Clustering

Clustering

General Idea of Clustering

- Similarity and distance measures

Prototype-based Clustering

- Classical c-means clustering
- Learning vector quantization
- Fuzzy c-means clustering
- Expectation maximization for Gaussian mixtures

Hierarchical Agglomerative Clustering

- Merging clusters: Dendrograms
- Measuring the distance of clusters
- Choosing the clusters

Density Based Clustering

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 $\forall \vec{x}, \vec{y}, \vec{z} \in \mathbb{R}^m$:

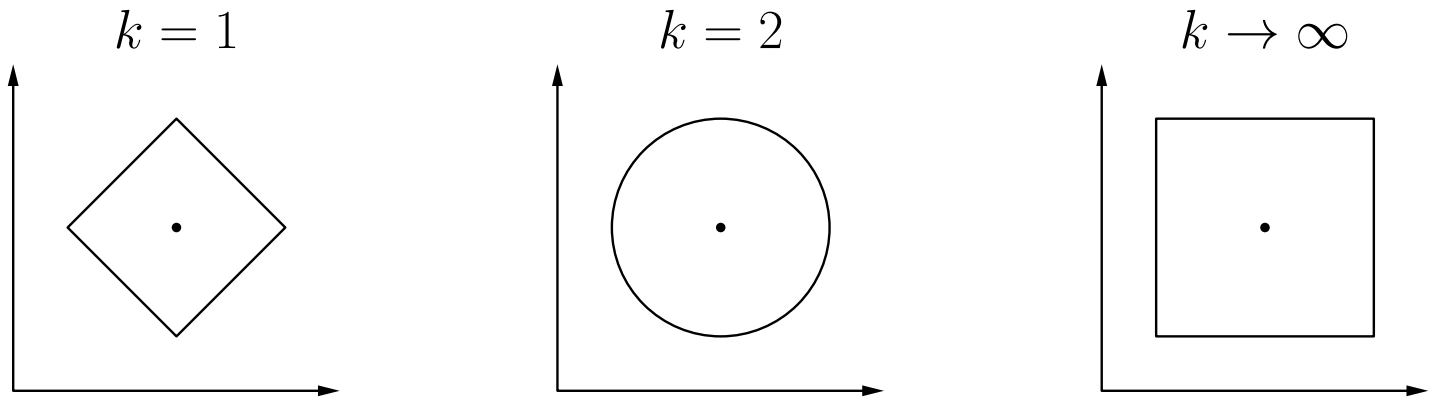
$$
\begin{array}{rll}
(i) & d(\vec{x}, \vec{y}) = 0 \quad \Leftrightarrow \quad \vec{x} = \vec{y}, & \\
(ii) & d(\vec{x}, \vec{y}) = d(\vec{y}, \vec{x}) & \text{(symmetry)}, \\
(iii) & d(\vec{x}, \vec{z}) \leq d(\vec{x}, \vec{y}) + d(\vec{y}, \vec{z}) & \text{(triangle inequality)}.
\end{array}
$$

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



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).

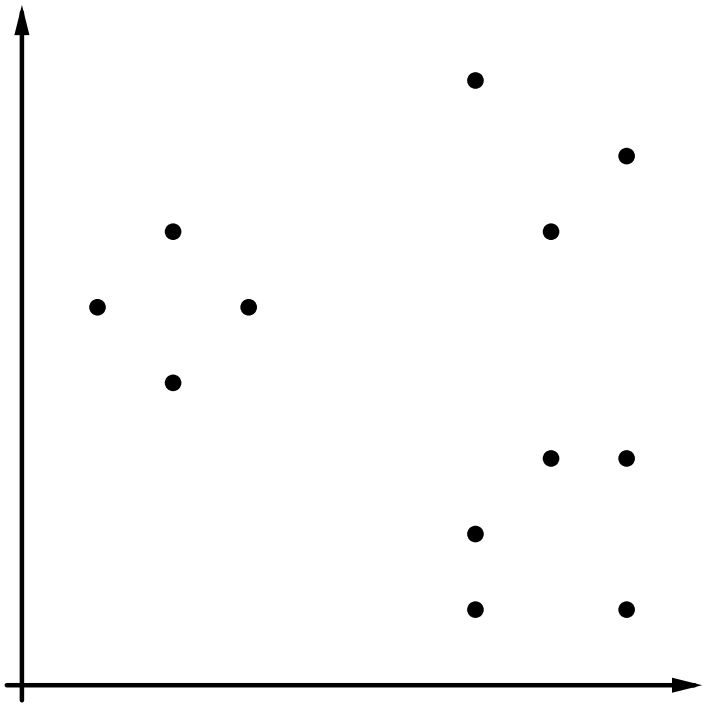
Data point assignment:
Assign each data point to the cluster center that is closest to it
(i.e. closer than any other cluster center).

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.

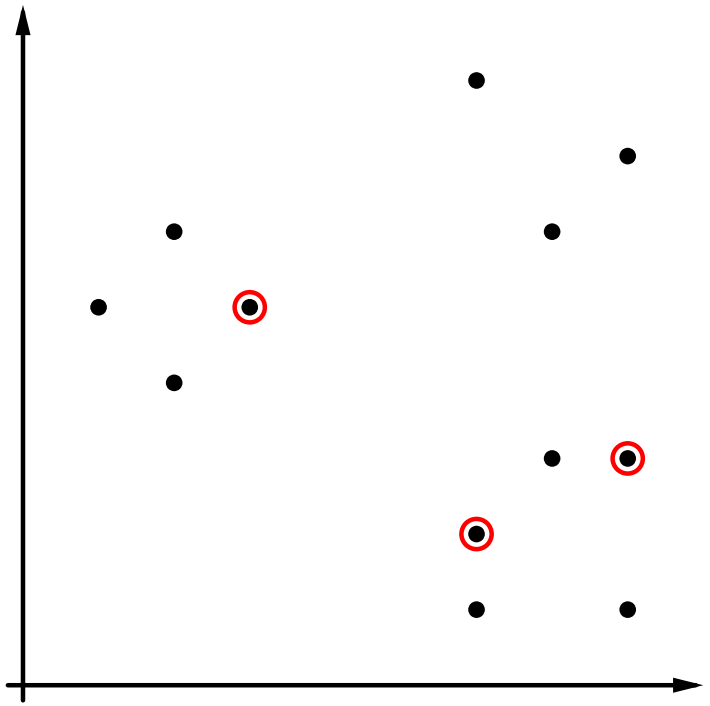
It can be shown that this scheme must converge,
i.e., the update of the cluster centers cannot go on forever.

c-Means Clustering: Example



Data set to cluster.

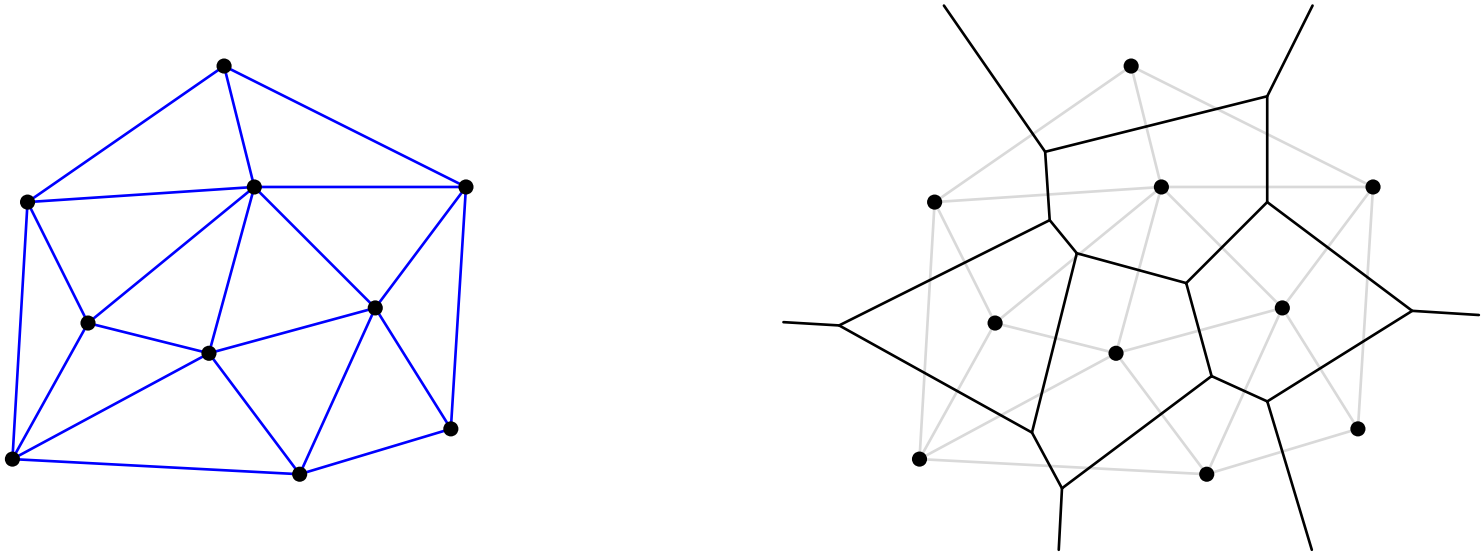
Choose $c = 3$ clusters.
(From visual inspection, can be difficult to determine in general.)



Initial position of cluster centers.

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

Delaunay Triangulations and Voronoi Diagrams



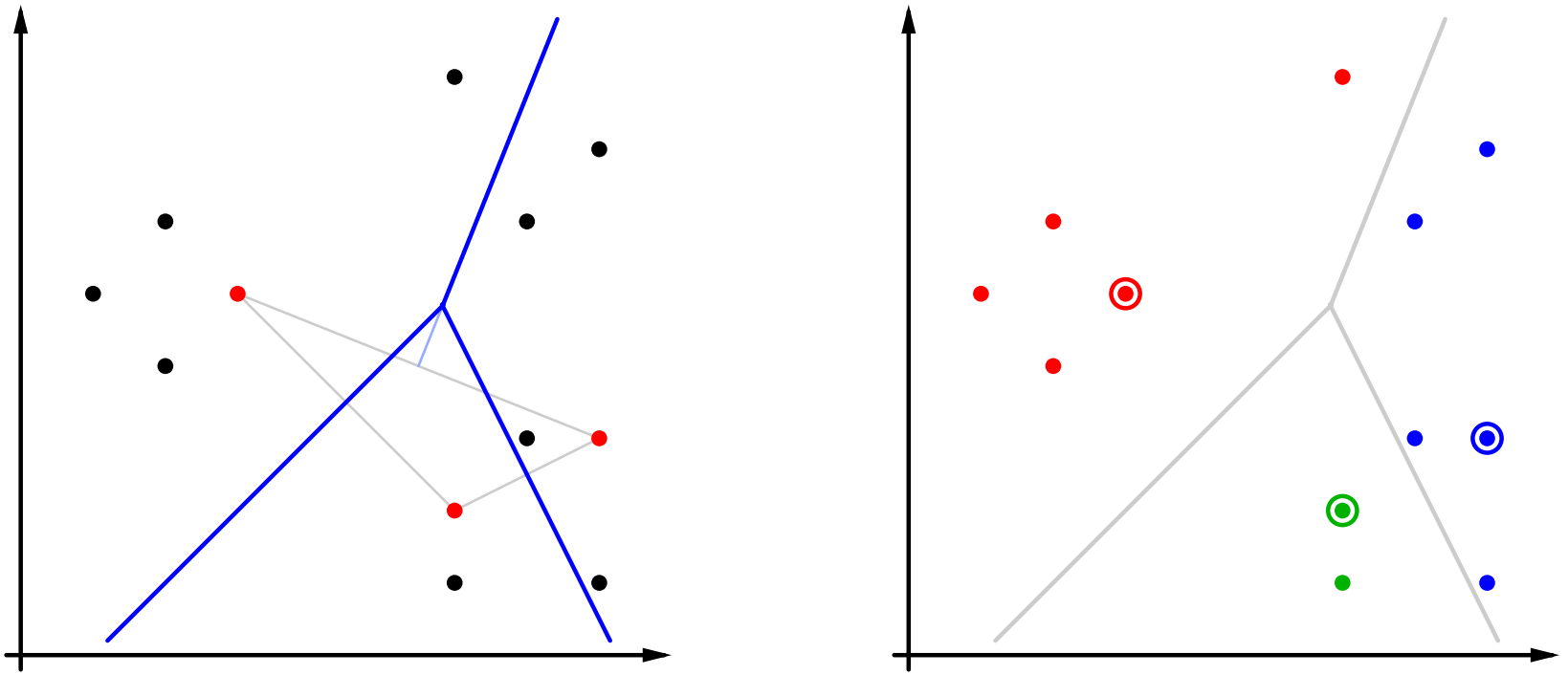
Dots represent cluster centers (quantization vectors).

Left: **Delaunay Triangulation**
(The circle through the corners of a triangle does not contain another point.)

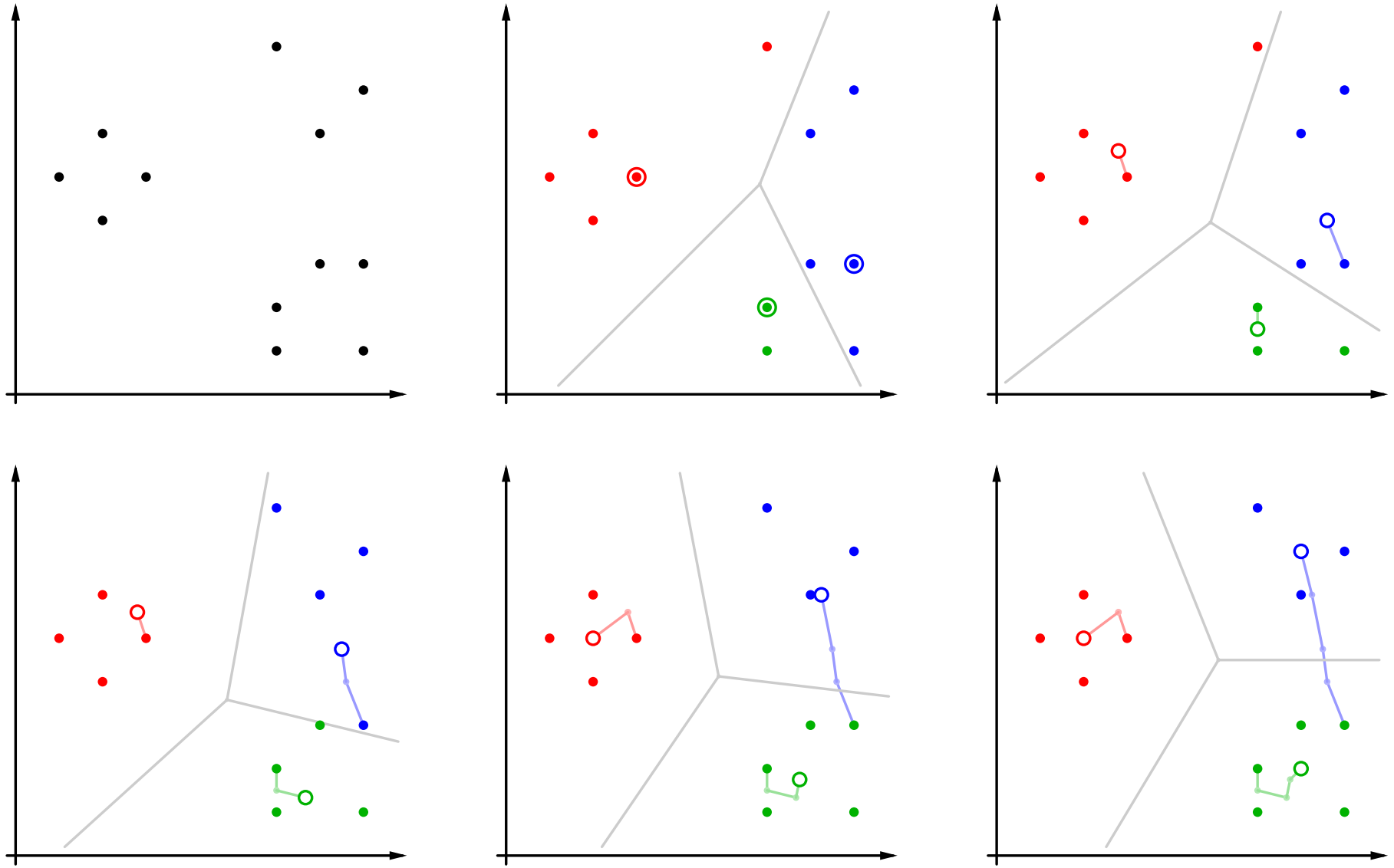
Right: **Voronoi Diagram**
(Midperpendiculars of the Delaunay triangulation: boundaries of the regions of points that are closest to the enclosed cluster center (Voronoi cells)).

Delaunay 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.)

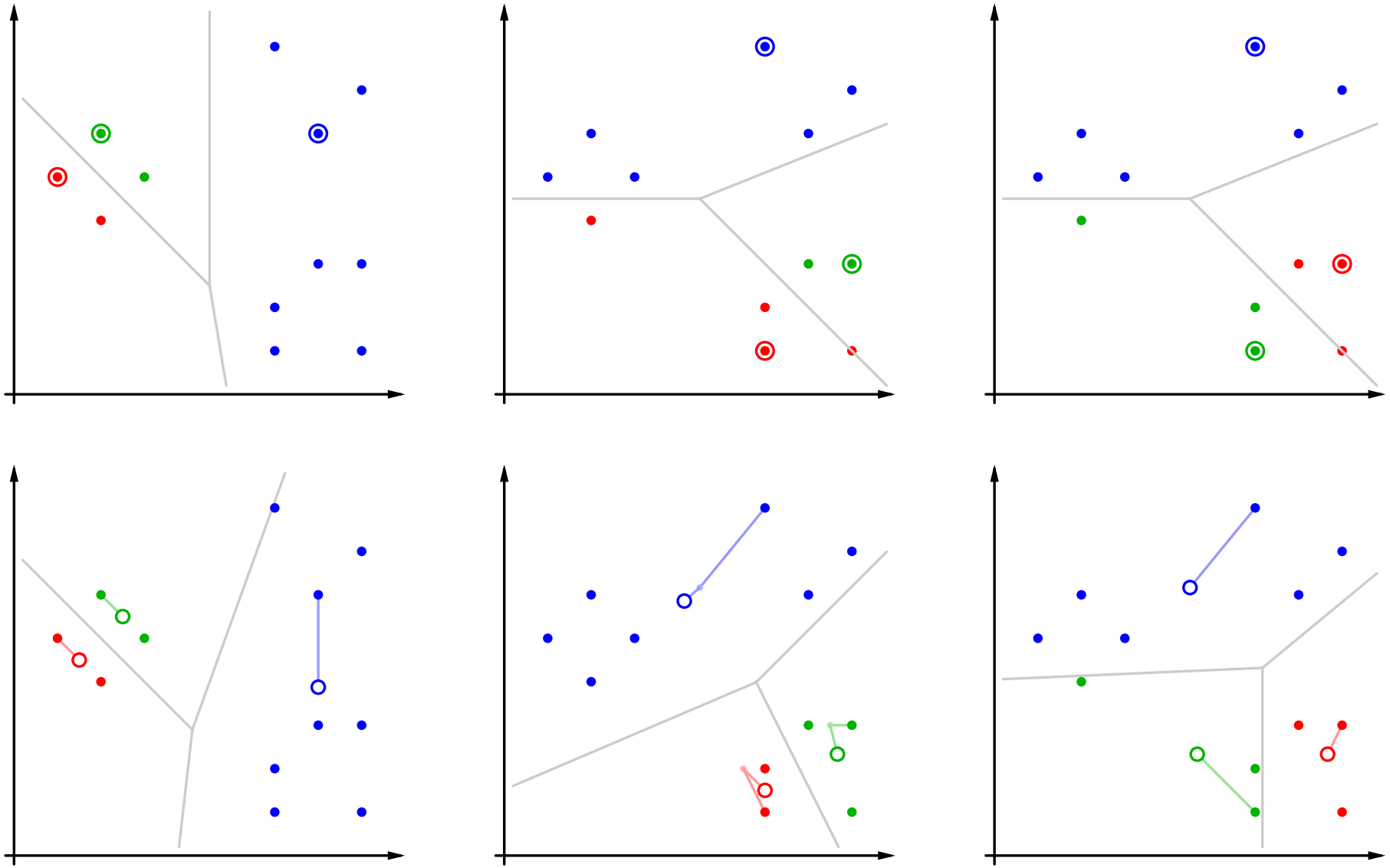
However: The clustering result is fairly **sensitive to the initial positions** of the cluster centers.

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



Learning Vector Quantization

Adaptation of reference vectors / codebook vectors

Like "online" c-means clustering (update after each data point).

For each training pattern find the closest reference vector.

Adapt only this reference vector (winner neuron).

For classified data the class may be taken into account.
(reference vectors are assigned to classes)

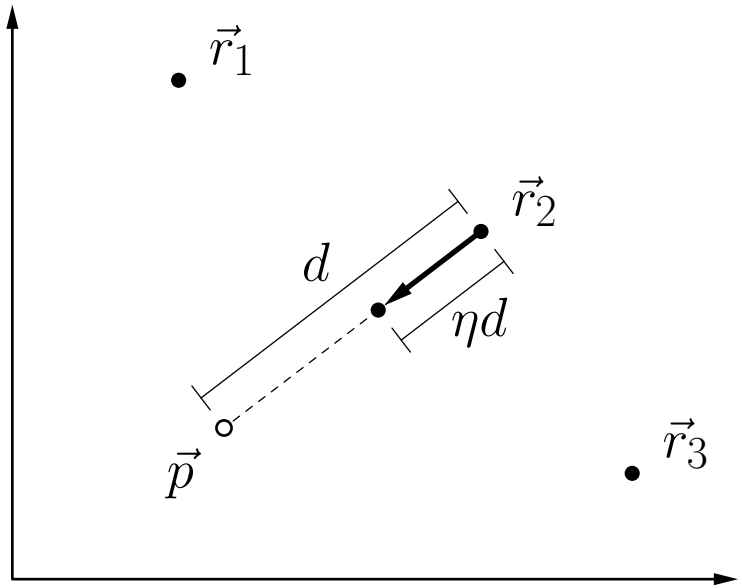
Attraction rule (data point and reference vector have same class)

$$\vec{r}^{\,(\text{new})} = \vec{r}^{\,(\text{old})} + \eta\Big(\vec{p} - \vec{r}^{\,(\text{old})}\Big),$$

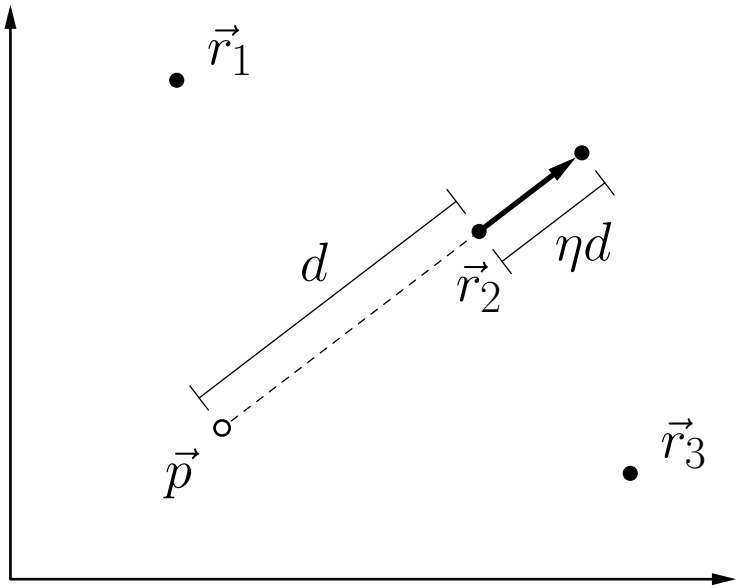
Repulsion rule (data point and reference vector have different class)

$$\vec{r}^{\,(\text{new})} = \vec{r}^{\,(\text{old})} - \eta\Big(\vec{p} - \vec{r}^{\,(\text{old})}\Big).$$

Adaptation of reference vectors / codebook vectors



attraction rule



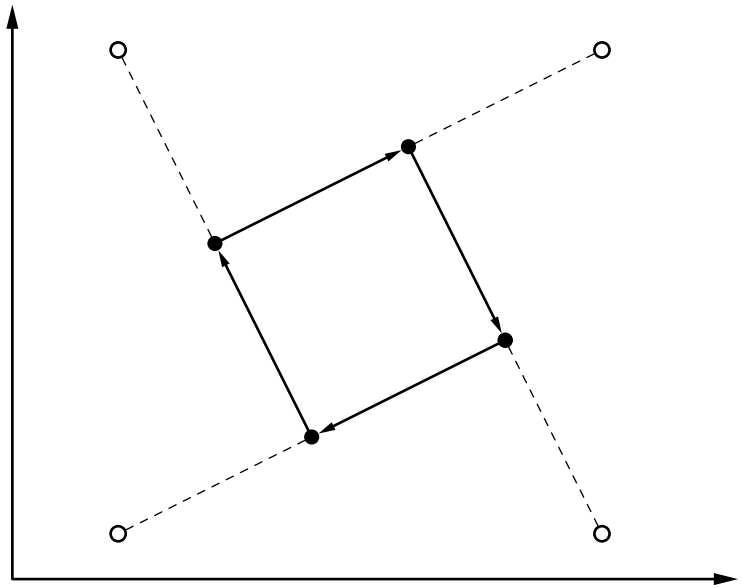
repulsion rule

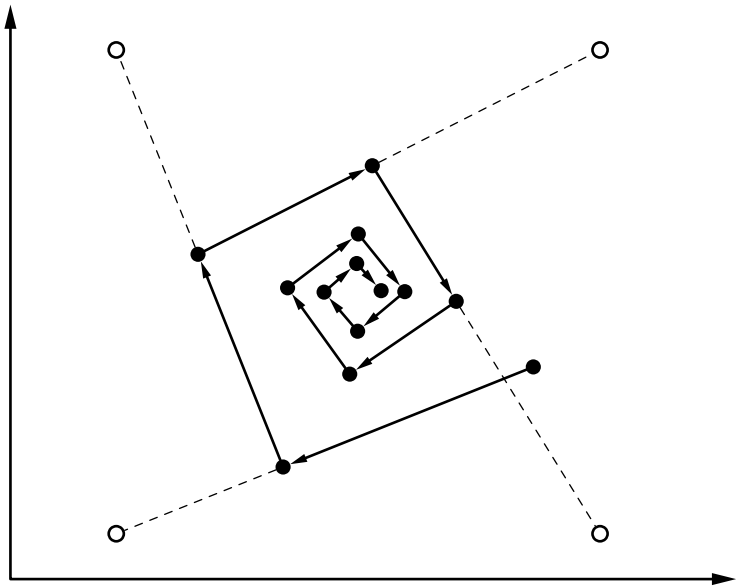
$\vec{p}$: data point, $\vec{r}_i$: reference vector

$\eta = 0.4$ (learning rate)

Learning Vector Quantization: Learning Rate Decay

Problem: fixed learning rate can lead to oscillations



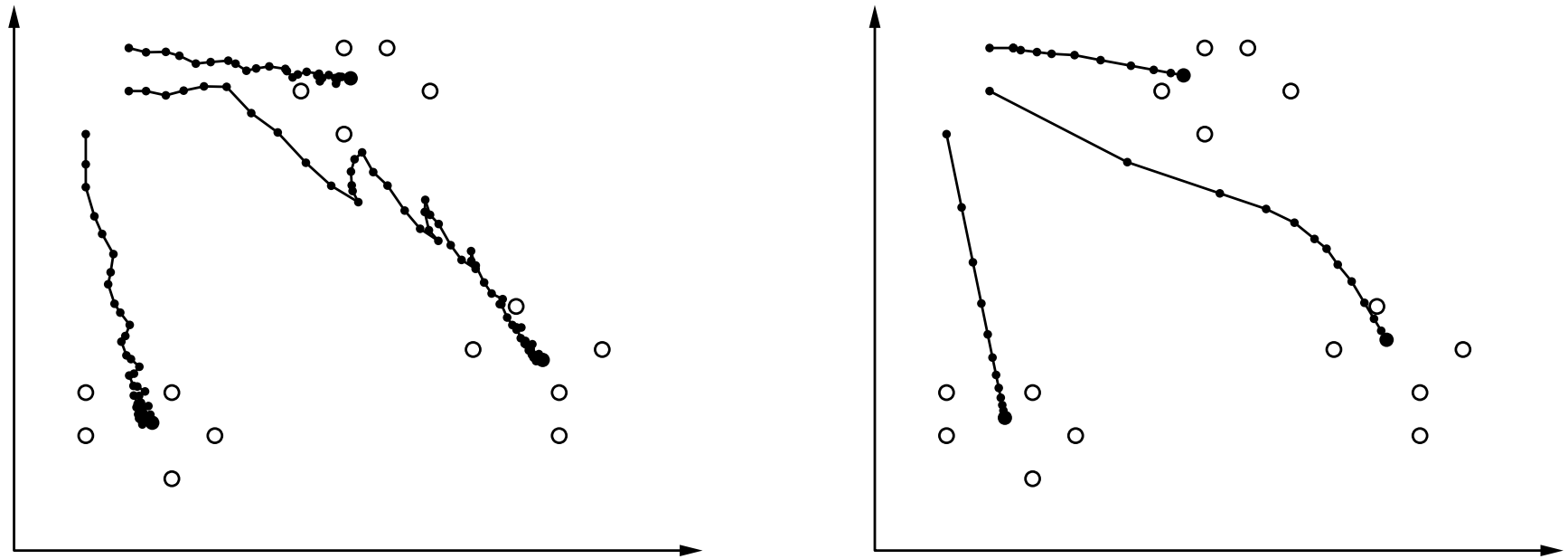


Solution: **time dependent learning rate**

$$\eta(t) = \eta_0 \alpha^t, \quad 0 < \alpha < 1, \qquad \text{or} \qquad \eta(t) = \eta_0 t^\kappa, \quad \kappa < 0.$$

Learning Vector Quantization: Example

Adaptation of reference vectors / codebook vectors



Left: Online training with learning rate $\eta = 0.1$,

Right: Batch training with learning rate $\eta = 0.05$.

Classical Logic: only truth values *true* and *false*.
Classical Algebra: either *is element* or *is not element*.

The bivalence of the classical theories is often not appropriate.

Illustrating example: **Sorites Paradoxon** (Greek *sorites*: heap)

1. One billion grains of sand is a heap of sand. *(true)*
2. If you remove a grain of sand from a heap of sand, a heap of sand remains. *(true)*

Thus it follows:

3. 999 999 999 grains are a heap of sand. *(true)*

Multiple repetition of the same conclusion finally leads to

4. 1 grain of sand is a heap of sand. *(false!)*

At which number of grains the conclusion does not preserve the truth?

Excursus: Short Introduction to Fuzzy Theory

Obviously: There is no exact determined number of grains of sand,
where the conclusion to the next smaller number is wrong.

Problem: Terms of natural language (e.g., "heap of sand", "bald headed", "warm", "fast", "high pressure", "light" etc.) are **vague**.

Note: Though vague terms may be *inexact*, but not *useless*.

Also for vague terms there exist situations/objects,
where they are *for sure applicable* and
where they are *for sure not applicable*.

Between this a **penumbra** (Latin *semi-shade*) of situations is located, where it is unclear, whether the terms are applicable, or whether they are only applicable with restrictions ("small heap of sand").

The fuzzy theory tries to model this penumbra mathematically
("soft transition" between *applicable* and *not applicable*).

Excursus: Fuzzy Logic

Fuzzy logic is an extension of the classical logic
to intermediate values between *true* and *false*.

As truth value, any value from the real interval $[0, 1]$ can appear,
whereas $0 \mathrel{\widehat{=}} \textit{false}$ and $1 \mathrel{\widehat{=}} \textit{true}$.

Thus necessary: **Extension of the logical operators**

◦ Negation	classical: $\neg a$,	fuzzy: $\sim a$	Fuzzy Negation
◦ Conjunction	classical: $a \wedge b$,	fuzzy: $\top(a, b)$	t-Norm
◦ Disjunction	classical: $a \vee b$,	fuzzy: $\bot(a, b)$	t-Conorm

Basic principles of the extension:

- For the extreme values 0 and 1, the operators shall behave the same way like the classical examples (marginal/edge constraints).
- For the intermediate values, the behavior shall be monotone.
- As far as possible the laws of the classical logical shall be preserved.

Fuzzy Clustering

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

Objective Function: (to be minimized)

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

$\mathbf{U} = [u_{ij}]$ 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.

$\mathbf{B} = \{\vec{\beta}_1, \ldots, \vec{\beta}_c\}$ is the set of cluster prototypes.

w is the so-called "fuzzifier" (the higher w, the softer the cluster boundaries).

Constraints:

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

Relation to Classical c-Means Clustering:

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

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

where $\forall i, j : u_{ij} \in \{0, 1\}$ (i.e. hard assignment of the data points) and the cluster prototypes $\vec{\beta}_i$ 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 $w = 2$.

Fuzzy Clustering: Alternating Optimization

Objective function: (to be minimized)

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

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).

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.

Fuzzy C-Means Algorithm: Euclidean distance

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

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):

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

Gustafson–Kessel Algorithm: Mahalanobis distance

$$d^2_{\mathrm{gk}}(\vec{x}_j, \vec{\beta}_i) = (\vec{x}_j - \vec{\mu}_i)^\top \mathbf{C}_i^{-1} (\vec{x}_j - \vec{\mu}_i) \quad \text{with} \quad \vec{\beta}_i = (\vec{\mu}_i, \mathbf{\Sigma}_i)$$

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

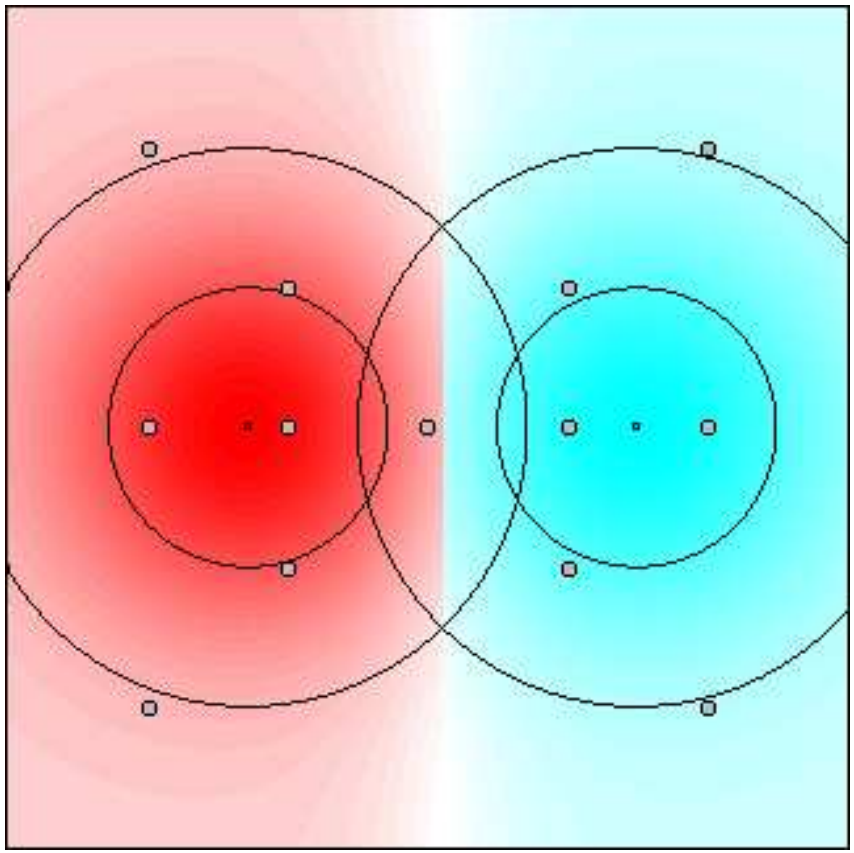
A similar derivation as for the fuzzy c-means algorithm
yields the same update rule for the cluster centers:

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

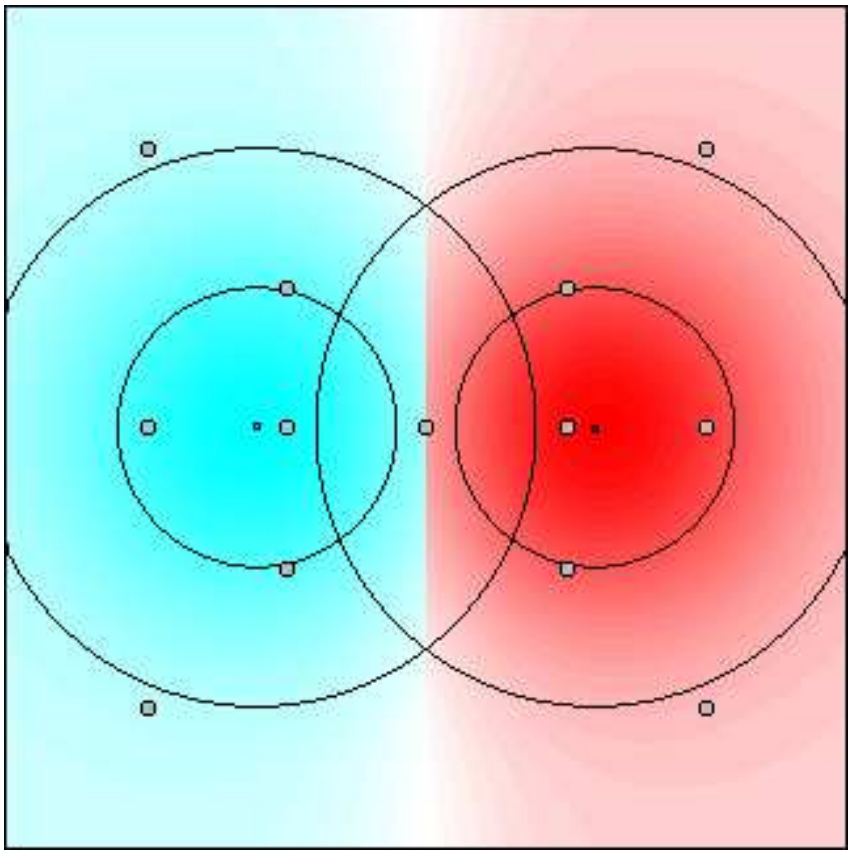
Update rule for the covariance matrices (m is the number of dimensions):

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

Fuzzy Clustering: Overlapping Clusters

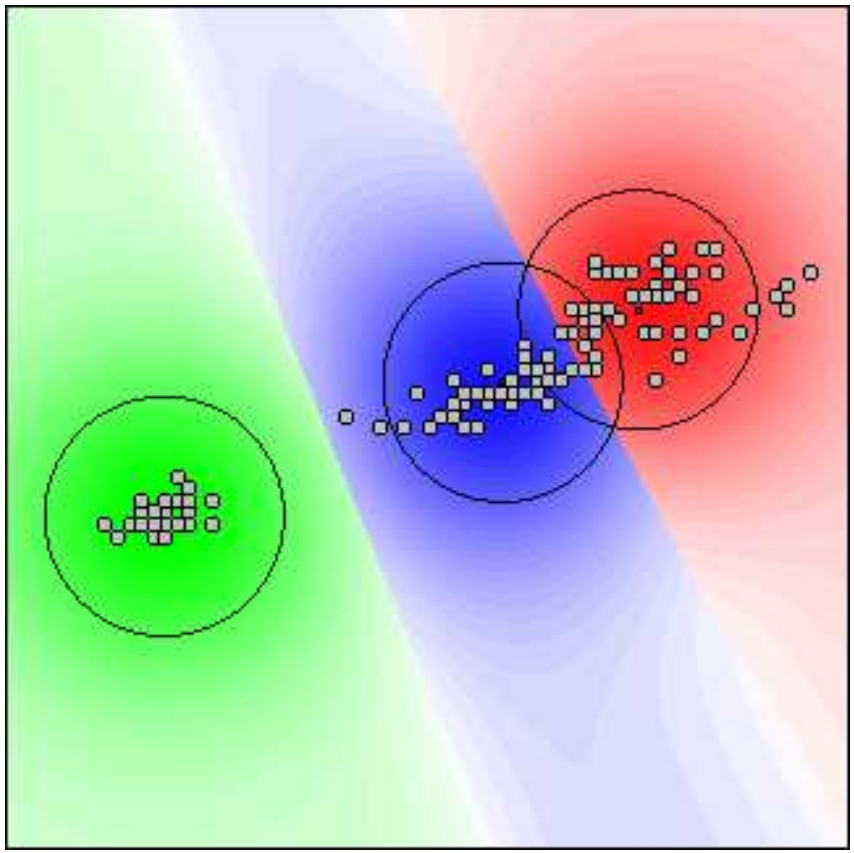


Classical c-Means

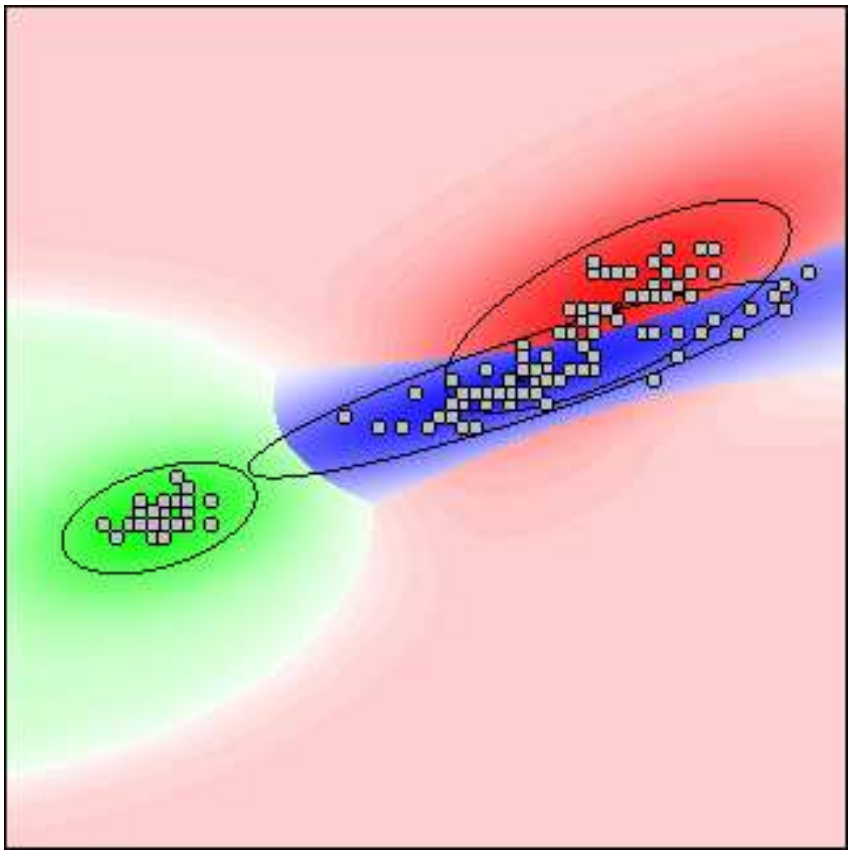


Fuzzy c-Means

Fuzzy Clustering of the Iris Data



Fuzzy c-Means



Gustafson–Kessel

Expectation Maximization: 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

$$\hat{\mathbf{C}} = \arg\max_{\mathbf{C}} E([\ln] L(\mathbf{X}, \vec{y}; \mathbf{C}) \mid \mathbf{X}; \mathbf{C}),$$

that is, maximize the expected likelihood

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

or, alternatively, maximize the expected log-likelihood

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

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.
Example $x = 2$: $y_0 = 1$, $y_1 = 1.5$, $y_2 \approx 1.41667$, $y_3 \approx 1.414216$, $y_4 \approx 1.414213$.

Heron's algorithm converges very quickly and is often used in pocket calculators and microprocessors to implement the square root.

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

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

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).

Justification of the last step on the previous slide:

$$\begin{aligned}
&\sum_{\vec{y}\in\{1,\ldots,c\}^n} \left(\prod_{l=1}^{n} p_{Y_l|\vec{X}_l}(y_l|\vec{x}_l;\mathbf{C}_k)\right) \sum_{j=1}^{n} \ln f_{\vec{X}_j,Y_j}(\vec{x}_j,y_j;\mathbf{C}) \\
&= \sum_{y_1=1}^{c} \cdots \sum_{y_n=1}^{c} \left(\prod_{l=1}^{n} p_{Y_l|\vec{X}_l}(y_l|\vec{x}_l;\mathbf{C}_k)\right) \sum_{j=1}^{n}\sum_{i=1}^{c} \delta_{i,y_j} \ln f_{\vec{X}_j,Y_j}(\vec{x}_j,i;\mathbf{C}) \\
&= \sum_{i=1}^{c}\sum_{j=1}^{n} \ln f_{\vec{X}_j,Y_j}(\vec{x}_j,i;\mathbf{C}) \sum_{y_1=1}^{c} \cdots \sum_{y_n=1}^{c} \delta_{i,y_j} \prod_{l=1}^{n} p_{Y_l|\vec{X}_l}(y_l|\vec{x}_l;\mathbf{C}_k) \\
&= \sum_{i=1}^{c}\sum_{j=1}^{n} p_{Y_j|\vec{X}_j}(i|\vec{x}_j;\mathbf{C}_k) \cdot \ln f_{\vec{X}_j,Y_j}(\vec{x}_j,i;\mathbf{C}) \\
&\quad \underbrace{\sum_{y_1=1}^{c} \cdots \sum_{y_{j-1}=1}^{c} \sum_{y_{j+1}=1}^{c} \cdots \sum_{y_n=1}^{c} \prod_{l=1,l\neq j}^{n} p_{Y_l|\vec{X}_l}(y_l|\vec{x}_l;\mathbf{C}_k)}_{= \prod_{l=1,l\neq j}^{n} \sum_{y_l=1}^{c} p_{Y_l|\vec{X}_l}(y_l|\vec{x}_l;\mathbf{C}_k) \;=\; \prod_{l=1,l\neq j}^{n} 1 \;=\; 1} .
\end{aligned}$$

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

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

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.

Hierarchical Agglomerative Clustering

Start with every data point in its own cluster.
(i.e., start with so-called **singletons**: single element clusters)

In each step merge those two clusters that are closest to each other.

Keep on merging clusters until all data points are contained in one cluster.

The result is a hierarchy of clusters that can be visualized in a tree structure
(a so-called **dendrogram** — from the Greek $\delta\acute{\varepsilon}\nu\tau\varrho\omega\nu$ (dendron): tree)

Measuring the Distances

- The distance between singletons is simply the distance between the (single) data points contained in them.
- However: How do we compute the distance between clusters that contain more than one data point?

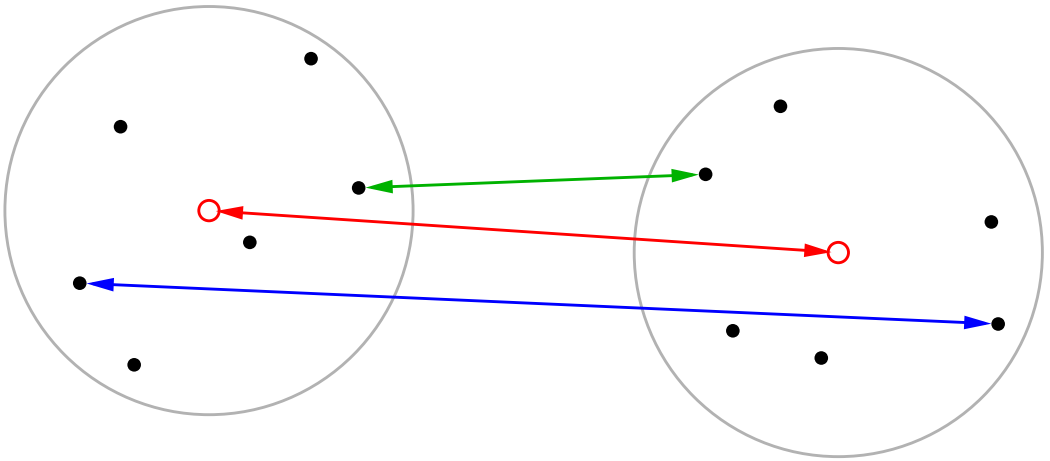
Measuring the Distance between Clusters

Centroid (red)
Distance between the centroids (mean value vectors) of the two clusters.

Average Linkage
Average distance between two points of the two clusters.

Single Linkage (green)
Distance between the two closest points of the two clusters.

Complete Linkage (blue)
Distance between the two farthest points of the two clusters.

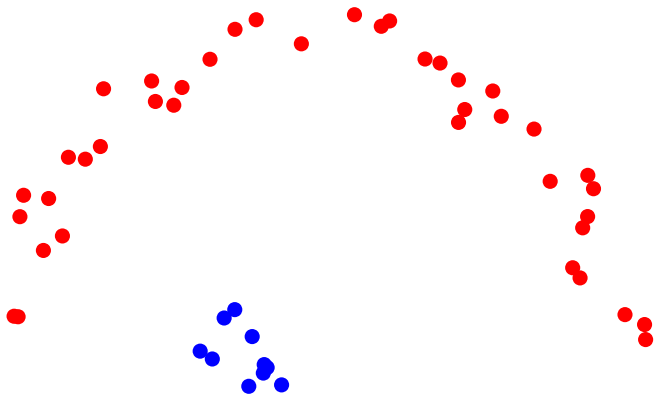


Measuring the Distance between Clusters

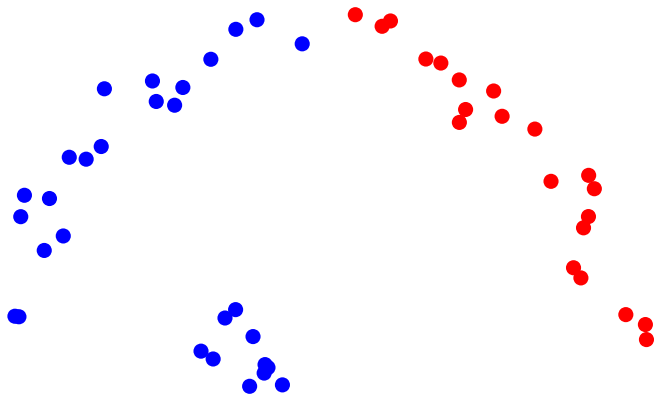
Single linkage can "follow chains" in the data
(may be desirable in certain applications).

Complete linkage leads to very compact clusters.

Average linkage also tends clearly towards compact clusters.



Single Linkage



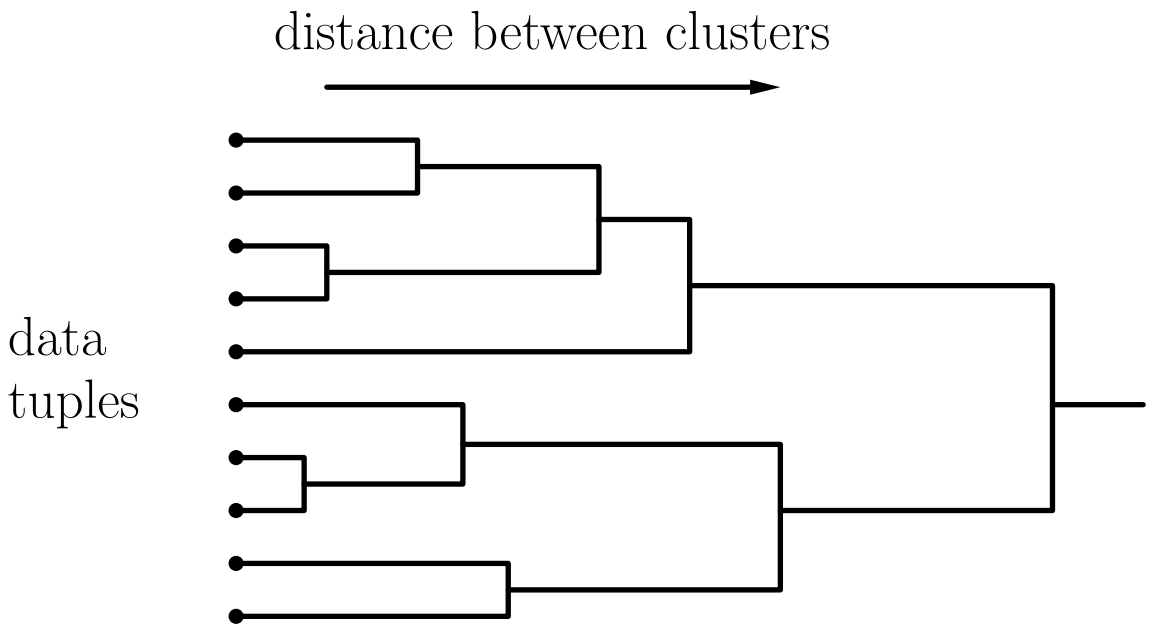
Complete Linkage

Dendrograms

The cluster merging process arranges the data points in a binary tree.

Draw the data tuples at the bottom or on the left
(equally spaced if they are multi-dimensional).

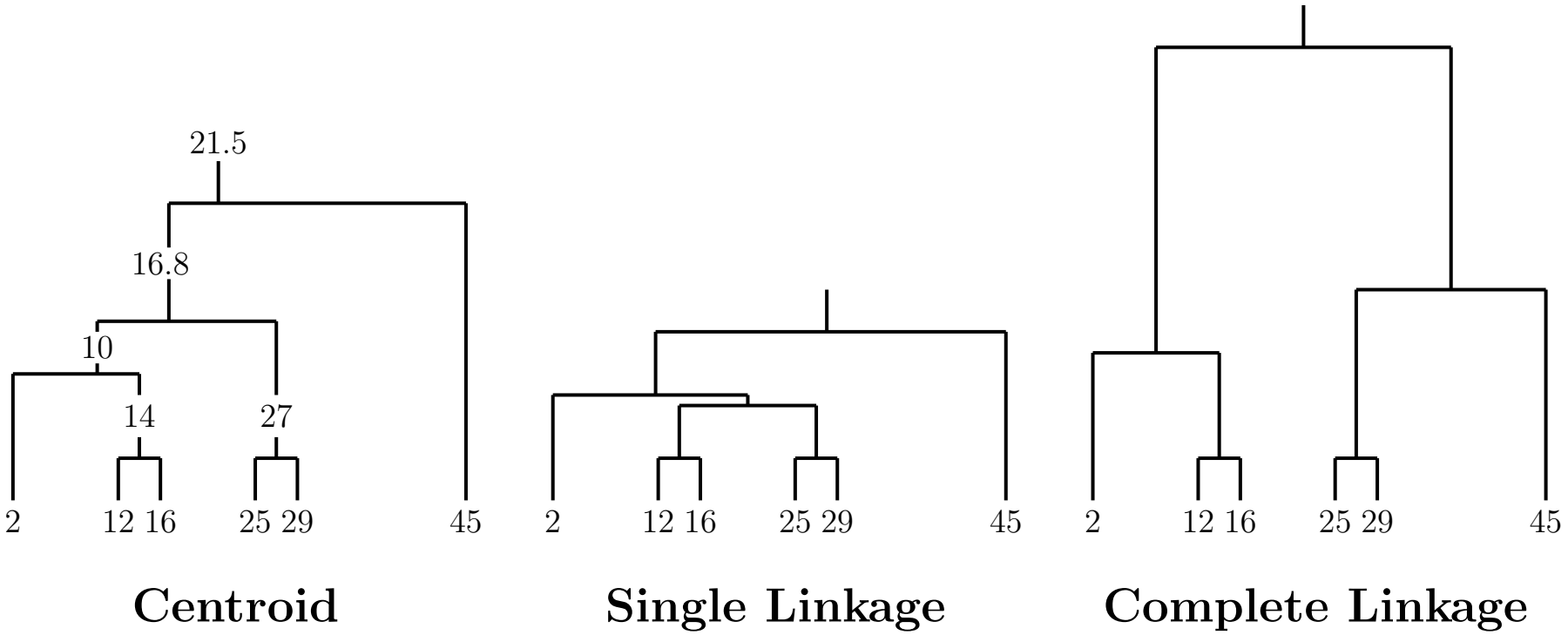
Draw a connection between clusters that are merged, with the distance to the data points representing the distance between the clusters.



Hierarchical Agglomerative Clustering

Example: Clustering of the 1-dimensional data set $\{2, 12, 16, 25, 29, 45\}$.

All three approaches to measure the distance between clusters lead to different dendrograms.



Implementation Aspects

Hierarchical agglomerative clustering can be implemented by processing the matrix $\mathbf{D} = (d_{ij})_{1 \leq i,j \leq n}$ containing the pairwise distances of the data points. (The data points themselves are actually not needed.)

In each step the rows and columns corresponding to the two clusters that are closest to each other are deleted.

A new row and column corresponding to the cluster formed by merging these clusters is added to the matrix.

The elements of this new row/column are computed according to

$$\forall k: \qquad d_{k*} = d_{*k} = \alpha_i d_{ik} + \alpha_j d_{jk} + \beta d_{ij} + \gamma |d_{ik} - d_{jk}|$$

i, j	indices of the two clusters that are merged
k	indices of the old clusters that are *not* merged
$*$	index of the new cluster (result of merger)
$\alpha_i, \alpha_j, \beta, \gamma$	parameters specifying the method (single linkage etc.)

The parameters defining the different methods are
(n_i, n_j, n_k are the numbers of data points in the clusters):

method	α_i	α_j	β	γ
centroid method	$\frac{n_i}{n_i+n_j}$	$\frac{n_j}{n_i+n_j}$	$-\frac{n_i n_j}{n_i+n_j}$	0
median method	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{4}$	0
single linkage	$\frac{1}{2}$	$\frac{1}{2}$	0	$-\frac{1}{2}$
complete linkage	$\frac{1}{2}$	$\frac{1}{2}$	0	$+\frac{1}{2}$
average linkage	$\frac{n_i}{n_i+n_j}$	$\frac{n_j}{n_i+n_j}$	0	0
Ward's method	$\frac{n_i+n_k}{n_i+n_j+n_k}$	$\frac{n_j+n_k}{n_i+n_j+n_k}$	0	0

Choosing the Clusters

Simplest Approach:

- Specify a minimum desired distance between clusters.
- Stop merging clusters if the closest two clusters are farther apart than this distance.

Visual Approach:

- Merge clusters until all data points are combined into one cluster.
- Draw the dendrogram and find a good cut level.
- Advantage: Cut need not be strictly horizontal.

More Sophisticated Approaches:

- Analyze the sequence of distances in the merging process.
- Try to find a step in which the distance between the two clusters merged is considerably larger than the distance of the previous step.
- Several heuristic criteria exist for this step selection.

Density Based Clustering: DBScan

Different notion of a cluster

- A cluster is a region in space where points a close to each other (dense).
- Points that are isolated from other points should be treated as outliers and not contribute to the clustering.

Neighborhoods and Density-Reachability

- The neighborhood of a point x are all points that lie closer to x than a certain threshold, formally: $N_\epsilon(x) = \{p \in X | d(x,p) \leq \epsilon\}$
- Points can act as cluster cores if their ϵ-neighborhood exceeds a minimum size: x is core $\leftrightarrow |N_\epsilon(x)| \geq MinPts$.
- A point p that lies on the border of a cluster might not be a core point but still belong to that cluster if there is at least one core point x such that $p \in N_\epsilon(x)$. Then, p is *directly density-reachable* from x.
- A point p is *density reachable* from another point x if there is a series of points $p_1, p_2, \ldots, p_n$ with $p = p_n$ and $x = p1$ if for every (p_i, p_{i+1}): p_{i+1} is directly density-reachable from p_i.

Density-Connected

- The former definition does not handle the case of two border points yet. We still need to connect these:
- Two points p and q are density connected if there exists a core point o such that p is density reachable from o and q is density reachable from o.

Cluster

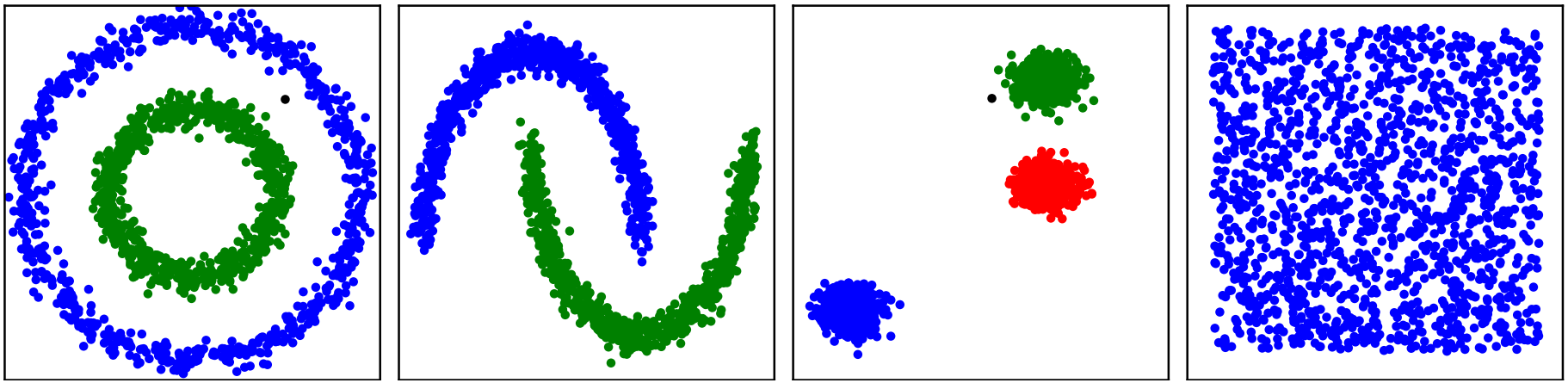
- A cluster C is a subset of all data points such that for every $p, q \in C$ holds: p and q are density connected.

Noise

- A point x is considered as noise if there is no cluster C which x belongs to.

Note: All the above definitions are always with respect to the user-defined parameters ϵ and $MinPts$!

Density Based Clustering - Example



Clusters can be of arbitrary shape and size.

DBScan can find the number of clusters without user interaction.

Can work on a distance matrix alone, no further information needed (beside ϵ and $MinPts$).

Main Problem: Clusters of different densities (would require seperate ϵs for each cluster).

Summary Clustering

Prototype-based Clustering

- Alternating adaptation of data point assignment and cluster parameters.
- Online or batch adaptation of the cluster center.
- Crisp or fuzzy/probabilistic assignment of a datum to a cluster.
- Local minima can pose a problem.
- Fuzzy/probabilistic approaches are usually more robust.

Hierarchical Agglomerative Clustering

- Start with singletons (one element clusters).
- Always merge those clusters that are closest.
- Different ways to measure the distance of clusters.
- Cluster hierarchy can be depicted as a dendrogram.

Summary Clustering

Density Based Clustering

- Find cluster core points.
- For each core point find its neighborhood.
- Merge overlapping neighborhoods (density-reachability).
- Points thats do not belong to any cluster are noise.