BD017 - Clustering and Classification with Applications in R

Berthold Lausen and Osama Mahmoud

Essex Big Data and Analytics Summer School
8th September 2016

Overview

What is multivariate data?

Our first example is a relatively small data set of anthropometric measurements - chest, waist and hip - of 10 man and 10 woman.

chest	*waist*	*hip*	*gender*	*chest*	*waist*	*hip*	*gender*
34	30	32	male	36	24	35	female
37	32	37	male	36	25	37	female
38	30	36	male	34	24	37	female
36	33	39	male	33	22	34	female
38	29	33	male	36	26	38	female
43	32	38	male	37	26	37	female
40	33	42	male	34	25	38	female
38	30	40	male	36	26	37	female
40	30	37	male	38	28	40	female
41	32	39	male	35	23	35	female

R data set **measure**: chest, waist and hip measurements in inches of 20 individuals (10 male and 10 female)
Hothorn & Everitt (2011, table 1.2).

Possible questions

1) Can we summarise the three measurements in one meaningful variable for each individual?
2) Can we identify groups of similar individuals?
3) Can we construct a meaningful classification (discrimination) rule for gender using the three anthropometric measurements?

Answers:

1) Can we summarise the three measurements in one meaningful variable for each individual?

Principal component analysis provides an answer (not covered in the course) and uses the three dimensional observations of chest, waist and hip measurements.

Answer: 2)

2) Can we identify groups of similar individuals?

Clustering methods (unsupervised learning) as *hierarchical clustering and k-means clustering* give possible solutions. Clustering methods use the three dimensional observations of chest, waist and hip measurements as information on each individual, but is assuming that we have **no further prior information** on groupings or clusters of the observations.

Answers: 3)

3) Can we construct a meaningful classification (discrimination) rule for gender using the three anthropometric measurements?

Classification methods (supervised learning) as *linear discriminant analysis, k-nearest neighbours, logistic classification and tree-based classification methods* provide possible solutions. The classification methods use the three dimensional observations of chest, waist and hip measurements as information on each individual and the gender variable as additional **binary class variable** of interest. The classification rule allows to predict the class membership of future observations with unknown gender.

Romano-British pottery data

Al2O3	*Fe2O3*	*MgO*	*CaO*	*Na2O*	*K2O*	*TiO2*	*MnO*	*BaO*	*kiln*
18.8	9.52	2.00	0.79	0.40	3.20	1.01	0.077	0.015	1
16.9	7.33	1.65	0.84	0.40	3.05	0.99	0.067	0.018	1
18.2	7.64	1.82	0.77	0.40	3.07	0.98	0.087	0.014	1
16.9	7.29	1.56	0.76	0.40	3.05	1.00	0.063	0.019	1
17.8	7.24	1.83	0.92	0.43	3.12	0.93	0.061	0.019	1
18.8	7.45	2.06	0.87	0.25	3.26	0.98	0.072	0.017	1
16.5	7.05	1.81	1.73	0.33	3.20	0.95	0.066	0.019	1
18.0	7.42	2.06	1.00	0.28	3.37	0.96	0.072	0.017	1
15.8	7.15	1.62	0.71	0.38	3.25	0.93	0.062	0.017	1
14.6	6.87	1.67	0.76	0.33	3.06	0.91	0.055	0.012	1
13.7	5.83	1.50	0.66	0.13	2.25	0.75	0.034	0.012	1
14.6	6.76	1.63	1.48	0.20	3.02	0.87	0.055	0.016	1
14.8	7.07	1.62	1.44	0.24	3.03	0.86	0.080	0.016	1
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.

R data set **pottery**: Romano-British pottery data (Hothorn & Everitt, 2011, table 1.3) from Tubb et al. (1980).

Questions

Our second multivariate data set shows results of chemical measurements of Romano-British pottery from three regions (region one contains kiln 1; region two kiln 2 and 3; region 3 kiln 4 and 5). We can ask as possible questions:

1) Can we summarise the nine chemical measurements in one or two meaningful variables for each individual?
2) Can we identify groups with similar chemical profiles of the pottery?
3) Can we construct a meaningful classification (discrimination) rule for region or kiln using the nine chemical measurements?

Study on US air polution

	SO2	*temp*	*manu*	*popul*	*wind*	*precip*	*predays*
Albany	46	47.6	44	116	8.8	33.36	135
Albuquerque	11	56.8	46	244	8.9	7.77	58
Atlanta	24	61.5	368	497	9.1	48.34	115
Baltimore	47	55.0	625	905	9.6	41.31	111
Buffalo	11	47.1	391	463	12.4	36.11	166
Charleston	31	55.2	35	71	6.5	40.75	148
Chicago	110	50.6	3344	3369	10.4	34.44	122
Cincinnati	23	54.0	462	453	7.1	39.04	132
Cleveland	65	49.7	1007	751	10.9	34.99	155
Columbus	26	51.5	266	540	8.6	37.01	134
Dallas	9	66.2	641	844	10.9	35.94	78
Denver	17	51.9	454	515	9.0	12.95	86
Des Moines	17	49.0	104	201	11.2	30.85	103
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.

R data set **USairpollution**: Air pollution in 41 US cities (Hothorn & Everitt, 2011, table 1.5) from Sokal & Rohlf (1981).

Study on US air polution

Our third multivariate data set shows data of a study on air polution with following variables:

SO2: SO_2 content of air in micrograms per cubic metre;
temp: average annual temperature in degrees fahrenheit;
...
predays: average number of days with precipitation per year.

We may fit a **multiple regression model** with SO_2 as response and *temp* to *predays* as predictor variables.
An alternative is a **tree-based regression model.**
Moreover, we may investigate similar groups of cities (**clustering methods**) or
if we can use the variables *temp* to *predays* to classify the cities into a group with high pollution and another with low pollution (**classification methods**).

Basic Concepts

Covariance

The *covariance* of two random variables X_i and X_j is a measure of linear dependence and defined by:

$$Cov(X_i, X_j) = E\left[(X_i - \mu_i)(X_j - \mu_j)\right]$$

with $\mu_i = E(X_i)$ and $\mu_j = E(X_j)$.
For $i = j$ we observe that the covariance is the variance of the random variable X_i.
The covariance of two random variables X_i and X_j is denoted by $\sigma_{i,j} = Cov(X_i, X_j)$ and the variance of a random variable X_i by $\sigma_i^2 = E\left[(X_i - \mu_i)^2\right]$.
Larger absolute values of the covariance imply a higher degree of linear dependence between the two variables.

Covariance matrix

For q-dimensional multivariate random vectors $\mathbf{X} = (X_1, \ldots, X_q)$ we have q variances σ_i^2 for $i = 1, \ldots, q$ and $q(q-1)/2$ covariances.
The variances and covariances can be arrange in $q \times q$ symmetric matrix Σ:

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \sigma_{1,2} & \cdots & \sigma_{1,q} \\ \sigma_{2,1} & \sigma_2^2 & \cdots & \sigma_{2,q} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{q,1} & \sigma_{q,2} & \cdots & \sigma_q^2 \end{bmatrix}$$

We call Σ the variance-covariance matrix or - simply - the covariance matrix.

Sample covariance matrix

For a sample of q-dimensional observations, $\mathbf{x_1}, \ldots, \mathbf{x_n}$ with $\mathbf{x_i} \in \mathbb{R}^q$ for $i = 1, \ldots, n$ we can estimate the covariance matrix by

$$\mathbf{S} = \frac{1}{n-1} \sum_{i=1}^{n} (\mathbf{x_i} - \bar{\mathbf{x}})(\mathbf{x_i} - \bar{\mathbf{x}})^t$$

with $\mathbf{x_i} = (x_{i,1}, x_{i,2}, \ldots, x_{i,q})$ is the q-dimensional (row) vector of the observations of the i-th individual and $\bar{\mathbf{x}} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x_i}$ is the mean vector of the n q-dimensional observations.
The diagonal of $\mathbf{S}$ contains the sample variances, s_i^2, of each variable (dimension) of the data set.

Correlation matrix

The covariance depends on the scale of each variable. To avoid such a dependence one can standardise the covariance by dividing by the product of the two standard deviations of the two variables X_i and X_j which defines the *correlation* $\rho_{i,j}$:

$$\rho_{i,j} = \frac{\sigma_{i,j}}{\sigma_i \sigma_j},$$

where $\sigma_i = \sqrt{\sigma_i^2}$.
The correlation is independent of the scales of the variables and lies between -1 and 1.

Correlation matrix

With q variables we have $q(q-1)/2$ distinct correlations which may be arranged in a symmetric $q \times q$ correlation matrix. The diagonal elements of the correlation matrix equal 1.
For a sample of q-dimensional observations we estimate the correlation matrix by the matrix with the pairwise Pearson's correlation coefficients and denote the sample correlation matrix by **R**.

We can rewrite **R** in terms of the sample covariance matrix **S**:

$$\mathbf{R} = \mathbf{D}^{-1/2}\mathbf{S}\mathbf{D}^{-1/2},$$

where $\mathbf{D}^{-1/2} = diag(1/s_1, \ldots, 1/s_q)$ and $s_i = \sqrt{s_i^2}$ denotes the sample standard deviation of variable i.

Euclidean distance

Often we are interested to measure how similar or how close are the multivariate observations. For example cluster analysis we use *distance* measures between the observations.
A common distance measure is the *Euclidean distance*:

$$d_{i,j} = \sqrt{\sum_{k=1}^{q} \left(x_{i,k} - x_{j,k}\right)^2},$$

where $x_{i,k}$ and $x_{j,k}$ are the values of the k-th variable of observations i and j.

The **dist** function in R computes the Euclidean distance.

R: Sample covariance matrix

Using R to calculate the covariance matrix for the **measure** data set.

```
> cov(measure[, c(''chest'', ''waist'', ''hips'')])
          chest     waist     hips
 chest 6.631579 6.368421 3.000000
 waist 6.368421 12.526316 3.578947
 hips 3.000000 3.578947 5.944737
```

For the females:

```
> cov(subset(measure, gender == ''female'')[,
+   c(''chest'', ''waist'', ''hips'')])
          chest     waist     hips
 chest 2.277778 2.166667 1.555556
 waist 2.166667 2.988889 2.755556
 hips 1.555556 2.755556 3.066667
```

For the males?

R: Correlation matrix

Using R to calculate the correlation for the **measure** data set.

```
> cor(measure[,c(''chest'', ''waist'',
''hips'')])
        chest     waist     hips
chest 1.0000000 0.6987336 0.4778004
waist 0.6987336 1.0000000 0.4147413
hips 0.4778004 0.4147413 1.0000000
```

R: Euclidean distance

Using R to calculate the distance matrix for the observations of the **measure** data set.

```
> x <- dist(scale(measure[, c(''chest'', ''waist'', ''hips'')],
+   center = FALSE))
>  as.dist(round(as.matrix(x), 2)[1:9, 1:9])
     1    2    3    4    5    6    7    8
 2 0.17
 3 0.15 0.08
 4 0.22 0.07 0.14
 5 0.11 0.15 0.09 0.22
 6 0.29 0.16 0.16 0.19 0.21
 7 0.32 0.16 0.20 0.13 0.28 0.14
 8 0.23 0.11 0.11 0.12 0.19 0.16 0.13
 9 0.21 0.10 0.06 0.16 0.12 0.11 0.17 0.09
```

Note, that we have scaled (R function **scale**) all observations by the standard deviations.

Cluster Analysis

K-means clustering

The chapter introduces K-means clustering and the following hierarchical clustering methods (see chapter 10 James, Witten, Hastie & Tibshirani, pages 385-413).

In contrast to classification problems we have learning data with clustering information, but the aim of the cluster analysis is to asses, if clusters explain the data and to estimate how many clusters explain the observed data.

The machine learning community uses the term *unsupervised learning* for cluster analysis and *supervised learning* for classification.

K-means clustering model

K-means clustering aims to partition a set of *n* observations with *p* dimensional observations into *K* distinct and non-overlapping clusters.

To do this one has to specify *k*, the number of clusters, first. Figure 10.5 (James, Witten, Hastie & Tibshirani, page 387) illustrates the difficulty to decide, if one needs 2, 3 or 4 clusters for a simulated sample of 2 dimensional observations.

K-means clustering model

A *partition* $C_1, C_2, \ldots, C_K$ of a set of indices of n observations into K clusters satisfies two properties:

1) $C_1 \cup C_2 \cup \ldots \cup C_K = \{1, 2, \ldots, n\}$; i.e. each observation belongs to at least one of the K clusters.
2) $C_k \cap C_{k'} = \emptyset$ for all $k \neq k'$; i.e. the clusters are non-overlapping.

For example, if the i-th observation is in k-th cluster, then $i \in C_k$. The idea of K-means clustering is that we can explain

the observations by K groups of observations which cluster around their group means. We aim to estimate a clustering, i.e. partition, of the n observations which has a *within-cluster variation* as small as possible.

K-means clustering objective function

The within-cluster variation of a cluster C_k is a measure $W(C_k)$ of the amount by which the observations differ from each other in cluster C_k.

Therefore, we aim minimise the overall within-cluster variation which is measured by the sum of all within-cluster variations:

$$\text{minimise}_{C_1,\ldots,C_K} = \left(\sum_{k=1}^{K} W(C_k) \right). \tag{1}$$

K-means clustering objective function

The minimisation of the within-cluster variation is a straightforward suggestion, but to do this we have to decide on the definition of the within-cluster variation of a cluster C_k.

Assuming that we have quantitative p-variate observations a common choice is the squared Euclidean distance:

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} \left(x_{i,j} - x_{i',j} \right)^2 , \qquad (2)$$

with $|C_k|$ denoting the size or number of observations in the k-th cluster.

K-means clustering objective function

We combine the overall within-cluster variation with the definition of the squared Euclidean distance and define the optimisation criterion for K-means clustering by:

$$\text{minimise}_{C_1,\ldots,C_K} = \left(\sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} \left(x_{i,j} - x_{i',j} \right)^2 \right). \qquad (3)$$

How can we minimise the objective function?

There are nearly K^n possibilities to allocate n observations to K distinct clusters.

K-means clustering algorithm

The following algorithm is relatively simple and guarantees to find a local minimum of the objective function (cf. James, Witten, Hastie & Tibshirani, 2013, page 388):

1) As initial cluster assignment randomly assign a number from 1 to K to each of the n observations.
2) Iterate until the cluster assignments converge, i.e. stop changing:
   a) For each of the K clusters compute the cluster centroid, which are the p dimensional mean vectors of the observations of each cluster.
   b) Assign each observation to the cluster whose centroid is *closest*, where *closest* is defined by the Euclidean distance.

K-means clustering algorithm

$$\frac{1}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{p}\left(x_{i,j}-x_{i',j}\right)^2 = 2\sum_{i\in C_k}\sum_{j=1}^{p}\left(x_{i,j}-\overline{x_{k,j}}\right)^2, \qquad (4)$$

where $\overline{x_{k,j}} = \frac{1}{|C_k|}\sum_{i\in C_k} x_{i,j}$ is the mean for feature j in cluster k.

The centroid of cluster k is given by the vector of the p feature means:

$$\left[\begin{array}{c} \frac{1}{|C_k|}\sum_{i\in C_k} x_{i,1} \\ \frac{1}{|C_k|}\sum_{i\in C_k} x_{i,2} \\ \vdots \\ \frac{1}{|C_k|}\sum_{i\in C_k} x_{i,p} \end{array}\right].$$

K-means clustering algorithm

The K-means clustering algorithm starts with the cluster means which minimise equation (4).

Reassigning observations to centroids which are closer than the centroid of the current cluster reduces the squared Euclidean distance of the objective function: $\sum_{j=1}^{p}\left(x_{i,j} - \overline{x_{k,j}}\right)^2$.

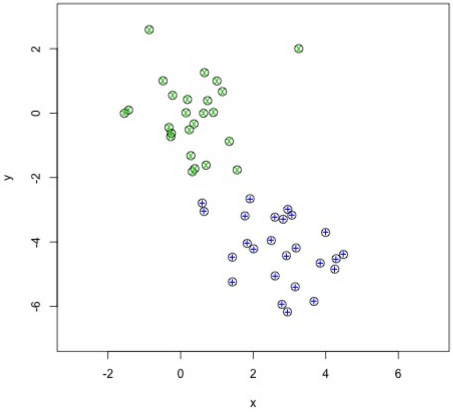
Consequently, the algorithm reduces the objective function in each iteration.

R: K-means clustering

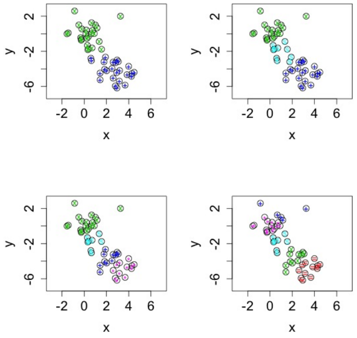
Using R we simulate two groups with bivariate normal distributed observations and apply K-means clustering with $K = 2$.

```
> set.seed(02022015)
> x <- matrix(rnorm(50*2), ncol=2)
> x[1:25,1] <- x[1:25,1]+3
> x[1:25,2] <- x[1:25,2]-4
> plot(x[1:25,],xlim=c(-3,7),ylim=c(-7,3),col=1,cex=1.5,xlab="x",ylab="y")
> points(x[26:50,],xlim=c(-3,7),ylim=c(-7,3),col=1,cex=1.5)
> my.km <- kmeans(x,centers=2,nstart=20)
> my.km$cluster
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2
> points(x[my.km$cluster==1,],xlim=c(-3,7),ylim=c(-7,3),col=3,pch="x",cex=1)
> points(x[my.km$cluster==2,],xlim=c(-3,7),ylim=c(-7,3),col=4,pch="+",cex=1)
```

R: *K*-means clustering



R: K-means clustering



The resulting K-means clusterings for $K \in \{2, 3, 4, 5\}$.

Hierarchies, ultrametrics and additive-tree metrics

K-means clustering comes with the disadvantage that one has to specify the number of clusters K.

Moreover, we assume that the data is generated by K different distributions, for example K different multivariate normal distributions with unknown mean vectors and and covariance matrices.

Hierarchical clustering fits a hierarchy to the n observations which allows for $n - 2$ non trivial clusters; i.e. clusters which have more than one observation and do not contain all observations.

Hierarchy and distance

A *hierarchy* H is a partially ordered set of subsets of the set of n indices or objects $O = \{1, 2, \dots n\}$:

$$H = \{C_1, C_2, \dots C_K\}$$

with $C_k \subseteq O$, moreover $C_k \subset C_{k'}$ or $C_k \supset C_{k'}$ or $C_k \cap C_{k'} = \emptyset$ for $k, k' \in \{1, 2, \dots K\}, k \neq k'$.

A *distance* on a set O is a function $d : O \times O \to \mathbb{R}$ and for $x, y, z \in O$ following conditions hold:

1) $d(x, y) \geq 0$ (nonnegativity);
2) $d(x, y) = 0$, if and only if $x = y$;
3) $d(x, y) = d(y, x)$ symmetry;
4) $d(x, z) \leq d(x, y) + d(y, z)$ triangle inequality.

Ultrametrics and additive-tree metrics

An *ultrametric* is defined by condition 1-3 and the so-called *ultrametric inequality* (and *three point condition*):

$$d(x,z) \leq max(d(x,y), d(y,z)),$$

for all $x, y, z \in O$.

An *additive tree metric* is defined by condition 1-3 and the so-called *additive tree inequality* (and *four point condition*):

$$d(u,v) + d(x,y) \leq max\left[d(u,x) + d(v,y), d(u,y) + d(v,x)\right],$$

for all $u, v, x, y \in O$.

Ultrametrics and additive-tree metrics

An **ultrametric** can be graphically represented by a **dendrogram**. The level of a node (cluster) defines the value of the ultrametric.

A given level of the ultrametric or dendrogram defines a partition of O into distinct clusters.

An **additive tree metric** can be defined by a weighted tree graph (**additive tree**). The sum of the nonnegative branch weights define the value of the additive tree metric and each interior branch define a bi-partition of O.

Examples: Hierarchical clustering

One could compute the **Euclidean distance** between n observations, fit an ultrametric to the measured distance and decide on the level of the ultrametric which defines a partition of the set O into K subsets of clusters.

Or, one estimates the **correlation matrix** R of the p dimensions as a similarity measures and uses $1 - r_{i,j}$ as distance measure, where $r_{i,j}$ correlation between the i-th and j-th dimension or variable.

Algorithm: Agglomerative hierarchical clustering

1) Begin with the set of n observations O and a distance (or dissimilarity) measure for all $n \times (n-1)/2$ pairs of objects;
2) For $i = n, n-1, \ldots, 2$:
    a) Examine all pairwise inter-cluster dissimilarities between the i clusters and identify the pair of clusters which are least dissimilar (or, most similar). The dissimilarity between the two joint clusters gives the height of the dendrogram at which the two clusters are joint.
    b) Update the distance between the new joint cluster and all other clusters with a given update or so called linkage function (for example complete-,single-,average- or centroid-linkage).

Linkage functions: Agglomerative hierarchical clustering

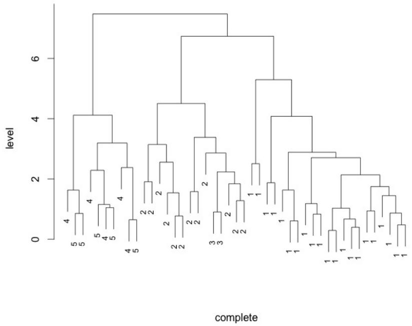
Complete or Maximal intercluster dissimilarity. Compute all dissimilarities between the two clusters C_i and C_j and record the largest of these dissimilarities as new value of the updated dissimilarity.

Single or Minimal intercluster dissimilarity. Compute all dissimilarities between the two clusters C_i and C_j and record the smallest of these dissimilarities as new value of the updated dissimilarity.

Average or Mean intercluster dissimilarity. Compute all dissimilarities between the two clusters C_i and C_j and record the average of these dissimilarities as new value of the updated dissimilarity.

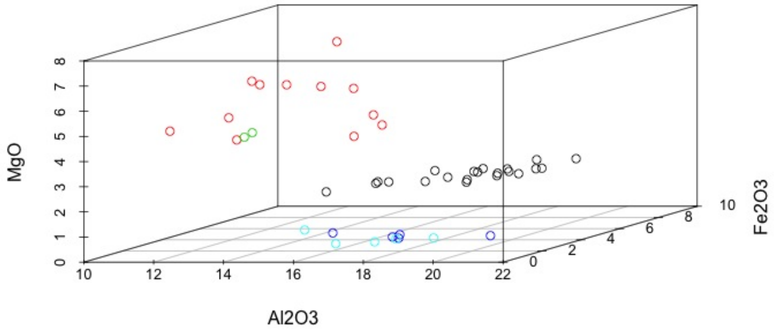
Centroid. Dissimilarity between centroid for cluster A and cluster B. Centroids can result in inversions (violation of ultrametric ineqality).

Complete-linkage as tree graph



Agglomerative hierarchical clustering with complete-linkage. .

Example: Pottery Data

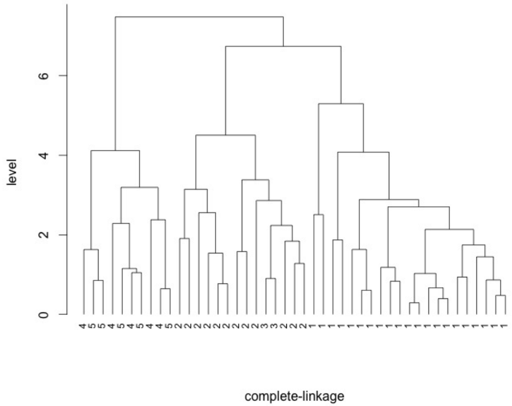


R: Agglomerative hierarchical clustering

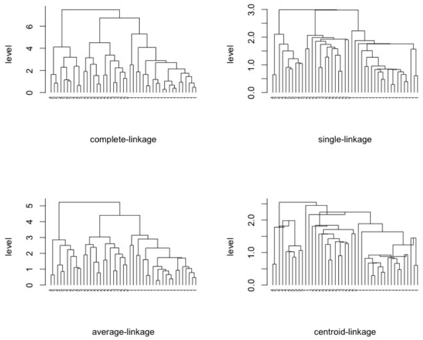
R data set **pottery**: Romano-British pottery data (Hothorn & Everitt, 2011, table 1.3) from Tubb et al. (1980).

```
> data(''pottery'', package = ''HSAUR2'')
> s.pottery <- scale(pottery[,c(1:9)])
> d.pottery <- dist(s.pottery)
> plot(hclust(d.pottery,
method=complete"),labels=(as.character(pottery$kiln)),
+   cex=.75, main=,xlab=complete-linkage",ylab=level",sub=)
> plot(hclust(d.pottery,
method=complete"),labels=(as.character(pottery$kiln)),
+   hang=-1,cex=.75,
main='''',xlab=''complete-linkage'',ylab=''level'',sub='''')
```

R: Agglomerative hierarchical clustering

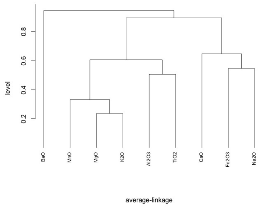


R: Agglomerative hierarchical clustering of observations



Agglomerative hierarchical clustering with complete-, single-, average- and centroid-linkage.

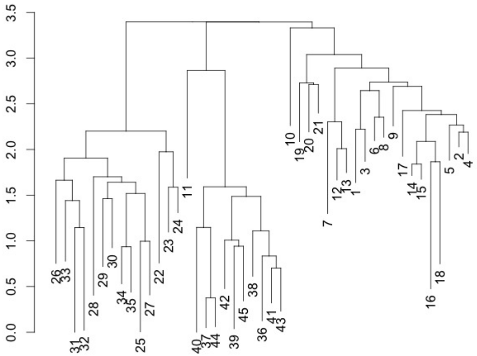
R: Agglomerative hierarchical clustering of variables



Agglomerative hierarchical clustering of the 9 variables (dimensions) with average–linkage of $1 - r^2_{i,j}$ as distance measure.

```
> d.pottery <- as.dist(1- cor(pottery[,c(1:9)])**2)
> plot(hclust(d.pottery, method=''average''),
+ hang=-1,cex=.75,
main='''',xlab=''average-linkage'',ylab=''level'',sub='''')
```

R: Additive tree of pottery data



Stability/confidence limits

Hierarchical cluster methods and tree methods are often deterministic algorithms and provide always a hierarchical clustering or a tree graph as outcome. Consequently, it is important to assess the underlying hierarchical structure or tree structure using some kind of stability analysis.

Common approaches are to apply resampling techniques as bootstrap and use the frequency of a partition or a cluster derived by the bootstrap samples as estimate of a naive confidence interval or stability measure.
Another idea is to model the distance data by an additive error model and to use a parametric bootstrap approach; for example $d(x, y) = d_u(x, y) + e(x, y)$ with d_u denoting the ultrametric and $e(x, y)$ i.i.d. random variables for $x, y \in O$ (see Degens, Lausen & Vach, 1990, Trees and Hierarchical Structures, Lecture Notes in Biomathematics Volume 84, 1990, pp 9-42).

Break

Classification

Classification or supervised learning

We introduce classification methods as logistic regression, Bayes classifier, and linear discriminant analysis (LDA) following James, Witten, Hastie & Tibshirani (2013, chap. 4).

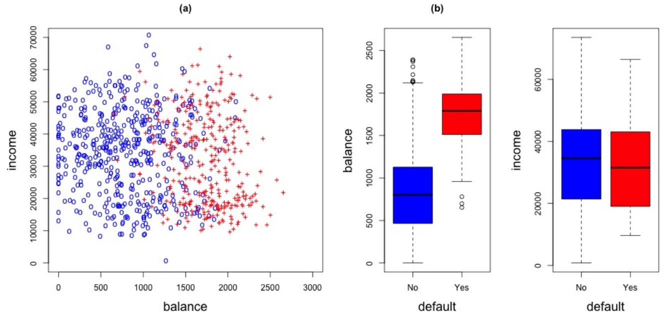
Logistic regression
Let $(x_1, y_1) \ldots (x_n, y_n)$ be a (learning) sample of n multivariate observations. The response variable y_i is a binary variable or a categorical variable with more than two categories. Our aim is to use the learning sample to construct a prediction rule of the predictor variables x_i for a future observations of the response variable y_i.

R: Logistic regression

R data set **Default** of the R package ÏSLR”. Simulated data set of credit card repayment defaults (James, Witten, Hastie & Tibshirani, 2013, p. 128).

```
> data(''Default'', package = ''ISLR)
> str(Default)
'data.frame': 10000 obs. of 4 variables:
 $ default: Factor w/ 2 levels ''No'',''Yes'': 1 1 1 1 1 1 1 ...
 $ student: Factor w/ 2 levels ''No'',''Yes'': 1 2 1 1 1 2 1 ...
 $ balance: num 730 817 1074 529 786 ...
 $ income : num 44362 12106 31767 35704 38463 ...
```

R: Logistic regression



(a) Default on monthly credit card payments: Scatter plot of the monthly credit card balance vs. annual income (no default - plot symbol blue ’o’; default - plot symbol red ’+’);
(b) boxplots of balance and annual income by default.

R: Logistic regression

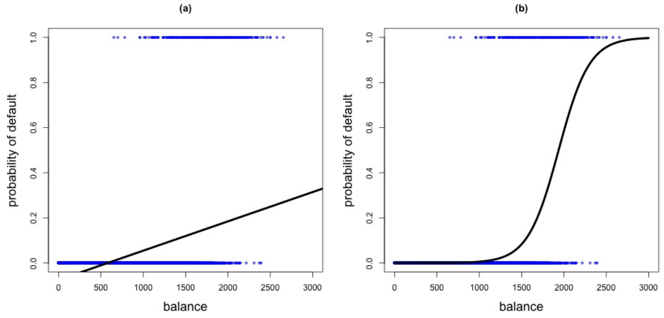
Figures indicates that the monthly credit card balance can be used to predict the individual defaults on the monthly payments.

We are interested to predict the conditional probability, $p(\text{balance})$, of a default on the monthly credit card repayments:

$$p(\text{balance}) = P(\text{default} = \text{"yes"}|\text{balance}). \tag{5}$$

As a classification rule one may predict default for $p(\text{balance}) > 0.5$ or - as a more conservative rule - $p(\text{balance}) > 0.1$.

R: Logistic regression



(a) Scatter plot and fitted least square estimate of linear regression line: Default on monthly credit card payments as response variable and monthly credit card balance as predictor variable;
(b) Scatter plot and fitted maximum likelihood estimate of the logistic regression function: Default on monthly credit card payments as **binary** response variable and monthly credit card balance as predictor variable.

Logistic regression

We overcome this problem by using a nonlinear or generalised linear model - so called logistic regression - for binary or binomial response variables.
For example:

$$P(Y_i = 1|x_i) = \frac{e^{-10.65+0.0055\ x_i}}{1 + e^{-10.65+0.0055\ x_i}}$$

where y_i denotes the binary response variable default on monthly credit card payments (0 - no default; 1 - default) and x_i denotes the monthly credit card balance as predictor variable. The logistic regression line can be seen as an estimate of the probability of a default and stays within the parameter space $[0, 1]$ which makes sense.

Logistic regression

$Y_1, Y_2, \cdots, Y_n$ independently $B(1, p(x_i))$ distributed with $x = (x_1, \cdots, x_n) \in \mathbb{R}^n$. If $p(x_i)$ is given by

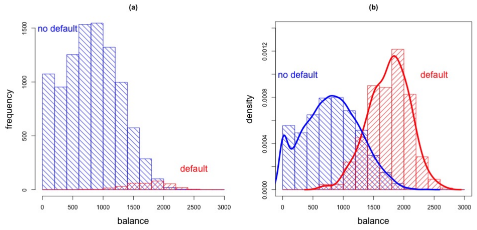
$$p(x_i) = \frac{e^{\beta_0 + \beta_1 x_i}}{1 + e^{\beta_0 + \beta_1 x_i}}$$

with $\beta_0, \beta_1 \in \mathbb{R}$, then Y_i are said to come from a *logistic regression model* and $p(x_i)$ is called the *logistic regression function*.

The *logit function*, $log\left(\frac{p(x_i)}{1-p(x_i)}\right)$, is linear in β_0 and β_1:

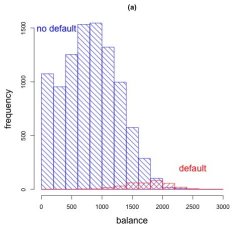
$$log\left(\frac{p(x_i)}{1 - p(x_i)}\right) = \beta_0 + \beta_1 x_i.$$

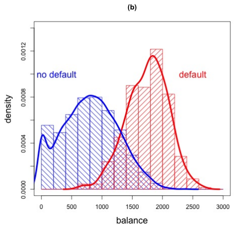
Bayes classifier



(a) Histogram of absolute frequencies of monthly credit card balance by the two classes no default or default on monthly credit card payments;
(b) Histogram of relative frequencies of monthly credit card balance and nonparametric kernel estimates of the probability density functions by the two classes no default or default on monthly credit card payments.

Bayes classifier





(a) illustrates the distribution of monthly credit card balance by the two classes no default or default on monthly credit card payments;
(b) illustrates the conditional distribution for the two classes no default or default on monthly credit card payments.

Bayes classifier

Suppose we want to classify an observation into one of K classes with $K \geq 2$. Let $f_k(X) = P(X = x|Y = k)$ denote the probability density function (pdf) of X for an observation that belongs to the k-th class. Let π_k be the **prior** or overall probability that Y comes from class k.
Consequently, $f_k(X)$ will be relatively large, if there is a high probability that an observation in the k-th class has $X = x$. On the other hand, $f_k(X)$ will be relatively small, if there is unlikely that an observation in the k-th class has $X = x$.
Using Bayes Theorem we get:

$$P(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)}. \tag{6}$$

Bayes classifier

Assuming a random sample of the population we estimate π_k by the relative frequencies of each class and assuming that we have a consistent estimate of the pdf $f_k(x)$ we can classify each observation by plugging the estimates in the Bayes formula (6) above.

We refer to $p_k(x) = P(Y = k|X = x)$ as the **posterior** probability of an observation belonging to class k given that $X = x$. The resulting classification rule or classifier is know as *Bayes classifier* which classifies an observation to the class with highest probability.

For the true, but in **general unknown**, π_k and $f_k(x)$ with $k = 1, \dots, K$ the *Bayes classifier has the lowest possible error rate*.

Linear discriminant analysis (LDA)

Assuming that we have one predictor variable and that $f_k(x)$ is the pdf of a normal distribution:

$$f_k(x) = \frac{1}{\sqrt{2\pi}\,\sigma_k} \exp\left[-\frac{1}{2\sigma_k^2}(x-\mu_k)^2\right],$$

where μ_k and σ_k^2 are the mean and the variance parameter of the normal distribution for the k-th class.

Assuming that $\sigma_1^2 = \sigma_2^2 = \cdots = \sigma_K^2 = \sigma^2$ equation (6) becomes

$$p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\,\sigma} \exp\left[-\frac{1}{2\sigma^2}(x-\mu_k)^2\right]}{\sum_{l=1}^{K} \pi_l \frac{1}{\sqrt{2\pi}\,\sigma} \exp\left[-\frac{1}{2\sigma^2}(x-\mu_l)^2\right]}. \qquad (7)$$

The Bayes classifier assigns an observation to the class k with largest $p_k(x)$.

Linear discriminant analysis (LDA)

Taking the logarithm of (7) and keeping parts of the equation which depend on k, this is equivalent to assigning the observation to the class were $\delta_k(x)$ is largest with

$$\delta_k(x) = x\,\frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\,\sigma^2} + \log(\pi_k). \tag{8}$$

.

Linear discriminant analysis (LDA)

For linear discriminant analysis we estimate $\mu_1, \ldots, \mu_K, \pi_1, \ldots, \pi_K$ and σ^2 by

$$\widehat{\mu_k} = \frac{1}{n_k} \sum_{\{i: y_i = k\}} x_i,$$

$$\widehat{\sigma^2} = \frac{1}{n - K} \sum_{k=1}^{K} \sum_{\{i: y_i = k\}} (x_i - \widehat{\mu_k})^2,$$

where n_k are the number of observations of class k and n is the number of all observations in the training sample;

$$\widehat{\pi_k} = \frac{n_k}{n}.$$

Linear discriminant analysis (LDA)

We derive the LDA classifier by plugging the estimates in equation (8) and assign an observation $X = x$ to class for which

$$\widehat{\delta_k}(x) = x\,\frac{\widehat{\mu_k}}{\widehat{\sigma^2}} - \frac{\widehat{\mu_k}^2}{2\,\widehat{\sigma^2}} + \log(\widehat{\pi_k}) \tag{9}$$

is largest.

R: Linear discriminant analysis (LDA)

Applying LDA to the Default data set and the computation of the confusion matrix.

```
> library(MASS)
> lda.Default <- lda(default ~ balance + income , data=Default)
> lda.Default
```

```
Call:
lda(default ~ balance + income, data = Default)
Prior probabilities of groups:
 No     Yes
0.9667 0.0333
Group means:
 balance income
No    803.9438 33566.17
Yes 1747.8217 32089.15
Coefficients of linear discriminants:
 LD1
balance 2.230835e-03
income  7.793355e-06
```

R: Linear discriminant analysis (LDA)

```
> default.predicted <- predict(lda.Default)$class
> table(default.predicted,Default$default)
default.predicted       No   Yes
                  No 9647 256
                  Yes   20   77
```

The predictions are presented as so called **confusion matrix**.

		True	default	status
		No	Yes	total
Predicted	No	9647	256	9903
default status	Yes	20	77	97
	total	9667	333	10000

We observe $20 + 256 = 276$ classification errors and an estimate of

misclassification error / **probability** : $276/10000 = 0.0276$;

sensitivity : $77/333 = 0.231$;

specificity : $9647/9667 = 0.998$.

Resampling methods

We introduce leave-one-out, *k*-fold cross-validation and bootstrap estimation of the misclassification error and other regression characteristics of interest.

In classification problems we are interested to estimate the *misclassification error err* of a given classification rule $\hat{c}$ for the class variable Y depending on a random vector of predictors X:

$$err\left(\hat{c}\right) = P\left(\hat{c}(\mathbf{X}) \neq Y\right) \ . \tag{10}$$

Our aim is to find a classifier with small misclassification error *err*. The Bayes classifier minimises *err*, but we do not know the underlying statistical model or parameters of the model. Consequently, we can not compute the Bayes classifier in applications in general.

Misclassification error

Let $(\mathbf{X}_1, Y_1), \dots (\mathbf{X}_n, Y_n)$ denote a *training sample* with $\mathbf{X}_i = (X_{i,1}, \dots, X_{i,m}) \in \mathbb{R}^m$, $Y_i \in \{0, 1\}$ and $i = 1, \dots, n$.

One possibility is to estimate the misclassification error *err* of the classification rule $\hat{c}$ using the training sample which was used to derive (to estimate) the classification rule $\hat{c}$. This estimate of the misclassification error is called the *training error rate* or *apparent error*:

$$\widehat{err_{apparent}}(\hat{c}) = \frac{1}{n} \sum_{i=1}^{n} I\left(\hat{c}(\mathbf{x_i}) \neq y_i\right), \tag{11}$$

where I denotes the indicator function which equals one, if the expression is true, and zero if it is false.

Misclassification error: Test sample

Sometimes we have an independent *test sample* $(\tilde{\mathbf{X}}_1, \tilde{Y}_1), \ldots (\tilde{\mathbf{X}}_{\tilde{n}}, \tilde{Y}_{\tilde{n}})$, where $\tilde{n}$ denotes the size of the test sample and we can compute the *test error* as estimator of the misclassification error:

$$\widehat{err_{test}}(\hat{c}) = \frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} I\left(\hat{c}(\tilde{\mathbf{x}}_\mathbf{i}) \neq \tilde{y}_i)\right). \tag{12}$$

The test error $\widehat{err_{test}}(\hat{c})$ is an unbiased estimator of the misclassification error of a classification rule trained on a sample of size n.

The apparent error $\widehat{err_{apparent}}(\hat{c})$ tends to underestimate the misclassification error.

Misclassification error: Validation set approach

1) divide the sample into two random partitions of equal size: a training sample of $n/2$ observations and a validation sample of $n/2$;
2) estimate the classification rule $\hat{c}$ on the training sample;
3) estimate the misclassification error *err* on the validation sample.

The resulting estimator of the misclassification error is unbiased for the misclassification error of a classification rule trained on a sample of size $n/2$, but exploits only 50% of the observations of the sample for the estimation of the classification rule and the misclassification error:

Misclassification error: Leave-one-out-cross-validation

Instead of creating two equal sized data sets - test and validation sample - we use a single observation as validation set and all remaining observations as training sample. Leaving all observation once out we get the LOOCV estimator of the misclassification error:

$$\widehat{err_{LOOCV}}(\hat{c}) = \frac{1}{n}\sum_{i=1}^{n} I\left(\hat{c}_{(i)}(\mathbf{x_i}) \neq y_i)\right), \tag{13}$$

where $\hat{c}_{(i)}$ denotes the classification rule estimated by the sample excluding the i-th observation.

Misclassification error: Leave-one-out-cross-validation

The LOOCV estimator $\widehat{err_{LOOCV}}(\hat{c})$ is an unbiased estimator of the misclassification error of a classification rule trained on a sample of size $n-1$:

$$E\left[\widehat{err_{LOOCV}}(\hat{c})\right] = \frac{1}{n}\sum_{i=1}^{n} E\left[I\left(\hat{c}_{(i)}(\mathbf{x_i}) \neq y_i\right)\right] = P\left(\hat{c}_{(.)}(\mathbf{X}) \neq Y\right).$$

Misclassification error: *k*-fold cross-validation

A disadvantage of LOOCV is that the $(\hat{c}_{(i)}$ are highly correlated. To reduce the correlation we partition the sample in k sets of approximately equal size:

$$U = \{1, 2, \ldots, n\} = \cup_{i=1}^{k} U_i \text{ and } \cap_{i=1}^{k} U_i = \emptyset.$$

The k-fold cross-validation estimator is defined as average of k estimators which use all k sets once as validation set and the data of the other $k - 1$ sets as corresponding training set:

$$\widehat{err_{CV}}(\hat{c}) = \frac{1}{k} \sum_{i=1}^{k} \frac{1}{|U_i|} \sum_{j \in U_i} I\left(\hat{c}_{(U_i)}(\mathbf{x_j}) \neq y_j\right), \tag{14}$$

where $\hat{c}_{(U_i)}$ denotes the classification rule estimated by the sample excluding the observations of set U_i.

Misclassification error: *k*-fold cross-validation

The k-fold cross-validation estimator $\widehat{err_{CV}}(\hat{c})$ is an unbiased estimator of the misclassification error of a classification rule trained on samples of approximately size of $n(1 - 1/k)$:

$$E\left[\widehat{err_{CV}(\hat{c})}\right] = \frac{1}{k}\sum_{i=1}^{k}\frac{1}{|U_i|}\sum_{j\in U_i} E\left[I\left(\hat{c}_{(U_i)}(\mathbf{x_j}) \neq y_j\right)\right] = P\left(\hat{c}_{(kCV)}(\mathbf{X}) \neq Y\right).$$

Misclassification error: The bootstrap

The (theoretical) bootstrap estimators are defined by the observed empirical distribution. Often we approximate these estimators by simulation using the empirical distribution.
The simulation is defined by :

1) generate B bootstrap samples of size n by sampling with replacement from the empirical distribution;
2) compute for each bootstrap sample U_i^*, $i = 1, \ldots, B$, a classifier $\hat{c}_{(U_i^*)}$ and a validation error using the observations of the so called out-of-bag (OOB) sample which is defined by all observations which are not in the i-th bootstrap sample.
3) the average of the B validation errors defines a bootstrap estimator of the misclassification error.

R: Misclassification error

Default data: 10 fold cross validation and bootstrap estimates of misclassification error.

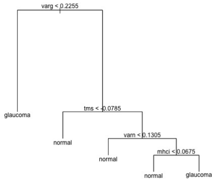
```
> library(ipred)

> mypredict.lda <- function(object, newdata)
+ predict(object, newdata = newdata)$class

> set.seed(25012015)
> errorest(default ~ ., data=Default, model=lda,
+   estimator = c(cv"),
+   est.para=control.errorest(k=10), predict= mypredict.lda)

Call:
errorest.data.frame(formula = default   ., data = Default, model
= lda,
 predict = mypredict.lda, estimator = c(cv"),
 est.para = control.errorest(k = 10))

 10-fold cross-validation estimator of misclassification error

Misclassification error: 0.0275
```

R: Misclassification error

```
> errorest(default   ., data=Default, model=lda,
+ estimator = c("boot"),
+ est.para=control.errorest(nboot=200), predict= mypredict.lda)

Call:
errorest.data.frame(formula = default ~ ., data = Default,
model = lda,
 predict = mypredict.lda, estimator = c("boot"), est.para =
control.errorest(nboot = 200))

 Bootstrap estimator of misclassification error
 with 200 bootstrap replications

Misclassification error: 0.0276
Standard deviation: 0
```

Tree-based classifiers

We use the **GlaucomaMVF** data set of the R package **ipred** to illustrate the construction of a decision tree. The data set has 170 observations in two classes (glaucoma patients, healthy subjects) and 66 predictors which are derived from a confocal laser scanning image of the optic nerve head, from a visual field test, a fundus photography and a measurement of the intraoccular pressure.



A classification tree for glaucoma diagnostics. The predictor variables measure characteristics of the morphology of the optic nerve head: varg - volume above reference global; tms - third moment superior; varn - volume above reference nasal; mhci - mean height contour inferior.

R: Tree-based classifiers

R stores the binary classification tree as follows:

```
node), split, n, deviance, yval, (yprob)
 * denotes terminal node

 1) root 170 235.700 glaucoma ( 0.50000 0.50000 )
   2) varg < 0.2255 77 63.160 glaucoma ( 0.85714 0.14286 ) *
   3) varg > 0.2255 93 94.170 normal ( 0.20430 0.79570 )
     6) tms < -0.0785 51 16.880 normal ( 0.03922 0.96078 ) *
     7) tms > -0.0785 42 56.690 normal ( 0.40476 0.59524 )
      14) varn < 0.1305 15 7.348 normal ( 0.06667 0.93333 ) *
      15) varn > 0.1305 27 36.500 glaucoma ( 0.59259 0.40741 )
        30) mhci < 0.0675 14 16.750 normal ( 0.28571 0.71429 ) *
        31) mhci > 0.0675 13 7.051 glaucoma ( 0.92308 0.07692 ) *
```

Tree-based classifiers

The underlying classification tree is build using recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has less than a given number of observations. Often cost complexity pruning combined with cross validation is used to identify the 'best' size of the tree.
Various splitting criteria are proposed in the literature. A popular one is based on the Gini index:

$$G = \sum_{k=1}^{K} \hat{p}_{m,k} \, (1 - \hat{p}_{m,k}),$$

where $\hat{p}_{m,k}$ is the estimated probability of class k at node m. The Gini index can be seen as a measure of total variance over K classes. Small values of the Gini index indicate the groups are dominated by one of the classes. The recursive partitioning algorithm choose splits at each step which result in the maximum reduction of the Gini index.

Advantages and disadvantages of trees

+ 1) Tree model is easy to understand and to explain.
+ 2) Humans are used to make binary decisions. Classification and regression trees can be seen a series of such binary decisions.
+ 3) The tree structure can be illustrated easily and supports the interpretation.
+ 4) Trees can handle predictors with different scales without introducing dummy variables.

- 1) Trees are unstable predictors or classifiers. Trees do not have the same predictive accuracies compared to other classifiers.

Bagging, random forests, boosting

Classification and regression trees (CART) suffer from high variance. *Bootstrap aggregation* (acronym *Bagging*) of classification trees is based on the idea that an average of unstable classifiers should result in a stable classifier.

A bagging tree classifier is computed by following steps

1) generate B bootstrap samples of size n with replacement from the learning sample $(\mathbf{X}_1, Y_1), \ldots (\mathbf{X}_n, Y_n)$;
2) compute for each bootstrap sample a tree classifier $\hat{c}^b$;
3) classify a new observation by the majority vote of the tree classifiers $\hat{c}^b$ for $b = 1, \ldots, B$.

For two classes we can estimate the probability of one class by the relative frequency of the class in the final partition (each element of the final partition is a leaf of the tree graph).

Bagging

The bagging estimator of the probability of one class for a new observation **x** can be defined by the average of the estimates of the *B* classification trees:

$$\hat{c}_{bag}(\mathbf{x}) = \frac{1}{B}\sum_{b=1}^{B}\hat{c}^{b}(\mathbf{x}).$$

Using a bootstrap method to define Bagging it is straightforward to use the out-of-bag (OOB) sample of each bootstrap sample for estimation of the misclassification error of the bagging classifier. For each observation *i* we estimate the classification or the probability of a class using roughly $B/3$ trees which do not include observation *i*.
Many studies demonstrate that bagging improves the accuracy of single trees (as CART) substantially.

Random Forest and Boosting

Apparently it works to average over unstable classifiers. *random forests* exploits this idea further and grows trees which consider at each split (node of the tree) a random subset of the predictor variables. The size of the random subset is one of the hyper parameters of the classification method random forests.

Another idea is the so called *boosting* classifier. Boosting is not defined by bootstrap sample, but by an iterative process. Each tree is grown using information from previously grown trees. .

R: Tree-based classifiers

We use the **GlaucomaMVF** data set of the R package **ipred** to illustrate random forests using the R function **randomForest.**

```
>  forest.glaucoma <- randomForest(Class ~ ., data=GlaucomaMVF[,c(1:62,67)])
>  forest.glaucoma
Call:
 randomForest(formula = Class ~ ., data = GlaucomaMVF[, c(1:62, 67)])
 Type of random forest: classification
 Number of trees: 500
No. of variables tried at each split: 7
 OOB estimate of error rate: 17.65%
Confusion matrix:
 glaucoma normal class.error
glaucoma 68 17 0.2000000
normal 13 72 0.1529412
>  mypredict.randomForest <- function(object, newdata)
+   predict(object, newdata = newdata, type = c(response"))
>  errorest(Class ~ ., data=GlaucomaMVF[,c(1:62,67)],model=randomForest,
+   estimator = cv", predict= mypredict.randomForest)
Call:
errorest.data.frame(formula = Class ~ ., data = GlaucomaMVF[,
 c(1:62, 67)], model = randomForest, predict = mypredict.randomForest,
 estimator = cv")

 10-fold cross-validation estimator of misclassification error

Misclassification error: 0.1882
```

R: Tree-based classifiers

```
 10-fold cross-validation estimator of misclassification error

Misclassification error: 0.1882

>   errorest(Class ~ ., data=GlaucomaMVF[,c(1:62,67)],model=randomForest,
+   estimator = "boot", predict= mypredict.randomForest)

Call:
errorest.data.frame(formula = Class ~ ., data = GlaucomaMVF[,
 c(1:62, 67)], model = randomForest, predict = mypredict.randomForest,
 estimator = "boot")

 Bootstrap estimator of misclassification error
 with 25 bootstrap replications
Misclassification error: 0.188
Standard deviation: 0.0073
```

Summary

- Introduction on Multivariate Data
- *k*-Means Clustering
- Hierarchical Clustering
- Classification
- Resampling Methods
- Tree-based Classifiers

References and examples of useful textbooks

Everitt, B., Hothorn, T. (2011), An Introduction to Applied Multivariate Analysis with R, Springer-Verlag.

Everitt, B., Landau, S., Leese, M., Stahl, D. (2011), Cluster Analysis, Wiley.

James, G., Witten, D., Hastie, T., Tibshirani, R. (2013), An Introduction to Statistical Learning with Applications in R, Springer-Verlag.

Thank you