Fuzzy and Probabilistic Clustering

Christian Borgelt

Intelligent Data Analysis and Graphical Models Research Unit
European Center for Soft Computing
c/ Gonzalo Gutiérrez Quirós s/n, 33600 Mieres, Spain

christian.borgelt@softcomputing.es
http://www.softcomputing.es/
http://www.borgelt.net/

Overview

- **General Idea of Clustering**
  - Group data points by similarity or distance
  - Similarity and distance measures
- **Classical c-Means Clustering**
  - Crisp assignment of data points to clusters
  - Alternating adaptation
- **Fuzzy c-Means Clustering**
  - Allow degrees of membership to a cluster
- **Expectation Maximization for Gaussian Mixtures**
  - Probabilistic data model and its estimation
- **Summary**

General Idea of Clustering

- Goal: Arrange the given data tuples into **classes** or **clusters**.
- Data tuples assigned to the same cluster should be as similar as possible.
- Data tuples assigned to different clusters should be as dissimilar as possible.
- Similarity is most often measured with the help of a distance function.
  (The smaller the distance, the more similar the data tuples.)
- Often: restriction to data points in $\mathbb{R}^m$ (although this is not mandatory).

$d : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}_0^+$ is a **distance function** if it satisfies $\forall \vec{x}, \vec{y}, \vec{z} \in \mathbb{R}^m$:

$$
\begin{array}{rll}
(i) & d(\vec{x}, \vec{y}) = 0 \quad \Leftrightarrow \quad \vec{x} = \vec{y}, & \\
(ii) & d(\vec{x}, \vec{y}) = d(\vec{y}, \vec{x}) & \text{(symmetry)}, \\
(iii) & d(\vec{x}, \vec{z}) \leq d(\vec{x}, \vec{y}) + d(\vec{y}, \vec{z}) & \text{(triangle inequality)}.
\end{array}
$$

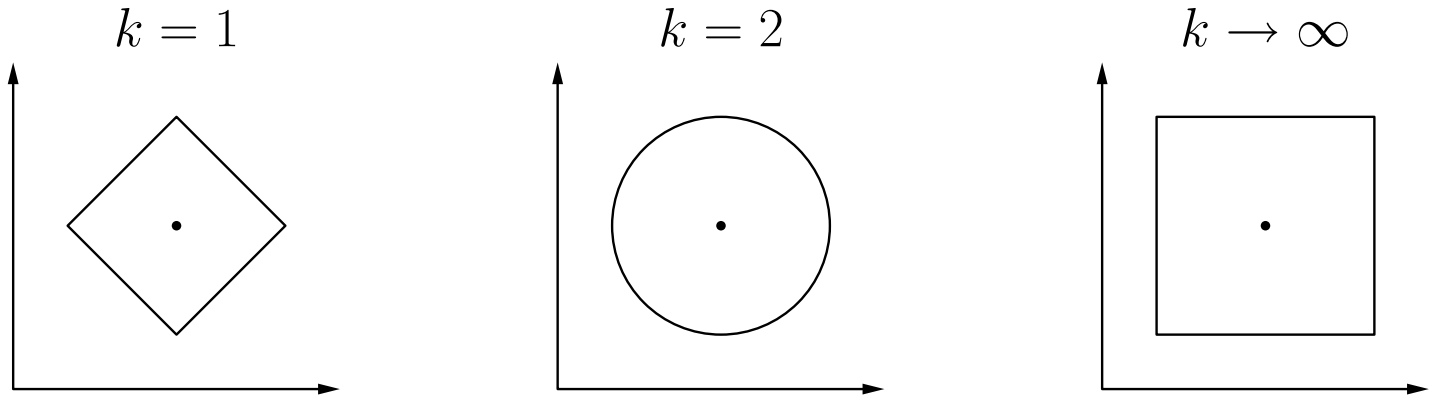
Distance Functions

Illustration of distance functions: Minkowski Family

$$d_k(\vec{x}, \vec{y}) = \left(\sum_{i=1}^{n} (x_i - y_i)^k \right)^{\frac{1}{k}}$$

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. $d_\infty(\vec{x}, \vec{y}) = \max_{i=1}^{n} \lvert x_i - y_i \rvert$.

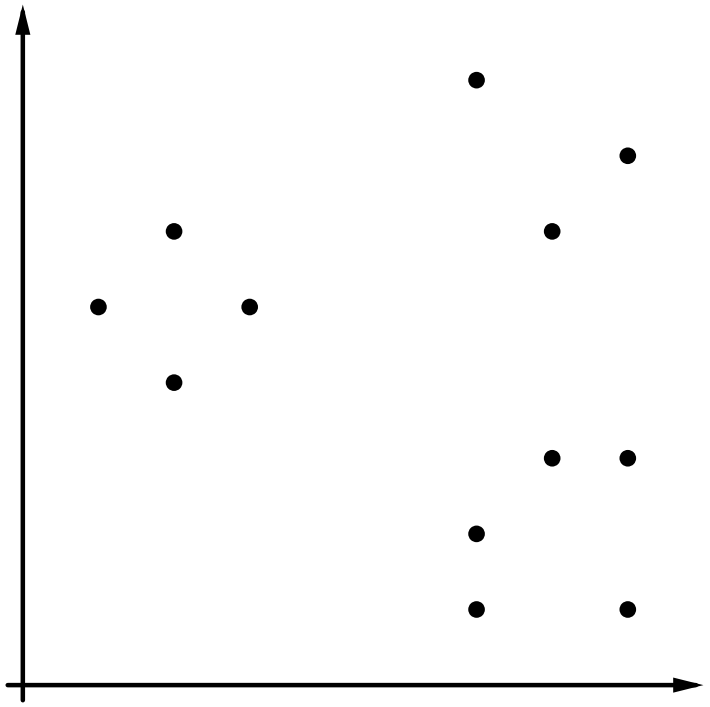


c-Means Clustering

- Choose a number c of clusters to be found (user input).
- Initialize the cluster centers randomly
  (for instance, by randomly selecting c data points).
- **Data point assignment**:
  Assign each data point to the cluster center that is closest to it
  (i.e. closer than any other cluster center).
- **Cluster center update**:
  Compute new cluster centers as the mean vectors of the assigned data points.
  (Intuitively: center of gravity if each data point has unit weight.)
- Repeat these two steps (data point assignment and cluster center update)
  until the clusters centers do not change anymore.

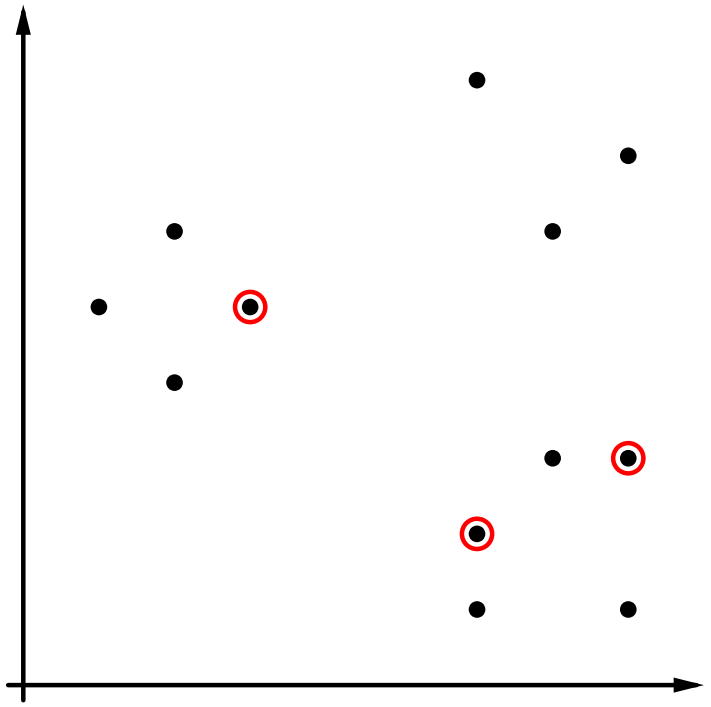
- It can be shown that this scheme must converge,
  i.e., the update of the cluster centers cannot go on forever.

c-Means Clustering: Example



Data set to cluster.

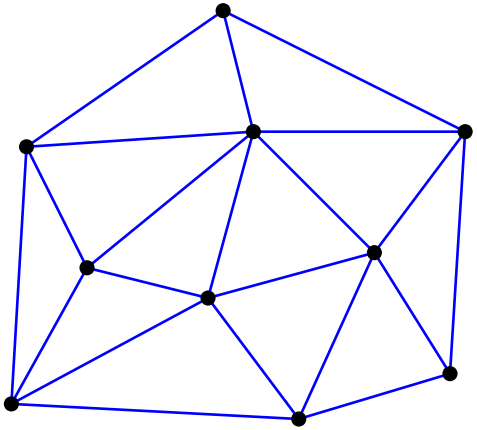
Choose $c = 3$ clusters.
(From visual inspection, can be difficult to determine in general.)

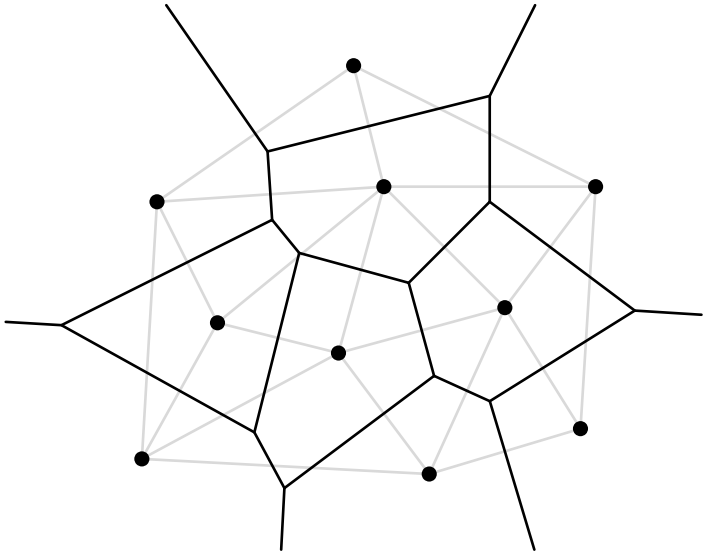


Initial position of cluster centers.

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

Delaunay Triangulations and Voronoi Diagrams

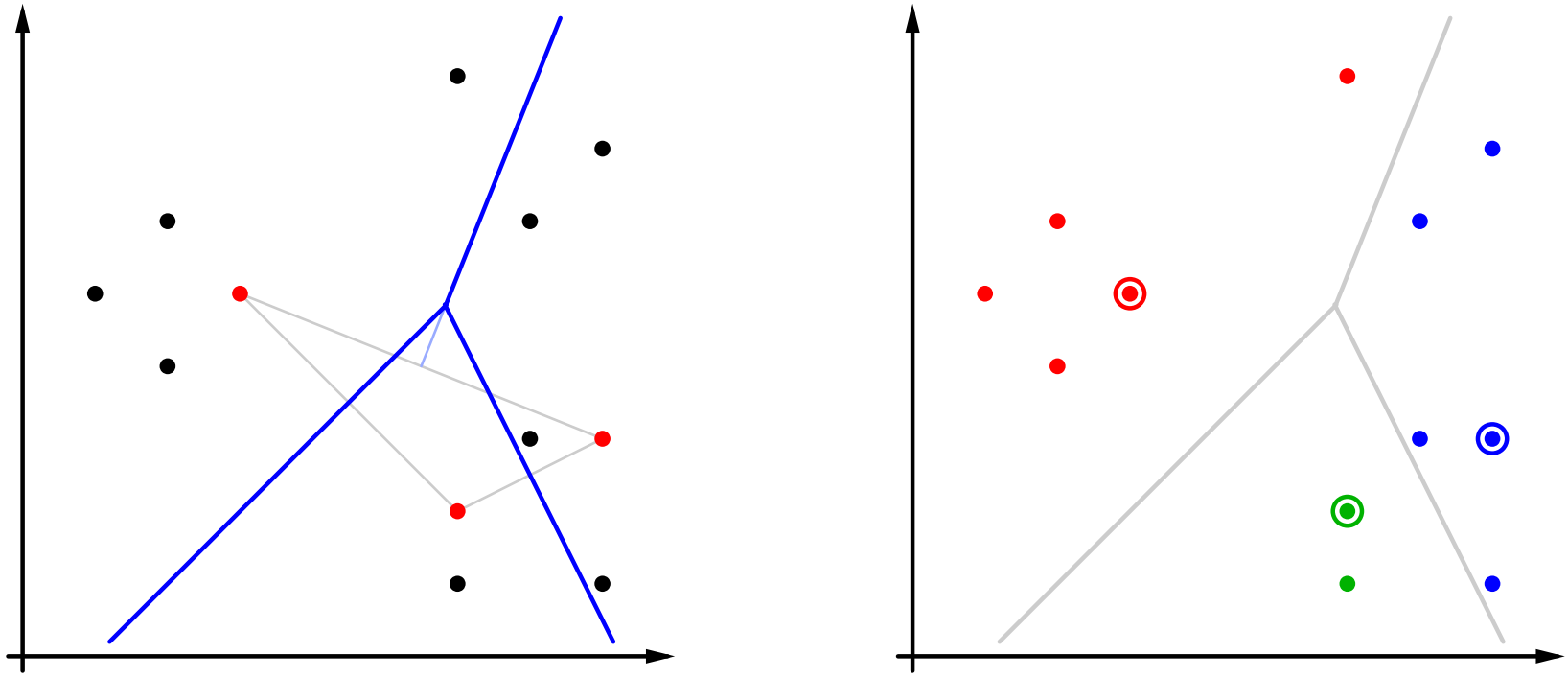




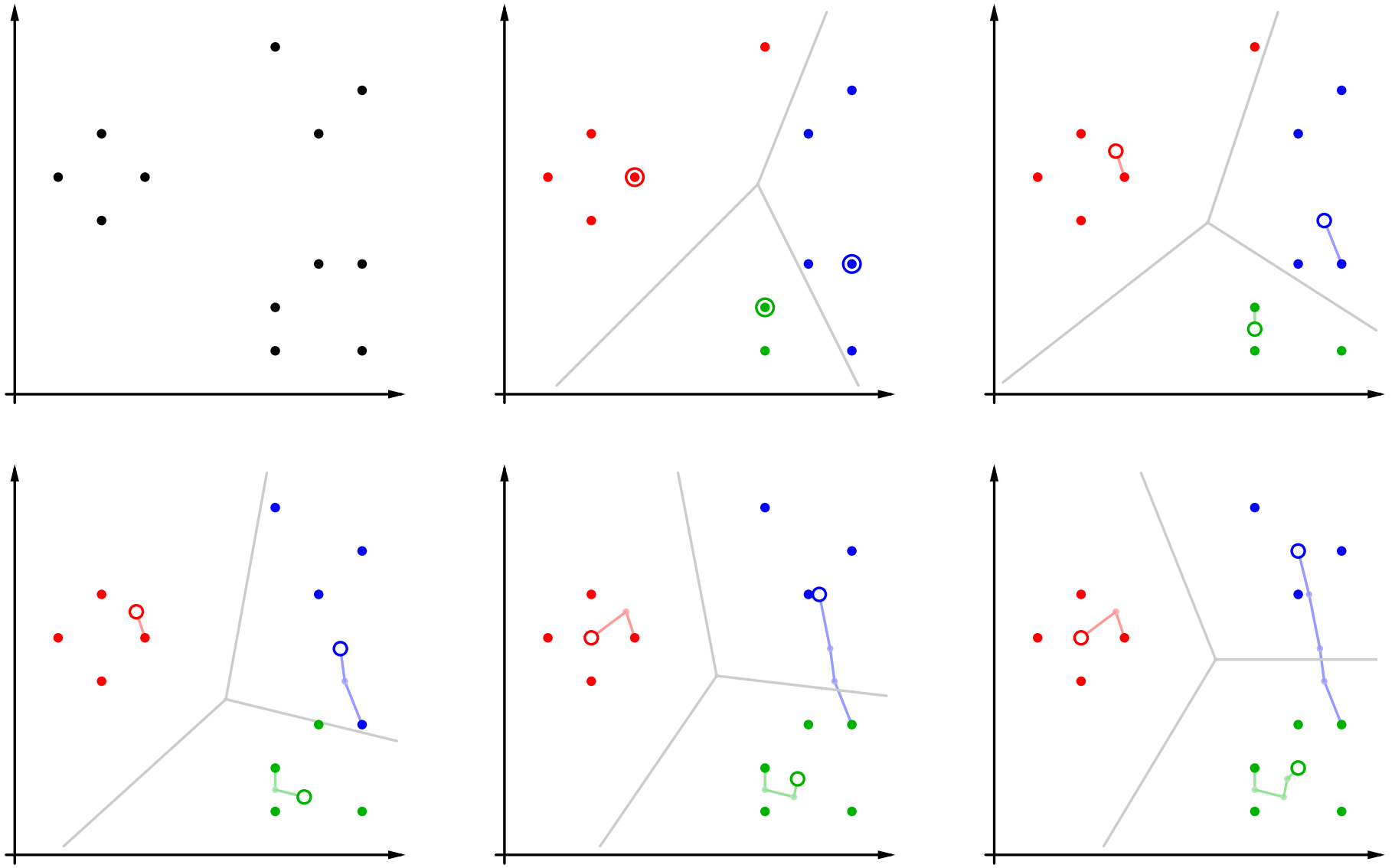
- Dots represent cluster centers (quantization vectors).
- Left: **Delaunay Triangulation**
  (The circle through the corners of a triangle does not contain another point.)
- Right: **Voronoi Diagram**
  (Midperpendiculars of the Delaunay triangulation: boundaries of the regions of points that are closest 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)



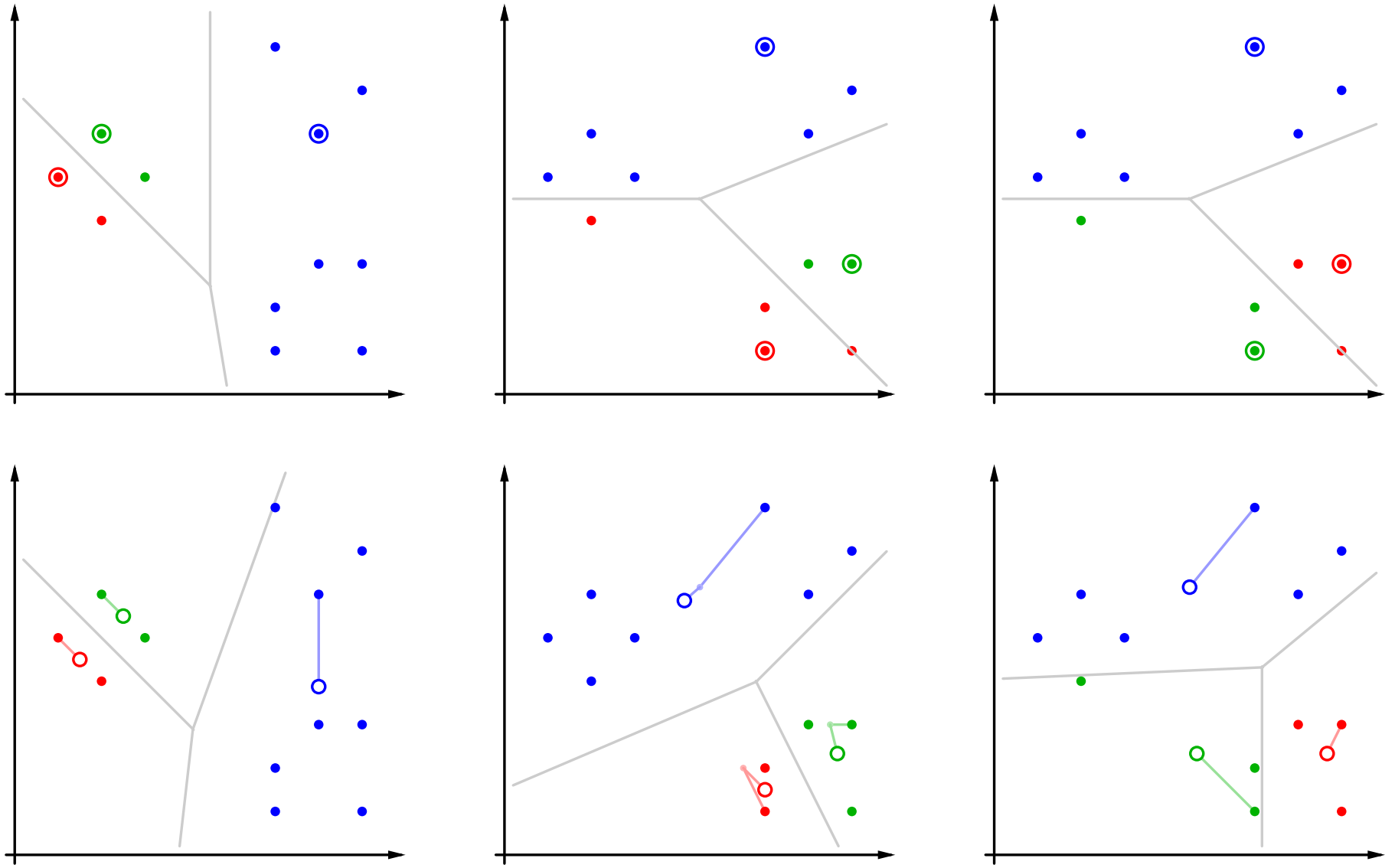
c-Means Clustering: Example



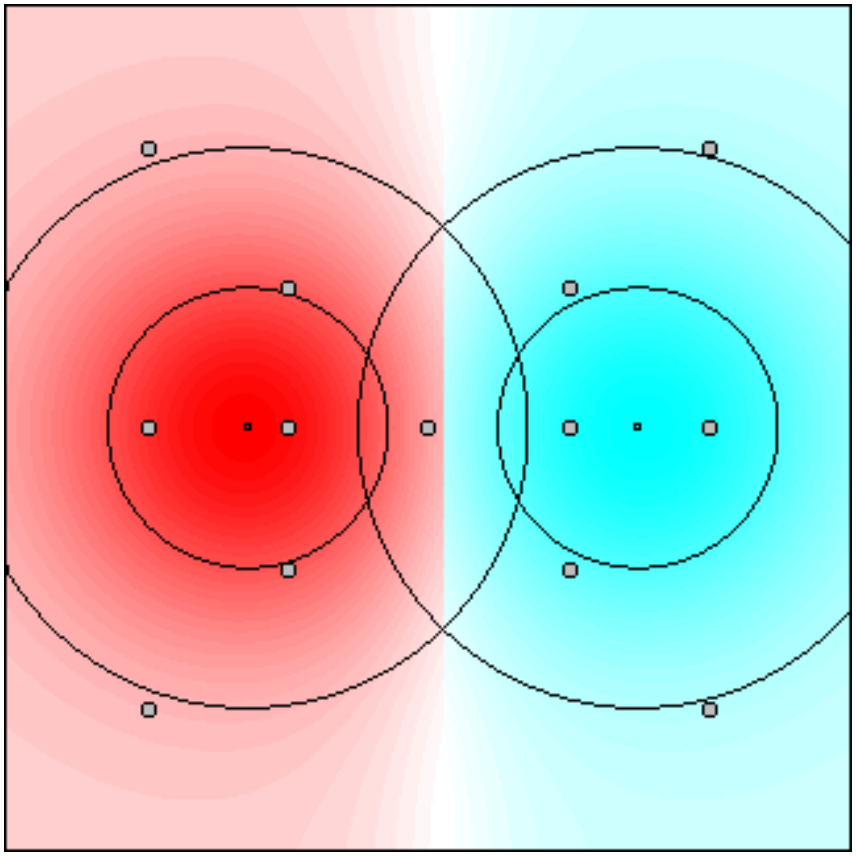
c-Means Clustering: Local Minima

- Clustering is successful in this example:
  The clusters found are those that would have been formed intuitively.
- Convergence is achieved after only 5 steps.
  (This is typical: convergence is usually very fast.)
- However: The clustering result is fairly **sensitive to the initial positions** of the 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: Can we determine the number of clusters automatically?
  (Some approaches exists, but none of them is too successful.)

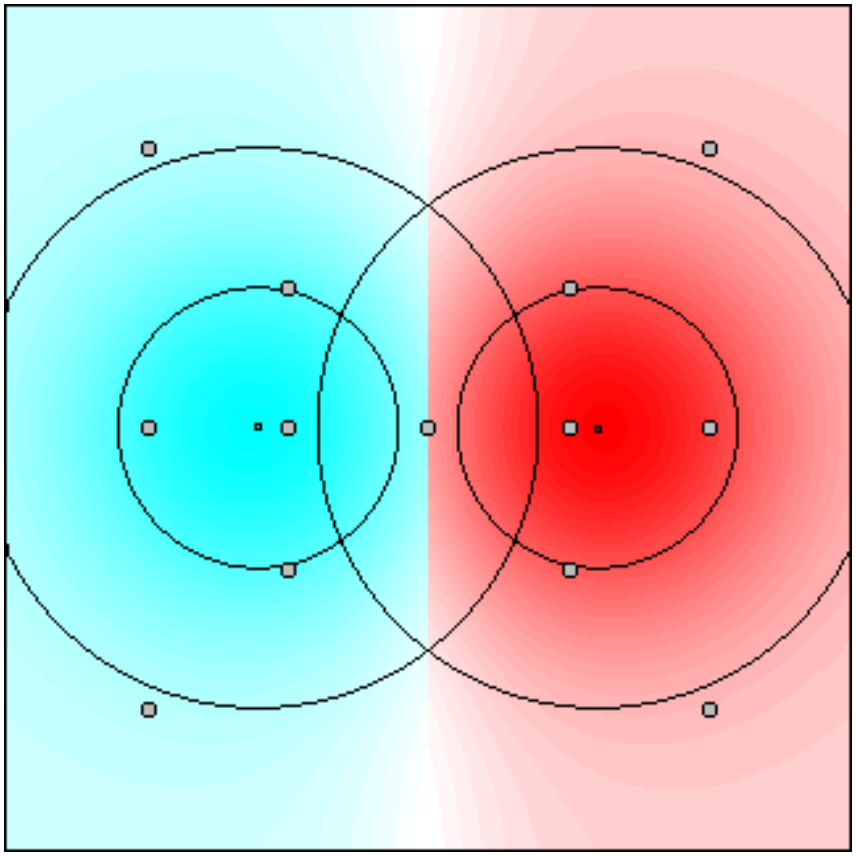
c-Means Clustering: Local Minima



Crisp Clustering: Overlapping Clusters



c-means clustering result



desired clustering result

Solution: allow degrees of membership of a datum to different clusters.

Fuzzy Clustering

- Allow degrees of membership of a datum to different clusters.
  (Classical c-means clustering assigns data crisply.)

Objective Function: (to be minimized)

$$J(\mathbf{X}, \mathbf{B}, \mathbf{U}) = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^{w} d^2(\beta_i, \vec{x}_j)$$

- $\mathbf{U} = [u_{ij}]$ is the $c \times n$ fuzzy partition matrix,

  $u_{ij} \in [0, 1]$ is the membership degree of the data point $\vec{x}_j$ to the i-th cluster.
- $\mathbf{B} = \{\beta_1, \ldots, \beta_c\}$ is the set of cluster prototypes.
- w is the so-called "fuzzifier" (the higher w, the softer the cluster boundaries).
- Constraints:

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

Fuzzy and Hard Clustering

Relation to Classical c-Means Clustering:

- c-means clustering can be seen as optimizing the objective function

$$J(\mathbf{X}, \mathbf{B}, \mathbf{U}) = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}\, d^2(\beta_i, \vec{x}_j),$$

  where $\forall i, j : u_{ij} \in \{0, 1\}$ (i.e. hard assignment of the data points) and the cluster prototypes β_i consist only of cluster centers.

- To obtain a fuzzy assignment of the data points, it is not enough to extend the range of values for the u_{ij} to the unit interval $[0, 1]$: The objective function J is optimized for a hard assignment (each data point is assigned to the closest cluster center).

- **Necessary for degrees of membership:**
  Apply a convex function $h : [0, 1] \to [0, 1]$ to the membership degrees u_{ij}.
  Most common choice: $h(u) = u^w$, usually with $w = 2$.

Reminder: Function Optimization

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

Often feasible approach:

- A necessary condition for a (local) optimum (maximum or minimum) is that the partial derivatives w.r.t. the parameters vanish (Pierre Fermat).
- Therefore: (Try to) solve the equation system that results 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:

1. Take the partial derivatives of the objective function and set them to zero:
$$\frac{\partial f}{\partial x} = 2x + y - 4 = 0, \qquad \frac{\partial f}{\partial y} = 2y + x - 5 = 0.$$
2. Solve the resulting (here: linear) equation system: $x = 1, \quad y = 2.$

Optimizing the Objective Function

Objective Function: (to be minimized)

$$J(\mathbf{X}, \mathbf{B}, \mathbf{U}) = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^{w} d^2(\beta_i, \vec{x}_j)$$

- For odd w there is no proper solution without the constraints:
  simply choose $\forall i, j : u_{ij} = -\infty$ (clearly not useful).
- If the u_{ij} are constrained to the interval $[0, 1]$,
  the optimal solution is clearly $\forall i, j : u_{ij} = 0$ (still not useful).
- Only with constraints a useful solution is obtained:

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

- For $w = 1$ the optimal solution has $\forall i, j : u_{ij} \in \{0, 1\}$
  (no advantage over crisp clustering).

Function Optimization with Constraints

Often a function has to be optimized subject to certain **constraints**.

Here: restriction to k **equality constraints** $C_i(\vec{x}) = 0$, $i = 1, \ldots, k$.

Note: the equality constraints describe a subspace of the domain of the function.

Problem of optimization with constraints:

- The gradient of the objective function f may vanish outside the constrained subspace, leading to an unacceptable solution (violating the constraints).
- At an optimum *in the constrained subspace* the derivatives need not vanish.

One way to handle this problem are **generalized coordinates**:

- Exploit the dependence between the parameters specified in the constraints to express some parameters in terms of the others and thus reduce the set $\vec{x}$ to a set $\vec{x}'$ of independent parameters (*generalized coordinates*).
- Problem: Can be clumsy and cumbersome, if possible at all, because the form of the constraints may not allow for expressing some parameters as proper functions of the others.

Function Optimization with Constraints

A much more elegant approach is based on the following nice insights:
Let $\vec{x}^*$ be a (local) optimum of $f(\vec{x})$ *in the constrained subspace*. Then:

- The gradient $\nabla_{\vec{x}} f(\vec{x}^*)$, if it does not vanish, must be **perpendicular** to the constrained subspace. (If $\nabla_{\vec{x}} f(\vec{x}^*)$ had a component in the constrained subspace, $\vec{x}^*$ would not be a (local) optimum in this subspace.)
- The gradients $\nabla_{\vec{x}} C_j(\vec{x}^*)$, $1 \leq j \leq k$, must all be **perpendicular** to the constrained subspace, because they are constant, namely 0, in this subspace. Together they span the subspace perpendicular to the constrained subspace.
- Therefore it must be possible to find values λ_j, $1 \leq j \leq k$, such that

$$\nabla_{\vec{x}} f(\vec{x}^*) + \sum_{j=1}^{s} \lambda_j \nabla_{\vec{x}} C_j(\vec{x}^*) = 0.$$

  If the constraints (and thus their gradients) are linearly independent, the values λ_j are uniquely determined. This equation can be used to **compensate the gradient** of $f(\vec{x}^*)$ so that it vanishes at $\vec{x}^*$.

Function Optimization: Lagrange Theory

As a consequence of these insights we obtain the

Method of Lagrange Multipliers:

Given:
- a function $f(\vec{x})$, which is to be optimized,
- k equality constraints $C_j(\vec{x}) = 0$, $1 \leq j \leq k$.

Procedure:

1. Construct the so-called **Lagrange function** by incorporating the equality constraints C_i, $i = 1, \ldots, k$, with (unknown) **Lagrange multipliers** λ_i:
$$L(\vec{x}, \lambda_1, \ldots, \lambda_k) = f(\vec{x}) + \sum_{i=1}^{k} \lambda_i C_i(\vec{x}).$$
2. Set the partial derivatives of the 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.

Function Optimization: Lagrange Theory

Observations:

- Due to the representation of the gradient of $f(\vec{x})$ at a local optimum $\vec{x}^*$ in the constrained subspace (see above) the gradient of L w.r.t. $\vec{x}$ vanishes at $\vec{x}^*$.

  $\rightarrow$ The standard approach works again!

- If the constraints are satisfied, the additional terms have no influence.

  $\rightarrow$ The original task is not modified (same objective function).

- Taking the partial derivative w.r.t. a Lagrange multiplier reproduces the corresponding equality constraint:

$$\forall j; 1 \leq j \leq k: \qquad \frac{\partial}{\partial \lambda_j} L(\vec{x}, \lambda_1, \ldots, \lambda_k) = C_j(\vec{x}),$$

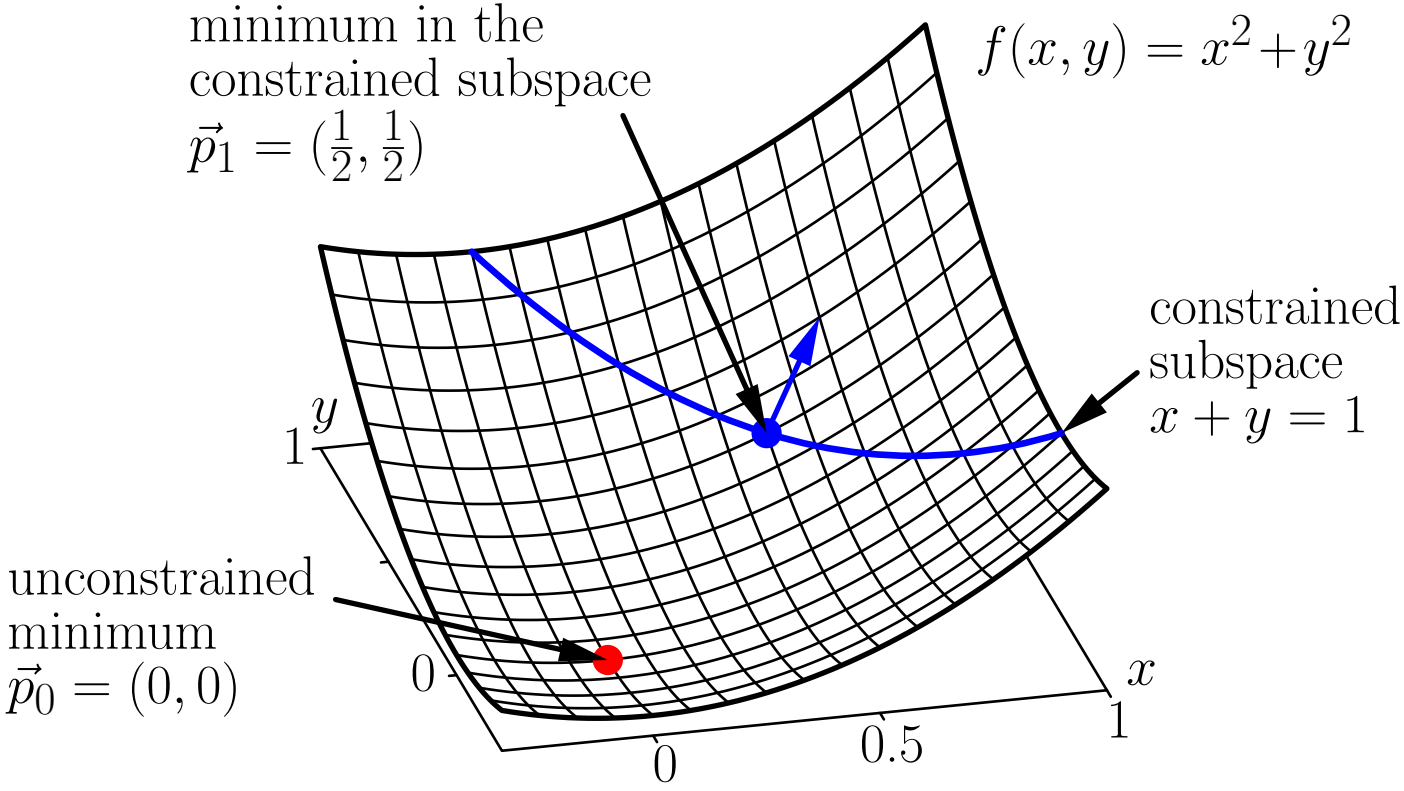
  $\rightarrow$ Constraints enter the equation system to solve in a natural way.

Remark:

- **Inequality** constraints can be handled with the **Kuhn–Tucker theory**.

Lagrange Theory: 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, and at the minimum in the constrained subspace the gradient does not vanish.

Lagrange Theory: Example 1

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

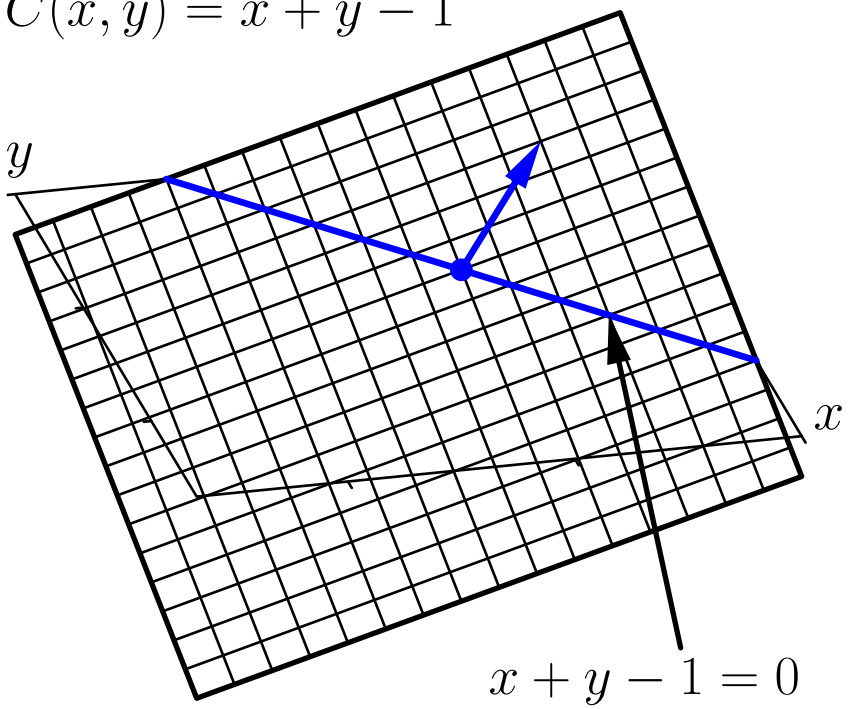
Solution procedure:

1. Rewrite the constraint, so that one side gets zero: $x + y - 1 = 0$.
2. Construct the Lagrange function by incorporating the constraint into the objective function with a Lagrange multiplier λ:
$$L(x, y, \lambda) = x^2 + y^2 + \lambda(x + y - 1).$$
3. Take the partial derivatives of the Lagrange function and set them to zero (necessary conditions for a minimum):
$$\frac{\partial L}{\partial x} = 2x + \lambda = 0, \qquad \frac{\partial L}{\partial y} = 2y + \lambda = 0, \qquad \frac{\partial L}{\partial \lambda} = x + y - 1 = 0.$$
4. Solve the resulting (here: linear) equation system:
$$\lambda = -1, \qquad x = y = \tfrac{1}{2}.$$

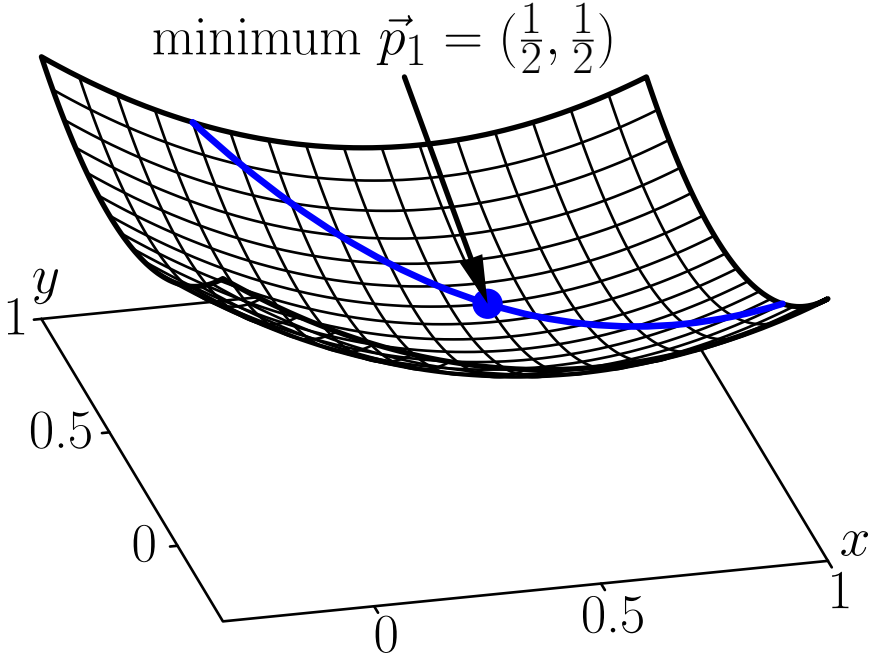
Lagrange Theory: Example 1

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 Lagrange function $L(x, y, \lambda)$
is the minimum of the objective function $f(x, y)$ 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:

$$\begin{array}{ll} \text{Maximize} & f(x, y, z) = xyz \\ \text{subject to} & 2xy + 2xz + 2yz = S. \end{array}$$

Solution procedure:

1. The constraint is $C(x, y, z) = 2xy + 2xz + 2yz - S = 0$.
2. The Lagrange function is
$$L(x, y, z, \lambda) = xyz + \lambda(2xy + 2xz + 2yz - S).$$
3. Taking the partial derivatives yields (in addition to the constraint):
$$\frac{\partial L}{\partial x} = yz+2\lambda(y+z) = 0, \quad \frac{\partial L}{\partial y} = xz+2\lambda(x+z) = 0, \quad \frac{\partial L}{\partial y} = xy+2\lambda(x+y) = 0.$$
4. The solution is: $\lambda = -\frac{1}{4}\sqrt{\frac{S}{6}}, \quad x = y = z = \sqrt{\frac{S}{6}}$ (i.e., the box is a cube).

Fuzzy Clustering: Alternating Optimization

Objective function: (to be minimized)

$$J(\mathbf{X}, \mathbf{B}, \mathbf{U}) = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^{w} d^2(\beta_i, \vec{x}_j)$$

Constraints:

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

- **Problem:** The objective function J cannot be minimized directly.
- Therefore: **Alternating Optimization**
  - Optimize membership degrees for fixed cluster parameters.
  - Optimize cluster parameters for fixed membership degrees.

    (Update formulae are derived by differentiating the objective function J)
  - Iterate until convergence (checked, e.g., by change of cluster center).

Fuzzy Clustering: Alternating Optimization

First Step: Fix the cluster parameters.

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

$$L(\mathbf{X}, \mathbf{B}, \mathbf{U}, \Lambda) = \underbrace{\sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^{w} \, d_{ij}^{2}}_{=J(\mathbf{X},\mathbf{B},\mathbf{U})} + \sum_{j=1}^{n} \lambda_j \left(1 - \sum_{i=1}^{c} u_{ij} \right),$$

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

$$\frac{\partial}{\partial u_{kl}} L(\mathbf{X}, \mathbf{B}, \mathbf{U}, \Lambda) = w \, u_{kl}^{w-1} \, d_{kl}^{2} - \lambda_l \stackrel{!}{=} 0,$$

which leads to

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

Fuzzy Clustering: Alternating Optimization

Summing these equations over the clusters (in order to be able to exploit the corresponding constraints on the membership degrees), we get

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

Consequently the λ_j, $1 \leq j \leq n$, are

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

Inserting this into the equation for the membership degrees yields

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

This update formula results regardless of the distance measure.

Standard Fuzzy Clustering Algorithms

Fuzzy C-Means Algorithm: $\beta_i = (\vec{\mu}_i)$ and Euclidean distance

$$d^2_{\mathrm{fcm}}(\beta_i, \vec{x}_j) = (\vec{x}_j - \vec{\mu}_i)^\top (\vec{x}_j - \vec{\mu}_i)$$

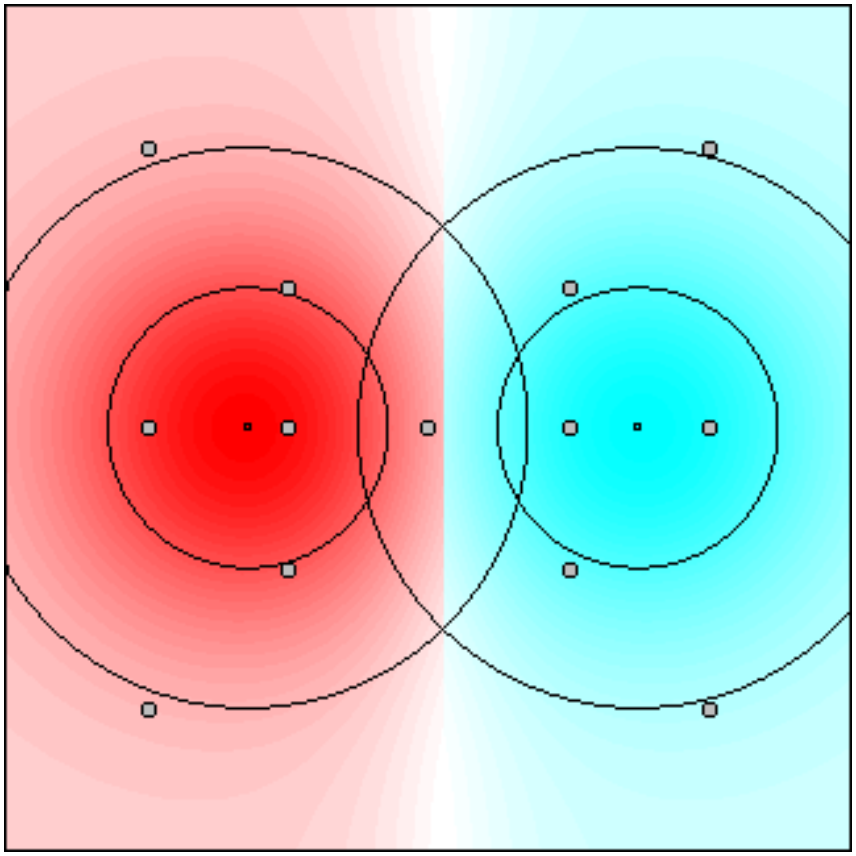
Necessary condition for a minimum: gradients w.r.t. cluster centers vanish.

$$\begin{aligned}
\nabla_{\vec{\mu}_k} J_{\mathrm{fcm}}(\mathbf{X}, \mathbf{B}, \mathbf{U}) &= \nabla_{\vec{\mu}_k} \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^w \, (\vec{x}_j - \vec{\mu}_i)^\top (\vec{x}_j - \vec{\mu}_i) \\
&= \sum_{j=1}^{n} u_{kj}^w \, \nabla_{\vec{\mu}_k} (\vec{x}_j - \vec{\mu}_k)^\top (\vec{x}_j - \vec{\mu}_k) \\
&= -2 \sum_{j=1}^{n} u_{kj}^w \, (\vec{x}_j - \vec{\mu}_k) \stackrel{!}{=} \vec{0}
\end{aligned}$$

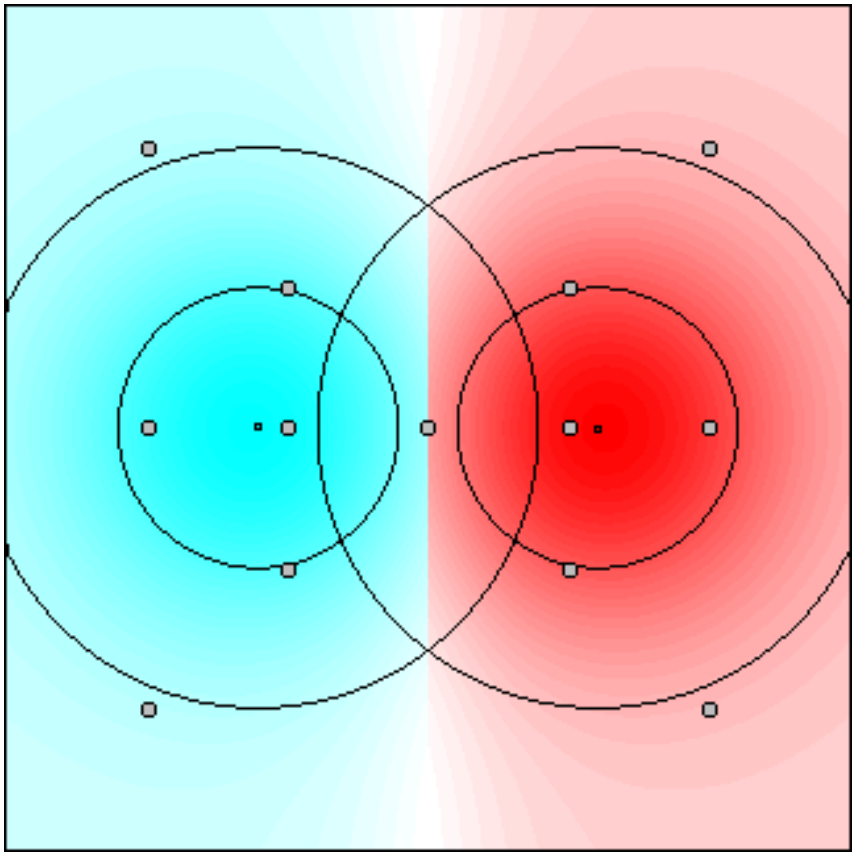
Resulting update rule for the cluster centers (**second step** of alt. optimization):

$$\forall i; 1 \leq i \leq c: \qquad \vec{\mu}_i = \frac{\sum_{j=1}^{n} u_{ij}^w \vec{x}_j}{\sum_{j=1}^{n} u_{ij}^w}$$

Fuzzy Clustering: Overlapping Clusters



Classical c-Means



Fuzzy c-Means

Standard Fuzzy Clustering Algorithms

Gustafson–Kessel Algorithm: $\beta_i = (\vec{\mu}_i, \mathbf{C}_i)$ and Mahalanobis distance

$$d_{\mathrm{gk}}^2(\beta_i, \vec{x}_j) = (\vec{x}_j - \vec{\mu}_i)^\top \mathbf{C}_i^{-1} (\vec{x}_j - \vec{\mu}_i)$$

Additional constraints: $|\mathbf{C}_i| = 1$ (all cluster have unit size).
These constraints are incorporated again by Lagrange multipliers.

A similar derivation as for the fuzzy c-means algorithm
yields the same update rule for the cluster centers:

$$\forall i; 1 \leq i \leq c: \qquad \vec{\mu}_i = \frac{\sum_{j=1}^{n} u_{ij}^w \vec{x}_j}{\sum_{j=1}^{n} u_{ij}^w}$$

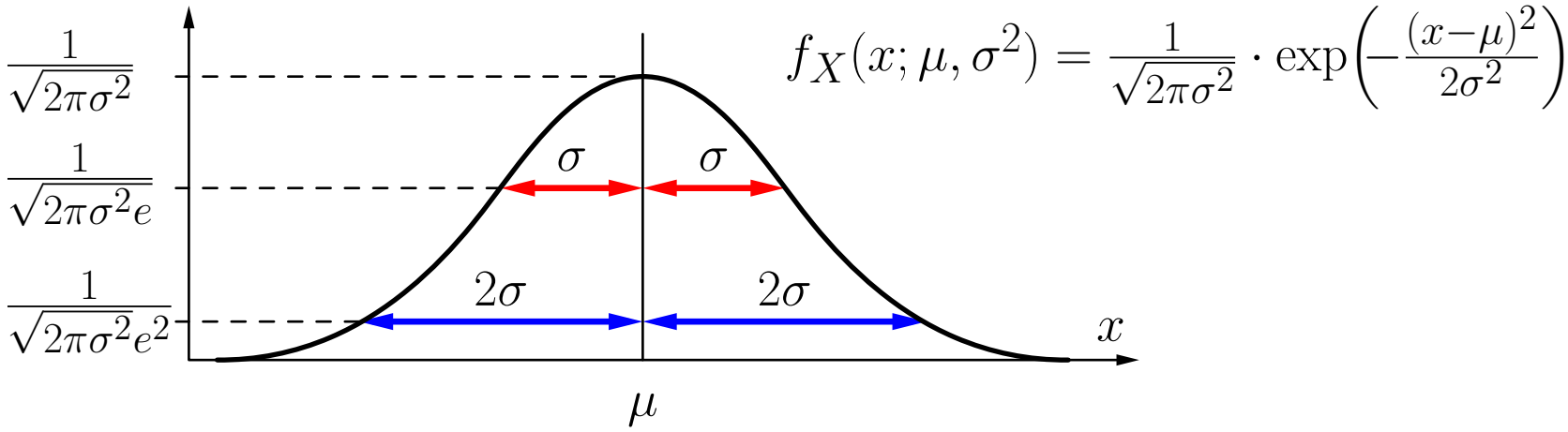
Update rule for the covariance matrices (m is the number of dimensions):

$$\mathbf{C}_i = \frac{1}{\sqrt[m]{|\mathbf{\Sigma}_i|}} \mathbf{\Sigma}_i \qquad \text{where} \qquad \mathbf{\Sigma}_i = \sum_{j=1}^{n} u_{ij}^w (\vec{x}_j - \vec{\mu}_i)(\vec{x}_j - \vec{\mu}_i)^\top.$$

Reminder: Variance and Standard Deviation

- **Special Case: Normal/Gaussian Distribution**

  The variance/standard deviation provides information about the height of the mode and the width of the curve.



- μ: expected value, estimated by mean value $\bar{x}$,
  σ^2: variance, estimated by (empirical) variance s^2,
  σ: standard deviation, estimated by (empirical) standard deviation s.
  Important: standard deviation has same unit as expected value.

Reminder: Covariance Matrix

- **Covariance Matrix** (for a two-dimensional data set)
  Compute variance formula with vectors (square: outer product $\vec{v}\vec{v}^\top$):

$$\mathbf{S} = \frac{1}{n-1}\sum_{i=1}^{n}\left(\begin{pmatrix} x_i \\ y_i \end{pmatrix} - \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix}\right)\left(\begin{pmatrix} x_i \\ y_i \end{pmatrix} - \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix}\right)^\top = \begin{pmatrix} s_x^2 & s_{xy} \\ s_{xy} & s_y^2 \end{pmatrix}$$

  where

$$s_x^2 = \frac{1}{n-1}\left(\sum_{i=1}^{n} x_i^2 - n\bar{x}^2\right) \qquad \text{(variance of } x\text{)}$$

$$s_y^2 = \frac{1}{n-1}\left(\sum_{i=1}^{n} y_i^2 - n\bar{y}^2\right) \qquad \text{(variance of } y\text{)}$$

$$s_{xy} = \frac{1}{n-1}\left(\sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y}\right) \qquad \text{(covariance of } x \text{ and } y\text{)}$$

Multivariate Normal Distribution

- A **univariate normal distribution** has the density function

$$f_X(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

μ:	expected value,	estimated by mean value $\bar{x}$,
σ^2:	variance,	estimated by (empirical) variance s^2,
σ:	standard deviation,	estimated by (empirical) standard deviation s.

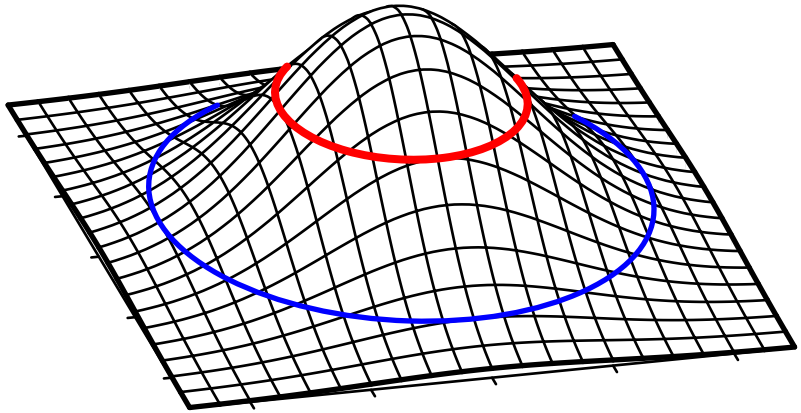
- A **multivariate normal distribution** has the density function

$$f_{\vec{X}}(\vec{x}; \vec{\mu}, \mathbf{\Sigma}) = \frac{1}{\sqrt{(2\pi)^m |\mathbf{\Sigma}|}} \cdot \exp\left(-\frac{1}{2}(\vec{x}-\vec{\mu})^\top \mathbf{\Sigma}^{-1}(\vec{x}-\vec{\mu})\right)$$

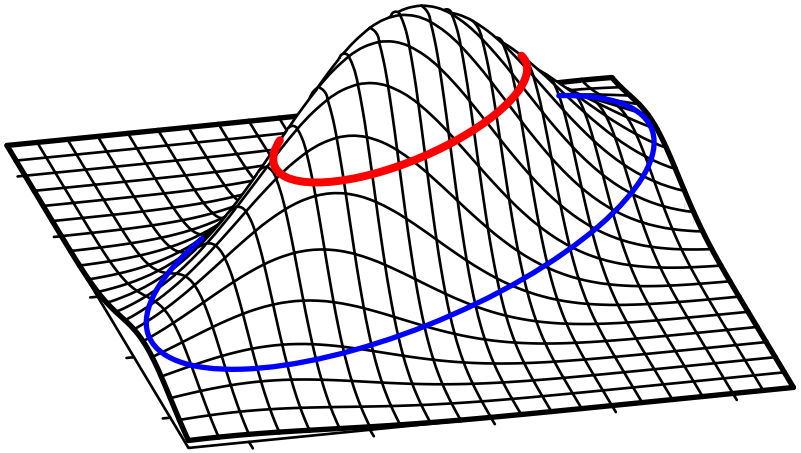
m:	size of the vector $\vec{x}$	(it is m-dimensional),
$\vec{\mu}$:	mean value vector,	estimated by (empirical) mean value vector $\bar{\vec{x}}$,
$\mathbf{\Sigma}$:	covariance matrix,	estimated by (empirical) covariance matrix $\mathbf{S}$,
$\lvert\mathbf{\Sigma}\rvert$:	determinant of the covariance matrix $\mathbf{\Sigma}$.	

Interpretation of a Covariance Matrix

- The variance/standard deviation relates the spread of the distribution to the spread of a **standard normal distribution** ($\sigma^2 = \sigma = 1$).
- The covariance matrix relates the spread of the distribution to the spread of a **multivariate standard normal distribution** ($\mathbf{\Sigma} = \mathbf{1}$).
- Example: bivariate normal distribution



standard



general

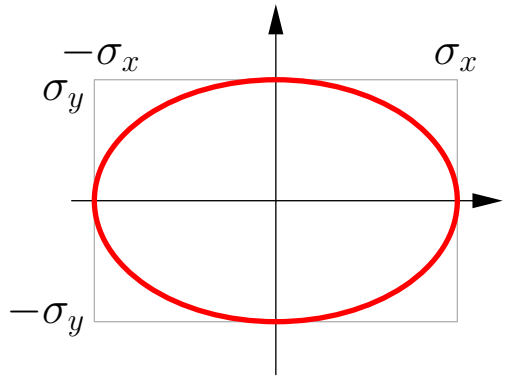
- **Question:** Is there a multivariate analog of standard deviation?

Interpretation of a Covariance Matrix

Question: Is there a multivariate analog of standard deviation?

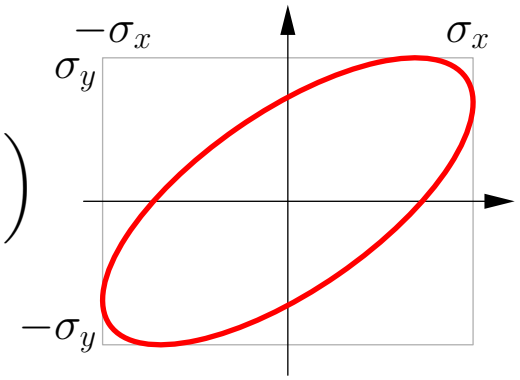
First insight:
If all covariances vanish,
the contour lines are axes-parallel ellipses.
The upper ellipse is inscribed into the
rectangle $[-\sigma_x, \sigma_x] \times [-\sigma_y, \sigma_y]$.

$$\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{pmatrix}$$



Second insight:
If the covariances do not vanish,
the contour lines are rotated ellipses.
Still the upper ellipse is inscribed into the
rectangle $[-\sigma_x, \sigma_x] \times [-\sigma_y, \sigma_y]$.

$$\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{pmatrix}$$



Consequence: A covariance matrix describes a scaling and a rotation.

Cholesky Decomposition

- Intuitively: **Compute an analog of standard deviation**.
- Let $\mathbf{S}$ be a symmetric, positive definite matrix (e.g. a covariance matrix). Cholesky decomposition serves the purpose to compute a "square root" of $\mathbf{S}$.
  - symmetric: $\forall 1 \leq i, j \leq m : s_{ij} = s_{ji}$
  - positive definite: for all m-dimensional vectors $\vec{v} \neq \vec{0}$ it is $\vec{v}^\top \mathbf{S}\, \vec{v} > 0$
- Formally: Compute a left/lower triangular matrix $\mathbf{L}$ such that $\mathbf{L}\mathbf{L}^\top = \mathbf{S}$. ($\mathbf{L}^\top$ is the transpose of the matrix $\mathbf{L}$.)

$$l_{ii} = \left(s_{ii} - \sum_{k=1}^{i-1} l_{ik}^2 \right)^{\frac{1}{2}}$$

$$l_{ji} = \frac{1}{l_{ii}} \left(s_{ij} - \sum_{k=1}^{i-1} l_{ik} l_{jk} \right), \qquad j = i+1, i+2, \ldots, m.$$

Cholesky Decomposition

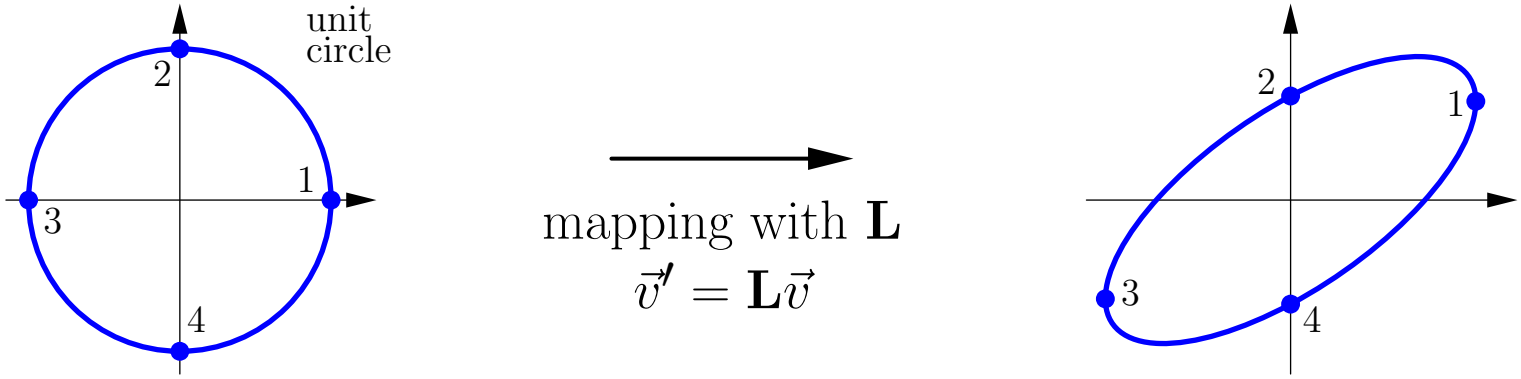
Special Case: Two Dimensions

- Covariance matrix

$$\mathbf{\Sigma} = \begin{pmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{pmatrix}$$

- Cholesky decomposition

$$\mathbf{L} = \begin{pmatrix} \sigma_x & 0 \\ \frac{\sigma_{xy}}{\sigma_x} & \frac{1}{\sigma_x}\sqrt{\sigma_x^2\sigma_y^2 - \sigma_{xy}^2} \end{pmatrix}$$



Eigenvalue Decomposition

- Also yields an **analog of standard deviation**.
- Computationally more expensive than Cholesky decomposition.
- Let $\mathbf{S}$ be a symmetric, positive definite matrix (e.g. a covariance matrix).
  - $\mathbf{S}$ can be written as

    $$\mathbf{S} = \mathbf{R}\ \mathrm{diag}(\lambda_1, \ldots, \lambda_m)\ \mathbf{R}^{-1},$$

    where the λ_j, $j = 1, \ldots, m$, are the eigenvalues of $\mathbf{S}$ and the columns of $\mathbf{R}$ are the (normalized) eigenvectors of $\mathbf{S}$.
  - The eigenvalues λ_j, $j = 1, \ldots, m$, of $\mathbf{S}$ are all positive and the eigenvectors of $\mathbf{S}$ are orthonormal ($\rightarrow \mathbf{R}^{-1} = \mathbf{R}^\top$).
- Due to the above, $\mathbf{S}$ can be written as $\mathbf{S} = \mathbf{T}\,\mathbf{T}^\top$, where

  $$\mathbf{T} = \mathbf{R}\ \mathrm{diag}\left(\sqrt{\lambda_1}, \ldots, \sqrt{\lambda_m}\right)$$

Eigenvalue Decomposition

Special Case: Two Dimensions

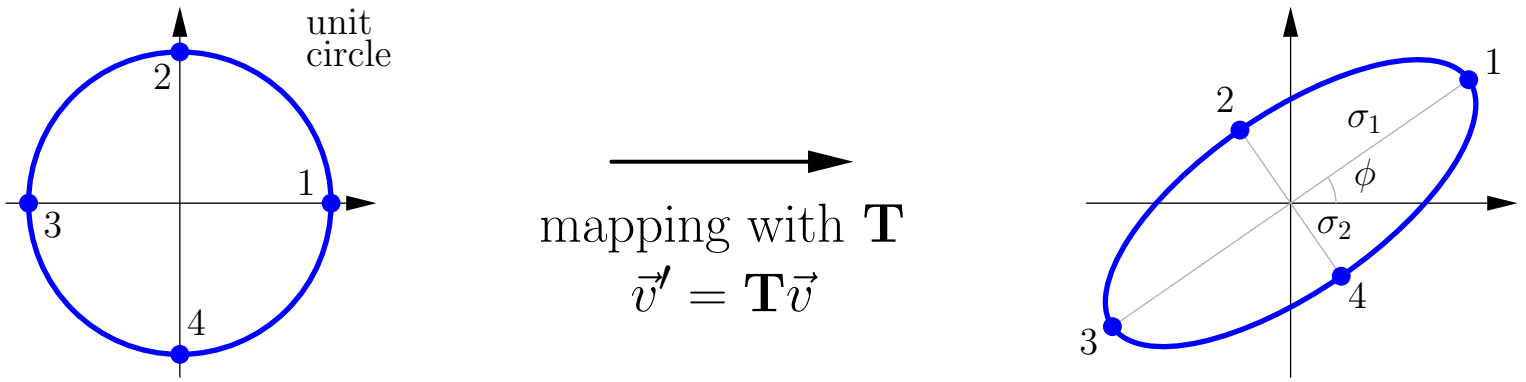
- Covariance matrix

$$\mathbf{\Sigma} = \begin{pmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{pmatrix}$$

- Eigenvalue decomposition

$$\mathbf{T} = \begin{pmatrix} c & -s \\ s & c \end{pmatrix} \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix},$$

$$s = \sin\phi, c = \cos\phi, \phi = \frac{1}{2}\arctan\frac{2\sigma_{xy}}{\sigma_x^2 - \sigma_y^2},$$
$$\sigma_1 = \sqrt{c^2\sigma_x^2 + s^2\sigma_y^2 + 2sc\sigma_{xy}},$$
$$\sigma_2 = \sqrt{s^2\sigma_x^2 + c^2\sigma_y^2 - 2sc\sigma_{xy}}.$$



Eigenvalue Decomposition

Eigenvalue decomposition enables us to write a covariance matrix $\mathbf{\Sigma}$ as

$$\mathbf{\Sigma} = \mathbf{T}\mathbf{T}^\top \qquad \text{with} \qquad \mathbf{T} = \mathbf{R}\,\mathrm{diag}\left(\sqrt{\lambda_1}, \ldots, \sqrt{\lambda_m}\right).$$

As a consequence we can write its inverse $\mathbf{\Sigma}^{-1}$ as

$$\mathbf{\Sigma}^{-1} = \mathbf{U}^\top\mathbf{U} \qquad \text{with} \qquad \mathbf{U} = \mathrm{diag}\left(\lambda_1^{-\frac{1}{2}}, \ldots, \lambda_m^{-\frac{1}{2}}\right)\mathbf{R}^\top.$$

$\mathbf{U}$ describes the inverse mapping of $\mathbf{T}$, i.e., rotates the ellipse so that its axes coincide with the coordinate axes and then scales the axes to unit length. Hence:

$$(\vec{x}-\vec{y})^\top\mathbf{\Sigma}^{-1}(\vec{x}-\vec{y}) \;=\; (\vec{x}-\vec{y})^\top\mathbf{U}^\top\mathbf{U}(\vec{x}-\vec{y}) \;=\; (\vec{x}\,'-\vec{y}\,')^\top(\vec{x}\,'-\vec{y}\,'),$$

where $\vec{x}\,' = \mathbf{U}\vec{x}$ and $\vec{y}\,' = \mathbf{U}\vec{y}$.

Result: $(\vec{x}-\vec{y})^\top\mathbf{\Sigma}^{-1}(\vec{x}-\vec{y})$ is equivalent to the squared **Euclidean distance** in the properly scaled eigensystem of the covariance matrix $\mathbf{\Sigma}$.

$$d(\vec{x},\vec{y}) = \sqrt{(\vec{x}-\vec{y})^\top\mathbf{\Sigma}^{-1}(\vec{x}-\vec{y})} \qquad \text{is called } \textbf{Mahalanobis distance}.$$

Eigenvalue Decomposition

Eigenvector decomposition also shows that the determinant of the covariance matrix $\mathbf{\Sigma}$ provides a measure of the (hyper-)volume of the (hyper-)ellipsoid. It is

$$|\mathbf{\Sigma}| = |\mathbf{R}|\,|\mathrm{diag}(\lambda_1, \ldots, \lambda_m)|\,|\mathbf{R}^\top| = |\mathrm{diag}(\lambda_1, \ldots, \lambda_m)| = \prod_{i=1}^{m} \lambda_i,$$

since $|\mathbf{R}| = |\mathbf{R}^\top| = 1$ as $\mathbf{R}$ is orthogonal with unit length columns, and thus

$$\sqrt{|\mathbf{\Sigma}|} = \prod_{i=1}^{m} \sqrt{\lambda_i},$$

which is proportional to the (hyper-)volume of the (hyper-)ellipsoid.

To be precise, the volume of the m-dimensional (hyper-)ellipsoid a (hyper-)sphere with radius r is mapped to with a covariance matrix $\mathbf{\Sigma}$ is

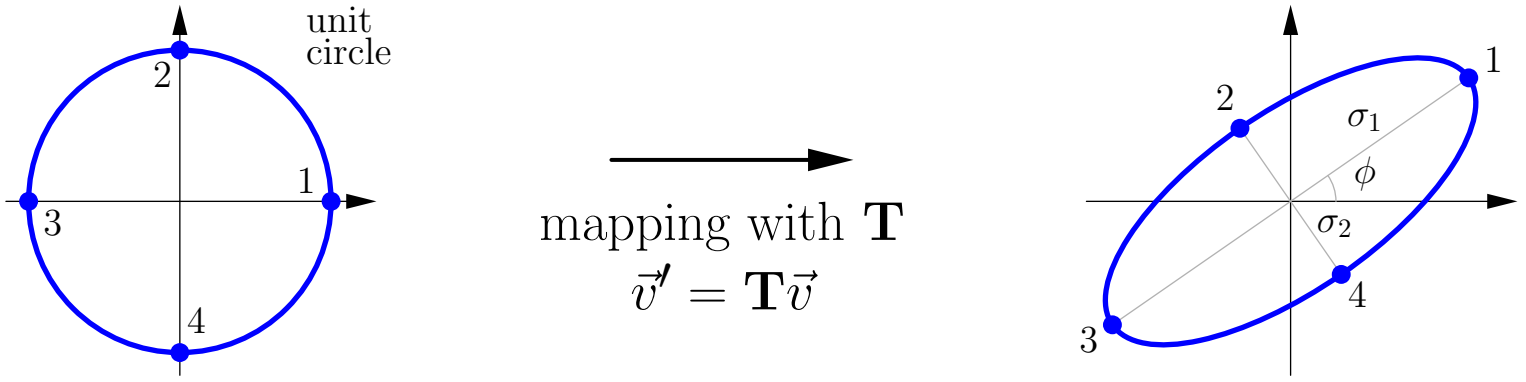
$$V_m(r) = \frac{\pi^{\frac{m}{2}} r^m}{\Gamma\left(\frac{m}{2}+1\right)} \sqrt{|\mathbf{\Sigma}|}, \quad \text{where} \quad \begin{array}{l} \Gamma(x) = \int_0^\infty e^{-t} t^{x-1}\,\mathrm{d}t, \quad x > 0, \\ \Gamma(x+1) = x \cdot \Gamma(x), \;\; \Gamma(\frac{1}{2}) = \sqrt{\pi}, \;\; \Gamma(1) = 1. \end{array}$$

Eigenvalue Decomposition

Special Case: Two Dimensions

- Covariance matrix and its eigenvalue decomposition:

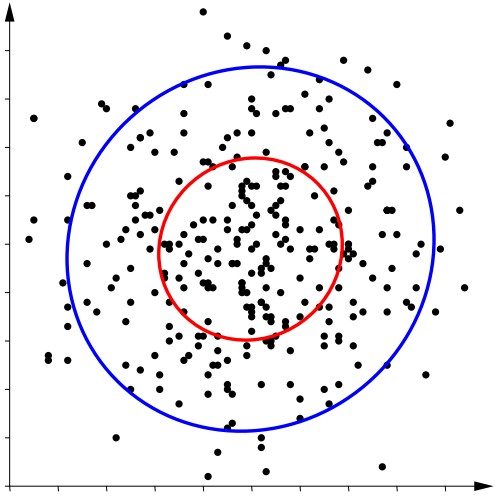
$$\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{pmatrix} \quad \text{and} \quad \mathbf{T} = \begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix} \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix}.$$



- The area of the ellipse, to which the unit circle (area π) is mapped, is

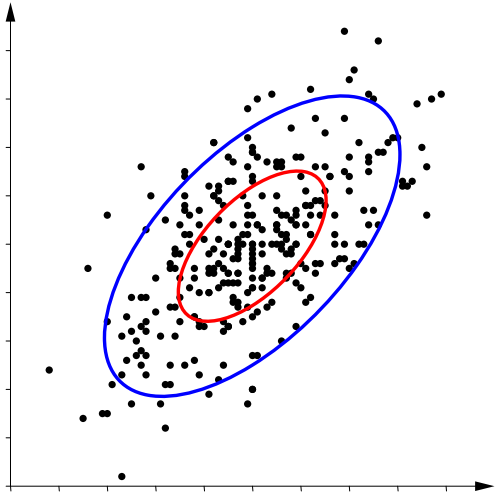
$$A = \pi\sigma_1\sigma_2 = \pi\sqrt{|\boldsymbol{\Sigma}|}.$$

Covariance Matrices of Example Data Sets



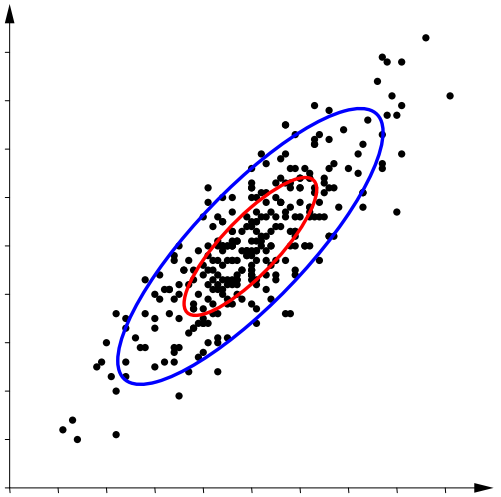
$$\boldsymbol{\Sigma} = \begin{pmatrix} 3.59 & 0.19 \\ 0.19 & 3.54 \end{pmatrix}$$

$$\mathbf{L} = \begin{pmatrix} 1.90 & 0 \\ 0.10 & 1.88 \end{pmatrix}$$



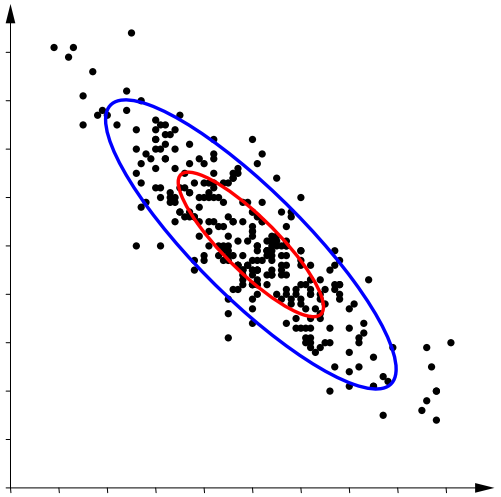
$$\boldsymbol{\Sigma} = \begin{pmatrix} 2.33 & 1.44 \\ 1.44 & 2.41 \end{pmatrix}$$

$$\mathbf{L} = \begin{pmatrix} 1.52 & 0 \\ 0.95 & 1.22 \end{pmatrix}$$



$$\boldsymbol{\Sigma} = \begin{pmatrix} 1.88 & 1.62 \\ 1.62 & 2.03 \end{pmatrix}$$

$$\mathbf{L} = \begin{pmatrix} 1.37 & 0 \\ 1.18 & 0.80 \end{pmatrix}$$



$$\boldsymbol{\Sigma} = \begin{pmatrix} 2.25 & -1.93 \\ -1.93 & 2.23 \end{pmatrix}$$

$$\mathbf{L} = \begin{pmatrix} 1.50 & 0 \\ -1.29 & 0.76 \end{pmatrix}$$

Covariance Matrix: Summary

- A covariance matrix provides information about the **height of the mode** and about the **spread/dispersion** of a multivariate normal distribution (or of a set of data points that are roughly normally distributed).
- A multivariate **analog of standard deviation** can be computed with Cholesky decomposition and eigenvalue decomposition.
  The resulting matrix describes the distribution's shape and orientation.
- The shape and the orientation of a two-dimensional normal distribution can be visualized as an **ellipse** (curve of equal probability density; similar to a **contour line** — line of equal height — on a map.)
- The shape and the orientation of a three-dimensional normal distribution can be visualized as an **ellipsoid** (surface of equal probability density).
- The (square root of the) **determinant** of a covariance matrix describes the spread of a multivariate normal distribution with a single value.
  It is a measure of the area or (hyper-)volume of the (hyper-)ellipsoid.

The Iris Data

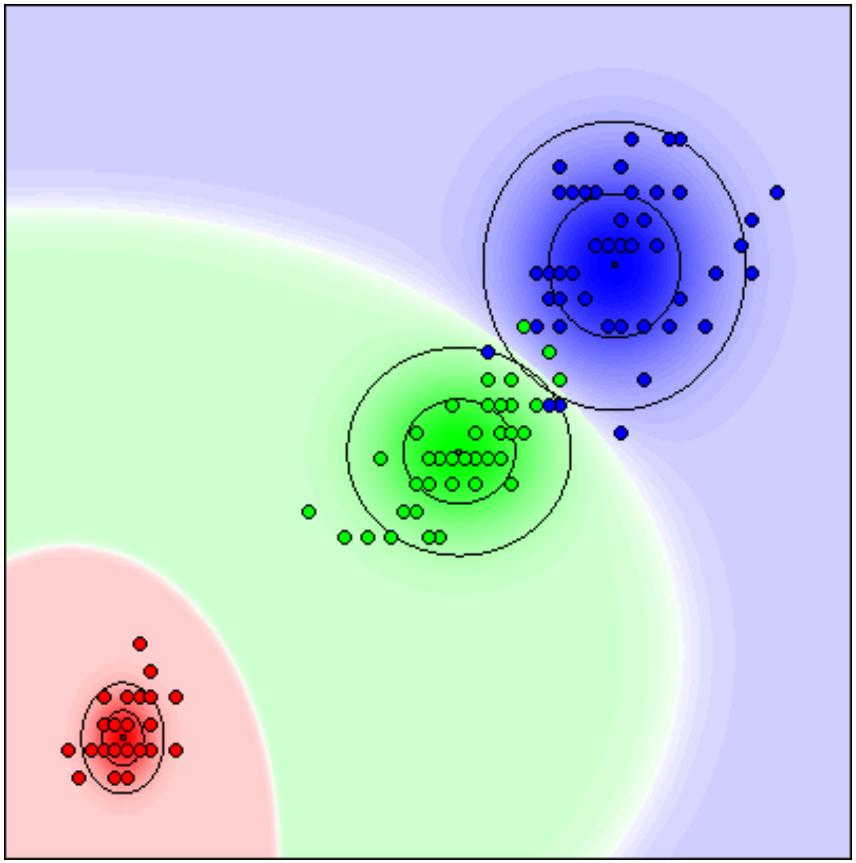
Iris setosa

Iris versicolor

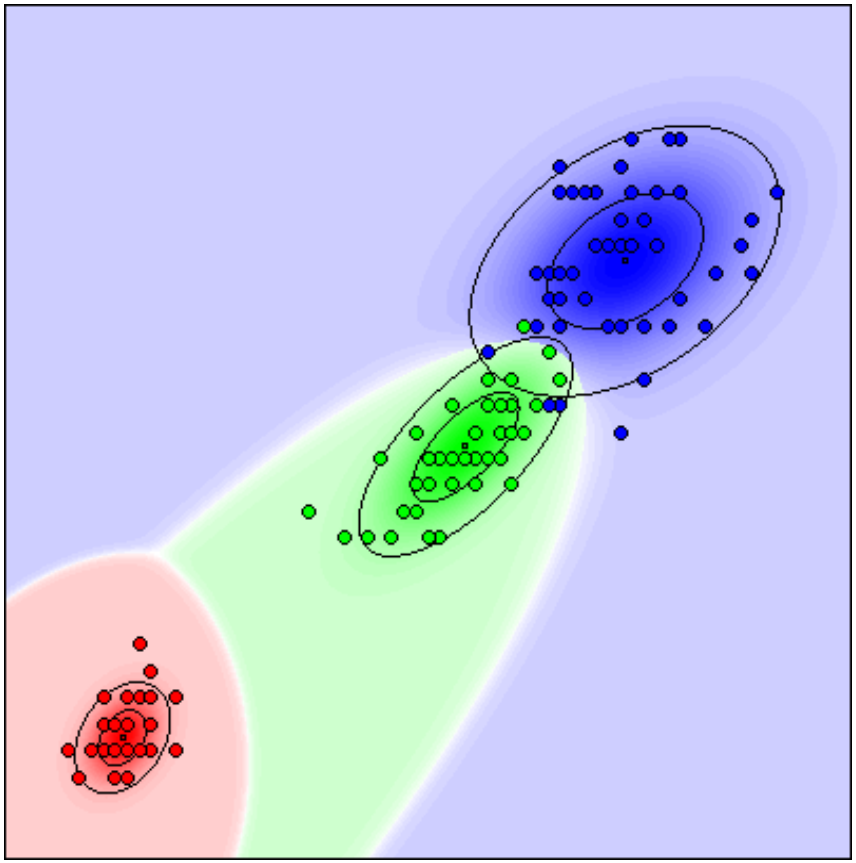
Iris virginica

- Used early on by Ronald Aylmer Fischer (famous statistician).
- 150 cases in total, 50 cases per Iris flower type.
- Measurements of sepal length and width and petal length and width (in cm).
- Most famous data set in pattern recognition and data analysis.

Bayes Classifiers for the Iris Data

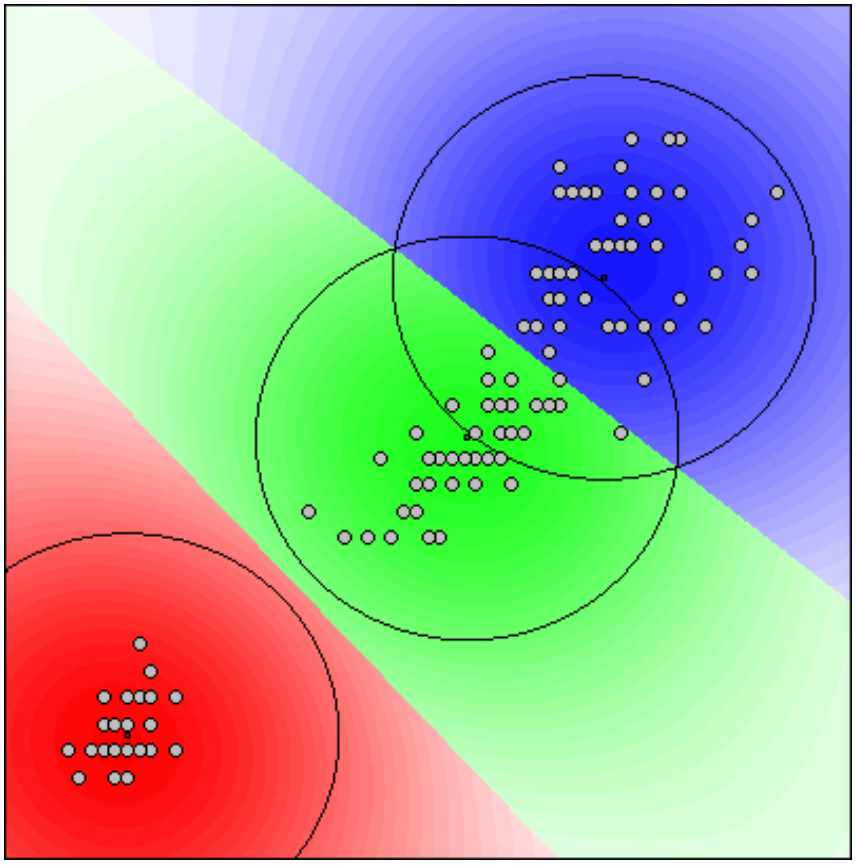


naive Bayes classifier

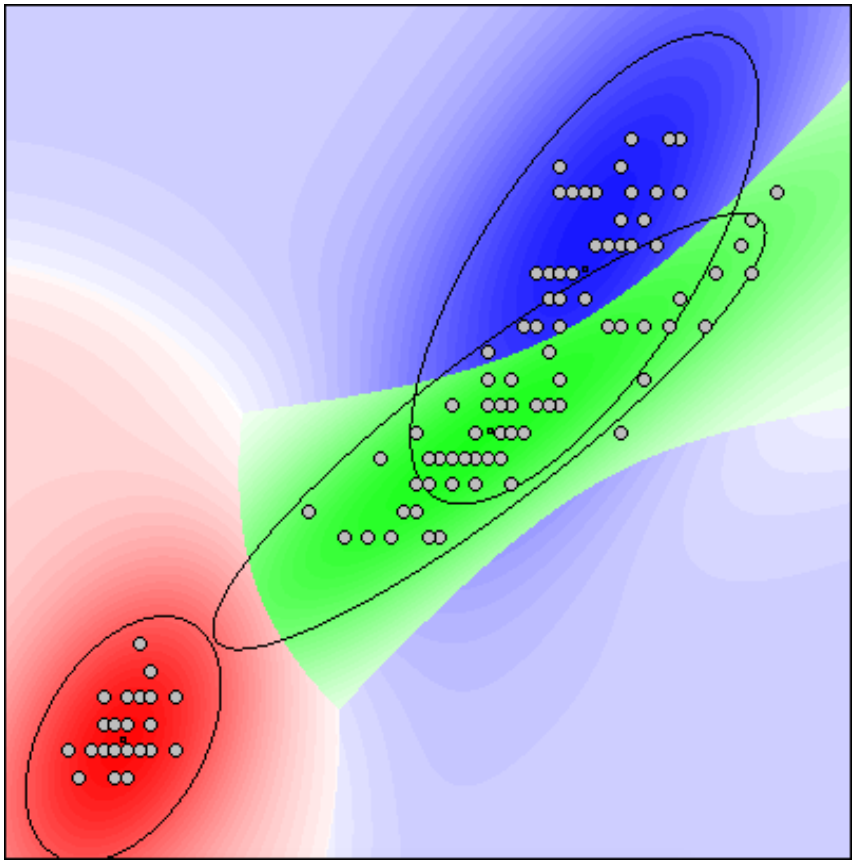


full Bayes classifier

Fuzzy Clustering of the Iris Data



Fuzzy c-Means



Gustafson–Kessel

Mixture of Gaussians

- **Assumption:** Data was generated by sampling a set of normal distributions. (The probability density is a mixture of Gaussian distributions.)
- **Formally:** We assume that the probability density can be described as

$$f_{\vec{X}}(\vec{x}; \mathbf{C}) = \sum_{y=1}^{c} f_{\vec{X},Y}(\vec{x}, y; \mathbf{C}) = \sum_{y=1}^{c} p_Y(y; \mathbf{C}) \cdot f_{\vec{X}|Y}(\vec{x}|y; \mathbf{C}).$$

$\mathbf{C}$	is the set of cluster parameters
$\vec{X}$	is a random vector that has the data space as its domain
Y	is a random variable that has the cluster indices as possible values (i.e., $\mathrm{dom}(\vec{X}) = \mathbb{R}^m$ and $\mathrm{dom}(Y) = \{1, \ldots, c\}$)
$p_Y(y; \mathbf{C})$	is the probability that a data point belongs to (is generated by) the y-th component of the mixture
$f_{\vec{X}\|Y}(\vec{x}\|y; \mathbf{C})$	is the conditional probability density function of a data point given the cluster (specified by the cluster index y)

Expectation Maximization

- **Basic idea:** Do a maximum likelihood estimation of the cluster parameters.
- **Problem:** The likelihood function,

$$L(\mathbf{X}; \mathbf{C}) = \prod_{j=1}^{n} f_{\vec{X}_j}(\vec{x}_j; \mathbf{C}) = \prod_{j=1}^{n} \sum_{y=1}^{c} p_Y(y; \mathbf{C}) \cdot f_{\vec{X}|Y}(\vec{x}_j|y; \mathbf{C}),$$

  is difficult to optimize, even if one takes the natural logarithm (cf. the maximum likelihood estimation of the parameters of a normal distribution), because

$$\ln L(\mathbf{X}; \mathbf{C}) = \sum_{j=1}^{n} \ln \sum_{y=1}^{c} p_Y(y; \mathbf{C}) \cdot f_{\vec{X}|Y}(\vec{x}_j|y; \mathbf{C})$$

  contains the natural logarithms of complex sums.
- **Approach:** Assume that there are "hidden" variables Y_j stating the clusters that generated the data points $\vec{x}_j$, so that the sums reduce to one term.
- **Problem:** Since the Y_j are hidden, we do not know their values.

Expectation Maximization

- **Formally:** Maximize the likelihood of the "completed" data set $(\mathbf{X}, \vec{y})$, where $\vec{y} = (y_1, \ldots, y_n)$ combines the values of the variables Y_j. That is,

$$L(\mathbf{X}, \vec{y}; \mathbf{C}) = \prod_{j=1}^{n} f_{\vec{X}_j, Y_j}(\vec{x}_j, y_j; \mathbf{C}) = \prod_{j=1}^{n} p_{Y_j}(y_j; \mathbf{C}) \cdot f_{\vec{X}_j | Y_j}(\vec{x}_j | y_j; \mathbf{C}).$$

- **Problem:** Since the Y_j are hidden, the values y_j are unknown (and thus the factors $p_{Y_j}(y_j; \mathbf{C})$ cannot be computed).

- **Approach to find a solution nevertheless:**
  - See the Y_j as random variables (the values y_j are not fixed) and consider a probability distribution over the possible values.
  - As a consequence $L(\mathbf{X}, \vec{y}; \mathbf{C})$ becomes a random variable, even for a fixed data set $\mathbf{X}$ and fixed cluster parameters $\mathbf{C}$.
  - Try to **maximize the expected value** of $L(\mathbf{X}, \vec{y}; \mathbf{C})$ or $\ln L(\mathbf{X}, \vec{y}; \mathbf{C})$ (hence the name **expectation maximization**).

Expectation Maximization

- **Formally:** Find the cluster parameters as

$$\hat{\mathbf{C}} = \operatorname*{argmax}_{\mathbf{C}} E([\ln] L(\mathbf{X}, \vec{y}; \mathbf{C}) \mid \mathbf{X}; \mathbf{C}),$$

that is, maximize the expected likelihood

$$E(L(\mathbf{X}, \vec{y}; \mathbf{C}) \mid \mathbf{X}; \mathbf{C}) = \sum_{\vec{y} \in \{1,\ldots,c\}^n} p_{\vec{Y}|\mathcal{X}}(\vec{y}\,|\mathbf{X}; \mathbf{C}) \cdot \prod_{j=1}^{n} f_{\vec{X}_j, Y_j}(\vec{x}_j, y_j; \mathbf{C})$$

or, alternatively, maximize the expected log-likelihood

$$E(\ln L(\mathbf{X}, \vec{y}; \mathbf{C}) \mid \mathbf{X}; \mathbf{C}) = \sum_{\vec{y} \in \{1,\ldots,c\}^n} p_{\vec{Y}|\mathcal{X}}(\vec{y}\,|\mathbf{X}; \mathbf{C}) \cdot \sum_{j=1}^{n} \ln f_{\vec{X}_j, Y_j}(\vec{x}_j, y_j; \mathbf{C}).$$

- Unfortunately, these functionals are still difficult to optimize directly.
- **Solution:** Use the equation as an iterative scheme, fixing $\mathbf{C}$ in some terms (iteratively compute better approximations, similar to Heron's algorithm).

Excursion: Heron's Algorithm

- **Task:** Find the square root of a given number x, i.e., find $y = \sqrt{x}$.
- **Approach:** Rewrite the defining equation $y^2 = x$ as follows:

$$y^2 = x \quad \Leftrightarrow \quad 2y^2 = y^2 + x \quad \Leftrightarrow \quad y = \frac{1}{2y}(y^2 + x) \quad \Leftrightarrow \quad y = \frac{1}{2}\left(y + \frac{x}{y}\right).$$

- Use the resulting equation as an iteration formula, i.e., compute the sequence

$$y_{k+1} = \frac{1}{2}\left(y_k + \frac{x}{y_k}\right) \qquad \text{with} \qquad y_0 = 1.$$

- It can be shown that $0 \leq y_k - \sqrt{x} \leq y_{k-1} - y_n$ for $k \geq 2$.
  Therefore this iteration formula provides increasingly better approximations of the square root of x and thus is a safe and simple way to compute it.
  Ex.: $x = 2$: $y_0 = 1$, $y_1 = 1.5$, $y_2 \approx 1.41667$, $y_3 \approx 1.414216$, $y_4 \approx 1.414213$.
- Heron's algorithm converges very quickly and is often used in pocket calculators and microprocessors to implement the square root.

Expectation Maximization

- **Iterative scheme for expectation maximization:**
  Choose some initial set $\mathbf{C}_0$ of cluster parameters and then compute

$$\begin{aligned}
\mathbf{C}_{k+1} &= \operatorname*{argmax}_{\mathbf{C}} E(\ln L(\mathbf{X}, \vec{y}; \mathbf{C}) \mid \mathbf{X}; \mathbf{C}_k) \\
&= \operatorname*{argmax}_{\mathbf{C}} \sum_{\vec{y} \in \{1,\ldots,c\}^n} p_{\vec{Y}|\mathcal{X}}(\vec{y}\,|\mathbf{X}; \mathbf{C}_k) \sum_{j=1}^{n} \ln f_{\vec{X}_j, Y_j}(\vec{x}_j, y_j; \mathbf{C}) \\
&= \operatorname*{argmax}_{\mathbf{C}} \sum_{\vec{y} \in \{1,\ldots,c\}^n} \left(\prod_{l=1}^{n} p_{Y_l|\vec{X}_l}(y_l|\vec{x}_l; \mathbf{C}_k) \right) \sum_{j=1}^{n} \ln f_{\vec{X}_j, Y_j}(\vec{x}_j, y_j; \mathbf{C}) \\
&= \operatorname*{argmax}_{\mathbf{C}} \sum_{i=1}^{c} \sum_{j=1}^{n} p_{Y_j|\vec{X}_j}(i|\vec{x}_j; \mathbf{C}_k) \cdot \ln f_{\vec{X}_j, Y_j}(\vec{x}_j, i; \mathbf{C}).
\end{aligned}$$

- It can be shown that each EM iteration increases the likelihood of the data and that the algorithm converges to a local maximum of the likelihood function (i.e., EM is a safe way to maximize the likelihood function).

Expectation Maximization

Justification of the last step on the previous slide:

$$\sum_{\vec{y}\in\{1,\ldots,c\}^n} \left(\prod_{l=1}^{n} p_{Y_l|\vec{X}_l}(y_l|\vec{x}_l;\mathbf{C}_k)\right) \sum_{j=1}^{n} \ln f_{\vec{X}_j,Y_j}(\vec{x}_j,y_j;\mathbf{C})$$

$$= \sum_{y_1=1}^{c} \cdots \sum_{y_n=1}^{c} \left(\prod_{l=1}^{n} p_{Y_l|\vec{X}_l}(y_l|\vec{x}_l;\mathbf{C}_k)\right) \sum_{j=1}^{n} \sum_{i=1}^{c} \delta_{i,y_j} \ln f_{\vec{X}_j,Y_j}(\vec{x}_j,i;\mathbf{C})$$

$$= \sum_{i=1}^{c} \sum_{j=1}^{n} \ln f_{\vec{X}_j,Y_j}(\vec{x}_j,i;\mathbf{C}) \sum_{y_1=1}^{c} \cdots \sum_{y_n=1}^{c} \delta_{i,y_j} \prod_{l=1}^{n} p_{Y_l|\vec{X}_l}(y_l|\vec{x}_l;\mathbf{C}_k)$$

$$= \sum_{i=1}^{c} \sum_{j=1}^{n} p_{Y_j|\vec{X}_j}(i|\vec{x}_j;\mathbf{C}_k) \cdot \ln f_{\vec{X}_j,Y_j}(\vec{x}_j,i;\mathbf{C})$$

$$\underbrace{\sum_{y_1=1}^{c} \cdots \sum_{y_{j-1}=1}^{c} \sum_{y_{j+1}=1}^{c} \cdots \sum_{y_n=1}^{c} \prod_{l=1,l\neq j}^{n} p_{Y_l|\vec{X}_l}(y_l|\vec{x}_l;\mathbf{C}_k)}_{= \prod_{l=1,l\neq j}^{n} \sum_{y_l=1}^{c} p_{Y_l|\vec{X}_l}(y_l|\vec{x}_l;\mathbf{C}_k) \;=\; \prod_{l=1,l\neq j}^{n} 1 \;=\; 1} .$$

Expectation Maximization

- The probabilities $p_{Y_j|\vec{X}_j}(i|\vec{x}_j; \mathbf{C}_k)$ are computed as

$$p_{Y_j|\vec{X}_j}(i|\vec{x}_j; \mathbf{C}_k) = \frac{f_{\vec{X}_j,Y_j}(\vec{x}_j, i; \mathbf{C}_k)}{f_{\vec{X}_j}(\vec{x}_j; \mathbf{C}_k)} = \frac{f_{\vec{X}_j|Y_j}(\vec{x}_j|i; \mathbf{C}_k) \cdot p_{Y_j}(i; \mathbf{C}_k)}{\sum_{l=1}^{c} f_{\vec{X}_j|Y_j}(\vec{x}_j|l; \mathbf{C}_k) \cdot p_{Y_j}(l; \mathbf{C}_k)},$$

  that is, as the relative probability densities of the different clusters (as specified by the cluster parameters) at the location of the data points $\vec{x}_j$.

- The $p_{Y_j|\vec{X}_j}(i|\vec{x}_j; \mathbf{C}_k)$ are the posterior probabilities of the clusters given the data point $\vec{x}_j$ and a set of cluster parameters $\mathbf{C}_k$.

- They can be seen as **case weights** of a "completed" data set:
  - Split each data point $\vec{x}_j$ into c data points $(\vec{x}_j, i)$, $i = 1, \ldots, c$.
  - Distribute the unit weight of the data point $\vec{x}_j$ according to the above probabilities, i.e., assign to $(\vec{x}_j, i)$ the weight $p_{Y_j|\vec{X}_j}(i|\vec{x}_j; \mathbf{C}_k)$, $i = 1, \ldots, c$.

Expectation Maximization: Cookbook Recipe

Core Iteration Formula

$$\mathbf{C}_{k+1} = \underset{\mathbf{C}}{\operatorname{argmax}} \sum_{i=1}^{c} \sum_{j=1}^{n} p_{Y_j|\vec{X}_j}(i|\vec{x}_j; \mathbf{C}_k) \cdot \ln f_{\vec{X}_j, Y_j}(\vec{x}_j, i; \mathbf{C})$$

Expectation Step

- For all data points $\vec{x}_j$:
  Compute for each normal distribution the probability $p_{Y_j|\vec{X}_j}(i|\vec{x}_j; \mathbf{C}_k)$
  that the data point was generated from it
  (ratio of probability densities at the location of the data point).
  $\rightarrow$ "weight" of the data point for the estimation.

Maximization Step

- For all normal distributions:
  Estimate the parameters by standard maximum likelihood estimation
  using the probabilities ("weights") assigned to the data points
  w.r.t. the distribution in the expectation step.

Expectation Maximization: Mixture of Gaussians

Expectation Step: Use Bayes' rule to compute

$$p_{C|\vec{X}}(i|\vec{x};\mathbf{C}) = \frac{p_C(i;\mathbf{c}_i)\cdot f_{\vec{X}|C}(\vec{x}|i;\mathbf{c}_i)}{f_{\vec{X}}(\vec{x};\mathbf{C})} = \frac{p_C(i;\mathbf{c}_i)\cdot f_{\vec{X}|C}(\vec{x}|i;\mathbf{c}_i)}{\sum_{k=1}^{c} p_C(k;\mathbf{c}_k)\cdot f_{\vec{X}|C}(\vec{x}|k;\mathbf{c}_k)}.$$

$\rightarrow$ "weight" of the data point $\vec{x}$ for the estimation.

Maximization Step: Use maximum likelihood estimation to compute

$$\varrho_i^{(t+1)} = \frac{1}{n}\sum_{j=1}^{n} p_{C|\vec{X}_j}(i|\vec{x}_j;\mathbf{C}^{(t)}), \qquad \vec{\mu}_i^{(t+1)} = \frac{\sum_{j=1}^{n} p_{C|\vec{X}_j}(i|\vec{x}_j;\mathbf{C}^{(t)})\cdot\vec{x}_j}{\sum_{j=1}^{n} p_{C|\vec{X}_j}(i|\vec{x}_j;\mathbf{C}^{(t)})},$$

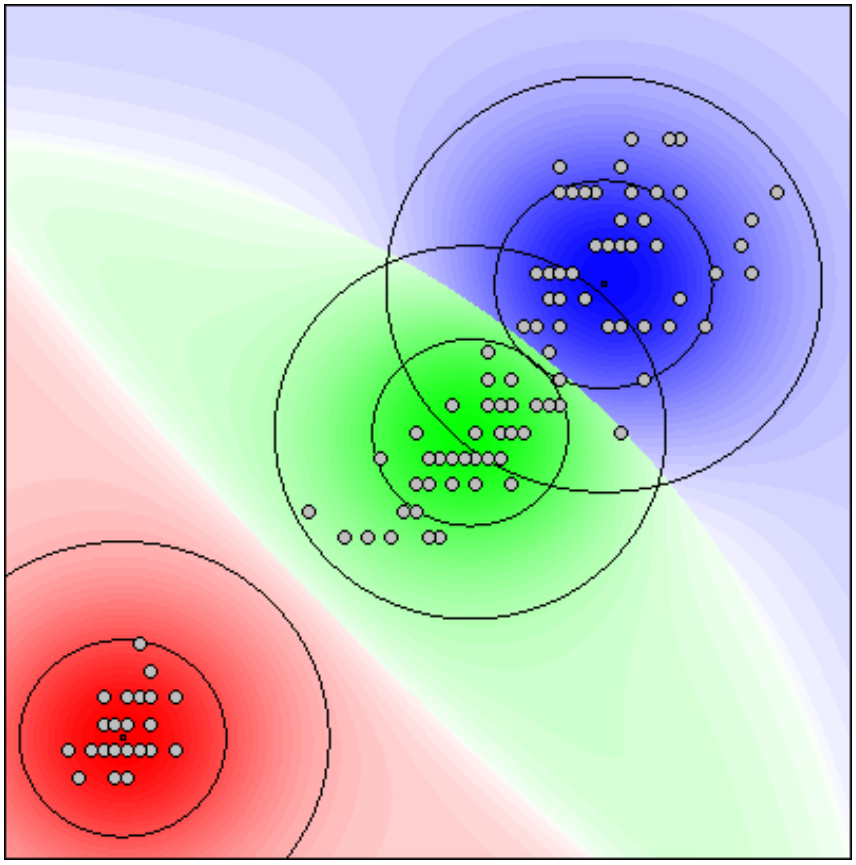
$$\text{and} \qquad \mathbf{\Sigma}_i^{(t+1)} = \frac{\sum_{j=1}^{n} p_{C|\vec{X}_j}(i|\vec{x}_j;\mathbf{C}^{(t)})\cdot\left(\vec{x}_j-\vec{\mu}_i^{(t+1)}\right)\left(\vec{x}_j-\vec{\mu}_i^{(t+1)}\right)^{\top}}{\sum_{j=1}^{n} p_{C|\vec{X}_j}(i|\vec{x}_j;\mathbf{C}^{(t)})}$$

Iterate until convergence (checked, e.g., by change of mean vector).

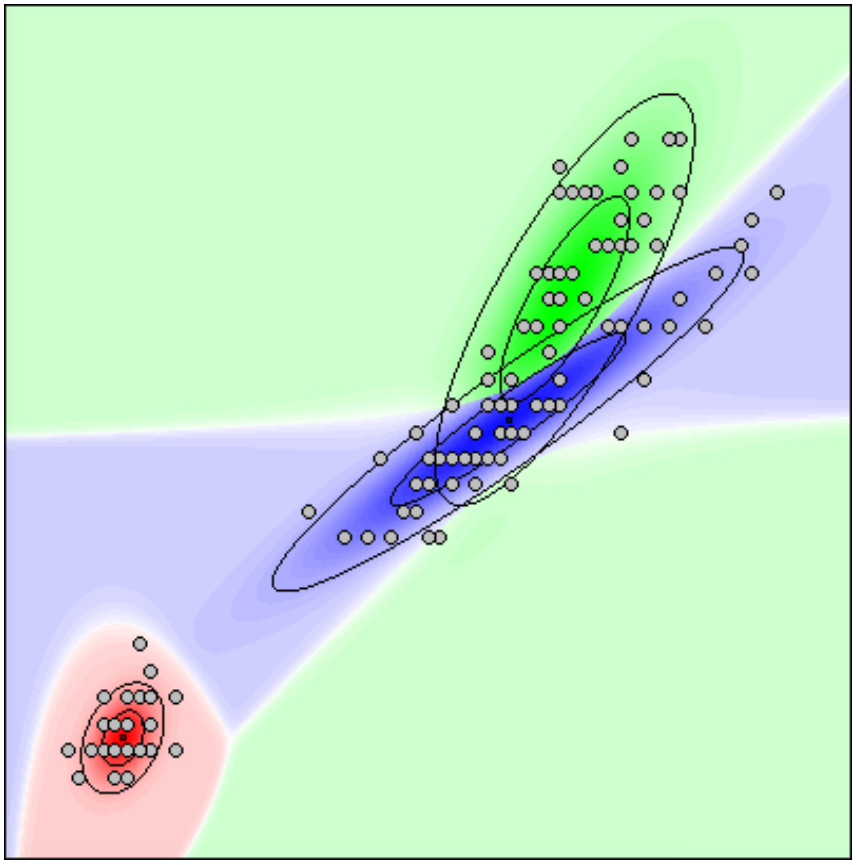
Expectation Maximization: Technical Problems

- If a fully general mixture of Gaussian distributions is used,
  the likelihood function is truly optimized if
  - all normal distributions except one are contracted to single data points and
  - the remaining normal distribution is the maximum likelihood estimate for the remaining data points.
- This undesired result is rare,
  because the algorithm gets stuck in a local optimum.
- Nevertheless it is recommended to take countermeasures,
  which consist mainly in reducing the degrees of freedom, like
  - Fix the determinants of the covariance matrices to equal values.
  - Use a diagonal instead of a general covariance matrix.
  - Use an isotropic variance instead of a covariance matrix.
  - Fix the prior probabilities of the clusters to equal values.

Expectation Maximization of the Iris Data

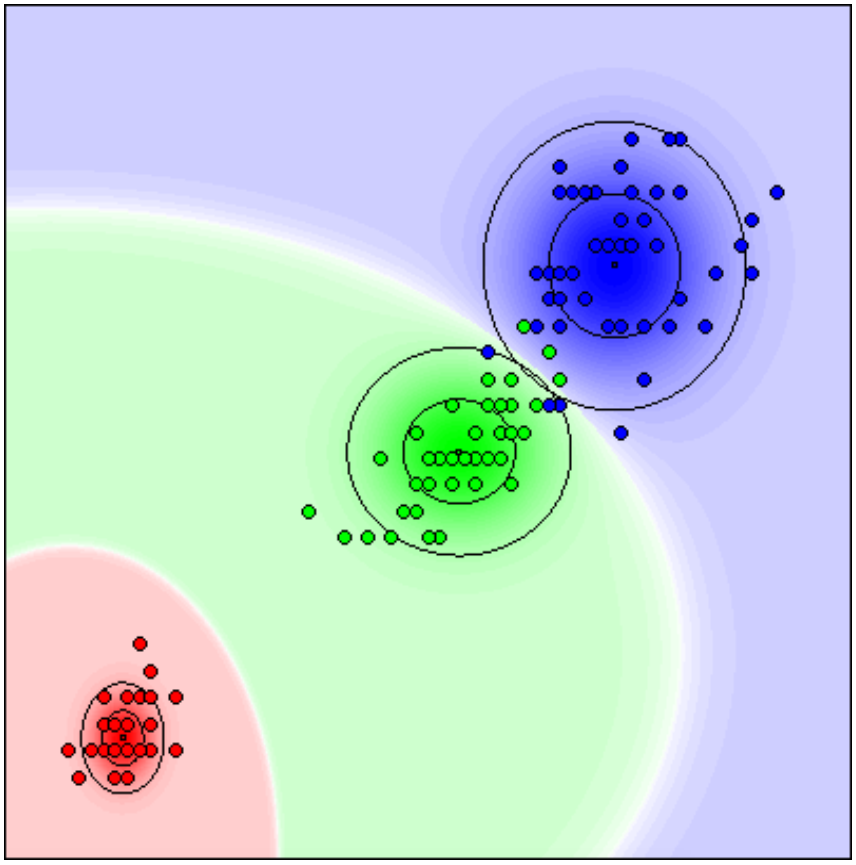


equal prior, spherical

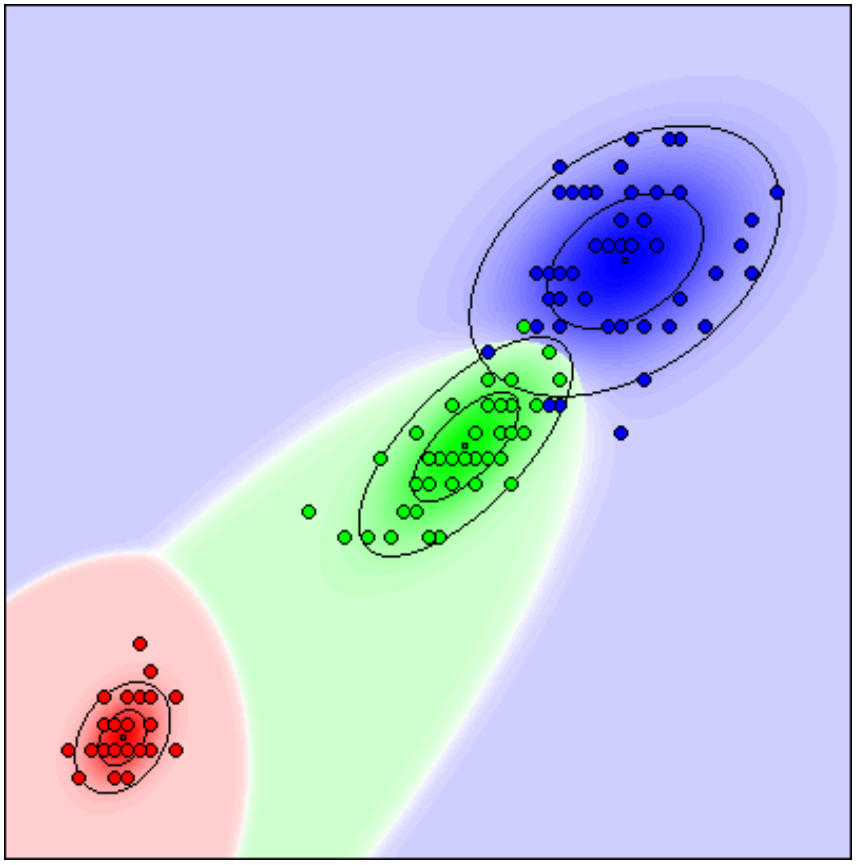


equal prior, ellipsoidal

Bayes Classifiers for the Iris Data

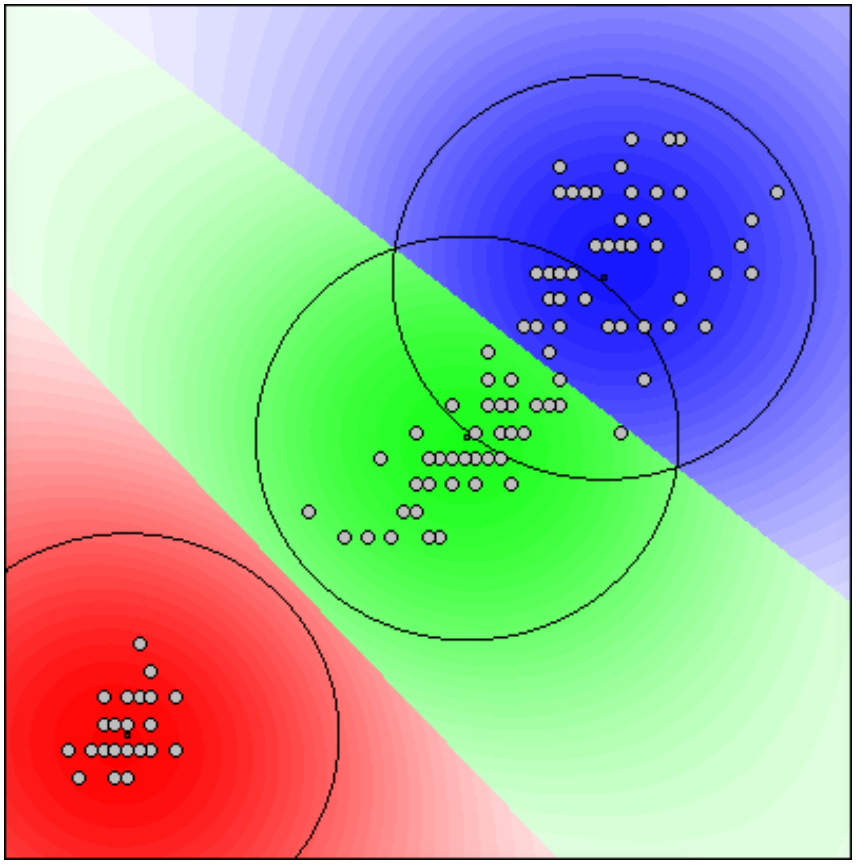


naive Bayes classifier

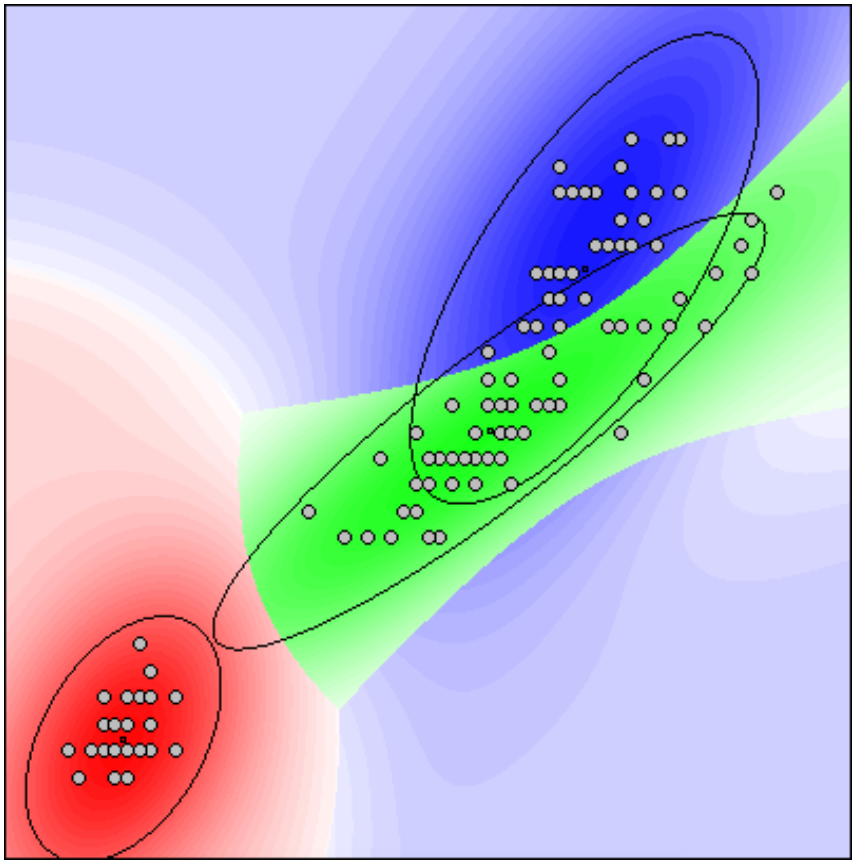


full Bayes classifier

Fuzzy Clustering of the Iris Data

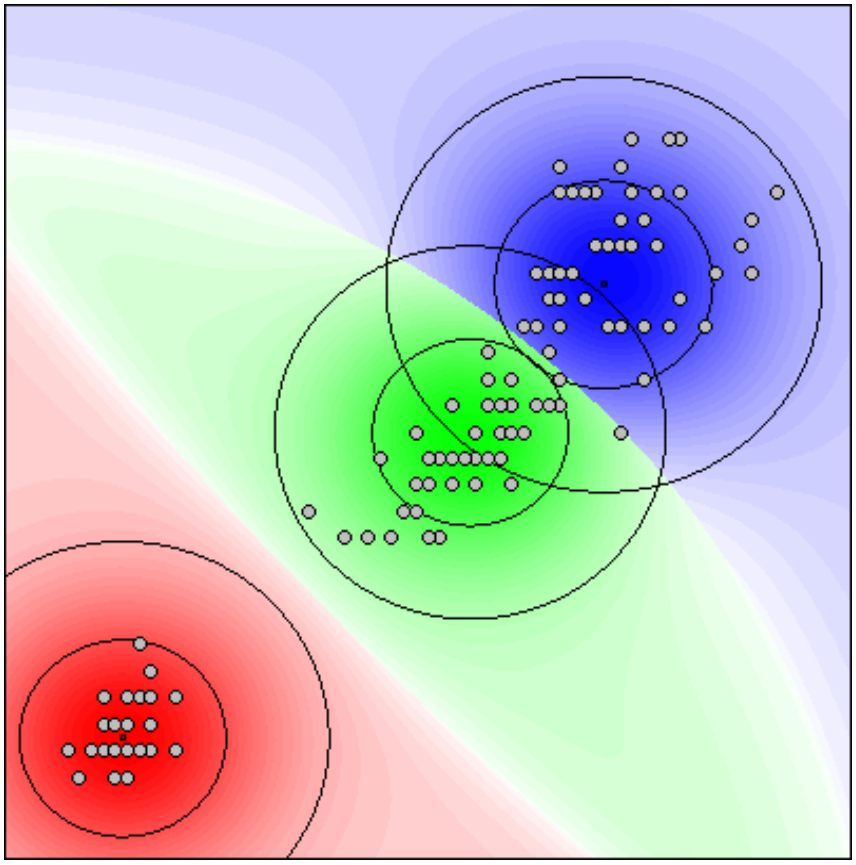


Fuzzy c-Means

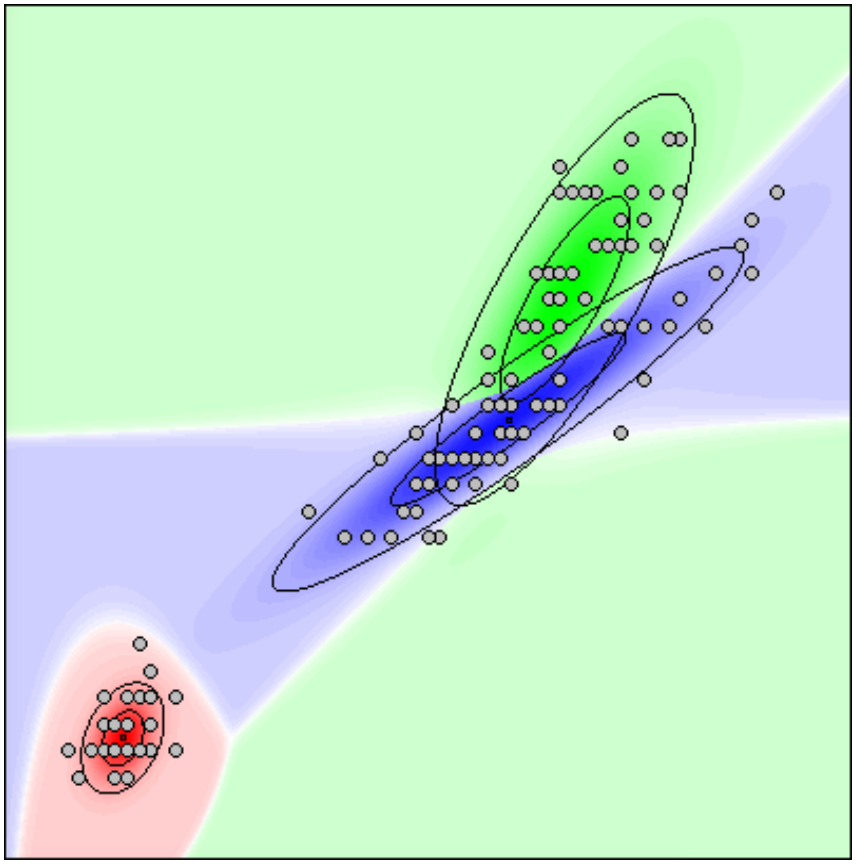


Gustafson–Kessel

Expectation Maximization of the Iris Data



equal prior, spherical



equal prior, ellipsoidal

Regularization: Constraining Shape and Size

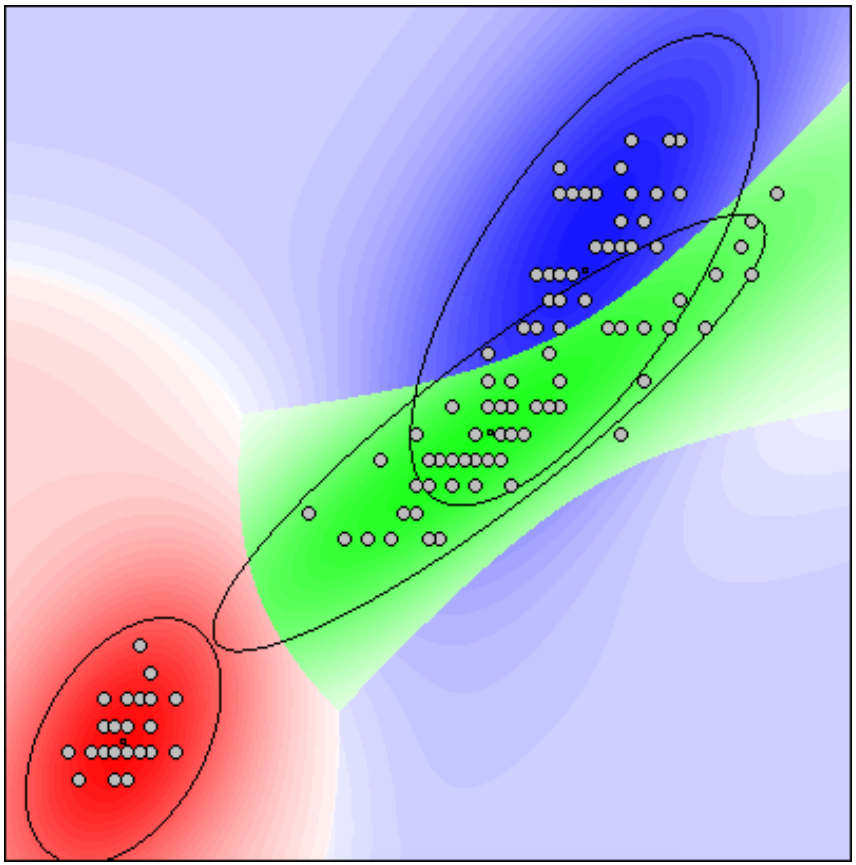
The more free parameters a cluster model has (size, shape, weight), the more difficult it becomes to estimate them robustly.

Possible problems:

- very long and thin ellisoids
- collapsing clusters

Possible solution:

- **Shape Regularization**
  "shift the eigenvalues"
- **Size Regularization**
  equalize the equiv. radius
- **Weight Regularization**
  equalize the data point weights



Shape Regularization: Eigenvalue Decomposition

- The shape of a cluster is described by its **covariance matrix**.
- Eigenvalue decomposition yields an **analog of standard deviation**.
- Let $\mathbf{S}$ be a symmetric, positive definite matrix (e.g. a covariance matrix).
  - $\mathbf{S}$ can be written as
    $$\mathbf{S} = \mathbf{R}\,\mathrm{diag}(\lambda_1, \ldots, \lambda_m)\,\mathbf{R}^{-1},$$
    where the λ_j, $j = 1, \ldots, m$, are the eigenvalues of $\mathbf{S}$ and the columns of $\mathbf{R}$ are the (normalized) eigenvectors of $\mathbf{S}$.
  - The eigenvalues λ_j, $j = 1, \ldots, m$, of $\mathbf{S}$ are all positive and the eigenvectors of $\mathbf{S}$ are orthonormal ($\rightarrow \mathbf{R}^{-1} = \mathbf{R}^\top$).
- Due to the above, $\mathbf{S}$ can be written as $\mathbf{S} = \mathbf{T}\,\mathbf{T}^\top$, where
  $$\mathbf{T} = \mathbf{R}\,\mathrm{diag}\left(\sqrt{\lambda_1}, \ldots, \sqrt{\lambda_m}\right)$$

Eigenvalue Decomposition

Special Case: Two Dimensions

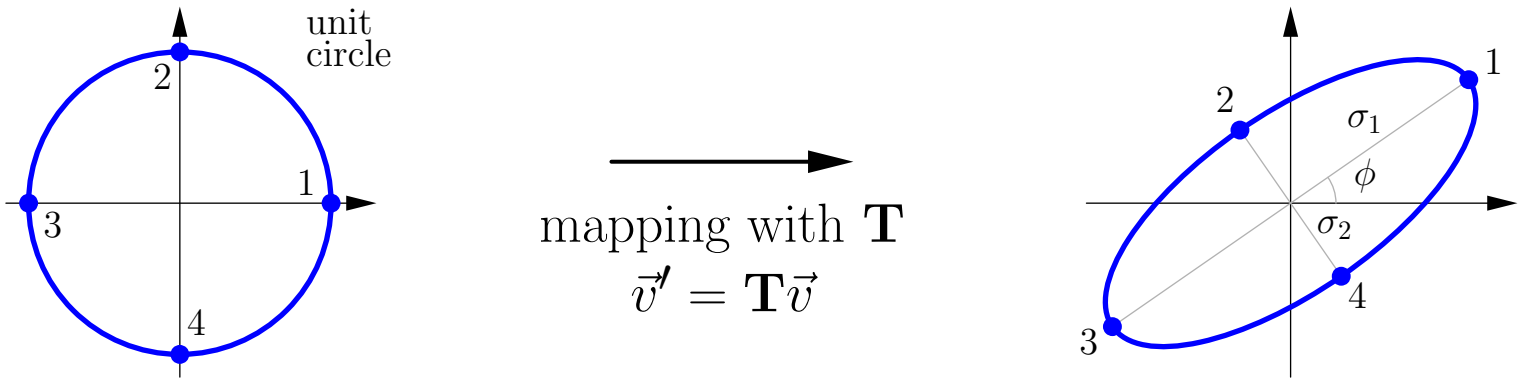
- Covariance matrix

$$\mathbf{\Sigma} = \begin{pmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{pmatrix}$$

- Eigenvalue decomposition

$$\mathbf{T} = \begin{pmatrix} c & -s \\ s & c \end{pmatrix} \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix},$$

$$s = \sin\phi, c = \cos\phi, \phi = \frac{1}{2}\arctan\frac{2\sigma_{xy}}{\sigma_x^2 - \sigma_y^2},$$
$$\sigma_1 = \sqrt{c^2\sigma_x^2 + s^2\sigma_y^2 + 2sc\sigma_{xy}},$$
$$\sigma_2 = \sqrt{s^2\sigma_x^2 + c^2\sigma_y^2 - 2sc\sigma_{xy}}.$$



Shape Regularization

Method 1: "Shift the eigenvalues" and renormalize.

The cluster-specific covariance matrices are adapted according to

$$\boldsymbol{\Sigma}_i^{(\mathrm{adap})} = \sigma_i^2 \cdot \frac{\mathbf{S}_i + h^2\mathbf{1}}{\sqrt[m]{|\mathbf{S}_i + h^2\mathbf{1}|}} = \sigma_i^2 \cdot \frac{\boldsymbol{\Sigma}_i + \sigma_i^2 h^2\mathbf{1}}{\sqrt[m]{|\boldsymbol{\Sigma}_i + \sigma_i^2 h^2\mathbf{1}|}}, \qquad \sigma_i^2 = \sqrt[m]{|\boldsymbol{\Sigma}_i|}, \quad \mathbf{S}_i = \frac{\boldsymbol{\Sigma}_i}{\sigma_i^2}.$$

The higher the value of h, the stronger the tendency towards spherical clusters.

Method 2: Constrain the axes ratio of the (hyper-)ellipsoid.

Let λ_k, $k = 1, \ldots m$, be the eigenvalues of the matrix $\boldsymbol{\Sigma}_i$. Set

$$h^2 = \begin{cases} 0, & \text{if } \dfrac{\max_{k=1}^{m}\lambda_k}{\min_{k=1}^{m}\lambda_k} \leq r^2, \\ \dfrac{\max_{k=1}^{m}\lambda_k - r^2 \min_{k=1}^{m}\lambda_k}{\sigma_i^2(r^2-1)}. & \text{otherwise,} \end{cases}$$

The value of r is the maximum accepted axes ratio of the (hyper-)ellipsoids.

Size Regularization

- The **size of a cluster** can be described in different ways, e.g., by its equivalent isotropic variance or its (hyper-)volume.

- Most of these measures can be specified by an exponent a of the **equivalent isotropic radius** of cluster i:

$$\sigma_i = \sqrt{\sigma_i^2} = \sqrt[2m]{|\boldsymbol{\Sigma}_i|}.$$

  That is, the size of cluster i is measured as σ_i^a, which means

$$\begin{aligned} a = 1: &\quad \text{equivalent isotropic radius,} \\ a = 2: &\quad \text{equivalent isotropic variance,} \\ a = m: &\quad \text{(hyper-)volume.} \end{aligned}$$

- In our approach a user has to specify a in order to state how he/she wants to measure the cluster size.

Size Regularization

Method 1: Bias the cluster size with a minimum (co-volume).

The equivalent isotropic radii σ_i are adapted according to

$$\sigma_i^{(\text{adap})} = \sqrt[a]{s \cdot \frac{\sum_{k=1}^{c} \sigma_k^a}{\sum_{k=1}^{c} (\sigma_k^a + b)} \cdot (\sigma_i^a + b)} = \sqrt[a]{s \cdot \frac{\sum_{k=1}^{c} \sigma_k^a}{cb + \sum_{k=1}^{c} \sigma_k^a} \cdot (\sigma_i^a + b)}.$$

Method 2: No renormalization, only scaling. (slightly more efficient)

The equivalent isotropic radii σ_i are adapted according to

$$\sigma_i^{(\text{adap})} = \sqrt[a]{s \cdot (\sigma_i^a + b)}.$$

Method 3: Constrain the size ratio of the clusters.

$$b = \begin{cases} 0, & \text{if } \frac{\max_{k=1}^{c} \sigma_k^a}{\min_{k=1}^{c} \sigma_k^a} \leq r, \\ \frac{\max_{k=1}^{c} \sigma_k^a - r \min_{k=1}^{c} \sigma_k^a}{r-1}, & \text{otherwise.} \end{cases}$$

Weight Regularization

Method 1: Bias the cluster weight with a minimum.

The cluster weights θ_i are adapted according to

$$\theta_i^{(\text{adap})} = \frac{\sum_{k=1}^{c} \theta_k}{\sum_{k=1}^{c} (\theta_k + b)} \cdot (\theta_i + b) = \frac{\sum_{k=1}^{c} \theta_k}{cb + \sum_{k=1}^{c} \theta_k} \cdot (\theta_i + b),$$

with a user-specified bias b.

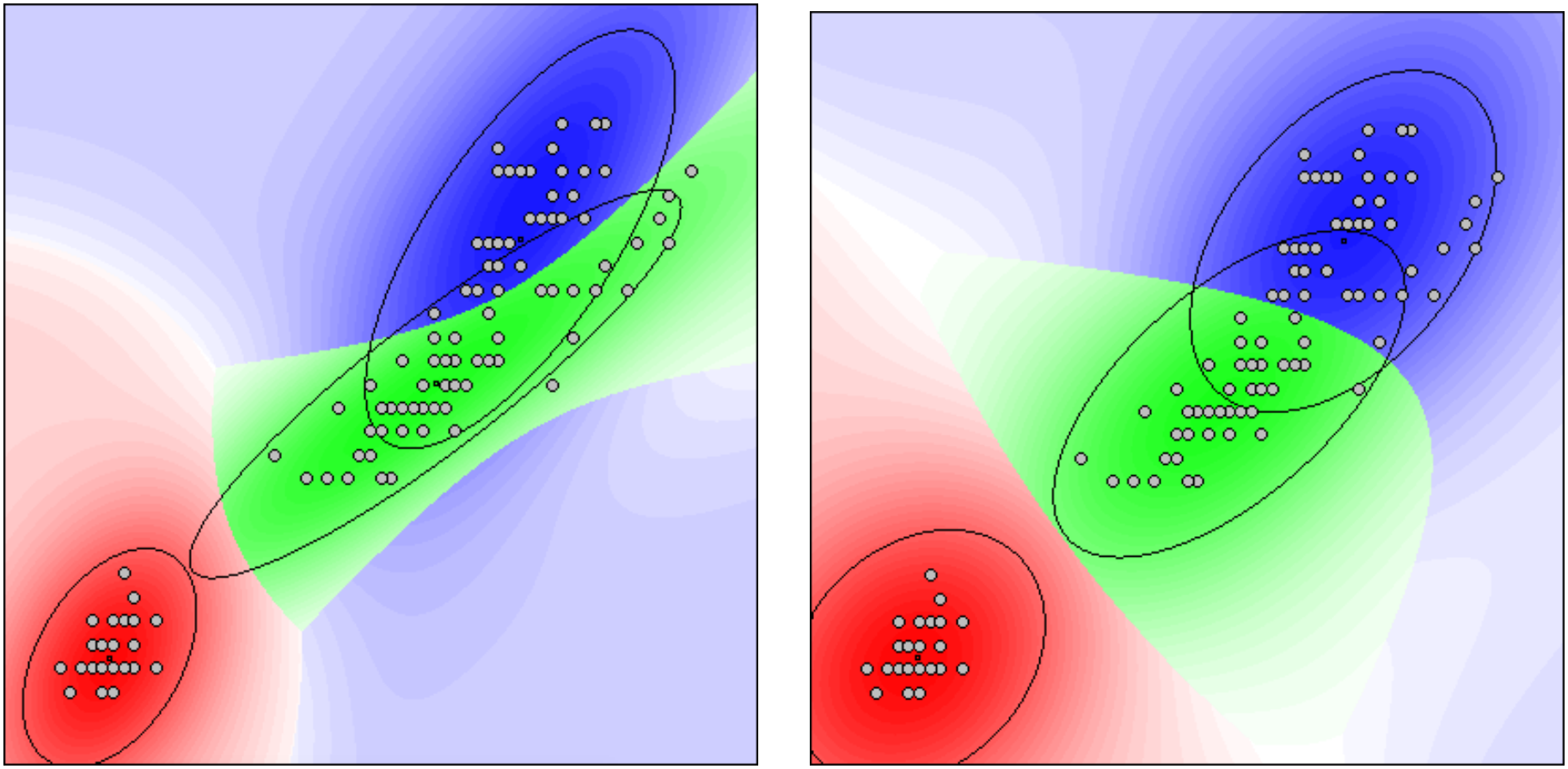
Method 2: Constrain the maximum weight ratio.

The value of the regularization parameter b is computed as

$$b = \begin{cases} 0, & \text{if } \dfrac{\max_{k=1}^{c} \theta_k}{\min_{k=1}^{c} \theta_k} \leq r, \\ \dfrac{\max_{k=1}^{c} \theta_k - r \min_{k=1}^{c} \theta_k}{r - 1}, & \text{otherwise,} \end{cases}$$

with a user-specified maximum weight ratio r.

Shape Regularization: Iris Data



Gustafson–Kessel clustering without and with regularization (method 2, $r = 4$)

Finding the Number of Clusters

- How to find the "best" number of clusters is a pressing problem in clustering, since prototype-based algorithms require the number of clusters as user input.
- **Most common approach**
  - Cluster the data set several times, each time with a different number of clusters from a user defined range, and
  - evaluate the results by *internal cluster evaluation measures* (e.g., partition entropy, Xie-Beni index, Fukuyama-Sugeno index, etc.).

  However, this approach is often unreliable.
- **Alternative approach**
  - Cluster several samples with the same number of clusters and
  - compare the results with *relative cluster evaluation measures* in order to assess the variability of the results.

  This approach has been studied fairly intensely in crisp clustering.

Dunn Index / Separation Index

Actually a family of indices, to be maximized, generally defined as

$$Q_{\text{Dunn}}(\mathbf{B}, \mathbf{U}, \mathbf{X}) = \frac{\min_{1 \leq i < k \leq c} d(\beta_i, \beta_k; \mathbf{U}, \mathbf{X})}{\max_{1 \leq i \leq c} S(\beta_i; \mathbf{U}, \mathbf{X})}$$

where $d(\beta_i, \beta_j; \mathbf{U}, \mathbf{X})$ is a cluster distance measure
and $S(\beta_l; \mathbf{U}, \mathbf{X})$ is a measure for the diameter of a cluster.

[Dunn 1973] originally used for the cluster distance measure

$$d(\beta_i, \beta_k; \mathbf{U}, \mathbf{X}) = \min_{1 \leq j,l \leq n: u_{ij} u_{kl} = 1} d(\vec{x}_j, \vec{x}_l),$$

that is, the smallest distance between two data points, one from each cluster, and for the diameter of a cluster

$$S(\beta_i; \mathbf{U}, \mathbf{X}) = \max_{1 \leq j,l \leq n: u_{ij} u_{il} = 1} d(\vec{x}_j, \vec{x}_l),$$

that is, as the largest distance between two data points from the cluster.

Dunn Index / Separation Index

Actually a family of indices, to be maximized, generally defined as

$$Q_{\text{Dunn}}(\mathbf{B}, \mathbf{U}, \mathbf{X}) = \frac{\min_{1 \leq i < k \leq c} d(\beta_i, \beta_k; \mathbf{U}, \mathbf{X})}{\max_{1 \leq i \leq c} S(\beta_i; \mathbf{U}, \mathbf{X})}$$

A better, that is, much more robust version was suggested by [Bezdek *et al.*1997]:

$$d(\beta_i, \beta_k; \mathbf{U}, \mathbf{X}) = \frac{1}{\left(\sum_{j=1}^{n} u_{ij}\right)\left(\sum_{j=1}^{n} u_{kj}\right)} \sum_{j=1}^{n} \sum_{l=1}^{n} u_{ij} u_{kl}\, d(\vec{x}_j, \vec{x}_l),$$

that is, as the average distance between two data points, one from each cluster.
In a matching fashion, the diameter of a cluster is defined as

$$S(\beta_i; \mathbf{U}, \mathbf{X}) = 2\, \frac{\sum_{j=1}^{n} u_{ij}\, d(\vec{x}_j, \vec{\mu}_i)}{\sum_{j=1}^{n} u_{ij}},$$

i.e., as twice the arithmetic mean of the data point distance from the cluster center, which may be seen as a kind of cluster radius.

Davies–Bouldin Index

Actually a family of indices, to be minimized, generally defined as

$$Q_{\mathrm{DB}}(\mathbf{B}, \mathbf{U}, \mathbf{X}) = \frac{1}{c} \sum_{i=1}^{c} \max_{\substack{1 \leq k \leq c \\ k \neq i}} \frac{S(\beta_i; \mathbf{U}, \mathbf{X}) + S(\beta_k; \mathbf{U}, \mathbf{X})}{d(\beta_i, \beta_k; \mathbf{U}, \mathbf{X})}$$

where $d(\beta_i, \beta_j; \mathbf{U}, \mathbf{X})$ is a cluster distance measure
and $S(\beta_l; \mathbf{U}, \mathbf{X})$ is a measure for the scatter within a cluster.

[Davies and Bouldin 1979] originally used for the cluster distance

$$d(\beta_i, \beta_k; \mathbf{U}, \mathbf{X}) = d(\vec{\mu}_i, \vec{\mu}_k) = \sqrt{(\vec{\mu}_i - \vec{\mu}_k)^\top (\vec{\mu}_i - \vec{\mu}_k)},$$

that is, the Euclidean distance of the cluster centers,
and for the scatter within a cluster

$$S(\beta_i; \mathbf{U}, \mathbf{X}) = \sqrt{\textstyle\sum_{j=1}^{n} u_{ij}\, d^2(\vec{x}_j, \vec{\mu}_i)} \;\; / \;\; \sqrt{\textstyle\sum_{j=1}^{n} u_{ij}},$$

that is, the quadratic mean of the data points distances from the cluster center.

Xie-Beni Index / Separation

The *Xie–Beni index* or *separation*, to be minimized, is defined as

$$Q_{\text{sep}}(\mathbf{B}, \mathbf{U}, \mathbf{X}) = \frac{\frac{1}{n} \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^{w} \, d^2(\vec{x}_j, \vec{\mu}_i)}{\min_{1 \leq i < k \leq n} d^2(\vec{\mu}_i, \vec{\mu}_k)}.$$

- Numerator contains the standard objective function of fuzzy clustering.
- Divided by n in order to remove a dependence on the number of data points.
- The result can be seen as an average (weighted) distance of the data points from the cluster centers.
- This can also be interpreted as a global / average scatter within clusters.
- Minimum distance between two cluster centers in denominator indicates how well the clusters are separated.

Fukuyama–Sugeno Index

The *Fukuyama–Sugeno index*, to be minimized, is defined as

$$\begin{aligned} Q_{\mathrm{FS}}(\mathbf{B}, \mathbf{U}, \mathbf{X}) &= \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^{w} \left(d^2(\vec{x}_j, \vec{\mu}_i) - d^2(\vec{\mu}_i, \vec{\mu}) \right) \\ &= \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^{w} \, d^2(\vec{x}_j, \vec{\mu}_i) - \sum_{i=1}^{c} d^2(\vec{\mu}_i, \vec{\mu}) \sum_{j=1}^{n} u_{ij}^{w}, \end{aligned}$$

where $\vec{\mu}$ is the global mean of the data points, that is,

$$\vec{\mu} = \frac{1}{n} \sum_{j=1}^{n} \vec{x}_j.$$

- Contains the standard objective function of fuzzy clustering.
- Modifies it by relating every single distance to a cluster center to the distance of this cluster center to the global mean.

Other Internal Cluster Evaluation Measures

- **Objective Function**
- **Partition Coefficient**
- **Partition / Classification Entropy**
- **Fuzzy Hyper-Volume**
- **Partition Density**
- and many more

Many of these internal cluster evaluation measures

- are not very robust (can change heavily with little change of the data)
- and yield unintuitive results for certain data sets.

Relative Cluster Evaluation Measures

- **Describing Clustering Results**

  A clustering is described by a $c \times n$ partition matrix $\mathbf{U} = (u_{ij})_{1 \leq i \leq c, 1 \leq j \leq n}$,

  where c is the number of clusters, n is the number of data points, and $u_{ij} \in [0, 1]$ is the degree of membership of the j-th data point to the i-th cluster.

- **Comparing Clustering Results**

  Given: two partition matrices $\mathbf{U}^{(1)}$ and $\mathbf{U}^{(2)}$.
  Desired: a measure of the similarity of the clustering results.

  - *Comparing Partition Matrices*

    Compare the partition matrices directly, usually row by row.

  - *Comparing Coincidence Matrices*

    Compute from each partition matrix a coincidence matrix $\mathbf{\Psi} = (\psi_{jl})_{1 \leq j,l \leq n}$ with $\psi_{jl} = \sum_{i=1}^{c} u_{ij} u_{il}$ and compare these.

    (The product of the membership degrees may be replaced by any t-norm.)

Comparing Partition Matrices

- Evaluation measures are based on **comparing rows (indiv. clusters)**. For each pair $(i,k) \in \{1,\ldots,c\}^2$ we compute

$$
\begin{aligned}
n_{11}^{(i,k)}(\mathbf{U}^{(1)},\mathbf{U}^{(2)}) &= \sum_{j=1}^{n} u_{ij}^{(1)} \cdot u_{kj}^{(2)}, \\
n_{01}^{(i,k)}(\mathbf{U}^{(1)},\mathbf{U}^{(2)}) &= \sum_{j=1}^{n} \left(1 - u_{ij}^{(1)}\right) \cdot u_{kj}^{(2)}, \\
n_{10}^{(i,k)}(\mathbf{U}^{(1)},\mathbf{U}^{(2)}) &= \sum_{j=1}^{n} u_{ij}^{(1)} \cdot \left(1 - u_{kj}^{(2)}\right), \\
n_{00}^{(i,k)}(\mathbf{U}^{(1)},\mathbf{U}^{(2)}) &= \sum_{j=1}^{n} \left(1 - u_{ij}^{(1)}\right) \cdot \left(1 - u_{kj}^{(2)}\right).
\end{aligned}
$$

- Since rows may be permuted, the **best permutation has to be found**. $O(c^3)$ using the *Hungarian method* for weighted bipartite matching problems. $\Pi(c)$ denotes the set of all permutations of the c numbers $1,\ldots,c$.

Comparing Partition Matrices

- **$\mathbf{F_1}$ measure**

$$F_1(\mathbf{U}^{(1)}, \mathbf{U}^{(2)}) = \max_{\varsigma \in \Pi(c)} \frac{1}{c} \sum_{i=1}^{c} \frac{2\,\pi_{i,\varsigma(i)} \rho_{i,\varsigma(i)}}{\pi_{i,\varsigma(i)} + \rho_{i,\varsigma(i)}}, \qquad \text{where}$$

$$\pi_{i,k} = \frac{n_{11}^{(i,k)}}{n_{01}^{(i,k)} + n_{11}^{(i,k)}} \quad \text{(precision)} \quad \text{and} \qquad \rho_{i,k} = \frac{n_{11}^{(i,k)}}{n_{10}^{(i,k)} + n_{11}^{(i,k)}} \quad \text{(recall)}.$$

- **(Cross-classification) Accuracy**

$$Q_{\text{acc}}(\mathbf{U}^{(1)}, \mathbf{U}^{(2)}) = \max_{\varsigma \in \Pi(c)} \frac{1}{cn} \sum_{i=1}^{c} \left(n_{00}^{(i,\varsigma(i))} + n_{11}^{(i,\varsigma(i))} \right).$$

- **Squared Difference of Membership Degrees**

$$Q_{\text{diff}}(\mathbf{U}^{(1)}, \mathbf{U}^{(2)}) = \min_{\varsigma \in \Pi(c)} \frac{1}{cn} \sum_{i=1}^{c} \sum_{j=1}^{n} \left(u_{ij}^{(1)} - u_{\varsigma(i)j}^{(2)} \right)^2.$$

Comparing Coincidence Matrices

- Compute from each partition matrix a coincidence matrix $\mathbf{\Psi} = (\psi_{jl})_{1 \leq j,l \leq n}$ with $\psi_{jl} = \sum_{i=1}^{c} u_{ij} u_{il}$. Then compute

$$
\begin{aligned}
N_{SS}(\mathbf{\Psi}^{(1)}, \mathbf{\Psi}^{(2)}) &= \sum_{j=2}^{n} \sum_{l=1}^{j-1} \psi_{jl}^{(1)} \psi_{jl}^{(2)}, \\
N_{SD}(\mathbf{\Psi}^{(1)}, \mathbf{\Psi}^{(2)}) &= \sum_{j=2}^{n} \sum_{l=1}^{j-1} \psi_{jl}^{(1)} \left(1 - \psi_{jl}^{(2)}\right), \\
N_{DS}(\mathbf{\Psi}^{(1)}, \mathbf{\Psi}^{(2)}) &= \sum_{j=2}^{n} \sum_{l=1}^{j-1} \left(1 - \psi_{jl}^{(1)}\right) \psi_{jl}^{(2)}, \\
N_{DD}(\mathbf{\Psi}^{(1)}, \mathbf{\Psi}^{(2)}) &= \sum_{j=2}^{n} \sum_{l=1}^{j-1} \left(1 - \psi_{jl}^{(1)}\right) \left(1 - \psi_{jl}^{(2)}\right),
\end{aligned}
$$

- All products (in the elements of the coincidence matrix as well as in the above quantities) may be replaced by any t-norm as they describe conjunctions.

Comparing Coincidence Matrices

- **Rand Statistic**

$$Q_{\text{Rand}}(\mathbf{\Psi}^{(1)}, \mathbf{\Psi}^{(2)}) = \frac{N_{SS} + N_{DD}}{N_{..}},$$

- **Jaccard Coefficient**

$$Q_{\text{Jaccard}}(\mathbf{\Psi}^{(1)}, \mathbf{\Psi}^{(2)}) = \frac{N_{SS}}{N_{SS} + N_{SD} + N_{DS}},$$

- **Folkes–Mallows Index**

$$Q_{\text{FM}}(\mathbf{\Psi}^{(1)}, \mathbf{\Psi}^{(2)}) = \frac{N_{SS}}{\sqrt{(N_{SS} + N_{SD})(N_{SS} + N_{DS})}},$$

- **Hubert Index**

$$Q_{\text{Hubert}}(\mathbf{\Psi}^{(1)}, \mathbf{\Psi}^{(2)}) = \frac{N_{..}N_{SS} - N_{S.}N_{.S}}{\sqrt{N_{S.}N_{.S}N_{D.}N_{.D}}},$$

Resampling Approaches: General Idea

- Resampling can be seen as a special **Monte Carlo method** (that is, a method for finding solutions to mathematical and statistical problems by (stochastic) simulation).
- A cluster model can usually be applied as a **classifier** and thus data points that have not been used to build the cluster model can be assigned to clusters.
- Hence **two different groupings of the same data set** can be obtained (e.g., one by clustering, the other by applying a cluster model as a classifier).
- By repeated comparisons of clustering results derived from several different samples drawn from the original data set, one can obtain an **assessment of the variability of the cluster structure**.
- It is plausible that the number of clusters that leads to the **least variability** is the "correct" or "best" number of clusters.

Resampling Approaches: Procedures

- **Bootstrapping**
  - Draw each sample with replacement from the given data set.
    (The same data point may appear multiple times in the sample.)
- **Subsampling**
  - Draw each sample without replacement from the given data set.
    (Do not allow duplicate occurrence of the same data point.)
  - Standard procedure: split data set into two subsets of roughly equal size.
- **Variability Evaluation**
  - Compare the clustering results on two samples, using the result on one as a classifier for the other (but only one way, not both).
  - Compare the clustering result on a sample with the result on the whole data set (using the result on the sample as a classifier).
  - Average over a sufficiently large number of comparisons.

Summary Fuzzy and Probabilistic Clustering

- **Prototype-based clustering**
  - Cluster center
  - Cluster shape and size parameters
  - Cluster weight / prior probability
- **Alternating optimization / estimation / adaptation**
  - Fix cluster parameters and optimize data point assignment.
  - Fix data point assignment and optimize cluster parameters.
  - Iterate until convergence.
- Crisp or fuzzy/probabilistic assignment of a datum to a cluster.
  - Fuzzy/probabilistic approaches are usually more robust.
- Local minima of the objective function can pose a problem.

Software

Some software for clustering can be found at

- `http://www.borgelt.net/cluster.html`
  Command line programs for fuzzy and probabilistic clustering
- `http://www.borgelt.net/clgui.html`
  Java-based graphical user interface for these programs
- `http://www.borgelt.net/bcview.html`
  Visualization program for Bayes classifiers and cluster sets
- `http://www.borgelt.net/ptless.html`
  Command line programs for prototype-less fuzzy clustering

Other software for data analysis can be found at

- `http://www.borgelt.net/software.html`