ML01 – Introduction to Machine Learning

Clustering

Thierry Denœux

tdenoeux@utc.fr
https://www.hds.utc.fr/~tdenoeux

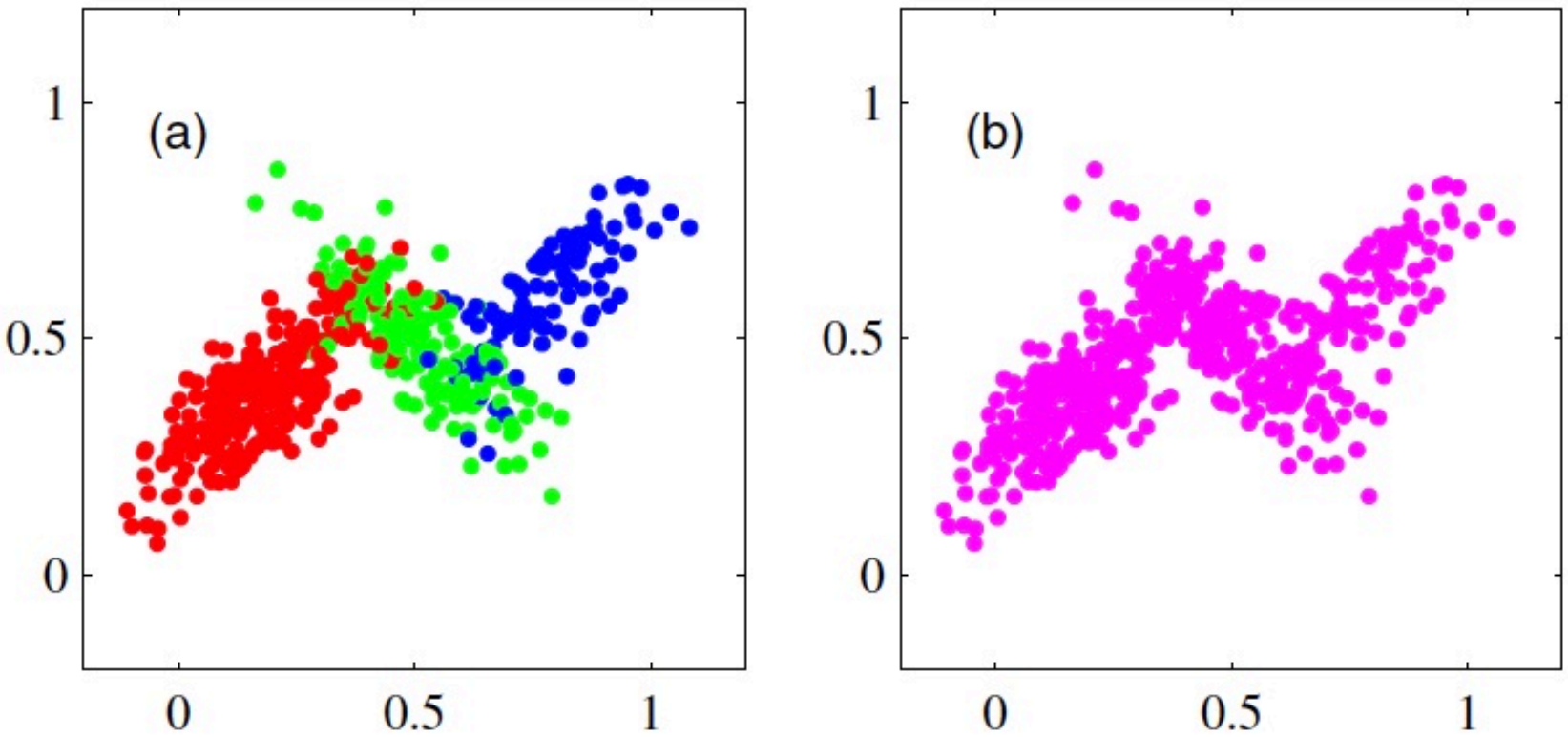
Université de technologie de Compiègne

Spring 2021

Supervised vs. unsupervised learning

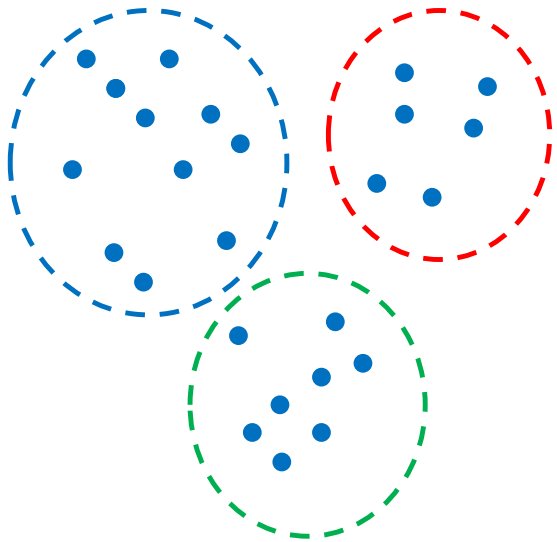
- The problems studied so far – classification and regression – are supervised learning problems:
  - We observe a vector $X = (X_1, \ldots, X_p)$ of predictors and a response variable Y for n individuals in a population (training set)
  - The goal is to predict Y from X for new data.
  - The performance of the prediction can be measured by the error on new data.
- In this chapter and the next we consider unsupervised learning problems in which there is no response variable, just a vector of predictors measured on a learning dataset.

Labeled vs. unlabeled data



Unsupervised learning tasks

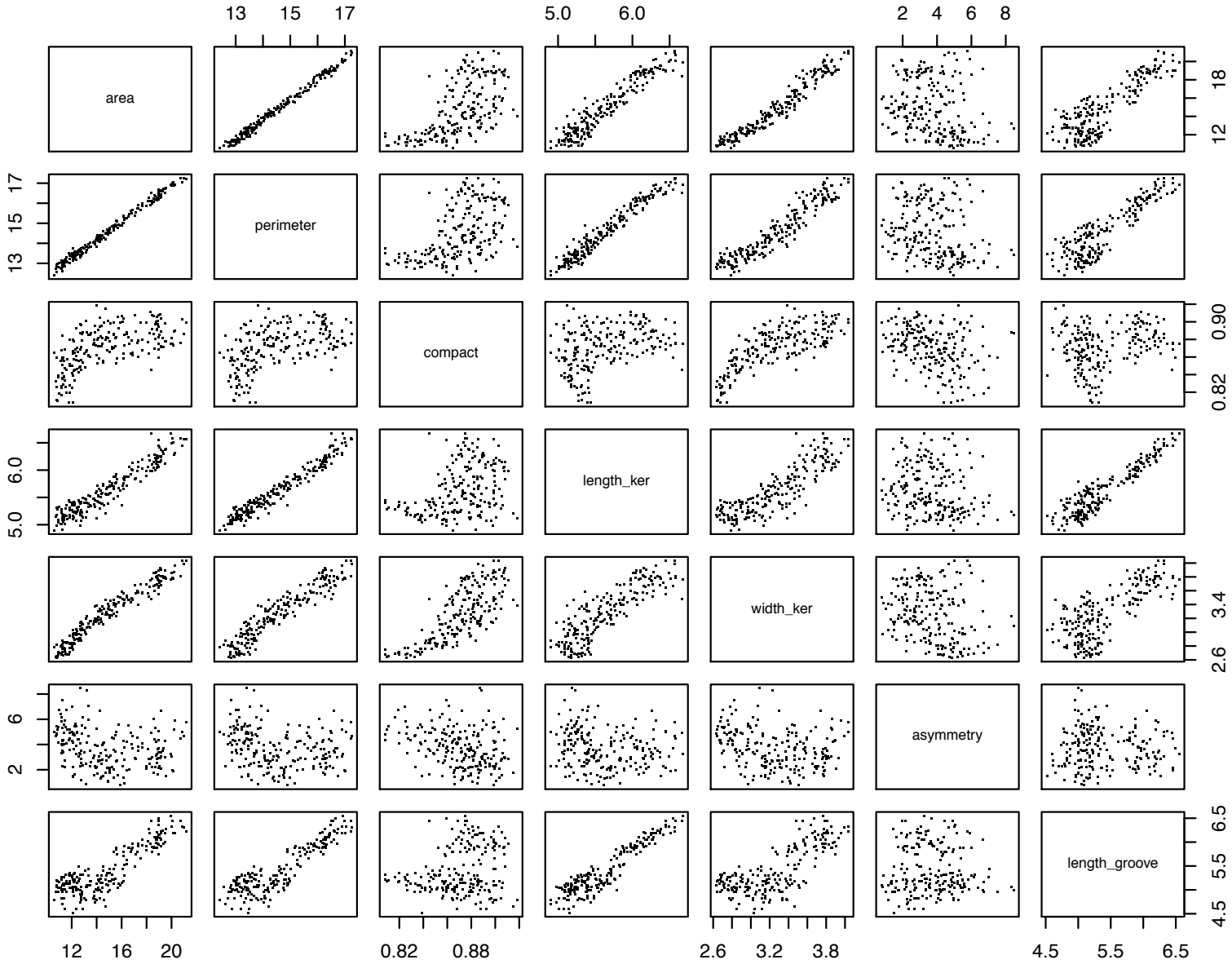
- Unsupervised learning tasks include:
  - Clustering: finding groups in data, such that observations within each group are similar, and observations from different groups are dissimilar
  - Feature extraction: finding a small number of features containing almost as much information as the original features
- In this chapter, we focus on the former problem.



Example: Seed data

- 210 kernels belonging to different varieties of wheat.
- Attributes: seven geometric parameters of wheat kernels
  1. Area
  2. Perimeter
  3. Compactness
  4. Length of kernel
  5. Width of kernel
  6. Asymmetry coefficient
  7. Length of kernel groove

Seeds dataset: graphical representation



Seed dataset: questions

Questions:

1. Are there different groups of seeds, possibly corresponding to different varieties of wheat?
2. How many groups are there?
3. How to assign observations to each group?

Overview

Partition

Definition

A partition of a set $\mathcal{X}$ is a collection of subsets $\mathcal{X}_1, \ldots, \mathcal{X}_c$ such that $\mathcal{X}_1 \cup \ldots \cup \mathcal{X}_c = \mathcal{X}$, and for any $i \neq j$, $\mathcal{X}_i \cap \mathcal{X}_j = \emptyset$. Each subset $\mathcal{X}_k$ is called a class, a group or a cluster.

- (Partitional) clustering aims at finding a partition of n observations (objects) in a dataset.
- Clustering is useful for data exploration and as a pre-processing step to classification. Examples of applications:
  - Marketing: define groups of customers with similar purchasing habits
  - System monitoring: define groups of measurements corresponding to a single state of the system or process under study
  - Image segmentation (define groups of pixels in an image corresponding to homogeneous or meaningful regions)
  - Etc.

Representation of a partition

- Let $\mathcal{X} = \{x_1, \ldots, x_n\}$ be a set of n objects. A partition in c groups can be represented in several ways:
  1. As a vector $\mathbf{y} = (y_1, \ldots, y_n)$ where $y_i = k$ if $x_i \in \mathcal{X}_k$
  2. As an $n \times c$ matrix $U = (u_{ik})$, where $u_{ik} = I(y_i = k)$
  3. As an $n \times n$ matrix $R = (r_{ij})$ such that $r_{ij} = I(y_i = y_j)$
- We will mainly use the 2nd representation.
- Matrix U verifies:

$$\sum_{k=1}^{c} u_{ik} = 1, \quad i = 1, \ldots, n.$$

- The number of observations in cluster k is

$$n_k = \sum_{i=1}^{n} u_{ik}, \quad k = 1, \ldots, c.$$

Partitional clustering algorithms

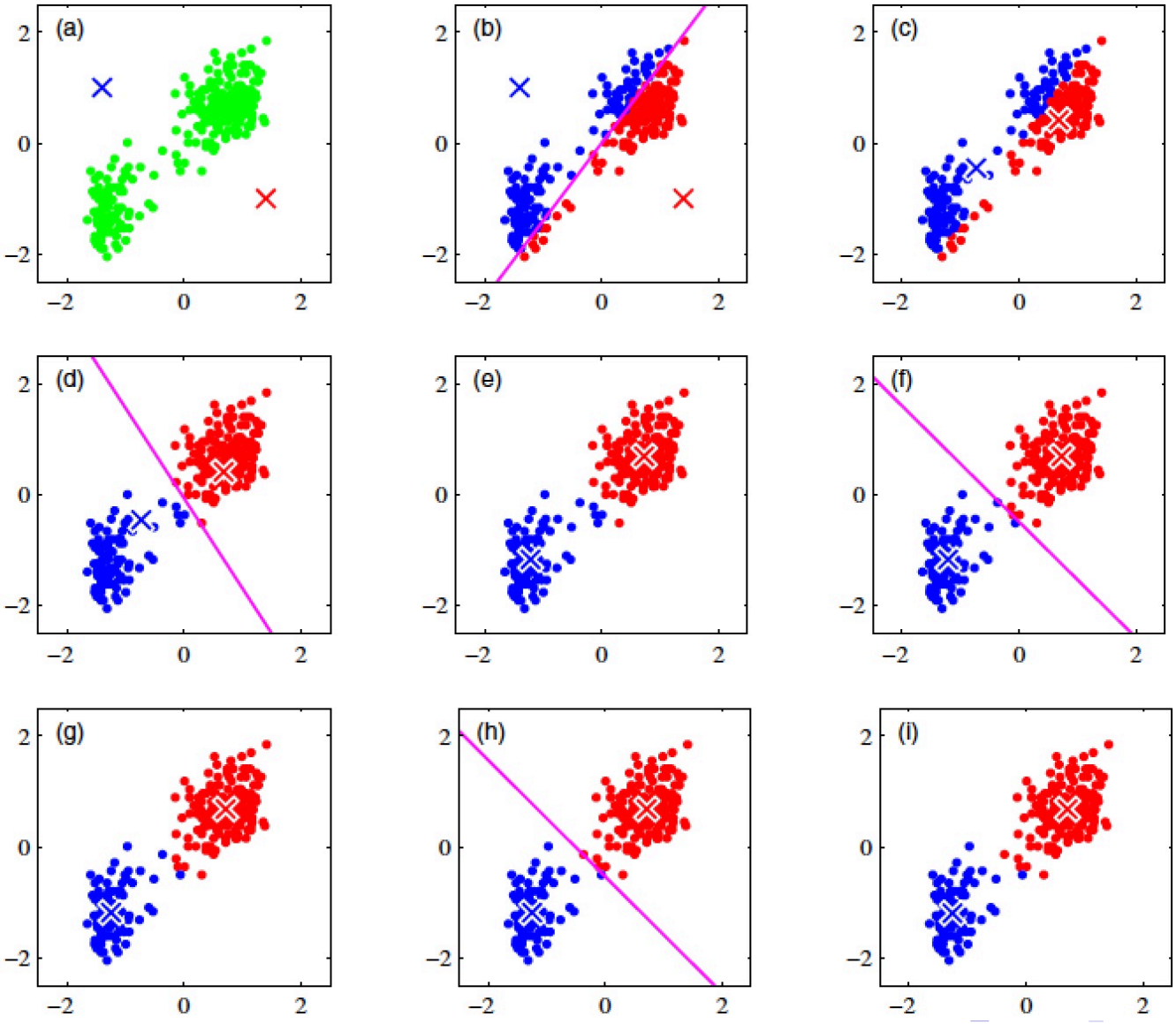
- There exist a lot of (partitional) clustering algorithms.
- The (hard) c-means (HCM) algorithm was introduced in the 1960's but it is still widely used today, because of its simplicity and speed.

Overview

(Hard) c-Means Algorithm

1. Fix the number c of clusters
2. Initialize cluster centers (prototypes) $v_1, \ldots, v_c$ randomly.
3. Compute distances $d_{ik} = \|x_i - v_k\|$ between each observation x_i and each prototype v_k, and assign each x_i to the cluster of its nearest prototype:
$$u_{ik} := I(\arg\min_{\ell} d_{i\ell} = k)$$
4. Recompute each prototype v_k as the center of mass of cluster k:
$$v_k := \frac{\sum_{i=1}^{n} u_{ik} x_i}{\sum_{i=1}^{n} u_{ik}}, \quad k = 1, \ldots, c$$
5. If the prototypes have not changed in the last iteration, stop. Otherwise, return to Step 3.

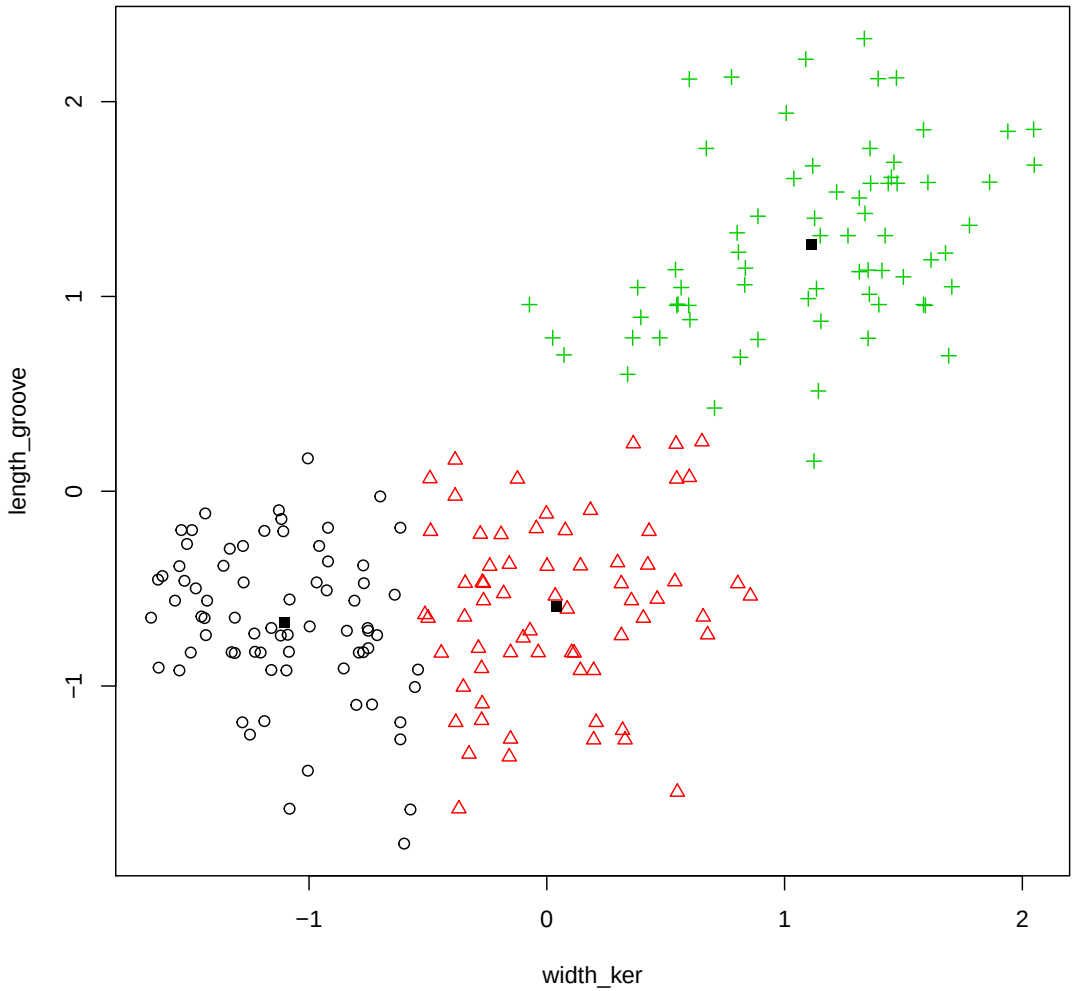
Illustration of the HCM algorithm



Example in R

```
data<-read.table(file="seeds.txt")
data[,1:7]<-scale(data[,1:7])
x<-data[,1:7]

c<-3
# HCM
km<-kmeans(x[,c(5,7)],centers=c,nstart=10)
plot(data[,c(5,7)],col=km$cluster,pch=km$cluster)
points(km$centers,pch=15)
```

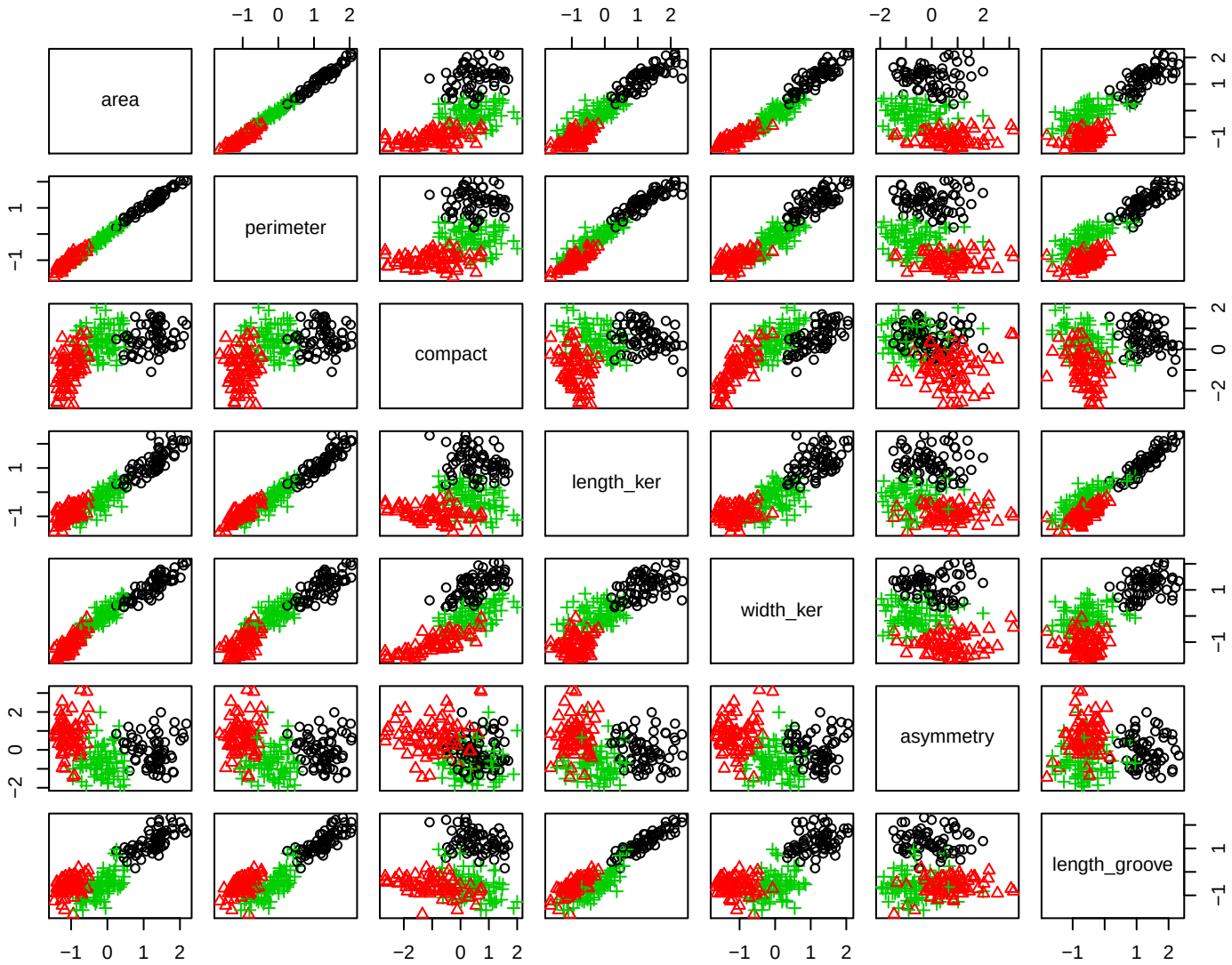
Result



Example in R (continued)

```
# HCM with 7 attributes
km<-kmeans(x,centers=c,nstart=10)
plot(data[,1:7],col=km$cluster,pch=km$cluster)
```

Result


area
perimeter
compact
length_ker
width_ker
asymmetry
length_groove

Why does it work?

- Let $U = (u_{ik})$ and $V = (v_1, \ldots, v_c)$. Consider the following cost function:

$$J_{\text{HCM}}(U, V) = \sum_{i=1}^{n} \sum_{k=1}^{c} u_{ik} d_{ik}^2,$$

  where $d_{ik} = \|x_i - v_k\|$ is the Euclidean distance between x_i and v_k.
- It can be shown that $J_{\text{HCM}}(U, V)$ decreases at each step of *HCM*.
- As a consequence, the algorithm converges to a local minimum of $J_{\text{HCM}}(U, V)$.

Proof that $J_{\text{HCM}}(U, V)$ decreases at each iteration

- The HCM algorithm alternates 2 steps:
  1. Update of U with V fixed
  2. Update of V with U fixed
- Step 1: the cost function can be written as

$$J_{\text{HCM}}(U, V) = \sum_{i=1}^{n} \underbrace{\sum_{k=1}^{c} u_{ik} d_{ik}^2}_{d_{i,k(i)}^2} = \sum_{i=1}^{n} d_{i,k(i)}^2$$

with $d_{i,k(i)} = \arg\min_\ell d_{i\ell}$. When updating U for fixed V, each $k(i)$ is chosen to minimize $d_{i,k(i)}^2$, and hence $J_{\text{HCM}}(U, V)$.

Proof that $J_{\text{HCM}}(U, V)$ decreases at each iteration (cont.)

- Step 2: the cost function can alternatively be written as

$$J_{\text{HCM}}(U, V) = \sum_{k=1}^{c} \underbrace{\sum_{i=1}^{n} u_{ik} d_{ik}^2}_{I(v_k)}$$

where

$$I(v_k) = \sum_{\{i|u_{ik}=1\}} (x_i - v_k)^T (x_i - v_k)$$

- We have

$$\frac{\partial I(v_k)}{\partial v_k} = 0 \Leftrightarrow v_k = \frac{1}{n_k} \sum_{\{i|u_{ik}=1\}} x_i.$$

so updating V in such a way that v_k is the center of cluster k minimizes $J_{\text{HCM}}(U, V)$.

Remarks

1. The initial prototypes may be initialized with randomly selected observations.
2. The final solution may depend on the initial prototypes. It is better to run the algorithm several times with different random initializations, and keep the best solution according to J_{HCM}.
3. The choice of the number c of clusters is a difficult problem in clustering. Graphical and numerical methods will be describe hereafter.

Overview

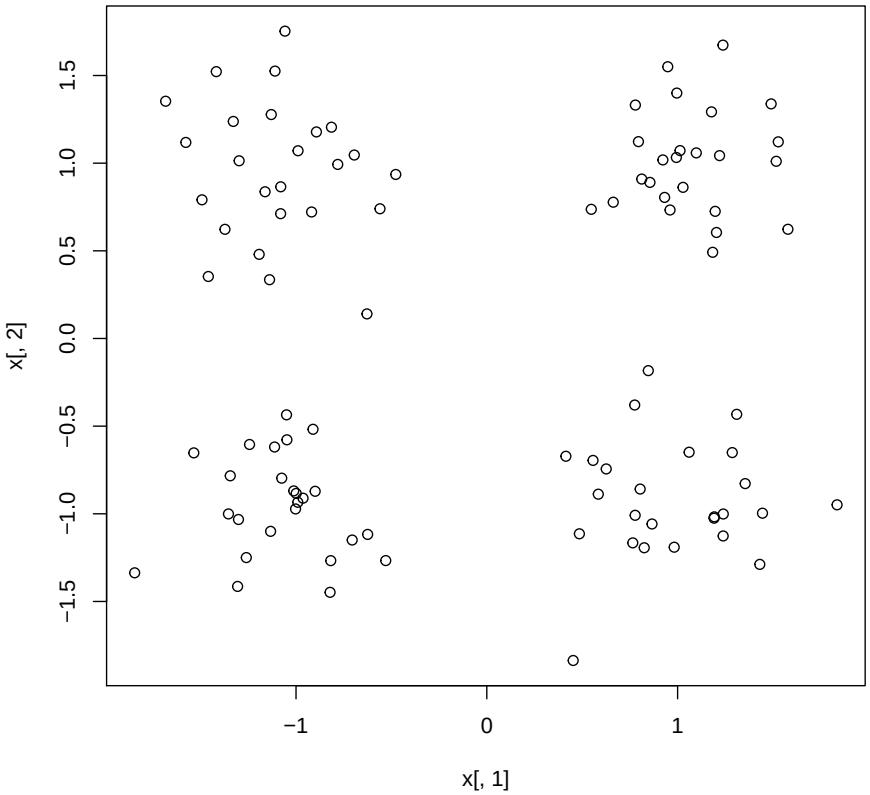
Quality of a partition

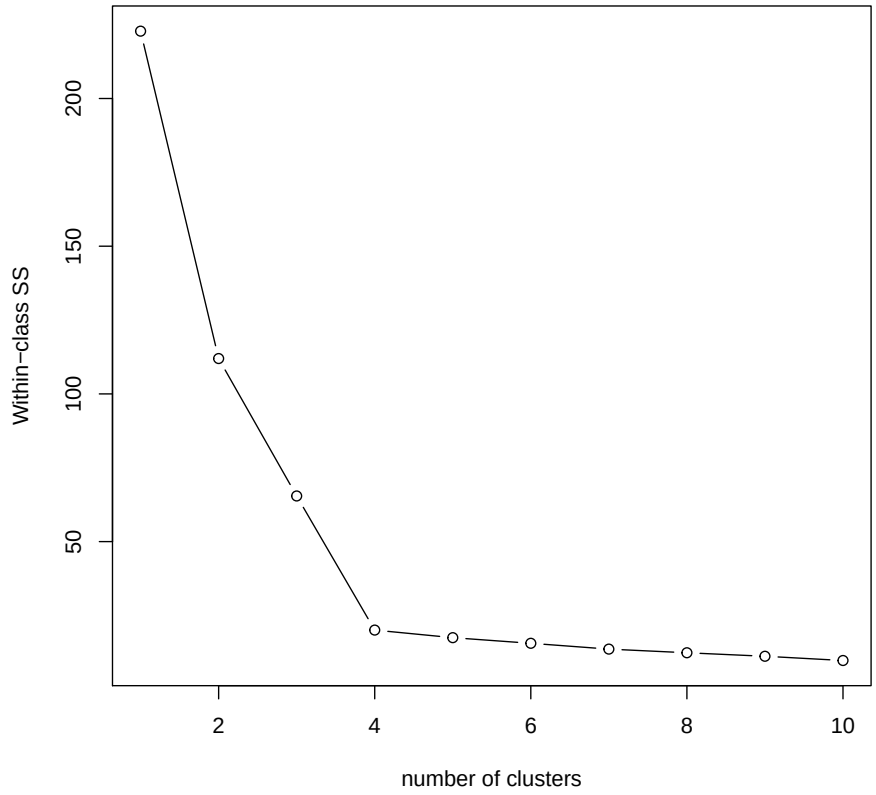
- After we have generated a partition using a clustering algorithm, we need ways to evaluate the validity/quality of this partition.
- If several partitions are generated (e.g., with different numbers of clusters), we need ways to compare them.
- The main approaches include:
  - Graphical representations
  - Internal indices

Graphical representations

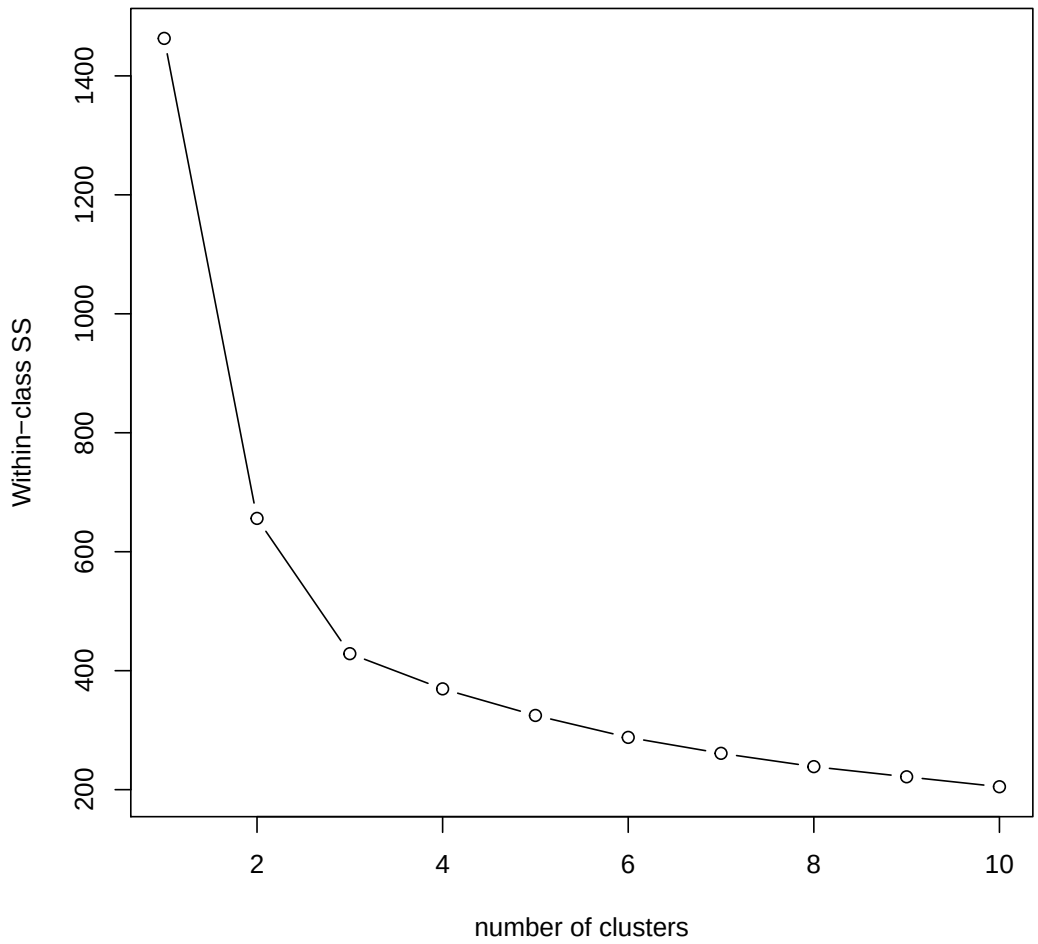
- If p is small (say, $p \leq 5$), we can visually inspect the data using 2D or 3D plots. If p is large, we need more other methods.
- A simple method to determine the number of clusters is to plot J_{HCM} as a function of c and see if the curve decreases more slowly beyond some value of c ("knee"). This method works only of the clusters are well separated (see next slides).
- The silhouette plot is a more sophisticated and widely used graphical representation.

The knee method for data with well-separated clusters





The knee method for the seeds data



Silhouette plot

- The silhouette plot is a graphical representation of data that can be used to visually evaluate the validity of a partition into clusters.
- For each object i in cluster $y_i = k(i)$, let a_i be the mean distance to the other objects in the same cluster

$$a_i = \frac{1}{n_{k(i)} - 1} \sum_{j \neq i, y_j = k(i)} d(i,j)$$

where $d(i,j)$ is the distance between objects i and j, y_j is the cluster of object j. (We assume $n_{k(i)} > 1$).

- Let b_i be the smallest mean distance of object i to all objects in any other cluster, to which i does not belong:

$$b_i = \min_{k \neq k(i)} \frac{1}{n_k} \sum_{j : y_j = k} d(i,j)$$

Silhouette (continued)

- We define the silhouette value of object i as

$$s_i = \frac{b_i - a_i}{\max(a_i, b_i)}$$

- We can see that

$$s_i = \begin{cases} 1 - a_i/b_i & \text{if } a_i < b_i \\ 0 & \text{if } a_i = b_i \\ b_i/a_i - 1 & \text{if } a_i > b_i \end{cases}$$

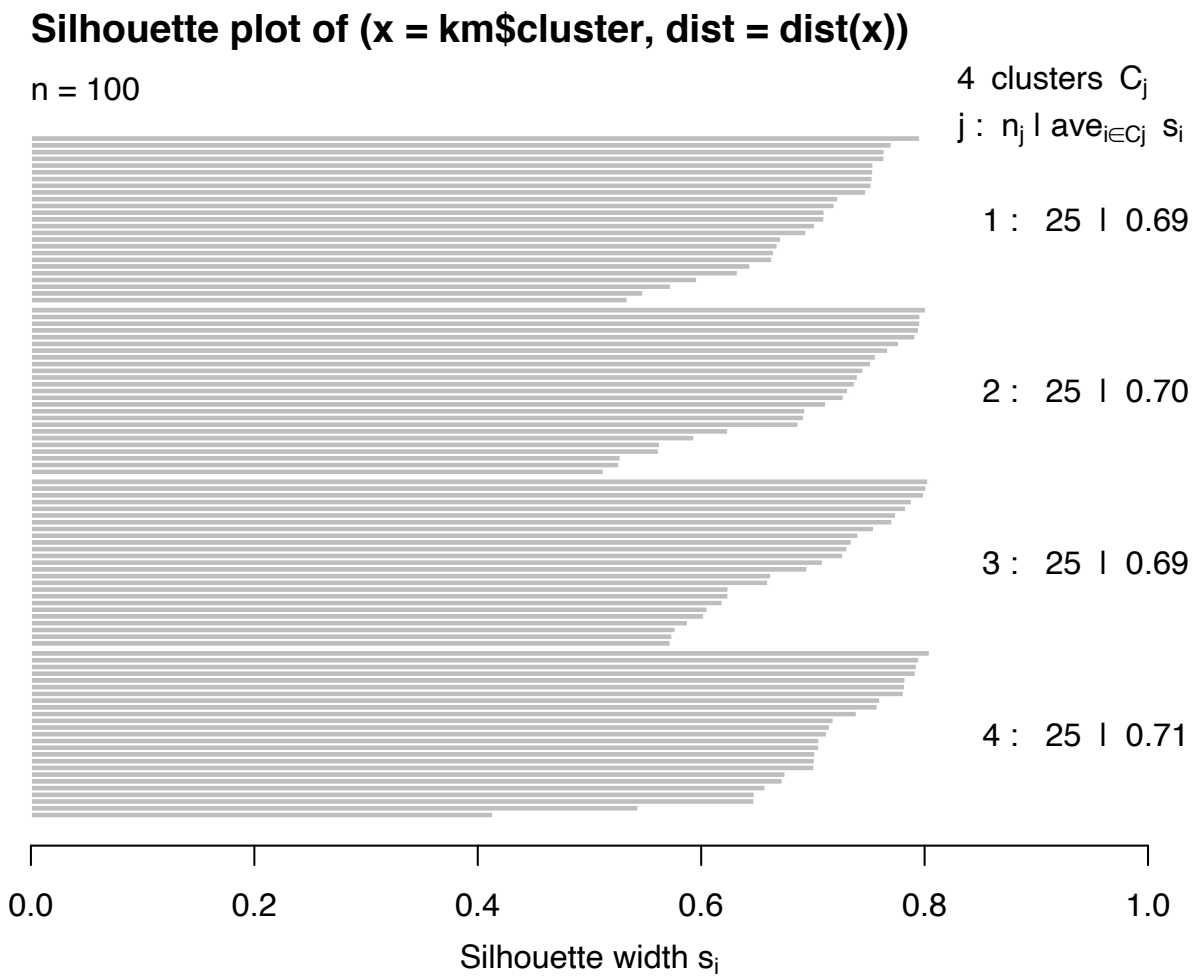
  and $-1 \leq s_i \leq 1$.

- Observations with a large s_i (close to 1) are well clustered, a small s_i (around 0) means that the observation lies between two clusters, and observations with $s_i < 0$ are probably placed in the wrong cluster.

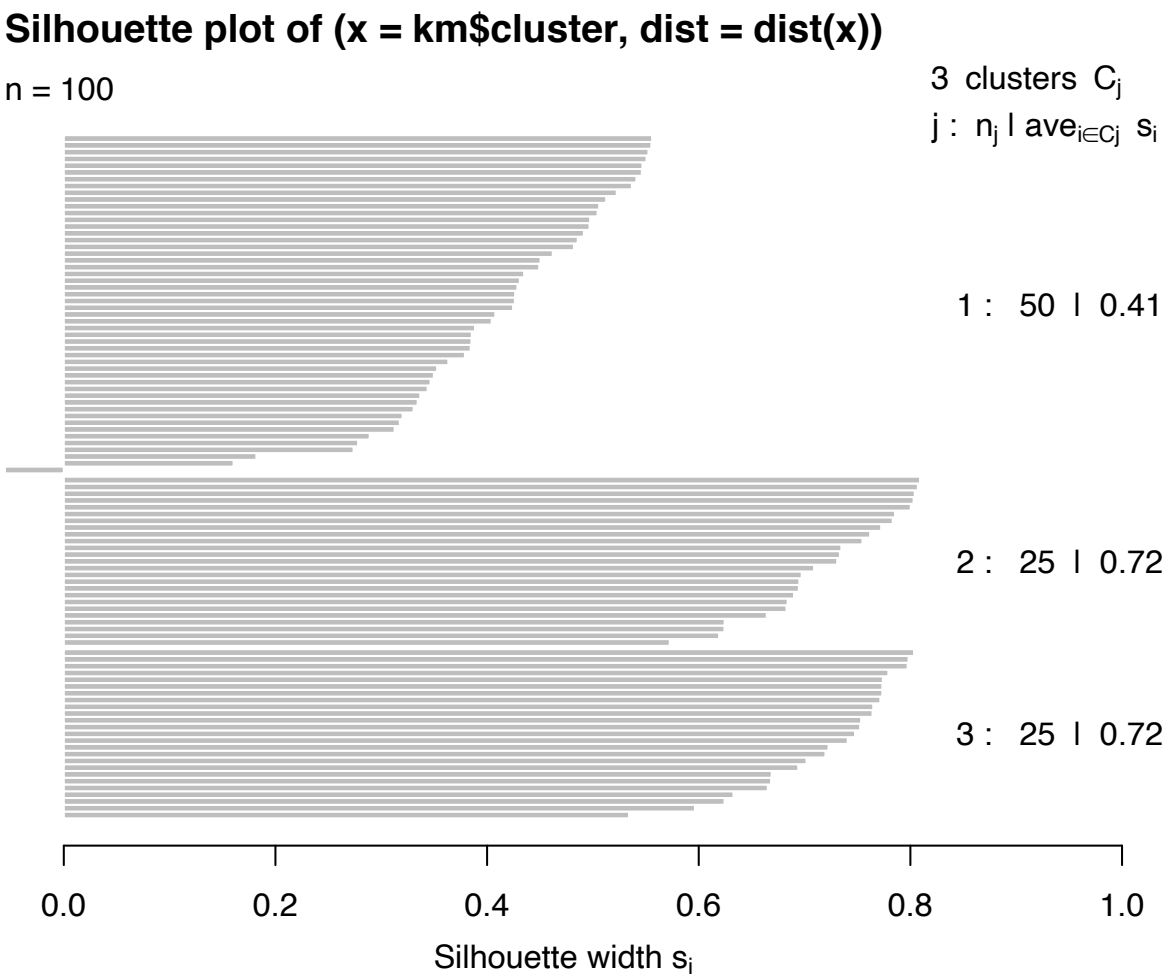
Silhouette plot in R

```
library(cluster)

km<-kmeans(x,centers=3,nstart=10)
D<- dist(x)
sil<-silhouette(km$cluster,D)
plot(sil)
```

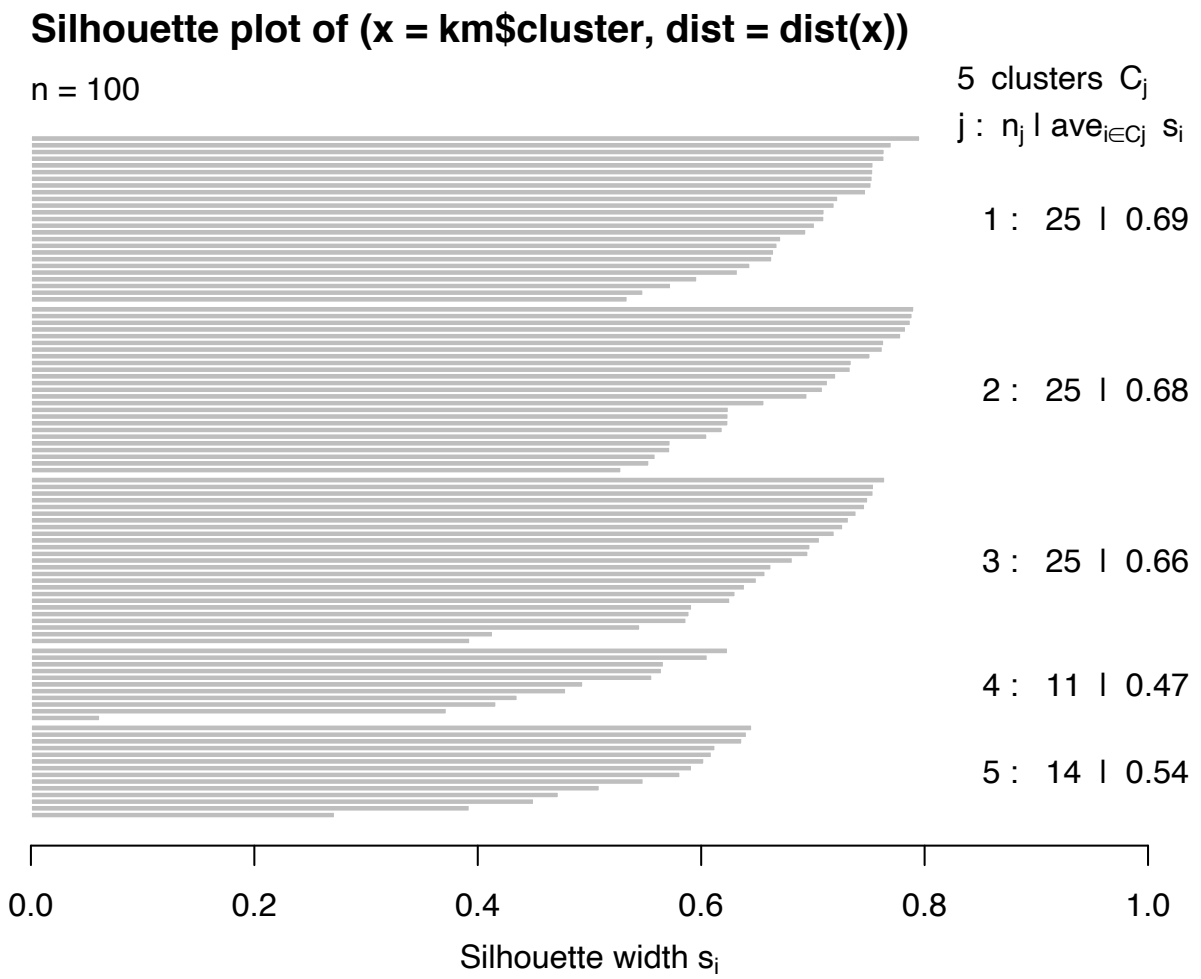
Silhouette plot of the 4-cluster synthetic data with $c = 4$



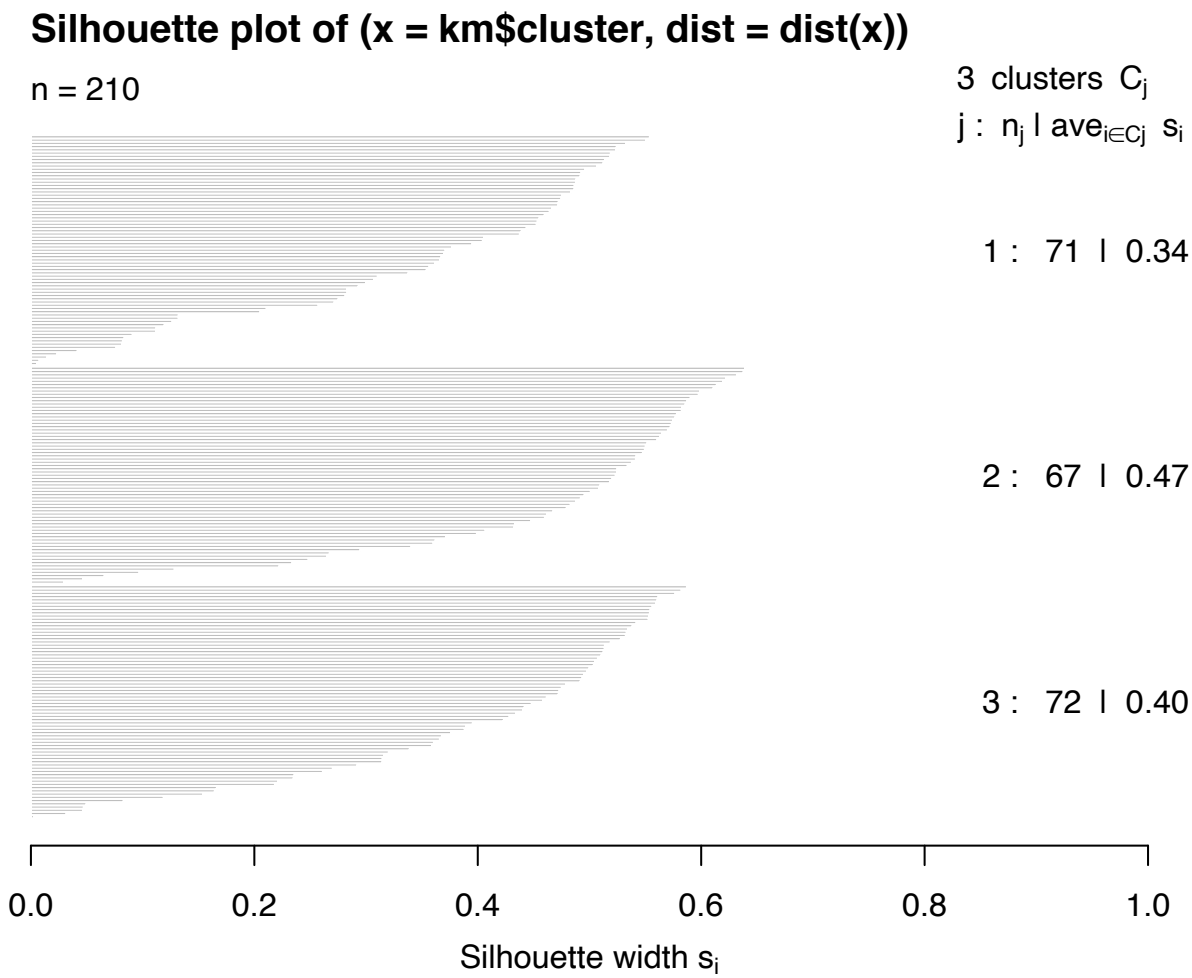
Silhouette plot of the 4-cluster synthetic data with $c = 3$



Silhouette plot of the 4-cluster synthetic data with $c = 5$



Silhouette plot of the seeds data with $c = 3$



Silhouette plot of the seeds data with $c = 4$

Silhouette plot of the seeds data with $c = 2$

Internal indices

- Internal indices mesure the "intrinsic" quality of a partition (how well-separated the clusters are), without comparing it to a ground-truth partition.
- We have seen that the mean silhouette value

$$\overline{s} = \frac{1}{n} \sum_{i=1}^{n} s_i$$

can be used as an internal index. (The larger $\overline{s}$, the better clustering).

- There exist many other internal indices. One of the most widely used is the Davies-Bouldin index.

Davis-Bouldin index

- Let $\overline{d}_k$ be the mean distance between an object of cluster k and the center v_k of that cluster:

$$\overline{d}_k = \frac{1}{n_k} \sum_{i=1}^{n} u_{ik} d_{ik}$$

  It is a measure of the scatter/spread of cluster k.

- A measure of within-to-between spread between clusters k and l is

$$R_{kl} = \frac{\overline{d}_k + \overline{d}_l}{d(v_k, v_l)}$$

  R_{kl} is small if clusters l and l are well separated.

Davis-Bouldin index (continued)

- The index of cluster k is

$$R_k = \max_{l \neq k} R_{kl}$$

  R_k is small if cluster k is well separated from all other clusters.
- The Davis-Bouldin (DB) index is defined as

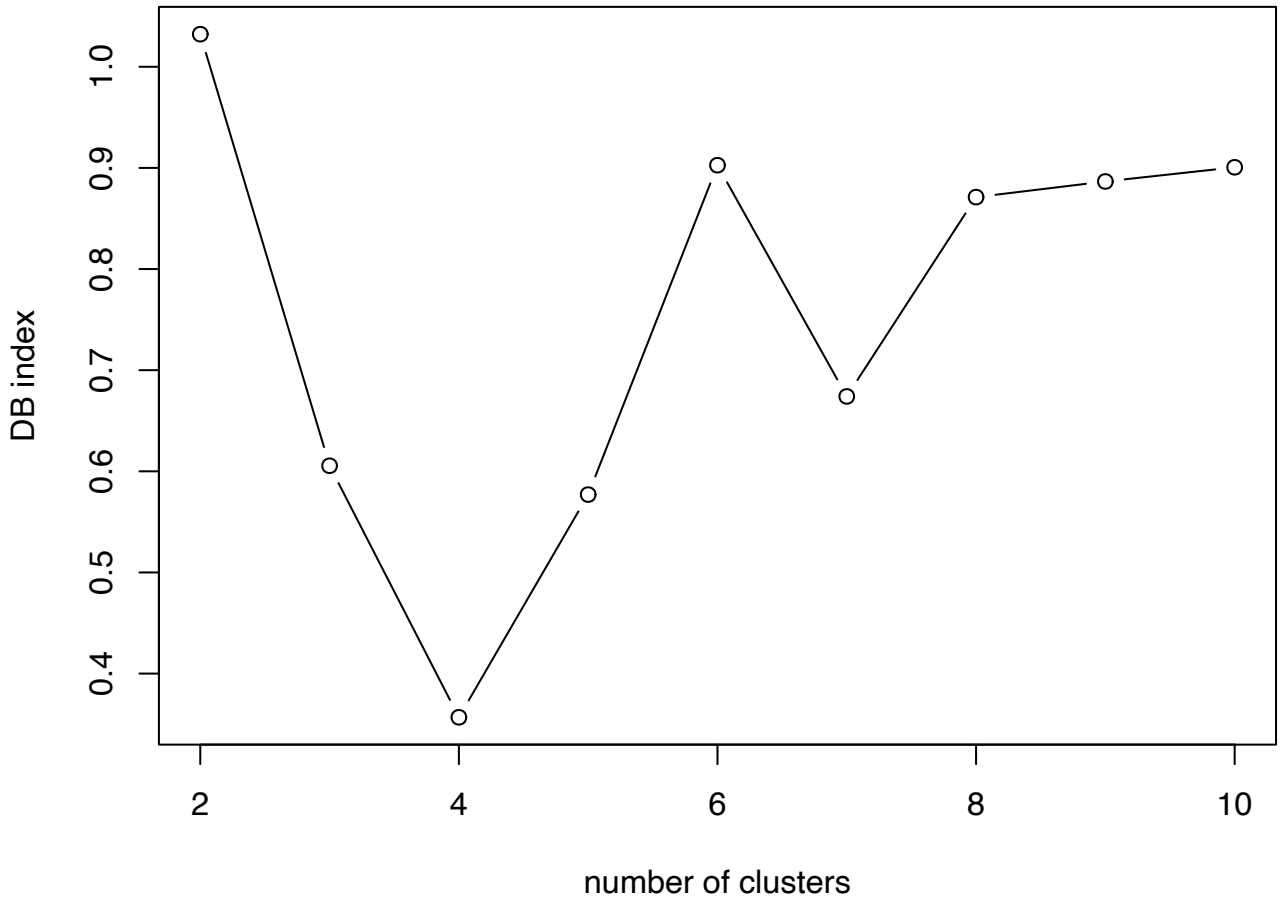
$$DB = \frac{1}{c} \sum_{k=1}^{c} R_k$$

- The smaller DB, the better clustering.

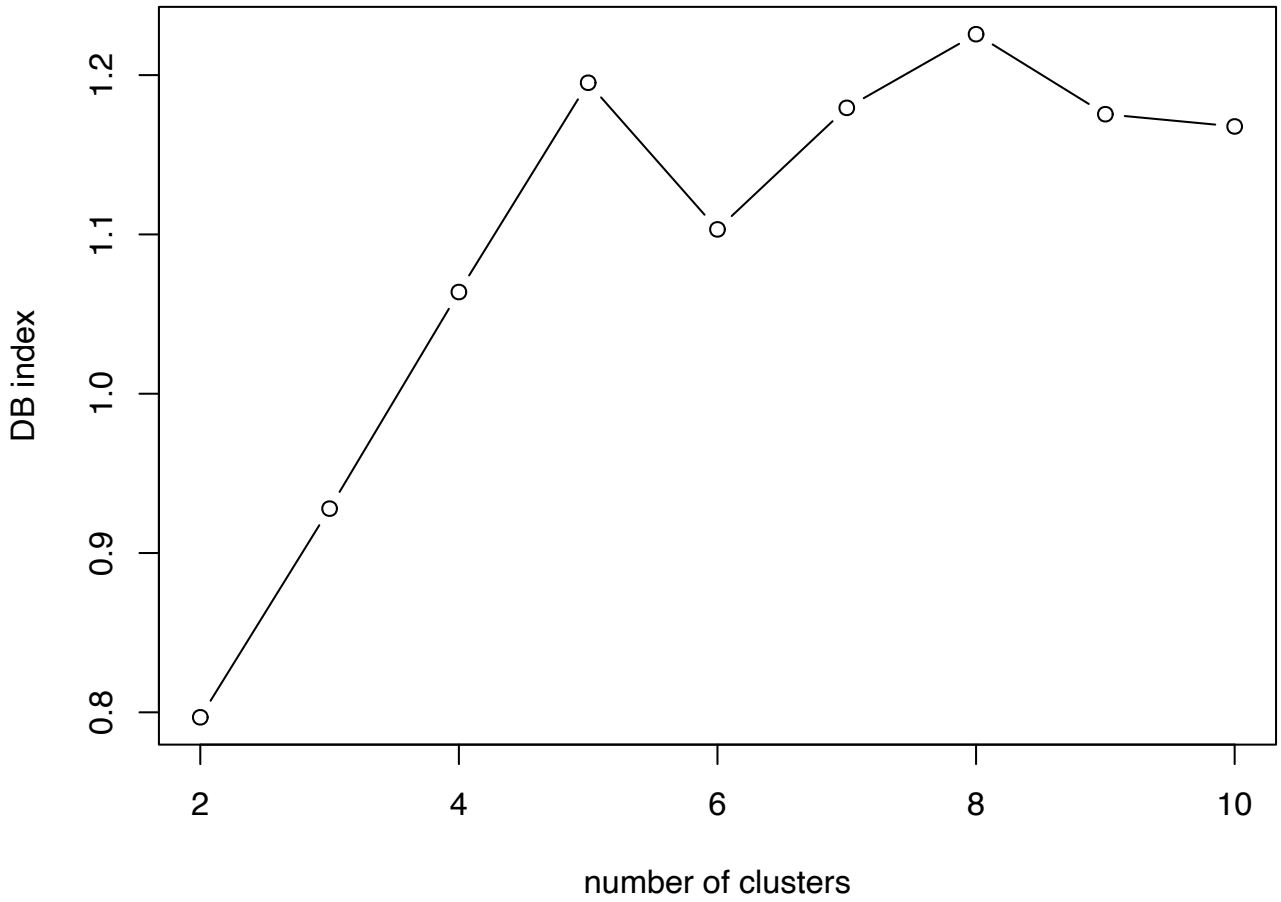
Davis-Bouldin index in R

```
library(clusterCrit)

C<- 2:10
N<-length(C)
DB<-rep(0,N)
for(i in 1:N){
km<-kmeans(x,centers=C[i],nstart=10)
DB[i]<-intCriteria(as.matrix(x), km$cluster, crit="Davies_Bouldin")
}
plot(C,DB,type="b",xlab="number of clusters",ylab="DB index")
```

DB index for the 4-cluster synthetic data



DB index for the seeds data

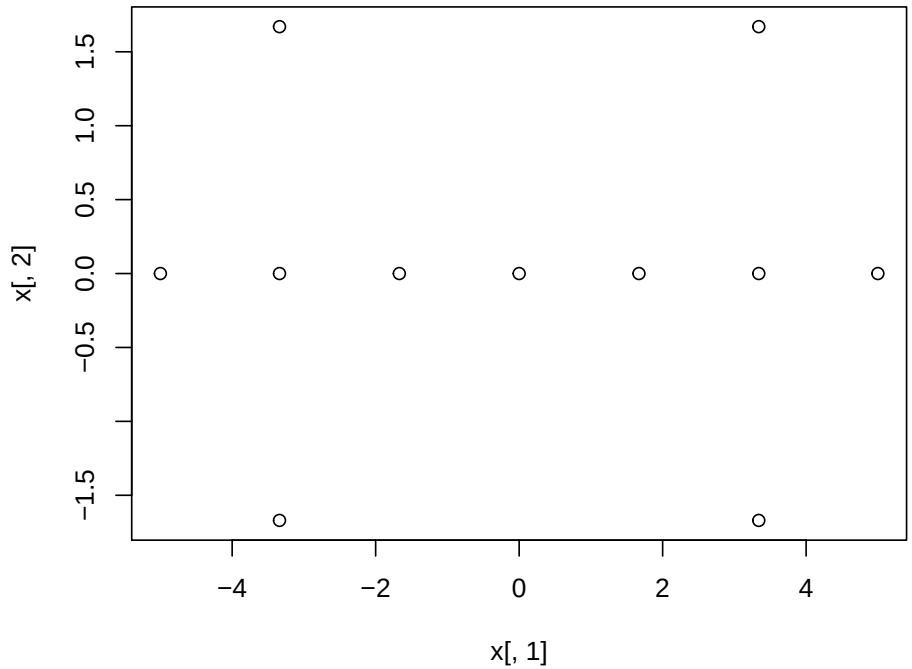


Overview

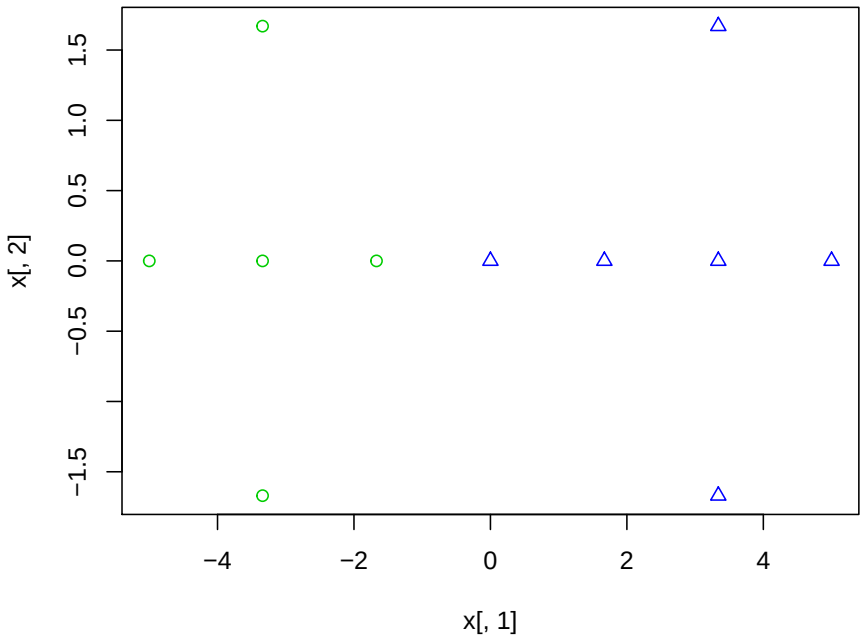
Overview

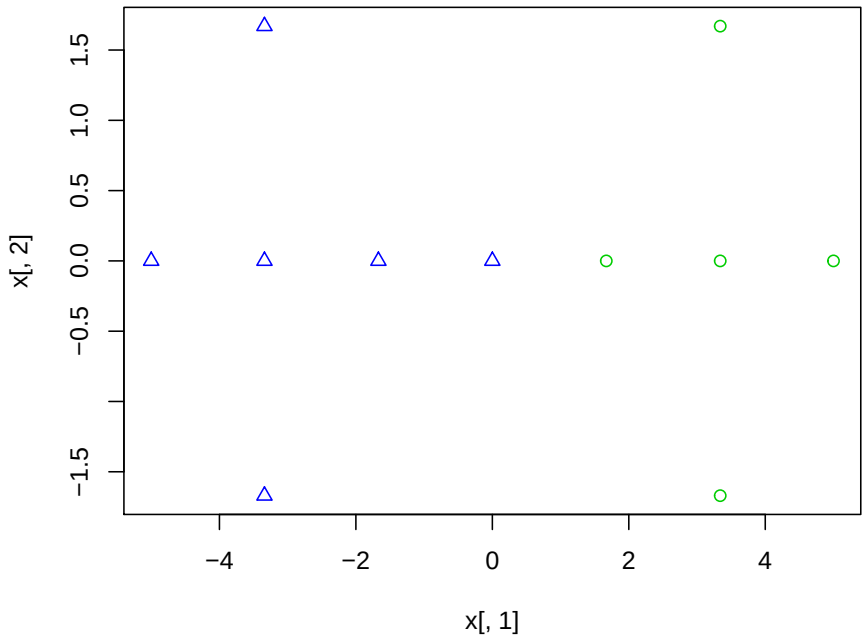
Limitation of partitional clustering

- In partitional clustering, an observation is assigned unambiguously to one and only one cluster.
- This may be arbitrary when the observation lies at the boundary between two or more clusters.
- Example ("butterfly" data"):



Butterfly data: 2 solutions with HCM





Fuzzy partition

- A fuzzy partition is described by an $n \times c$ matrix $U = (u_{ik})$, where $u_{ik} \in [0, 1]$ is the degree of membership of observation i to cluster k.
- We still impose

$$\sum_{k=1}^{c} u_{ik} = 1.$$

- Each cluster becomes a fuzzy set of observations.
- How to generate a fuzzy partition?

Overview

Fuzzy c-means (FCM)

- We consider the following optimization problem:

$$J_{\text{FCM}}(U, V) = \sum_{i=1}^{n} \sum_{k=1}^{c} u_{ik}^{\beta} d_{ik}^2$$

with $d_{ik} = \|x_i - v_k\|$, subject to the constraints

$$\sum_{k=1}^{c} u_{ik} = 1, \quad i = 1, \ldots, n$$

$$u_{ik} \geq 0, \quad i = 1, \ldots, n \text{ and } k = 1, \ldots, c$$

- With $\beta = 1$, the solution is the same as that of HCM.
- To obtain a fuzzy partition, we need to set $\beta > 1$ (default: $\beta = 2$).

Solution of the optimization problem

As with HCM, we start with randomly selected prototypes $v_1, \ldots, v_c$ and we use a grouped coordinate descent strategy by alternating 2 steps

1. Minimize $J_{\text{FCM}}(U, V)$ with respect to U for fixed V
2. Minimize $J_{\text{FCM}}(U, V)$ with respect to V for fixed U

until some stopping criterion is met, for instance

$$\max |U^{(t+1)} - U^{(t)}| < \epsilon$$

or

$$\max |V^{(t+1)} - V^{(t)}| < \epsilon$$

Minimization of $J_{\text{FCM}}(U, V)$ w.r.t. U for fixed V

- We can write the cost function as

$$J_{\text{FCM}}(U, V) = \sum_{i=1}^{n} \underbrace{\sum_{k=1}^{c} u_{ik}^{\beta} d_{ik}^2}_{J_i(u_{i\cdot})} = \sum_{i=1}^{n} J_i(u_{i\cdot}),$$

  with $u_{i\cdot} = (u_{i1}, \ldots, u_{ic})$.
- We can minimize each function J_i independently, s.t. $\sum_k u_{ik} = 1$ (and $u_{ik} \geq 0$ but we ignore these constraints)

Lagrange multipliers

- The method of Lagrange multipliers is a general method for solving constrained optimization problems of the form

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & g(x) = 0 \end{array} \tag{1}$$

- We consider the Lagrange function

$$\mathcal{L}(x, \lambda) = f(x) - \lambda g(x)$$

and we solve the equations

$$\frac{\partial \mathcal{L}}{\partial x}(x, \lambda) = 0, \quad g(x) = 0 \tag{2}$$

- Under some conditions, the solution of (2) gives us the solution of the optimization problem (1).

Minimization of $J_{\text{FCM}}(U, V)$ w.r.t. U for fixed V (cont.)

- We minimize

$$J_i = \sum_{k=1}^{c} u_{ik}^{\beta} d_{ik}^2 \quad \text{subject to} \quad \sum_{k} u_{ik} = 1$$

- The Lagrange function is

$$\mathcal{L}(u_{i\cdot}, \lambda) = \sum_{k=1}^{c} u_{ik}^{\beta} d_{ik}^2 - \lambda \left(\sum_{k=1}^{c} u_{ik} - 1 \right)$$

Minimization of $J_{\text{FCM}}(U, V)$ w.r.t. U for fixed V (cont.)

- The solution must verify

$$\frac{\partial \mathcal{L}}{\partial u_{ik}} = \beta u_{ik}^{\beta-1} d_{ik}^2 - \lambda = 0, \quad k = 1, \ldots, c$$

$$\sum_{k=1}^{c} u_{ik} = 1$$

- After some manipulation we get

$$\boxed{u_{ik} = \frac{d_{ik}^{-2/(\beta-1)}}{\sum_{\ell=1}^{c} d_{i\ell}^{-2/(\beta-1)}}, \quad k = 1, \ldots, c}$$

Proof

Minimization of $J_{\text{FCM}}(U, V)$ w.r.t. V for fixed U

- We write the cost function as

$$J_{\text{FCM}}(U, V) = \sum_{k=1}^{c} \underbrace{\sum_{i=1}^{n} u_{ik}^{\beta} (x_i - v_k)^T (x_i - v_k)}_{I(v_k)} = \sum_{k=1}^{c} I(v_k)$$

- We can minimize each $I(v_k)$ w.r.t. v_k independently. Solving

$$\frac{\partial I}{\partial v_k} = 0$$

we get

$$\boxed{v_k = \frac{\sum_{i=1}^{n} u_{ik}^{\beta} x_i}{\sum_{i=1}^{n} u_{ik}^{\beta}}, \quad k = 1, \ldots, c}$$

Proof

FCM algorithm

1. Initialize prototypes $V = (v_1, \ldots, v_c)$ randomly.
2. Update U for fixed V:

$$u_{ik} = \frac{d_{ik}^{-2/(\beta-1)}}{\sum_{\ell=1}^{c} d_{i\ell}^{-2/(\beta-1)}} \quad \text{for all } i, k$$

3. Update V for fixed U:

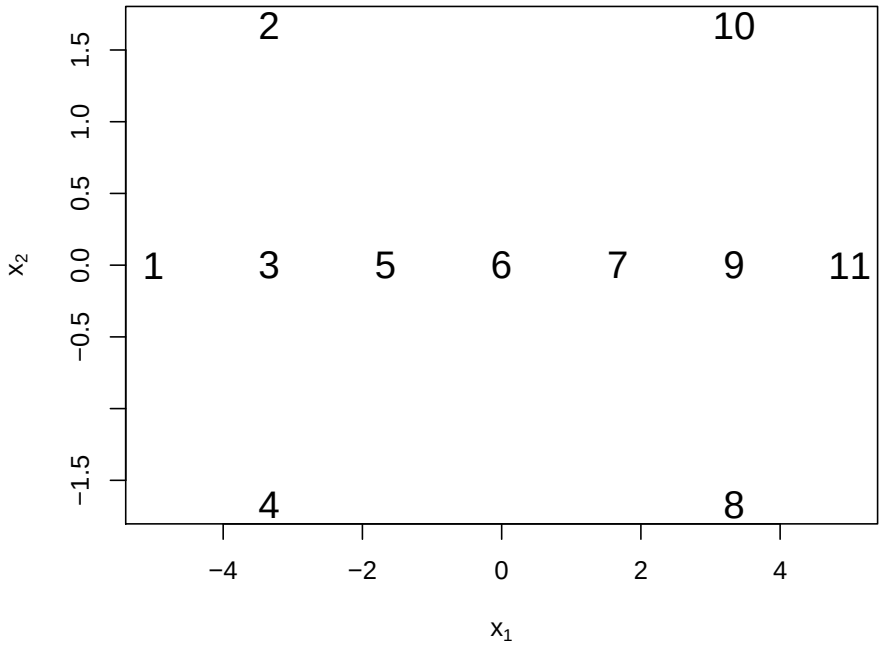
$$v_k = \frac{\sum_{i=1}^{n} u_{ik}^{\beta} x_i}{\sum_{i=1}^{n} u_{ik}^{\beta}} \quad \text{for all } k$$

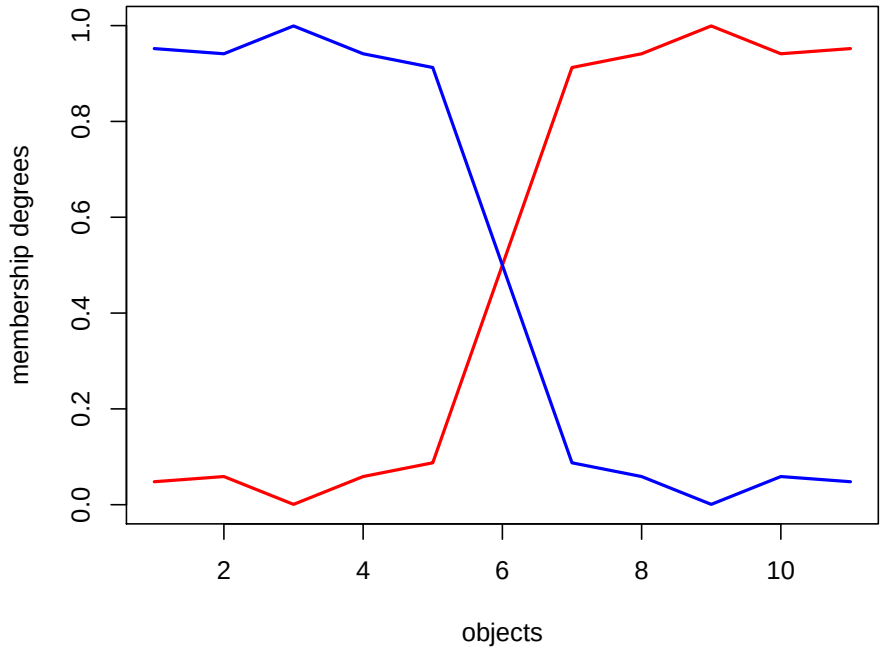
4. Return to Step 2 while the change in V or U is greater than some threshold.

Example in R: butterfly data

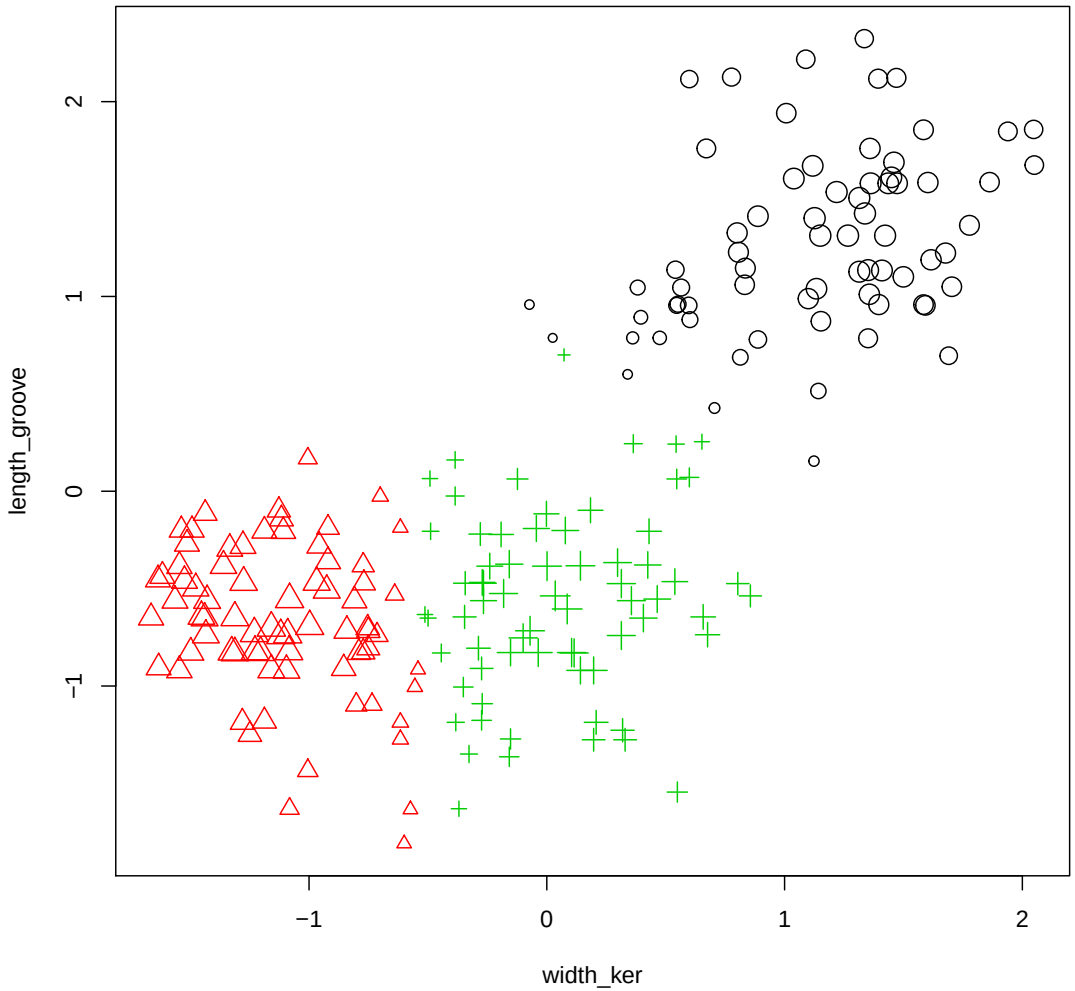
```
library(fclust)

fm<-FKM(x,2,RS=5)

plot(1:11,fm$U[,1],type="l",ylim=c(0,1),xlab="objects",
ylab="membership degrees",col="red",lwd=2)

lines(1:11,fm$U[,2],lty=1,col="blue",lwd=2)
```

Result


2
10
1
3
5
6
7
9
11
4
8
1.5
1.0
0.5
0.0
−0.5
−1.5
x2
−4
−2
0
2
4
x1


membership degrees
1.0
0.8
0.6
0.4
0.2
0.0
2
4
6
8
10
objects

Result with the Seeds data



Overview

Overview

Gaussian Mixture Model

- In LDA and QDA, we assume that the conditional density of input vector X given $Y = k$ is multivariate Gaussian

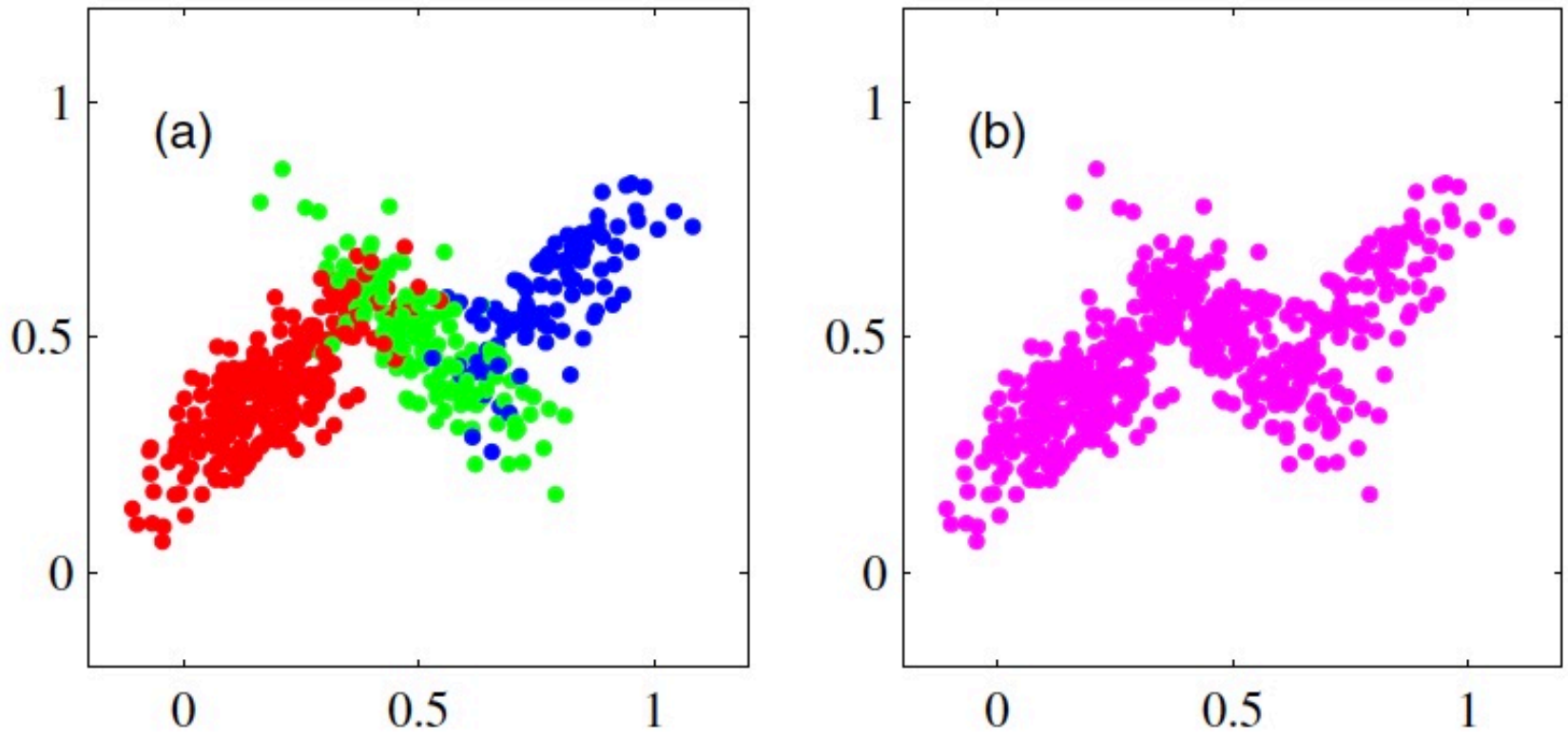
$$\phi(x; \mu_k, \mathbf{\Sigma}_k) = \frac{1}{(2\pi)^{p/2}|\mathbf{\Sigma}_k|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_k)^T \mathbf{\Sigma}_k^{-1}(x-\mu_k)\right)$$

  (with $\mathbf{\Sigma}_k = \mathbf{\Sigma}$ in the case of LDA)
- The marginal density of X is then a mixture of c Gaussian densities:

$$p(x) = \sum_{k=1}^{c} p(x \mid Y = k) P(Y = k) = \sum_{k=1}^{c} \pi_k \phi(x; \mu_k, \mathbf{\Sigma}_k)$$

- This is called a Gaussian Mixture Model (GMM).

Supervised vs. unsupervised learning



Gaussian Mixture Models

- GMMs are widely used in Machine Learning for
  - Density estimation
  - Classification (modeling complex-shaped class distributions)
  - Regression (accounting for different linear relations within subgroups of a population)
  - Clustering (finding groups in data)
  - etc.
- Here, we consider the application to clustering.

Application of GMM's to clustering

- We assume the data to be generated by a mixture of c multivariate normal distributions $\mathcal{N}(\mu_k, \mathbf{\Sigma}_k)$ with proportions π_k. Each cluster corresponds to a component of the mixture.
- The pdf of X_i is

$$p(x_i; \theta) = \sum_{k=1}^{c} \pi_k \phi(x_i; \mu_k, \mathbf{\Sigma}_k),$$

  where θ is the vector of parameters.
- Each cluster is characterized by a center μ_k, a covariance matrix $\mathbf{\Sigma}_k$ and a proportion π_k.
- The method makes it possible to find non-spherical clusters (in contrast to HCM and FCM, which can only find spherical clusters).

utseus 上海大学中欧工程技术学院

Application of GMM's to clustering (continued)

- After we have computed an estimate $\widehat{\theta}$ of θ, we obtain a fuzzy partition of the data by defining the degree of membership u_{ik} of object i to cluster k as the estimated posterior probability

$$u_{ik} = P_{\widehat{\theta}}[Y_i = k \mid x_i]$$

- We also get a hard partition by assigning each vector to the class with maximum estimated posterior probability:

$$\widehat{y}_i = \arg\max_k u_{ik}.$$

Overview

Parameter estimation

- The parameter vector θ is estimated by maximizing the likelihood function

$$L(\theta) = \prod_{i=1}^{n} p(x_i; \theta) = \prod_{i=1}^{n} \sum_{k=1}^{c} \pi_k \phi(x_i; \mu_k, \mathbf{\Sigma}_k)$$

- Maximizing this function is a difficult problem. We used a general algorithm called Expectation-Maximization (EM).
- It is an iterative that alternates 2 steps: expectation (E-step) and maximization (M-step).

EM algorithm

1. Set $t := 0$ and initialize $\theta^{(0)}$ randomly.
2. E-step: Compute

$$u_{ik}^{(t)} = \mathbb{P}_{\theta^{(t)}}[Y_i = k \mid x_i] = \frac{\phi(x_i; \mu_k^{(t)}, \mathbf{\Sigma}_k^{(t)})\pi_k^{(t)}}{\sum_{\ell=1}^{c} \phi(x_i; \mu_\ell^{(t)}, \mathbf{\Sigma}_\ell^{(t)})\pi_\ell^{(t)}}$$

3. M-step: Let $n_k^{(t)} = \sum_{i=1}^{n} u_{ik}^{(t)}$. Compute the estimates

$$\pi_k^{(t+1)} = \frac{n_k^{(t)}}{n}, \quad \mu_k^{(t+1)} = \frac{1}{n_k^{(t)}} \sum_{i=1}^{n} u_{ik}^{(t)} x_i$$

$$\mathbf{\Sigma}_k^{(t+1)} = \frac{1}{n_k^{(t)}} \sum_{i=1}^{n} u_{ik}^{(t)} (x_i - \mu_k^{(t+1)})(x_i - \mu_k^{(t+1)})^T$$

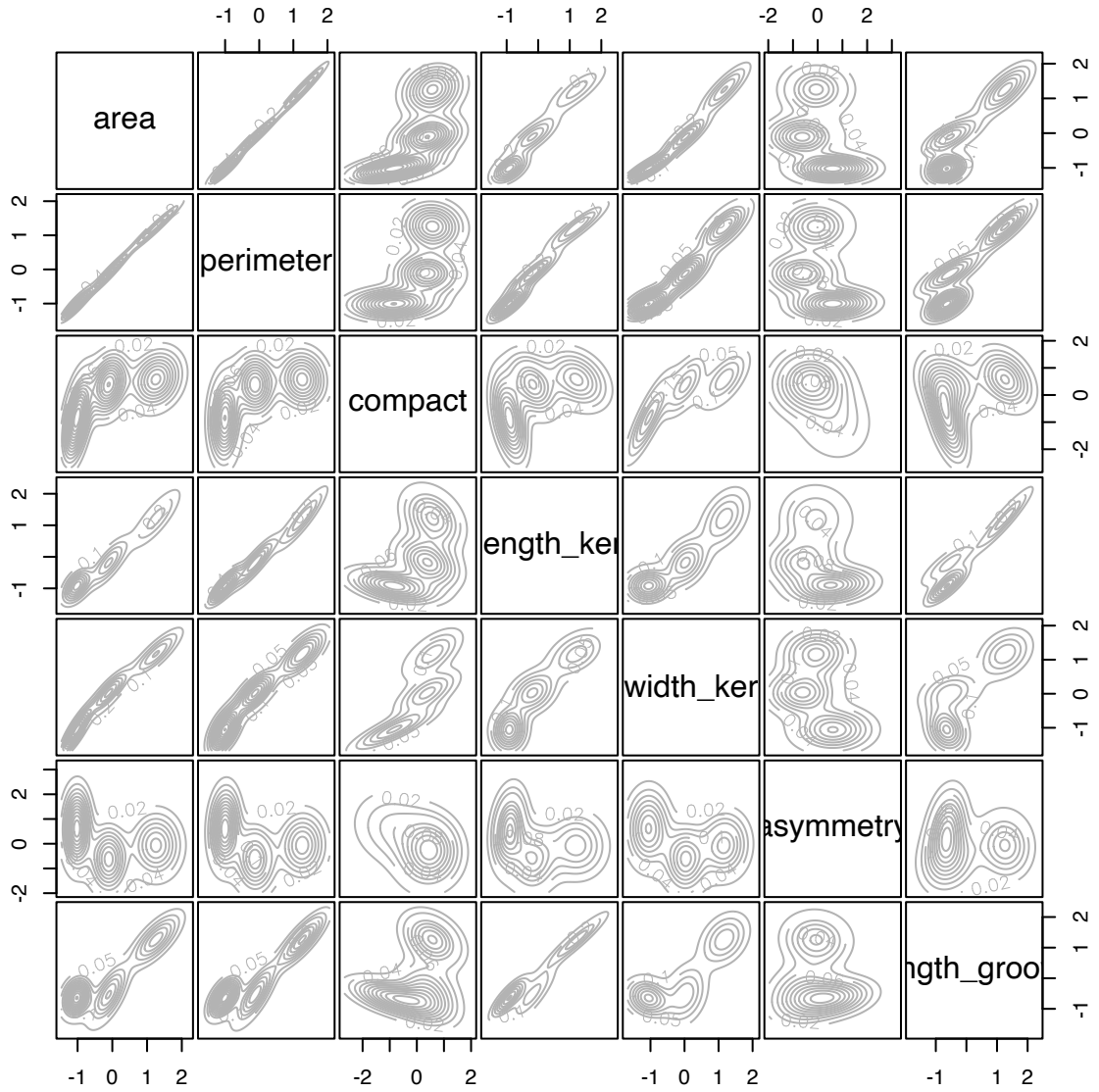
4. If $\|\theta^{(t+1)} - \theta^{(t)}\| \leq \epsilon$, stop. Else, set $t := t + 1$ and go back to step 2.

utseus 上海工学电数工程技术学院

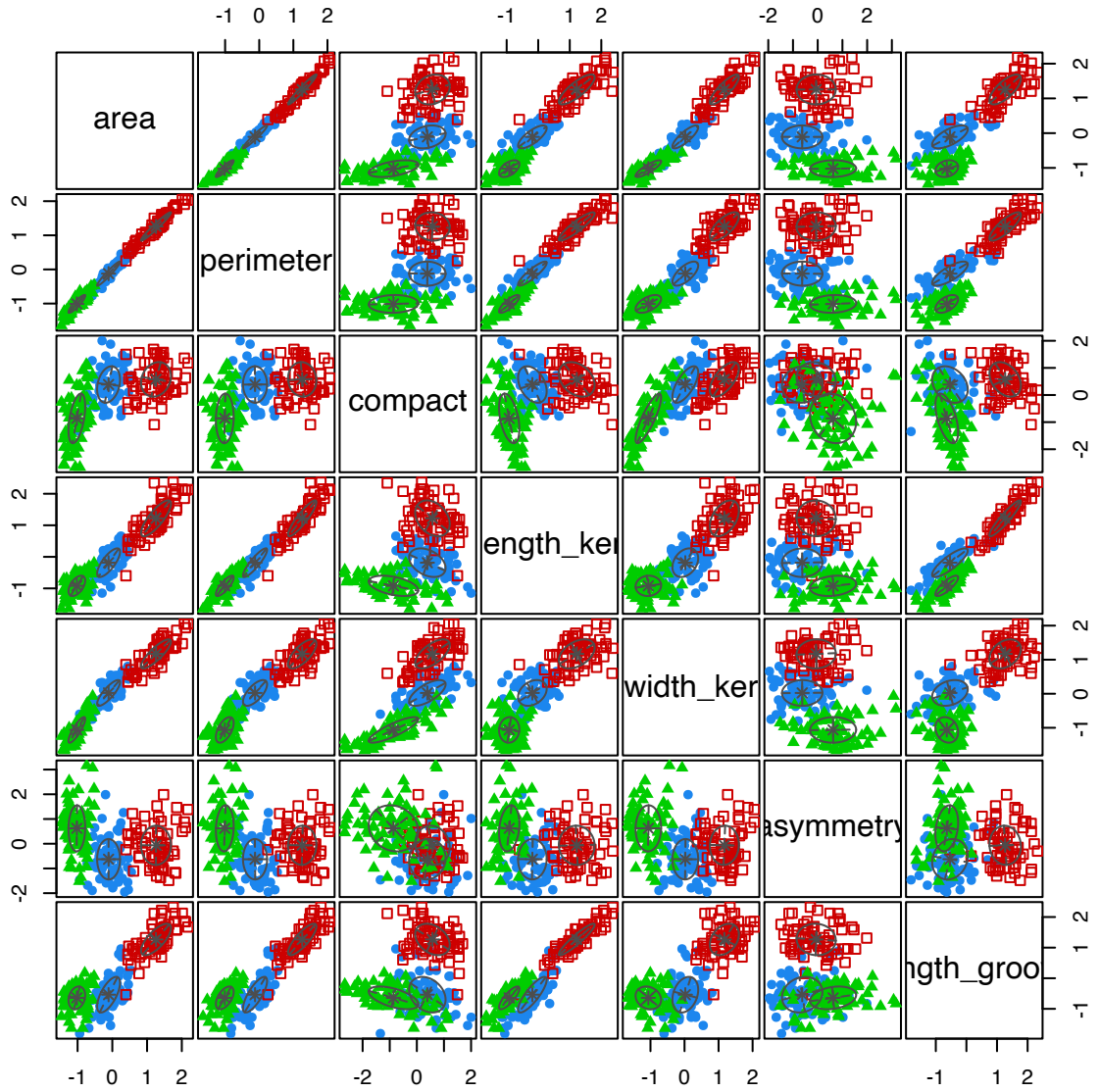
GMM with the package mclust

```
library(mclust)

fit <- Mclust(x,G=3,modelNames="VVV")
plot(fit)
```

Result


area
perimeter
compact
ength_ker
width_ker
symmetry
ngth_groo

Result

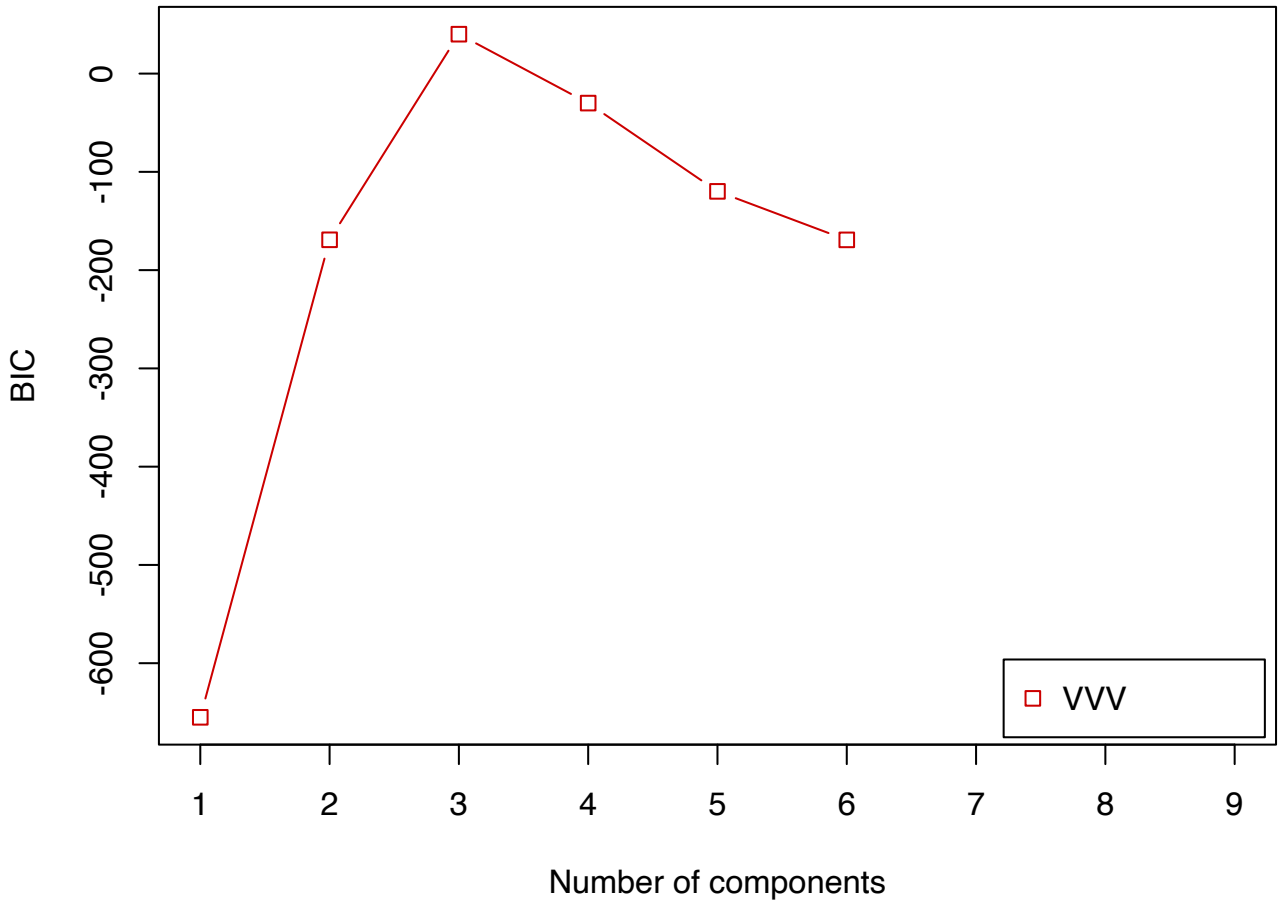

area
perimeter
compact
ength_ke
width_ker
symmetry
ngth_groo

Choosing the number of clusters

- Choosing the number of clusters is a model selection problem.
- We can use the BIC criterion. (Reminder: $BIC = -2\ell(\widehat{\theta}) + d\log(n)$; actually, `Mclust` computes $-BIC$).
- in R:

```
fit <- Mclust(x,modelNames="VVV")
plot(fit,what="BIC")
```

Choosing the number of clusters



Overview

Notion of hierarchy

- We have seen that determining the "best" number of clusters is a difficult problem in partitional and fuzzy custering.
- As an alternative, we can build a sequence of partitions $P_n, P_{n-1}, \ldots, P_2, P_1$, such that
  - Each partition P_k has exactly k clusters
  - Each partition P_k is obtained by merging 2 clusters in partition P_{k+1}
- Such a sequence of partitions is called a hierarchy.

Distance vs. dissimilarity

Definition (Distance)

A distance on a set X is a function $d : X \to \mathbb{R}_+$ verifying the following properties

1. $d(x, y) = 0 \Leftrightarrow x = y$
2. $d(x, y) = d(y, x)$ *[symmetry]*
3. $d(x, z) \leq d(x, y) + d(y, z)$ *[triangular inequality]*

Definition (Dissimilarity)

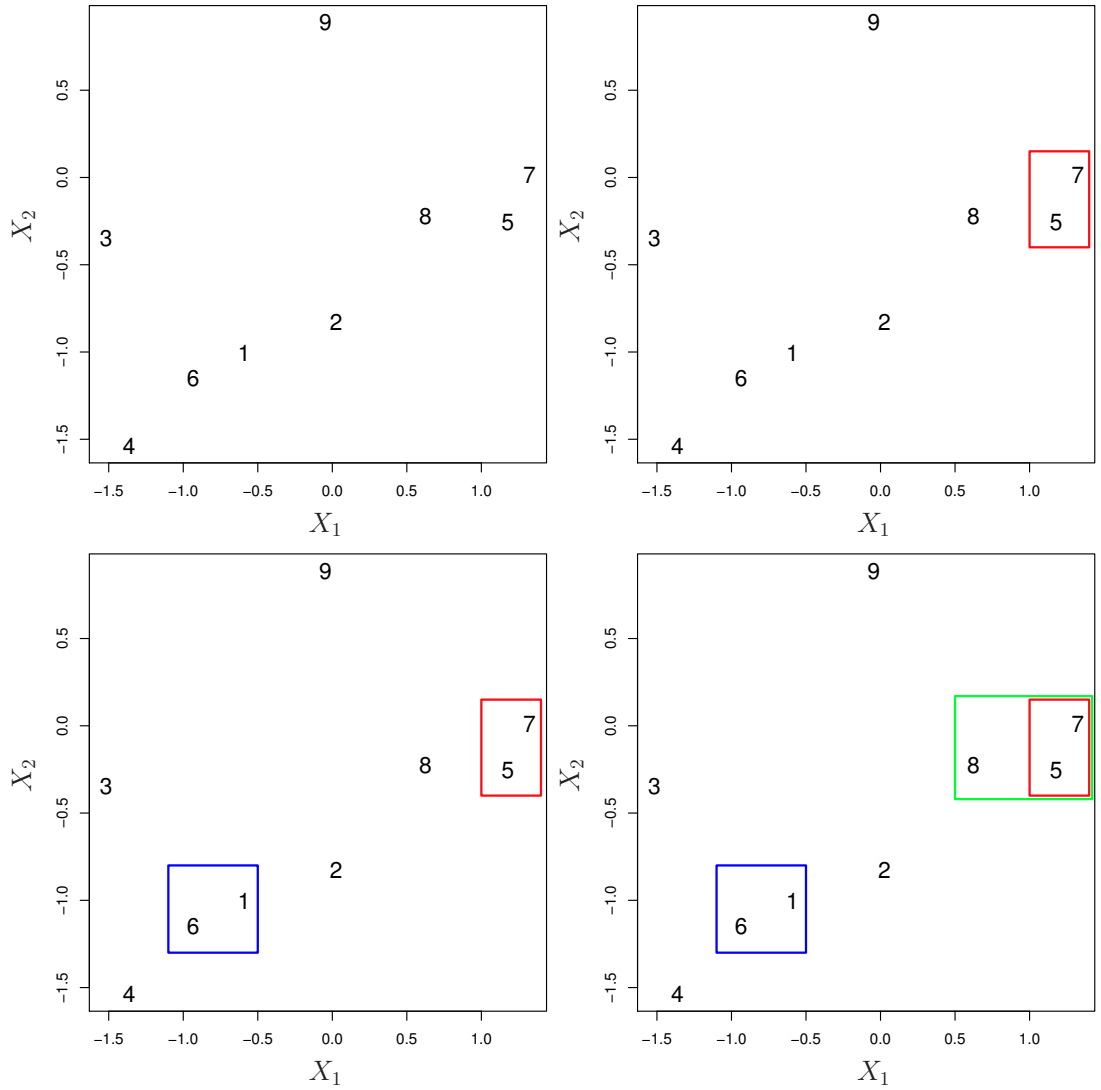
A dissimilarity measure on a set X is a function $d : X \to \mathbb{R}_+$ verifying the following properties

1. $d(x, x) = 0$
2. $d(x, y) = d(y, x)$ *[symmetry]*

Hierarchical clustering

- Hierarchical clustering methods do not require the user to supply a number of clusters, but only a way to compute the dissimilarity between two clusters, based on dissimilarities among the observations in the two clusters.
- They produce hierarchical representations in which the clusters at each level of the hierarchy are created by merging clusters at the next lower level.
- At the lowest level, each cluster contains a single observation. At the highest level there is only one cluster containing all of the data.

Example



Principle

- Agglomerative clustering algorithms begin with every observation representing a singleton cluster.
- At each of the $n - 1$ steps the closest (most similar) two clusters are merged into a single cluster, producing one less cluster at the next higher level.
- Therefore, a measure of dissimilarity between two clusters (groups of observations) must be defined.

Distances between clusters

- Let G and H represent two groups. The dissimilarity $d(G, H)$ between G and H is computed from the set of pairwise observation dissimilarities $d_{ii'}$ where one member of the pair i is in G and the other i' is in H.
- Three main definitions:

  Single linkage

$$d(G, H) = \min_{i \in G, i' \in H} d_{ii'}$$

  Complete linkage

$$d(G, H) = \max_{i \in G, i' \in H} d_{ii'}$$

  Average linkage

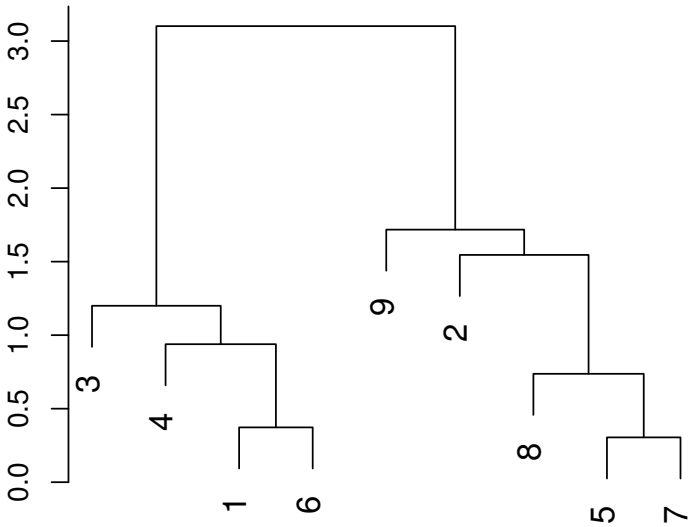
$$d(G, H) = \frac{1}{|G||H|} \sum_{i \in G, i' \in H} d_{ii'}$$

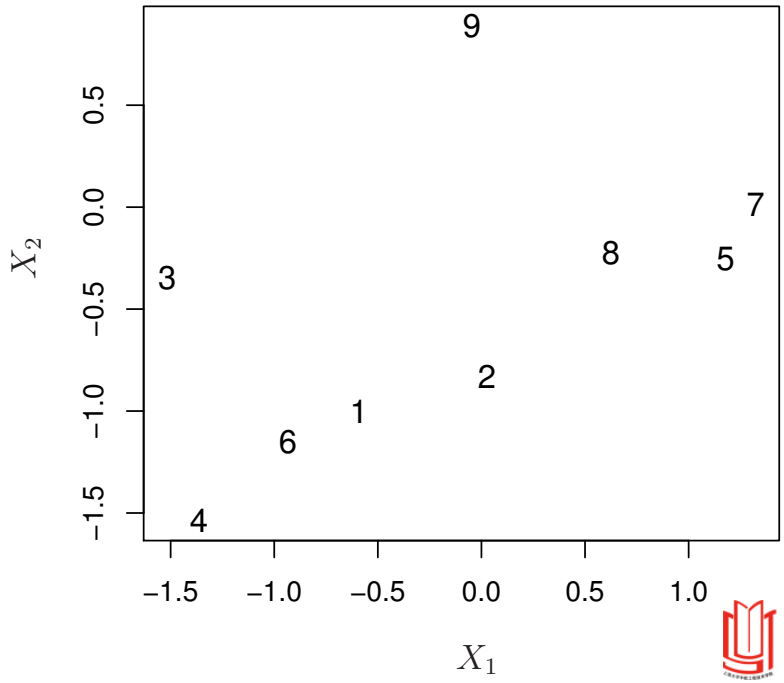
Criteria for selecting a distance measure

- If the data dissimilarities $d_{ii'}$ exhibit a strong clustering tendency, with each of the clusters being compact and well separated from others, then all three methods produce similar results. To the extent this is not the case, results will differ.
- Single linkage tends to created elongated clusters due to the chaining effect (observations linked by a series of close intermediate observations).
- In contrast, complete linkage tends to produce compact and small clusters.
- Group average clustering represents a compromise between the two extremes of single and complete linkage: it attempts to produce relatively compact clusters that are relatively far apart.

Dendrogram

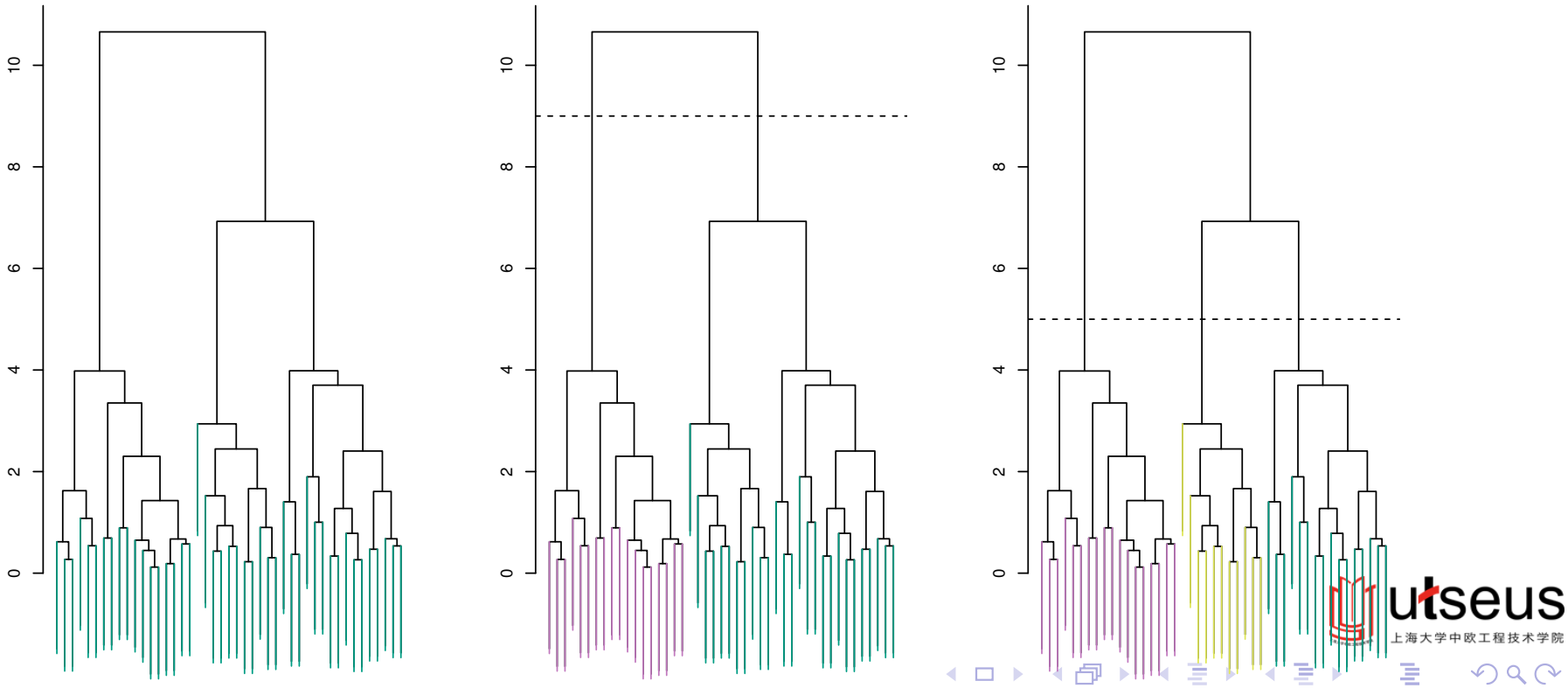
The hierarchy can be plotted as a tree such that the height of each node is proportional to the value of the intergroup dissimilarity between its two sons. The terminal nodes representing individual observations are all plotted at zero height. This is called a dendrogram.





Cutting the dendrogram

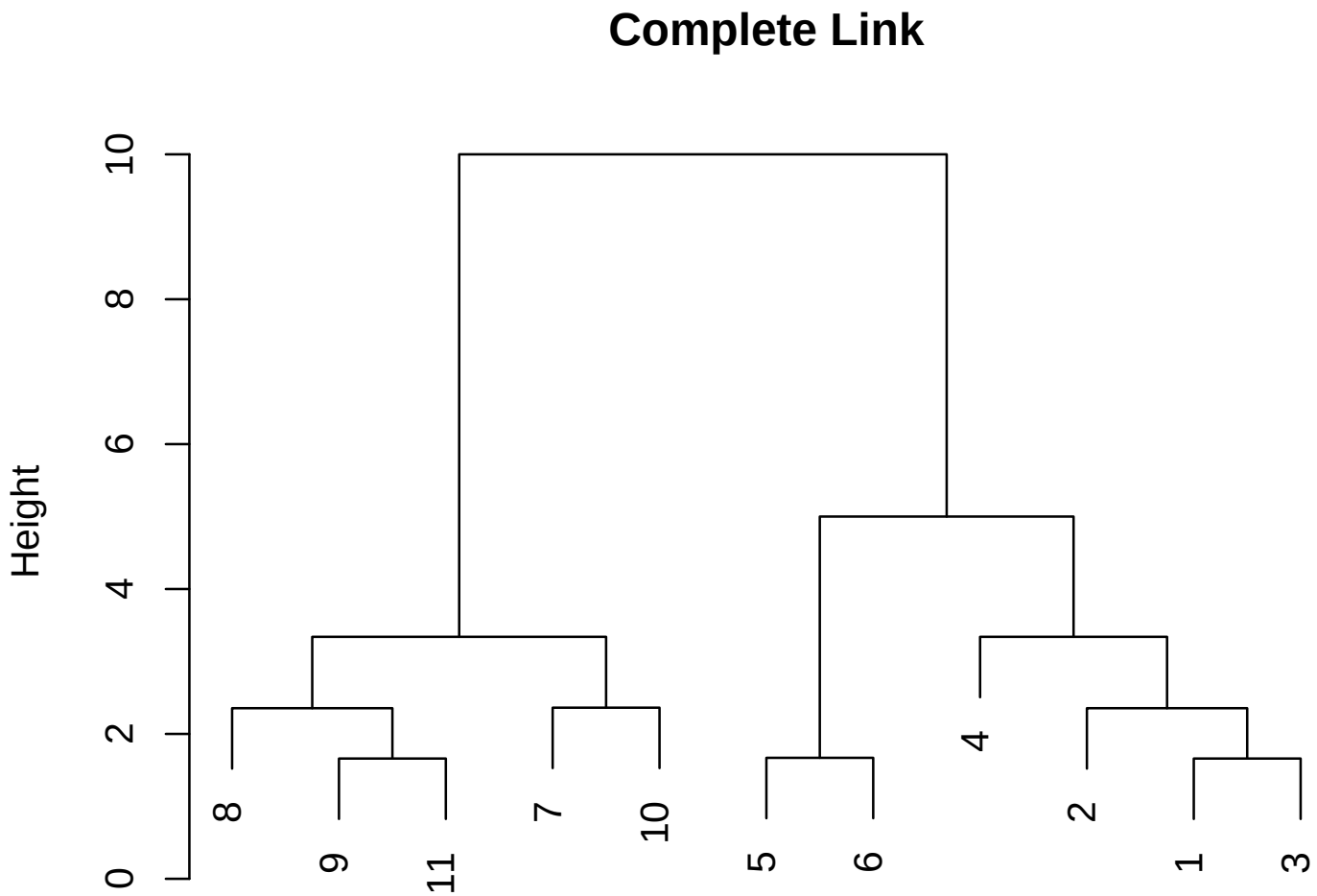
Cutting the dendrogram horizontally at a particular height partitions the data into disjoint clusters. Groups that merge at high values, relative to the merger values of the subgroups contained within them lower in the tree, are candidates for natural clusters.



Example in R: Butterfly data

```
D<-dist(x)

h<-hclust(D,method="complete")

plot(h,main="Complete Link")
```

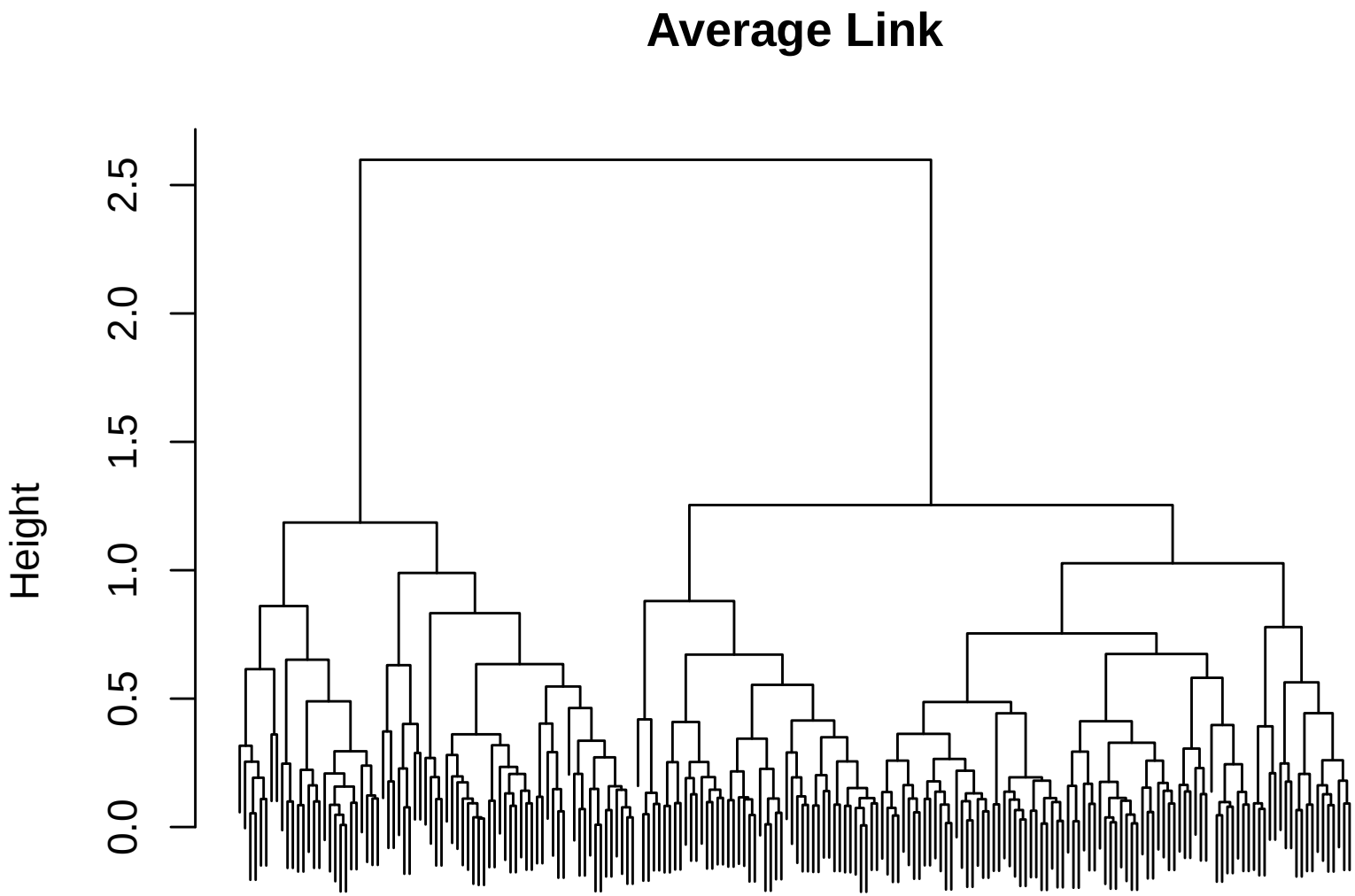
Result



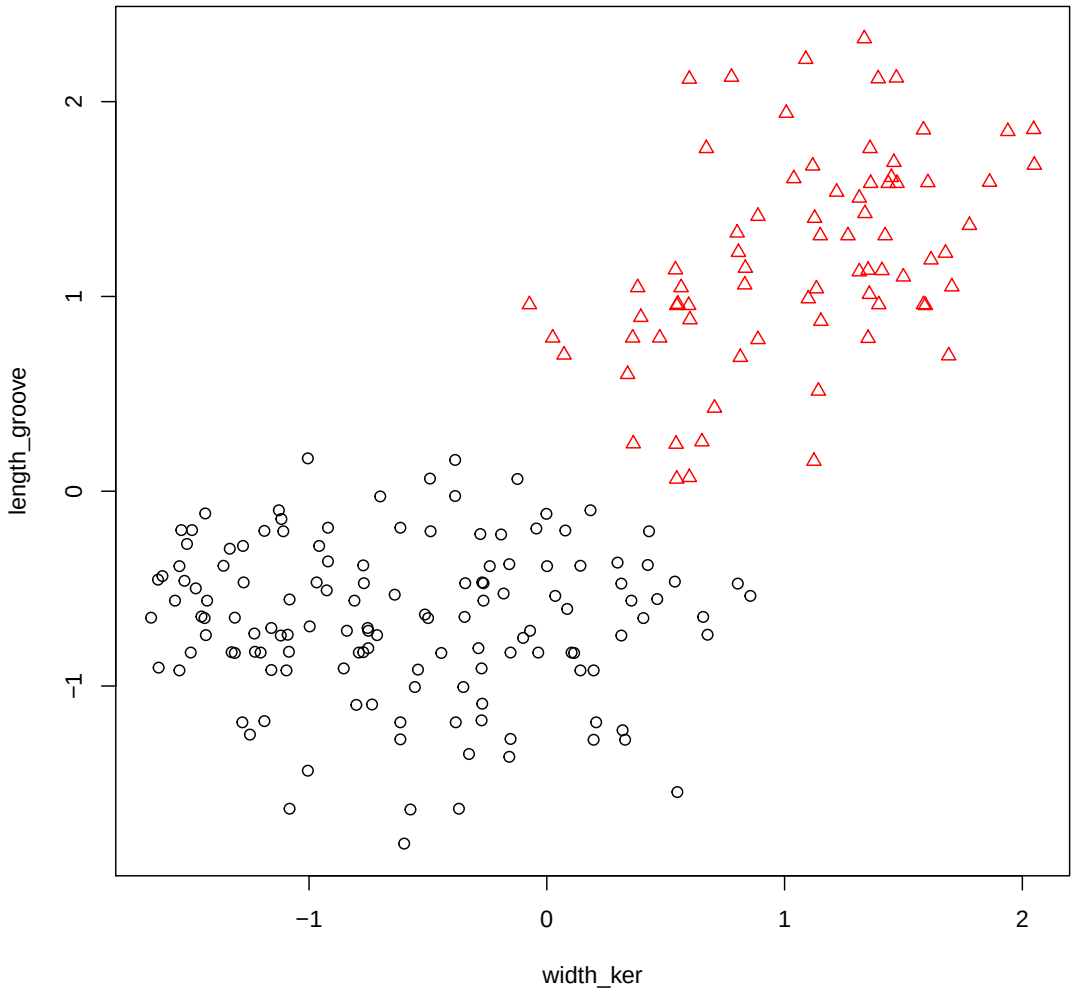
Example in R: Seeds data

```
D<-dist(x[,c(5,7)])

h<-hclust(D,method="average")
plot(h,labels=FALSE,main="Average Link")

y<-cutree(h, k = 2)
plot(data[,c(5,7)],col=y,pch=y)

y<-cutree(h, k = 3)
plot(data[,c(5,7)],col=y,pch=y)
```

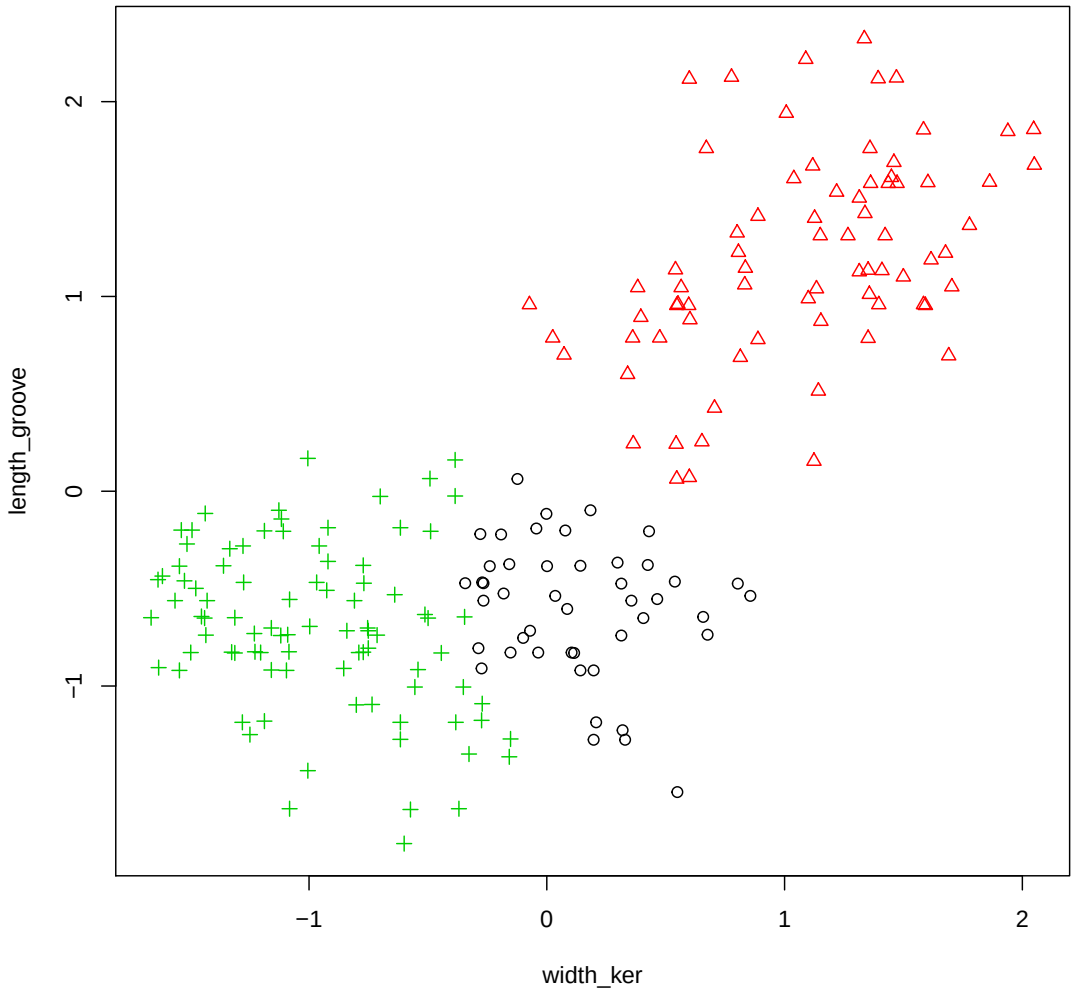
Result: dendrogram



Result: partition in 2 clusters



Result: partition in 3 clusters



Derivation of the FCM algorithm (1/2)

- From $\beta u_{ik}^{\beta-1} d_{ik}^2 - \lambda = 0$, we get

$$u_{ik} = \left(\frac{\lambda}{\beta d_{ik}^2} \right)^{1/(\beta-1)} = \lambda^{1/(\beta-1)} \beta^{1/(\beta-1)} d_{ik}^{-2/(\beta-1)}$$

- From $\sum_k u_{ij}$,

$$\lambda^{1/(\beta-1)} \beta^{1/(\beta-1)} \sum_k d_{ik}^{-2/(\beta-1)} \Rightarrow \lambda^{1/(\beta-1)} = \frac{1}{\beta^{1/(\beta-1)} \sum_k d_{ik}^{-2/(\beta-1)}}$$

- Finally,

$$u_{ik} = \frac{d_{ik}^{-2/(\beta-1)}}{\sum_\ell d_{i\ell}^{-2/(\beta-1)}}$$

Back

Derivation of the FCM algorithm (2/2)

- We have

$$I(v_k) = \sum_{i=1}^{n} u_{ik}^{\beta} (x_i - v_k)^T (x_i - v_k)$$

$$= -2 \left(\sum_i u_{ik}^{\beta} x_i \right)^T v_k + \left(\sum_i u_{ik}^{\beta} \right) v_k^T v_k + C$$

- Consequently,

$$\frac{\partial I(v_k)}{\partial v_k} = 2 \left(\sum_i u_{ik}^{\beta} x_i \right) + 2 v_k$$

and

$$\frac{\partial I(v_k)}{\partial v_k} = 0 \Leftrightarrow v_k = \frac{\sum_{i=1}^{n} u_{ik}^{\beta} x_i}{\sum_{i=1}^{n} u_{ik}^{\beta}}$$

Back