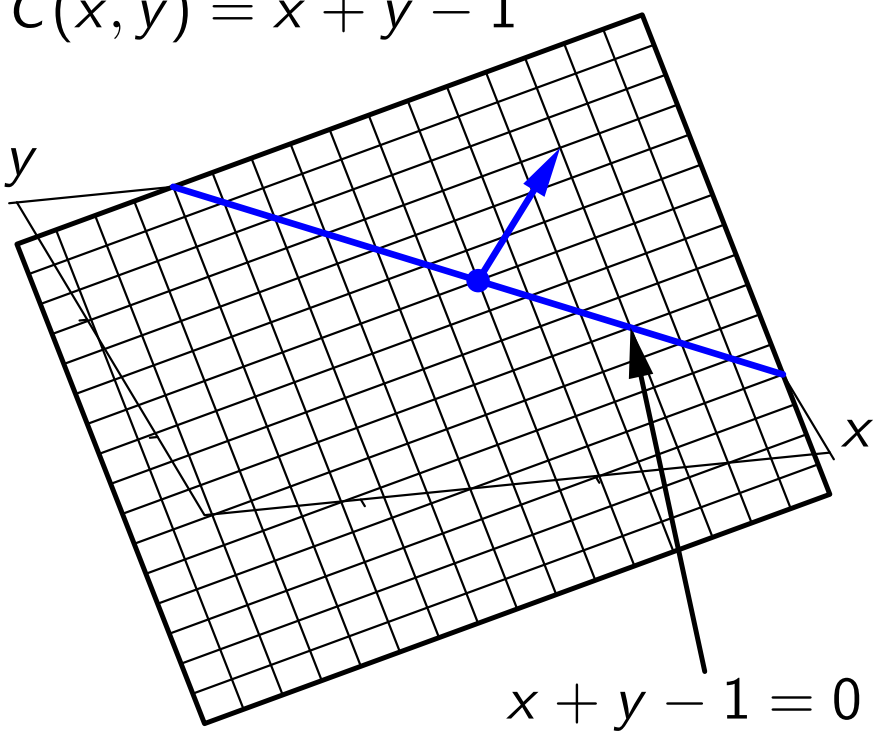


The unconstrained minimum is not in the constrained subspace. At the minimum in the constrained subspace the gradient does not vanish.

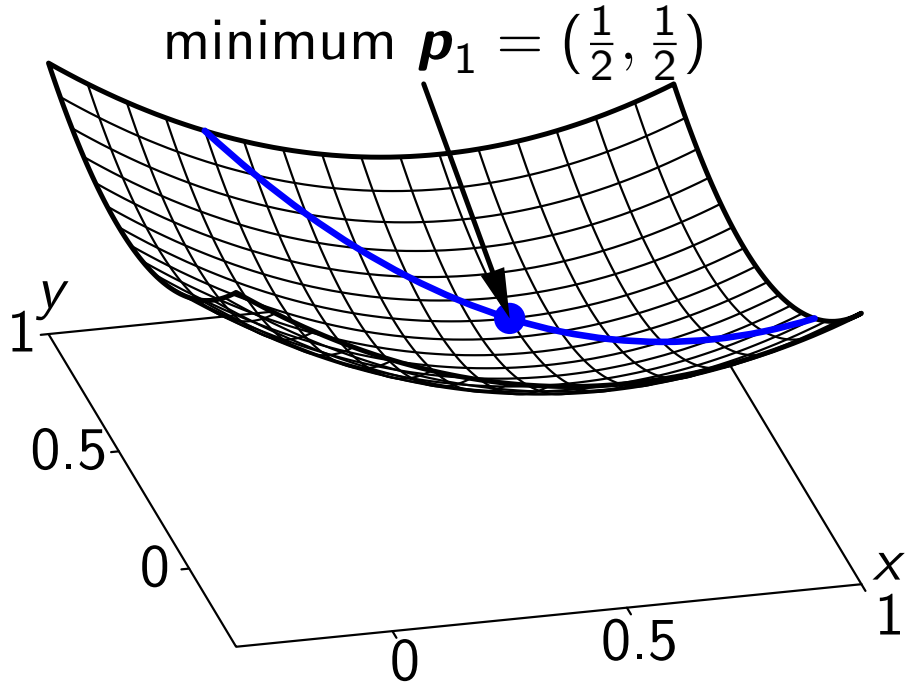
Lagrange Theory: Revisited Example 1

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

$C(x, y) = x + y - 1$



$L(x, y, -1) = x^2 + y^2 - (x + y - 1)$



The gradient of the constraint is perpendicular to the constrained subspace. The (unconstrained) minimum of the $L(x, y, \lambda)$ is the minimum of $f(x, y)$ in the constrained subspace.

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):

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

- 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).

Optimizing the Membership Degrees

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 [Bezdek, 1981]:

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

That is independent of any distance measure.

First Step: Fix the cluster parameters

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.

Optimizing the Cluster Prototypes

The update formula j_C depend on both

- cluster parameters (location, shape, size) and
- the distance measure.

Thus the general update formula cannot be given.

For the basic fuzzy c-means model,

- the cluster centers serve as prototypes, and
- the distance measure is an induced metric by the inner product.

Thus the **second step** (*i.e.* the derivations of J_f *w.r.t.* the centers) yields [Bezdek, 1981]

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

Discussion: Fuzzy c-means

It is initialized with randomly placed cluster centers.

The updating in AO scheme stops if

- the number of iterations exceeds some predefined limit
- or the changes in the prototypes $\leq$ some termination accuracy.

FCM is stable and robust.

Compared to hard c-means, it's

- quite insensitive to initialization and
- not likely to get stuck in local minimum.

FCM converges in a saddle point or minimum (but not in a maximum) [Bezdek, 1981].

Outline

Rule Generation by Fuzzy Clustering

apply fuzzy clustering to $\mathcal{X} \Rightarrow$ fuzzy partition matrix $U = [u_{ij}]$

use obtained $U = [u_{ij}]$ to define membership functions

usually $\mathcal{X}$ is multidimensional

How to specify meaningful labels for multidim. membership functions?

Extend u_{ij} to Continuous Membership Functions

assigning labels for one-dimensional domains is easier

- project U down to $X_1, \ldots, X_p$ axis, respectively
- only consider **upper envelope** of membership degrees
- **linear interpolate** membership values membership functions
- cylindrically extend membership functions

original clusters are interpreted as conjunction of cyl. extensions

- *e.g.* , cylindrical extensions "x_1 is low", "x_2 is high"
- multidimensional cluster label "x_1 is low and x_2 is high"

labeled clusters = **classes** characterized by labels

every cluster = one fuzzy rule

Convex Completion [Höppner et al., 1999]

problem of this approach: **non-convex** fuzzy sets

having upper envelope, compute **convex completion**

we denote $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n,\ \boldsymbol{p}_1 \leq \ldots, \boldsymbol{p}_k$ as ordered projections of $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ and $\mu_{i1}, \ldots, \mu_{in}$ as respective membership values

eliminate each point $(\boldsymbol{p}_t, \mu_{it}),\ t = 1, \ldots, n$, for which two limit indices $t_l, t_r = 1, \ldots, n,\ t_l < t < t_r$, exist such that

$$mu_{it} < \min\{mu_{it_l},\ mu_{it_r}\}$$

after that: apply linear interpolation of remaining points

Example: The Iris Data

© **Iris Species Database** `http://www.badbear.com/signa/`

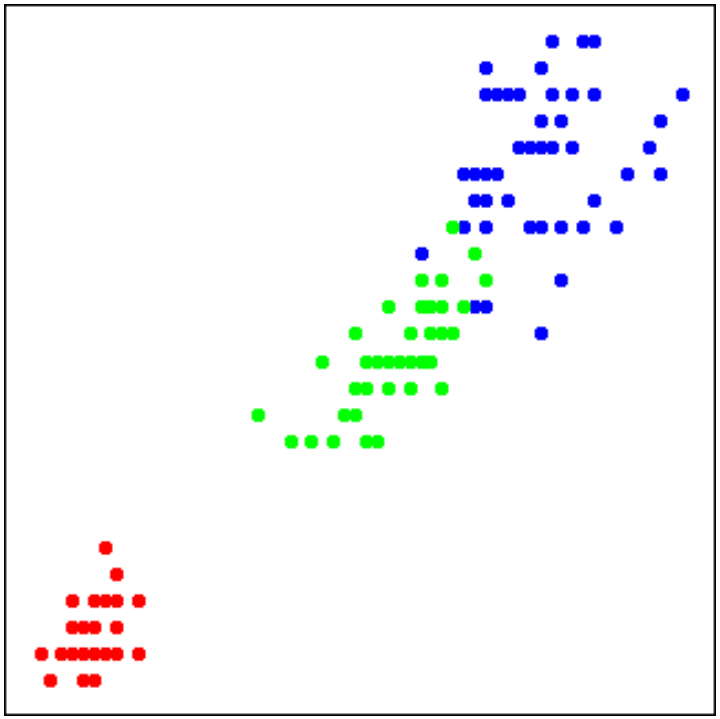
Iris setosa

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.

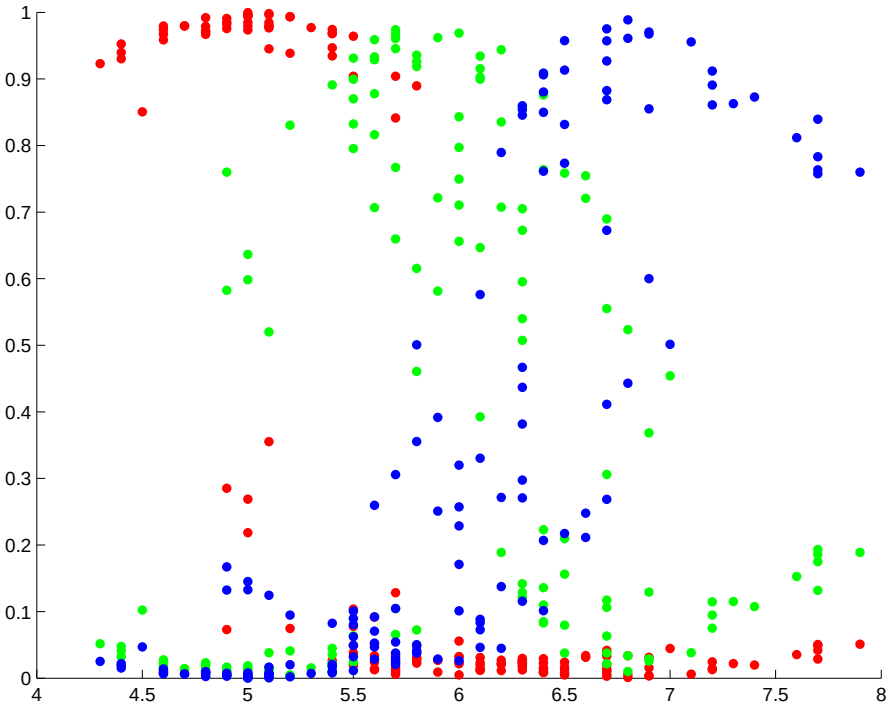
Example: The Iris Data



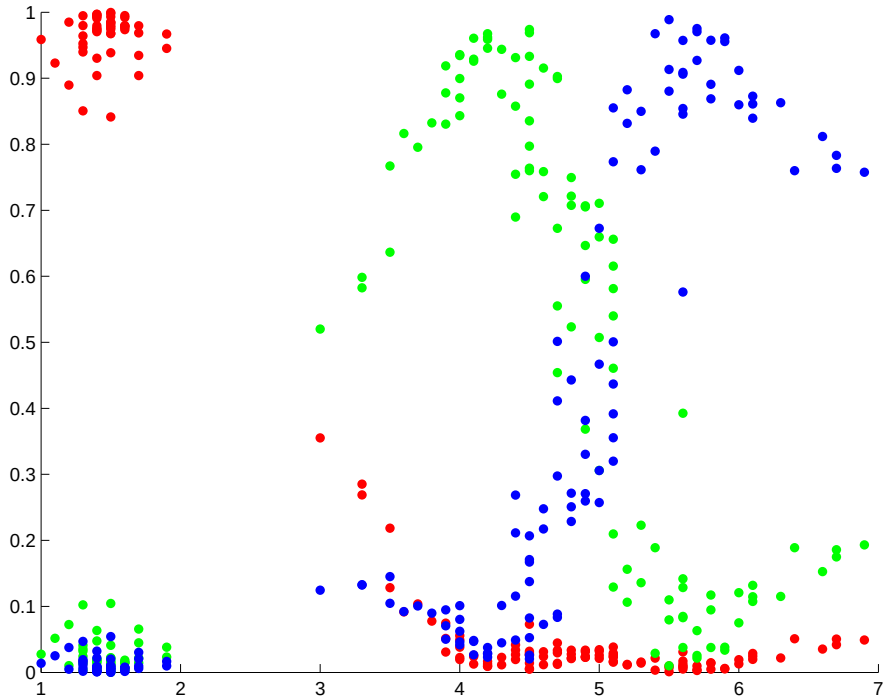
shown: sepal length and petal length

Iris setosa (red), Iris versicolor (green), Iris virginica (blue)

1. Membership Degrees from FCM



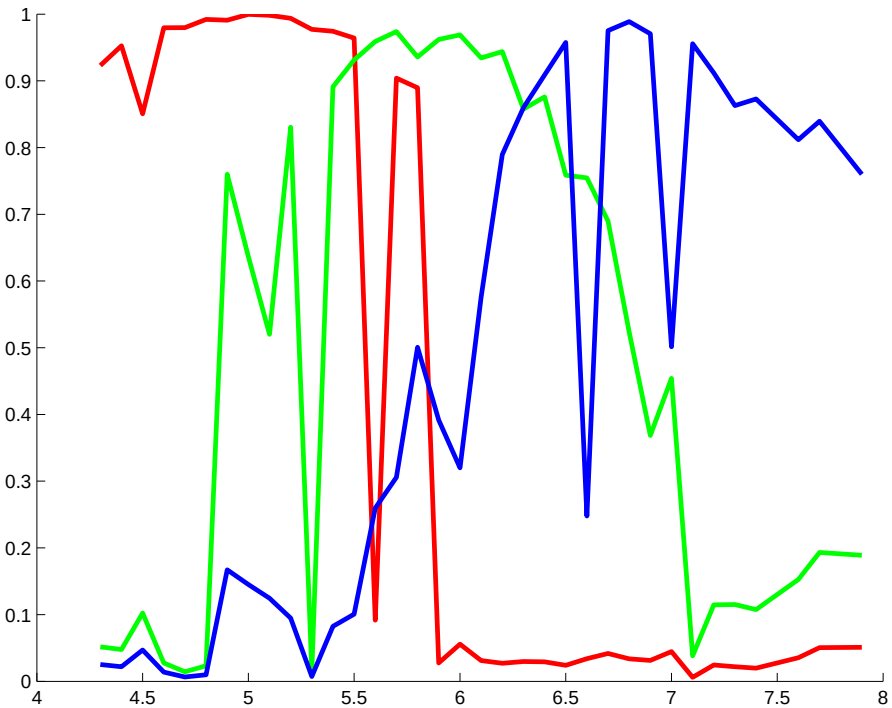
sepal length



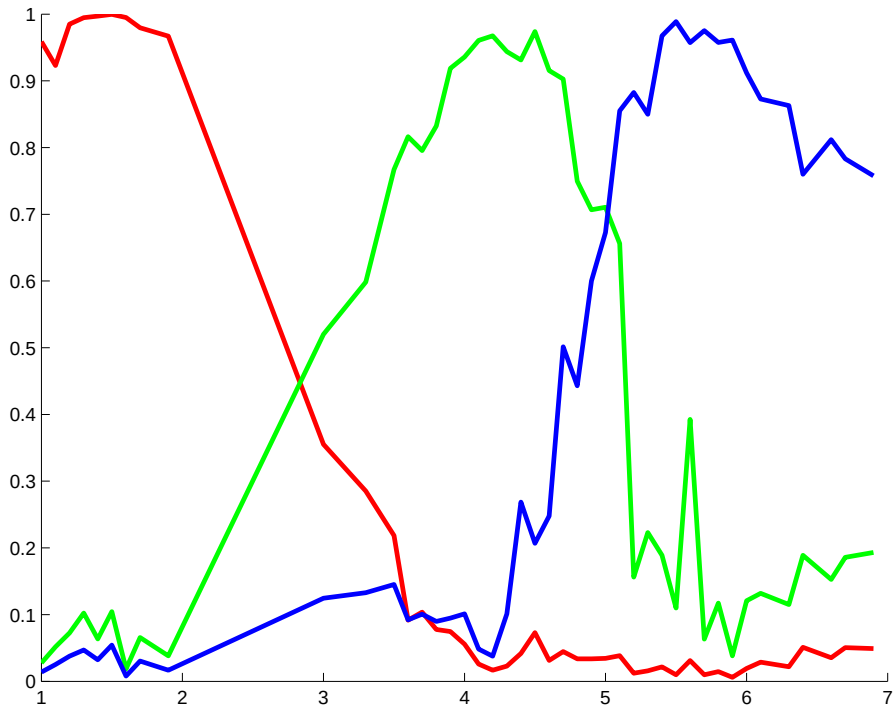
pedal length

raw, unmodified membership degrees

2. Upper Envelope



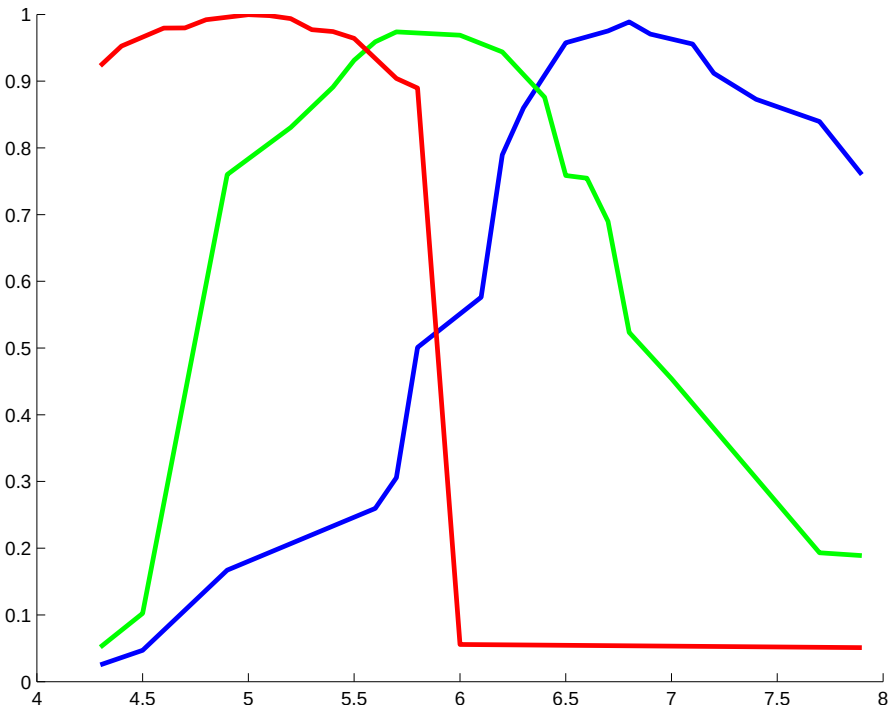
sepal length



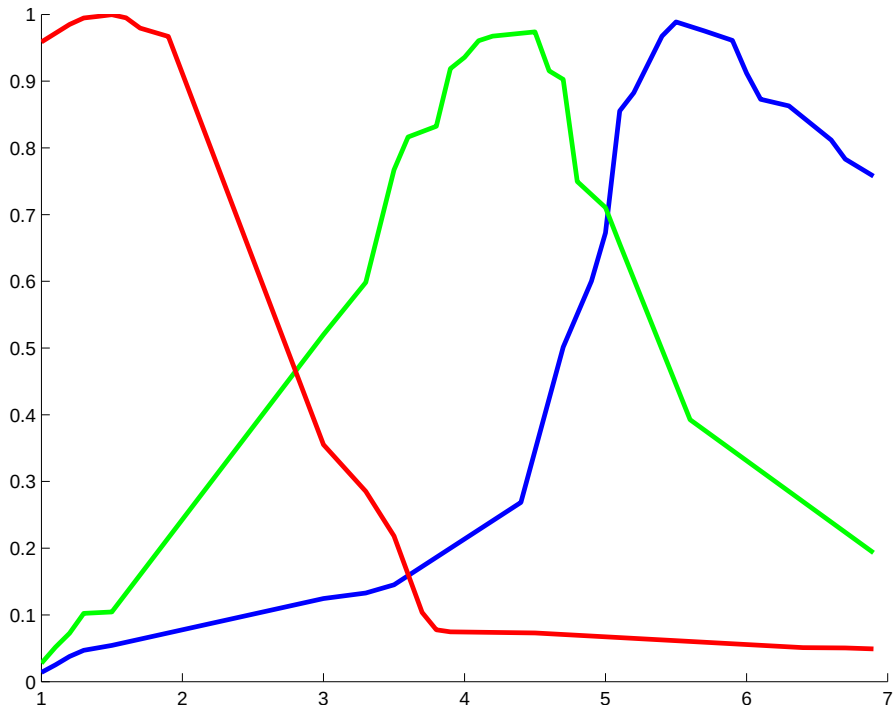
pedal length

for every attribute value and cluster center, only consider maximum membership degree

3. Convex Completion



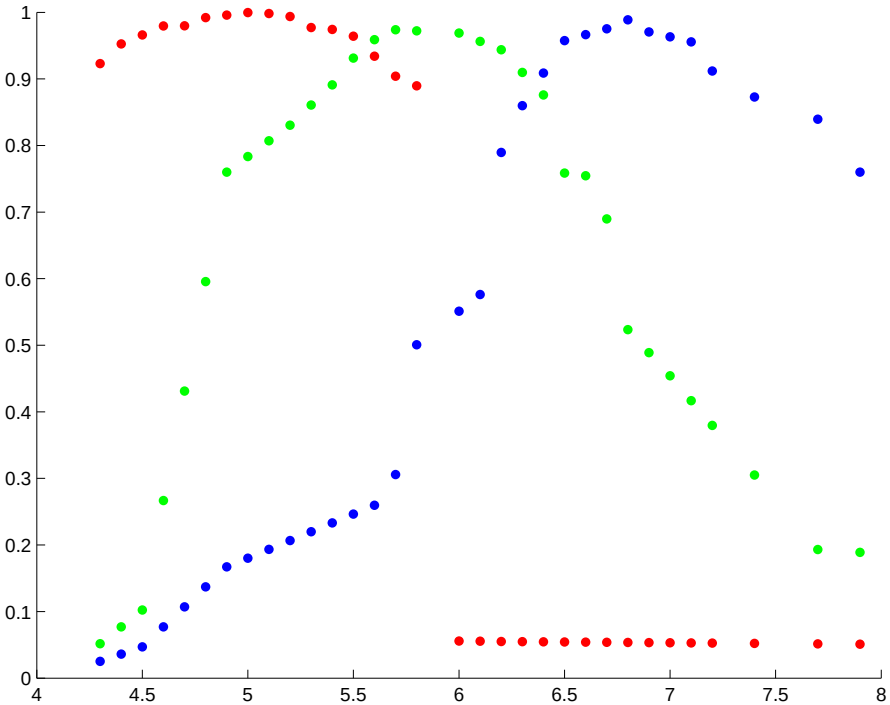
sepal length



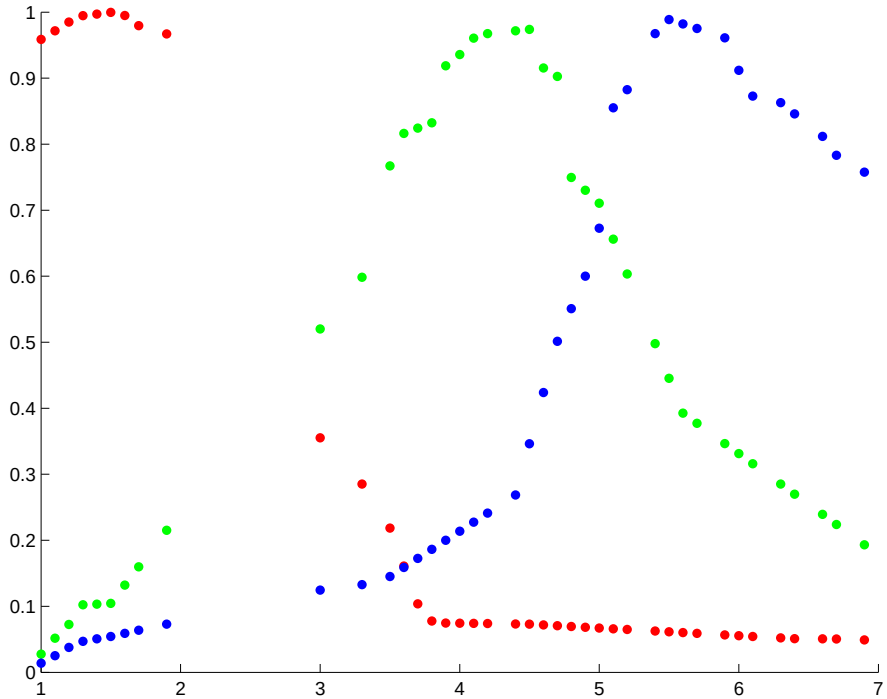
pedal length

convex completion removes spikes [Höppner et al., 1999]

4. Linear Interpolation



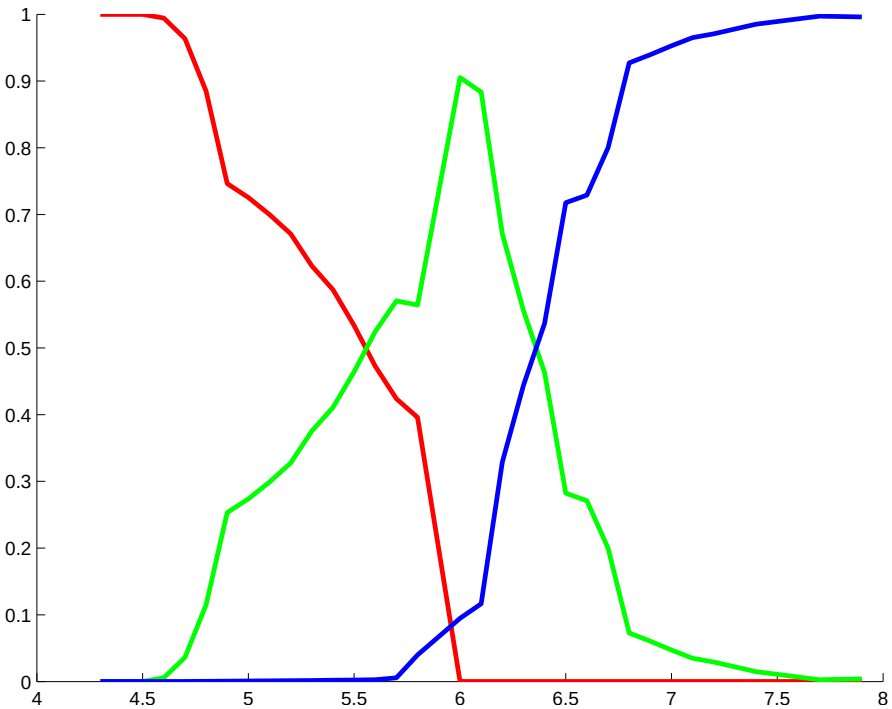
sepal length



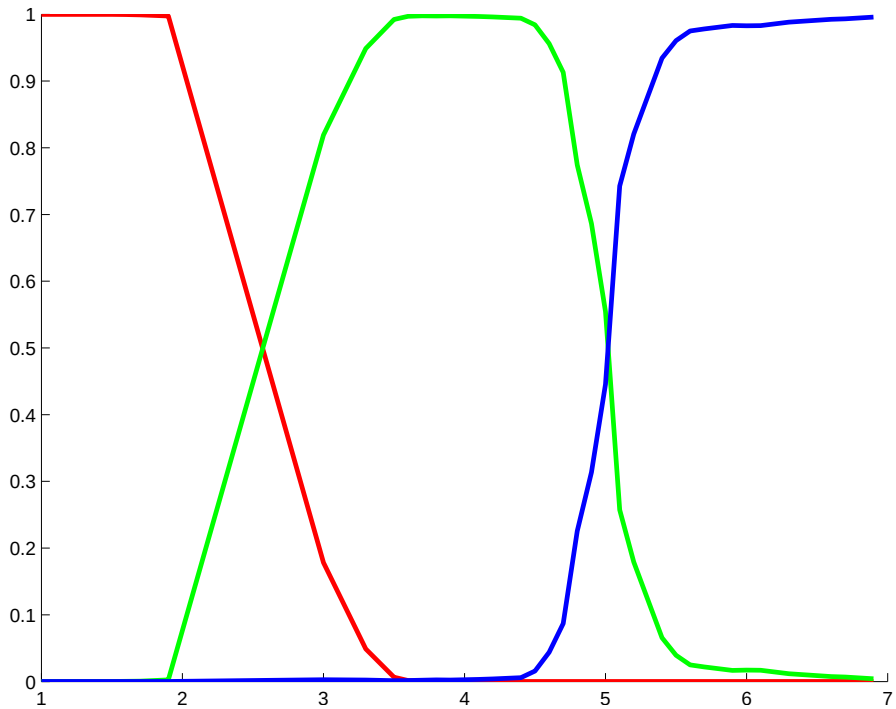
pedal length

interpolation for missing values (needed for normalization)

5. Stretched and Normalized Fuzzy Sets



sepal length

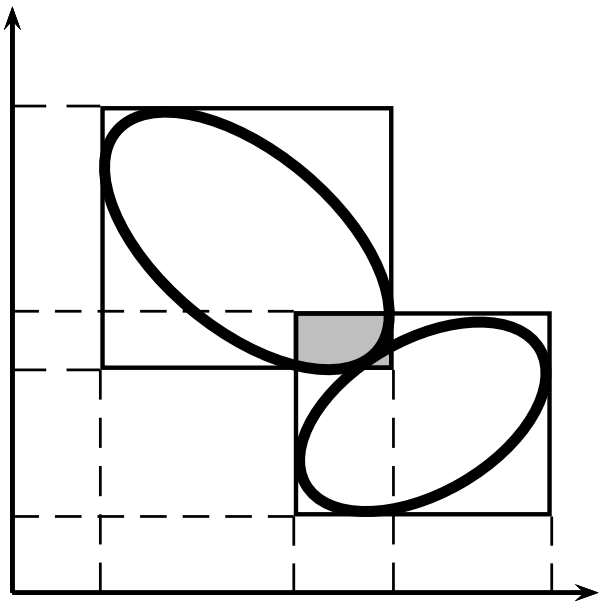


pedal length

every $\mu_i(x_j) \mapsto \mu_i(x_j)^5$ (extends core and support)

normalization has been performed finally

Information Loss from Projection



rule derived from fuzzy cluster represents approximation of cluster

information gets lost by projection

- cluster shape of FCM is spherical
- cluster projection leads to hypercube
- hypercube contains hypersphere

loss of information can be kept small using axes-parallel clusters

References I


Bezdek, J. (1981).
Pattern Recognition With Fuzzy Objective Function Algorithms.
Plenum Press, New York, NY, USA.


Couso, I., Dubois, D., and Sánchez, L. (2014).
Random sets and random fuzzy sets as ill-perceived random variables.


Höppner, F., Klawonn, F., Kruse, R., and Runkler, T. (1999).
Fuzzy Cluster Analysis: Methods for Classification, Data Analysis and Image Recognition.
John Wiley & Sons Ltd, New York, NY, USA.