Data Mining

Condensed Draft of a Lecture starting in Summer Term 2010
as developed up to November 2009

apl. Prof. Dr.-Ing. habil.
Rainer Knauf
クナフ･ライナー･ 准教授

Ilmenau University of Technology
Faculty of Computer Science and Automation
Chair of Artificial Intelligence
イルミナウ工科大学, 計算機科学･自動制御学部･人工知能講座長
Germany ドイツ

visiting professor at
Tokyo Denki University
School of Information Environment
東京電機大学･ 情報環境学部
Japan 日本

rainer.knauf@tu-ilmenau.de

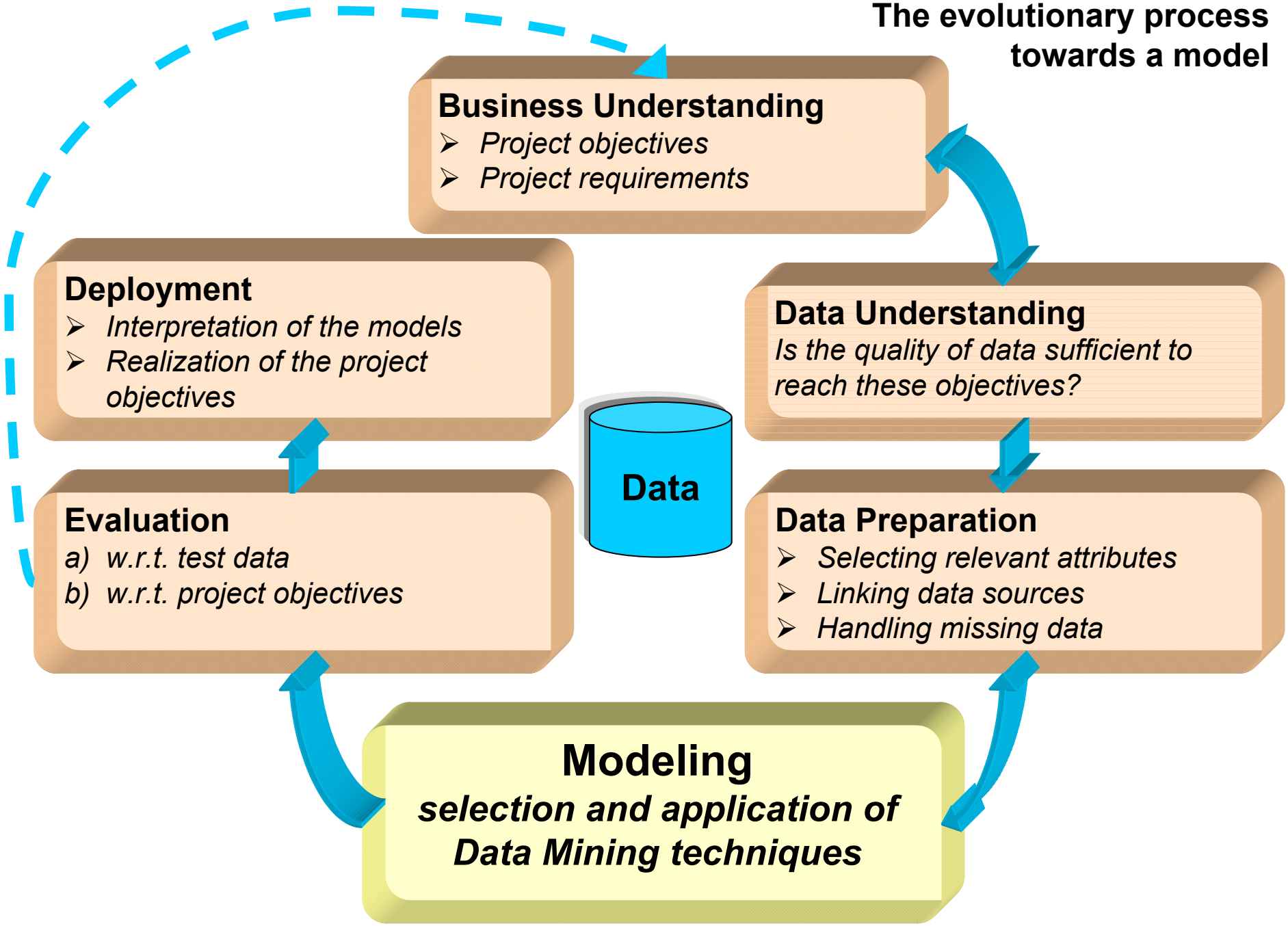
0. Introduction

Motivation

- Huge Data Bases my contain „hidden“ information on regularities
- Manual data analyses costs o lot of resources (time, money, …)
- Many Data Bases haven‘t been analyzed so far

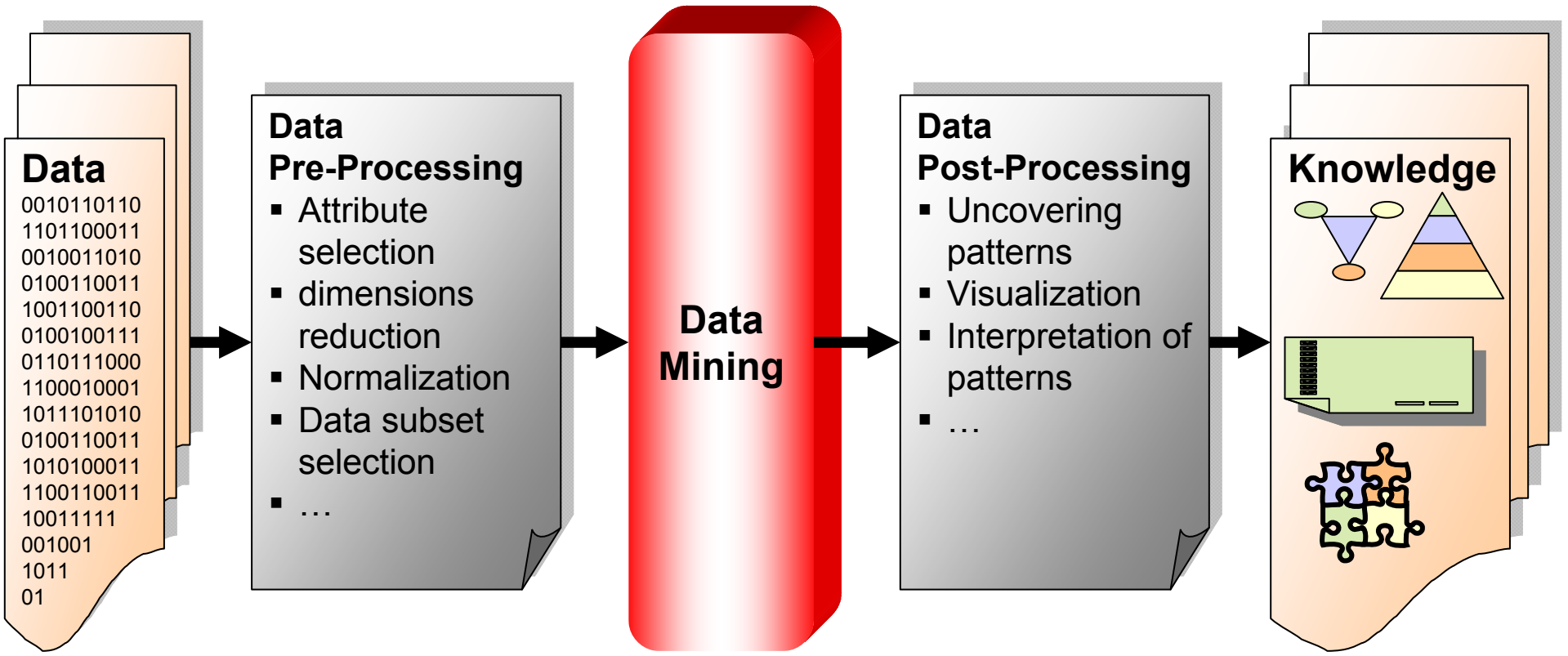
⇨ **Data Mining, i.e.**

- Extraction of implicit, (so far) unknown, potentially useful information from Data Bases
- Automatic exploration and analysis of Data Bases with the objective of uncovering patterns


The evolutionary process towards a model
Business Understanding
Project objectives
Project requirements
Data Understanding
Is the quality of data sufficient to reach these objectives?
Data Preparation
Selecting relevant attributes
Linking data sources
Handling missing data
Modeling
selection and application of Data Mining techniques
Evaluation
a) w.r.t. test data
b) w.r.t. project objectives
Deployment
Interpretation of the models
Realization of the project objectives
Data

The Aim of the Game

- ➢ Uncover abstract information (patterns, correlations, …) in huge Data Bases
- ➢ Building models of regularities, coherencies, ...
- = **„Knowledge Discovery in Databases“**



Application fields

Economy

- *Analysis of shopping behavior, data collection by costumer cards, e.g.*

Medicine

- *Extraction of typical symptoms associated to pathologic phenomena*
- *Empiric exploration of treatment effects*

Government

- *Uncovering corruption and tax evasion by analyzing monetary flows*
- *Analysis of civil and immigration data for use in terror prevention*

Science

- *Interpretation of geo data for predicting natural disasters*
- *Exploring genetic patterns for pathologic predispositions*

Mobile Communication

- ***Exploring typical costumer behaviors for resource management***
- ***Exploring typical costumer behaviors in exceptional states (power cut-off, natural disaster, ...)***

Typical Problem Classes

Prediction Models

- *Prediction of an attribute value based on other attribute values by*
  - *classification (discrete values)*
  - *regression (continuous values)*

Association Analysis

- *Exploration of associated properties within Data Bases (frequently visited web pages by the same kind of costumers, goods that are frequently purchased along with each other, …)*

Cluster Analysis

- *Partitioning data into clusters with similar properties (similar costumer behavior for marketing purposes, similar learner behavior in learning environments, similar serving behavior in the Internet for search machines, ...)*

Anomaly Analysis

- *Identification of Data Objects, which differ significantly from remaining data objects and thus, provide a hint for an anomaly:*
  - *diseases*
  - *technical malfunctions (in a mobile communication network, e.g.)*
  - *frauds (with Credit Cards, banking transactions, insurance frauds, …),*
  - *upcoming natural disasters (climate data)*

1. Data

1.1 Forms of Data

- Data Set: Set of Data Objects **DS: = {DO_1, DO_2, …, DO_n}**
- Data Object is represented by attribute-value pairs **DO_i:= {[A_i^1,W_i^1],[A_i^2,W_i^2], …}**

example: Data Set on examination students‘ results at Tokyo Denki University

	student ID	semester	course	units	rating
DO_1	SIE0019	1	Curriculum Planning	1	A
DO_2	SIE0019	1	Workshop	1	S
DO_3	SIE0019	1	Speaking & Writing in English I	2	B
DO_4	SIE0019	1	Basic Mathematics A	3	A
DO_5	SIE0019	1	European, American, and Asian Studies	4	S
DO_6	SIE0019	1	Architectural Design Practice	4	A
…	…	…	…	…	…
DO_{41}	SIE0019	7	Chinese I	2	C
DO_{42}	SIE0019	7	Course Project A	4	A
DO_{43}	SIE0019	7	Electronics B	4	A
DO_{44}	SIE0020	1	Curriculum Planning	1	A
DO_{45}	SIE0020	1	Computer Literacy	2	A
…	…	…	…	…	…

Properties of Attribute-Values

Attribute Types are classified according to whether or not it's values can be:

- discriminated: **= ≠**
- ordered: **≤ ≥**
- summed up: **+ -**
- multiplied: *** /**

		discriminate	order	sum	multiply
qualitative	nominal attributes	+	-	-	-
	ordinal attributes	+	+	-	-
quantitative	interval attributes	+	+	+	-
	rational attributes	+	+	+	+

Attribute Types

Attribute Typ		Description	Ope-ratio-ns	Examples
Qualitative	**nominal**	enumerable values	=, ≠	Post Index, student ID, eye color, codes
	ordinal	enumerable values with an ordering relation in-between	above ones, ≤, ≥ <, >	students‘ performance ratings (1,..5, S, A, ...,E), verbal descriptions of quantitative properties (hard, medium, soft),
Quantitative	**interval**	numeric values, for which sums and differences have an interpretation	above ones, +, -	temperature in °*C* or °*F*, calendar data
	rational	numeric values, for which (besides sums and differences) products and quotients have an interpretation	above ones, /, *	temperature in *K*, length, weight, age, money amounts, electric currency, …

1.2 Similarity Measures

1.2.1 Similarity of attributes

similarity: $s(x,y)$

dissimilarity: $d(x,y)$

possible approaches for attribute values x and y

attribute type	**dissimilarity**	**similarity**
nominal	$d = \begin{cases} 0, \text{if } x = y \\ 1, \text{if } x \neq y \end{cases}$	$s = \begin{cases} 1, \text{if } x = y \\ 0, \text{if } x \neq y \end{cases}$
ordinal	$d = \frac{\lvert x - y \rvert}{n-1}$	$s = 1 - d$
interval **rational**	$d = \lvert x - y \rvert$	$s = -d \quad s = \frac{1}{1+d} \quad s = e^{-d}$ $s = 1 - \frac{d - \min\_d}{\max\_d - \min\_d}$

1.2.2 (Dis-) similarity of Data Objects $x=[x_1, \ldots, x_n]$ and $y=[y_1, \ldots, y_n]$

Minkowski Distance (Generalization of Euclidean Distance)

$$d(x,y) = \left(\sum_{k=1}^{n} |x_k - y_k|^r \right)^{\frac{1}{r}}$$

Typically used special cases

- $r = 1$ Manhatten (city block) Distance (L1 norm)

  for binary attributes also called **Hamming Distance**

  - sum of die distances in each dimension $1 \ldots n$
  - for binary attributes, it is the number of bits, in which DO are different

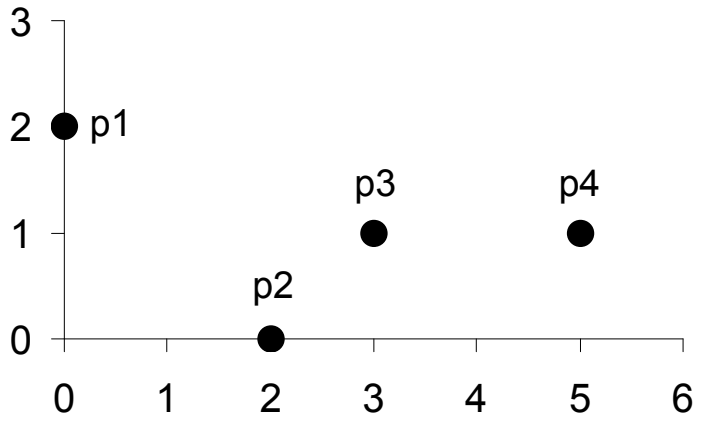
- $r = 2$ Euclidian Distance (L_2 norm)
  - for numeric attributes (ordinal or rational)
  - geometric distance of dots in n - dim. Euclidian space
  - normalization necessary, if dimensions have different value ranges (e.g. for a fair comparison of documents in a document-term matrix)

- $r \rightarrow \infty$ Supremum (L_∞ norm)
  - is maximum difference between any component of the vectors:

    $d(x,y) = max_{k=1 \ldots n} |x_k - y_k|$

Dissimilarity (distance) between $[x_1, \ldots, x_n]$ and $[y_1, \ldots, y_n]$ - examples

point	*x*	*y*
p1	*0*	*2*
p2	*2*	*0*
p3	*3*	*1*
p4	*5*	*1*



L_1	p1	p2	p3	p4
p1	***0***	***4***	***4***	***6***
p2	***4***	***0***	***2***	***4***
p3	***4***	***2***	***0***	***2***
p4	***6***	***4***	***2***	***0***

L_2	p1	p2	p3	p4
p1	***0***	***2,828***	***3,162***	***5,099***
p2	***2,828***	***0***	***1,414***	***3,162***
p3	***3,162***	***1,414***	***0***	***2***
p4	***5,099***	***3,162***	***2***	***0***

L_∞	p1	p2	p3	p4
p1	***0***	***2***	***3***	***5***
p2	***2***	***0***	***1***	***3***
p3	***3***	***1***	***0***	***2***
p4	***5***	***3***	***2***	***0***

Similarity of binary objects $[x_1, x_2, \ldots, x_n]$ and $[y_1, y_2, \ldots, y_n]$ $(x_i, y_i \in \{0,1\})$

- f_{00} : number of attributes with $x_i = 0$ and $y_i = 0$
- f_{01} : number of attributes with $x_i = 0$ and $y_i = 1$
- f_{10} : number of attributes with $x_i = 1$ and $y_i = 0$
- f_{11} : number of attributes with $x_i = 1$ and $y_i = 1$

Similarity measures

- Simple Matching Coefficient SMC

$$SMC = \frac{\text{number of equally valued attributes}}{\text{number of all attributes}} = \frac{f_{00} + f_{11}}{f_{00} + f_{01} + f_{10} + f_{11}}$$

drawback:
objects with many 0-values are considered very similar ⇨

- Jaccard coefficient J

$$J = \frac{\text{number of 1-valued attributes in both objects}}{\text{number of all non 0-valued attributes in both objects}} = \frac{f_{11}}{f_{01} + f_{10} + f_{11}}$$

Similarity of non-binary arrays (DO) $[x_1, x_2, \ldots, x_n]$ and $[y_1, y_2, \ldots, y_n]$ (1)

Simple matching	Cosine coefficient
$sim(x,y) = \sum_{k=1}^{n} x_k y_k$	$sim(x,y) = \frac{\sum_{k=1}^{n} x_k y_k}{\sqrt{\sum_{k=1}^{n} x_k^2} * \sqrt{\sum_{k=1}^{n} y_k^2}}$
▪ only those attributes have influence, which are different from 0 in both arrays ▪ Attributes with high values have higher influence on similarity	▪ cosine of the angle between *x* and *y* ⇨ *cos(0) = 1* for arrays with same direction ⇨ *cos(90°) = 0* for orthogonal arrays ⇨ *cos(180°) = -1* for reversely directed arrays ▪ same as normalized correlation coefficient ▪ same as simple matching for normalized arrays

Similarity of non-binary arrays (DO) $[x_1, x_2, \ldots, x_n]$ and $[y_1, y_2, \ldots, y_n]$ (2)

Dice coefficient	(extended Jaccard coefficient) Tanimoto coefficient	Overlap coefficient
$sim(x,y) = \frac{2\sum_{k=1}^{n} x_k y_k}{\sum_{k=1}^{n} x_k + \sum_{k=1}^{n} y_k}$	$sim(x,y) = \frac{\sum_{k=1}^{n} x_k y_k}{\sum_{k=1}^{n} x_k^2 + \sum_{k=1}^{n} y_k^2 - \sum_{k=1}^{n} x_k y_k}$	$sim(x,y) = \frac{\sum_{k=1}^{n} \min(x_k, y_k)}{\min(\sum_{k=1}^{n} x_k, \sum_{k=1}^{n} y_k)}$
▪ Includes the fraction of common entries: • sum of common entries $\neq 0$ related to • sum of all entries $\neq 0$ ▪ factor 2 for making the value range between 0.0 and 1.0 (binary arrays)	▪ Punishes a small number of common entries more than Dice coefficient: • the less there are common entries $\neq 0$, the larger the denominator	▪ Measures inclusion ▪ reaches 1.0, if each dimension $\neq 0$ in X is $\neq 0$ in Y, too, and vise versa (binary arrays) ▪ Reaches 1.0, if the values of all attributes in one array are ≤ the values in the other array: $sim(x,y)=1$, if $\forall i\ (x_i \leq y_i)$ or $\forall i\ (y_i \leq x_i)$

	d 1	d 2	d 3	d 4	d 5	d 6
Toyota	*1*	*1*	*2*	*1*	*1*	*1*
Honda	*1*	*1*	*2*	*0*	*2*	*0*
hybrid	*1*	*1*	*2*	*1*	*3*	*3*
Super	*1*	*1*	*2*	*0*	*4*	*0*
Diesel	*1*	*1*	*2*	*1*	*5*	*5*
Akku	*1*	*1*	*2*	*0*	*6*	*0*

Similarity measures compared to each other

Simple matching

	d1	d2	d3	d4	d5	d6
d1	-	6.0	12.0	3.0	21.0	9.0
d2	6.0	-	12.0	3.0	21.0	9.0
d3	12.0	12.0	-	6.0	42.0	18.0
d4	3.0	3.0	6.0	-	9.0	9.0
d5	21.0	21.0	42.0	9.0	-	35.0
d6	9.0	9.0	18.0	9.0	35.0	-

Dice

	d1	d2	d3	d4	d5	d6
d1	-	1.000	1.333	0.666	1.555	1.200
d2	1.000	-	1.333	0.666	1.555	1.200
d3	1.333	1.333	-	0.800	2.545	1.714
d4	0.666	0.666	0.800	-	0.750	1.500
d5	1.555	1.555	2.545	0.750	-	2.333
d6	1.200	1.200	1.714	1.500	2.333	-

Cosine

	d1	d2	d3	d4	d5	d6
d1	-	1.000	1.000	0.707	0.898	0.621
d2	1.000	-	1.000	0.707	0.890	0.621
d3	1.000	1.000	-	0.707	0.898	0.621
d4	0.707	0.707	0.707	-	0.544	0.878
d5	0.898	0.898	0.898	0.544	-	0.620
d6	0.621	0.621	0.621	0.878	0.620	-

Tanimoto (extended Jaccard)

	d1	d2	d3	d4	d5	d6
d1	-	1.000	2.000	0.500	3.500	1.500
d2	1.000	-	2.000	0.500	3.500	1.500
d3	2.000	2.000	-	0.666	-4.66	6.000
d4	0.500	0.500	0.666	-	0.600	3.000
d5	3.500	3.500	-4.66	0.600	-	-7.00
d6	1.500	1.500	6.000	3.000	-7.00	-

Overlap

	d1	d2	d3	d4	d5	d6
d1	-	1.000	1.000	1.000	1.000	0.500
d2	1.000	-	1.000	1.000	1.000	0.500
d3	1.000	1.000	-	1.000	0.916	0.555
d4	1.000	1.000	1.000	-	1.000	1.000
d5	1.000	1.000	0.916	1.000	-	1.000
d6	0.500	0.500	0.555	1.000	1.000	-

Correlation

- degree of **linear** (only!) interdependence between 2 attributes
- Typically used for time series of 2 attributes, e.g.
  - monthly average temperature throughout the year
  - hourly determined stock price throughout a trading day
- $-1 \leq corr(x,y) \leq +1$
  - $corr(x,y) = -1$ ⇨ perfect negative linear interdependence
  - $corr(x,y) = 0$ ⇨ no linear interdependence
  - $corr(x,y) = +1$ ⇨ perfect positive linear interdependence

$$\text{corr}(x,y) = \frac{\text{covariance}(x,y)}{\text{stdt\_abw}(x) \bullet \text{std\_abw}(y)} = \frac{s_{xy}}{s_x s_y}$$

$$= \frac{\frac{1}{n-1}\sum_{k=1}^{n}(x_k - \bar{x})(y_k - \bar{y})}{\sqrt{\frac{1}{n-1}\sum_{k=1}^{n}(x_k - \bar{x})^2}\sqrt{\frac{1}{n-1}\sum_{k=1}^{n}(y_k - \bar{y})^2}}$$

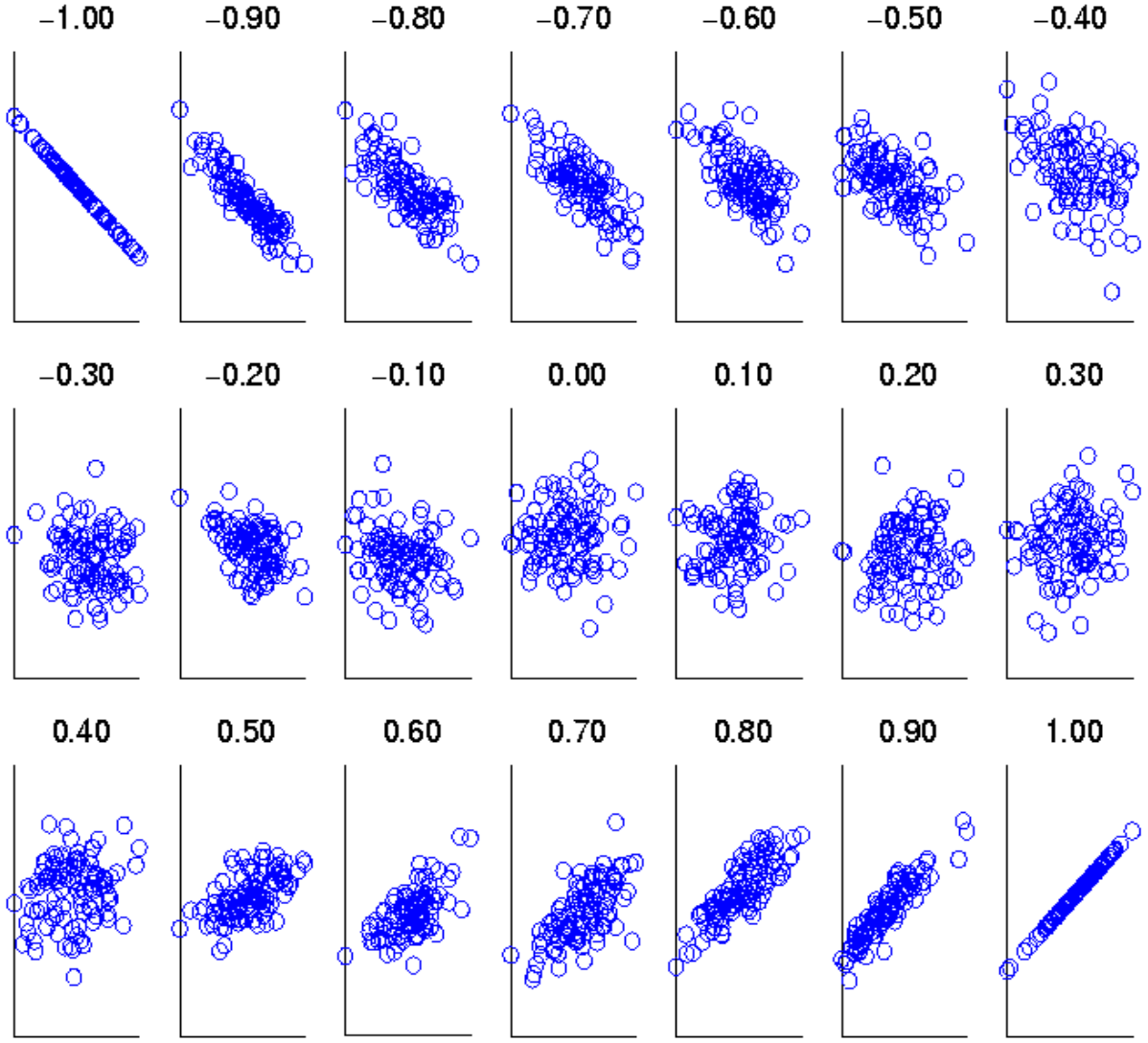
$$= \frac{\sum_{k=1}^{n}(x_k - \bar{x})(y_k - \bar{y})}{\sqrt{\sum_{k=1}^{n}(x_k - \bar{x})^2 \sum_{k=1}^{n}(y_k - \bar{y})^2}}$$

examples

x=[-3,6,0,3,-6]
y=[1,-2,0,-1,2]
⇒ *corr(x,y) = -1*, denn $y = -\frac{1}{3}x$

x=[3,6,0,3,6]
y=[1,2,0,1,2]
⇒ *corr(x,y) = 1*, denn $y = \frac{1}{3}x$

Correlation
examples: dispersions at correlation values between -1 and 1



Auto correlation

correlation between subsequent time ranges values within a time series of values

2. Classification

given

- set of data objects (DO) with known class membership (**training set**)
- each DO consists of **attributes**
- One of the attributes is the **class**

looked for

- **model**, which models the class as a function of the attributes

objective

- classify DO with unknown class as correct as possible

evaluation of the model

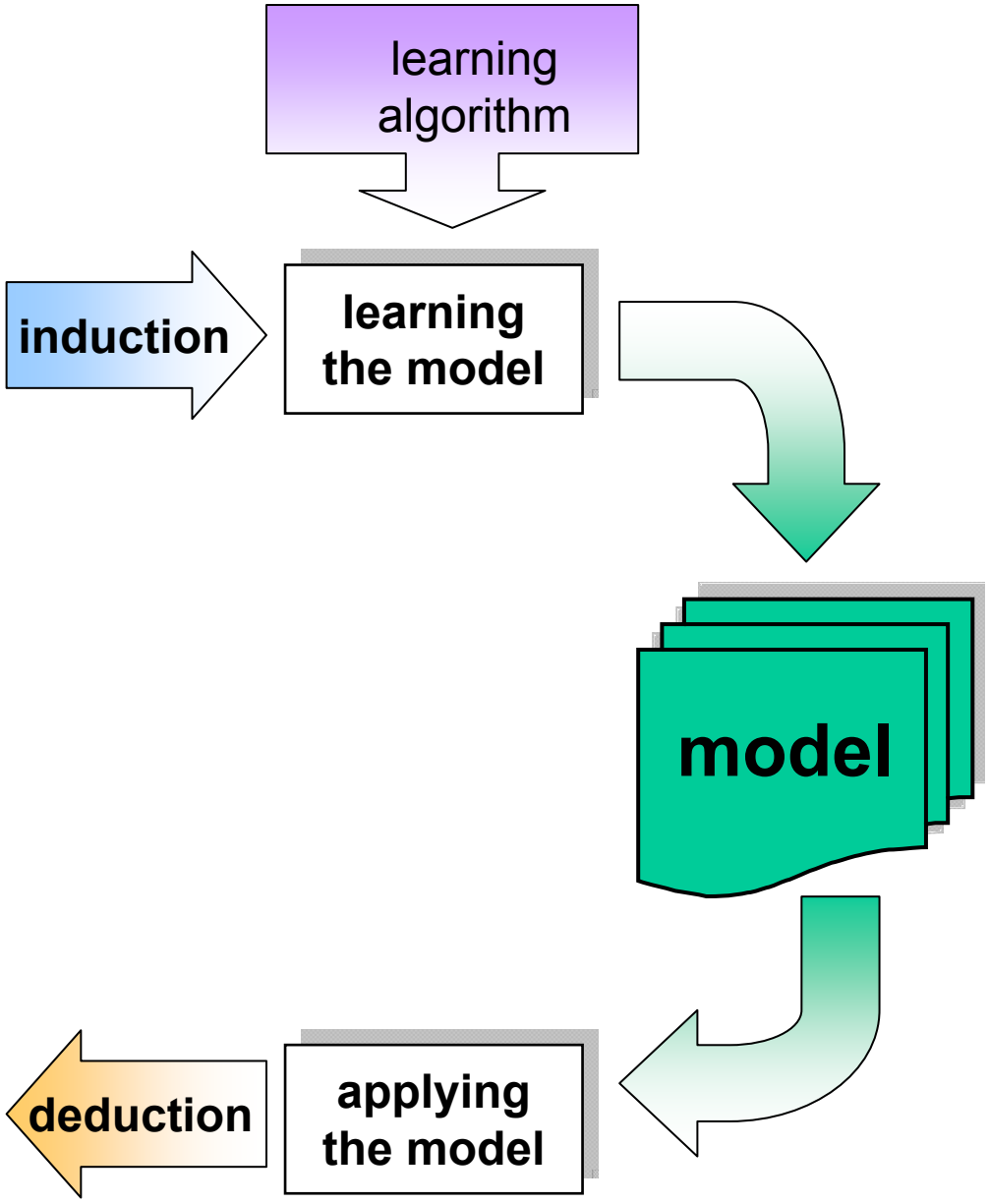
- a **test set** (DO with known classes) is classified by the model
- typically, all given DO with known class is grouped into
  - a trainings set for building the model and
  - a test set for validating the model

Training Set

id	attribute1	attribute2	attribute3	class
1	yes	large	125 k	no
2	no	medium	100 k	no
3	no	small	70 k	no
4	yes	medium	120 k	no
5	no	large	95 k	yes
6	no	medium	60 k	no
7	yes	large	220 k	no
8	no	small	85 k	yes
9	no	medium	75 k	no
10	no	small	90 k	yes

Test Set

id	attribute1	attribute2	attribute3	class
11	no	small	55 k	?
12	yes	medium	80 k	?
13	yes	larg	110 k	?
14	no	small	95 k	?
15	no	large	67 k	?



Classification techniques

- Decision trees
- Rule based methods
- Nearest-Neighbor (NN) classification
- Deriving a most likely class according to Bayes
- Support Vector Machines
- Neuronal Nets

Evaluation of the model

- **Accuracy** (fraction of correct results in the test set)
- **Error rate** (fraction of wrong results in the test set)

for binary classification

		derived class	
		0	1
real class	0	f_{00}	f_{01}
	1	f_{10}	f_{11}

$$accuracy = \frac{f_{00} + f_{11}}{f_{00} + f_{01} + f_{10} + f_{11}}$$

$$error\ rate = \frac{f_{01} + f_{10}}{f_{00} + f_{01} + f_{10} + f_{11}}$$

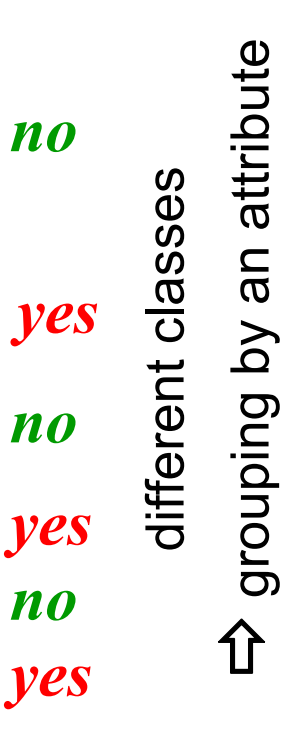
2.1 Induction of decision trees

Hunt‘s algorithm

1. If all DO belong to the same class, they form a leave of the decision tree.
2. Otherwise,
   (1) choose an attribute,
   (2) group the DO according to their values of the chosen attribute and form a successor-node out of each occurring value, and
   (3) apply the algorithm recursively to these successor nodes and the remaining attributes

example: missing credit clearances

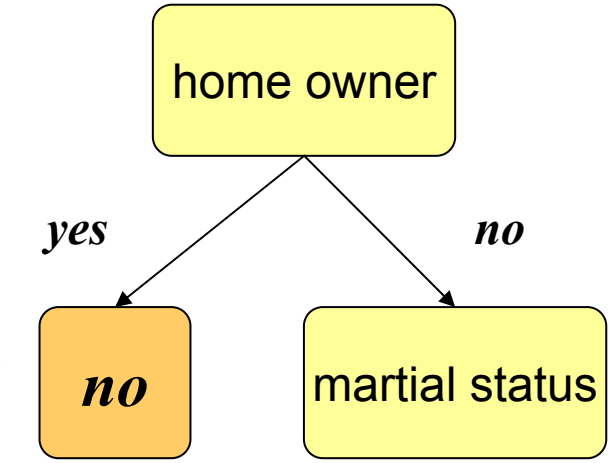
	binary	*discrete nominal*	*continuous rational*	*class*
ID	home owner	martial status	annual income	debt claim
1	*yes*	*single*	*125 k*	***no***
2	*no*	*married*	*100 k*	***no***
3	*no*	*single*	*75 k*	***no***
4	*yes*	*married*	*120 k*	***no***
5	*no*	*divorced*	*95 k*	***yes***
6	*no*	*married*	*60 k*	***no***
7	*yes*	*divorced*	*220 k*	***no***
8	*no*	*single*	*85 k*	***yes***
9	*no*	*married*	*75 k*	***no***
10	*no*	*single*	*90 k*	***yes***



- chosen attribute: home owner
- value range: *{yes, no}*

home owner = ***yes***

ID	martial status	annual income	debt claim
1	*single*	*125 k*	*no*
4	*married*	*120 k*	*no*
7	*divorced*	*220 k*	*no*



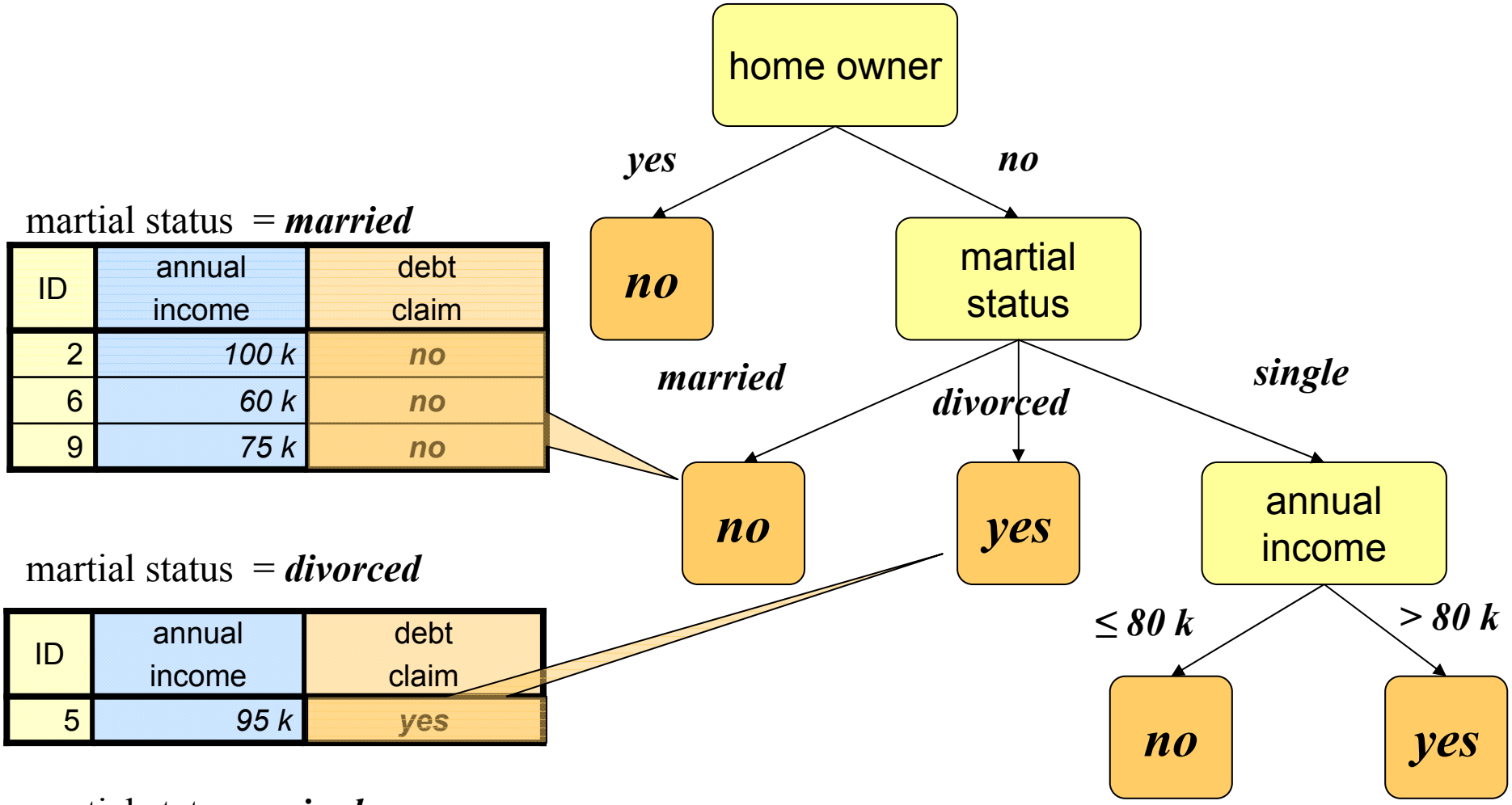
home owner = ***no***

ID	martial status	annual income	debt claim
2	*married*	*100 k*	***no***
3	*single*	*75 k*	***no***
5	*divorced*	*95 k*	***yes***
6	*married*	*60 k*	***no***
8	*single*	*85 k*	***yes***
9	*married*	*75 k*	***no***
10	*single*	*90 k*	***yes***

no ***yes*** ***no*** ***yes*** ***no*** ***yes***

different classes

⇧ grouping by an attribute



martial status = ***married***

ID	annual income	debt claim
2	100 k	no
6	60 k	no
9	75 k	no

martial status = ***divorced***

ID	annual income	debt claim
5	95 k	yes

martial status = ***single***

ID	annual income	debt claim
3	75 k	no
8	85 k	yes
10	90 k	yes

no

yes

different classes

⇨ grouping by an attribute

approach for handling numerical attribute:

- estimate the border betwen ***no*** and ***yes*** as the mean value between 75 and 85

drawbacks of Hunt‘s algorithm

- Works with a complete result (i.e. delivers a class for each possible new case), only if each combination is present in the training set
- Works only for „crisp“ class membership results in the training set and delivers only „crisp“ solutions when using the decision tree

Additional decision tree construction rules are needed for these two cases:

case # 1
For a potential attribute value there is no DO in the training set.

possible solution:
Attaching a leaf containing the class of the most DO in the training set

case # 2
All DO of a group have identical attribute values, but their classes are different.

possible solution:
Attaching a leaf containing the class of the most DO in this group

open issues

(1) exact specification of the splittings

a) How to determine the attribute values at the edges, in particular for continuous attributes?

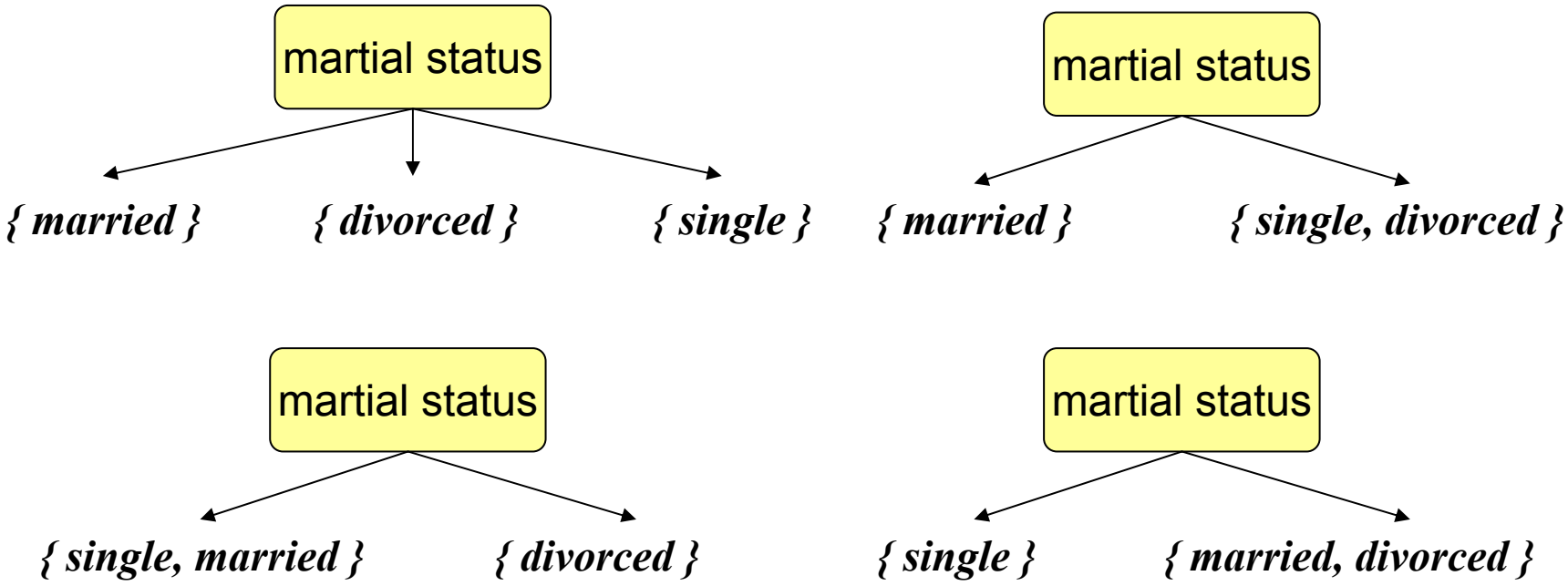
b) Ho to find a „best attribute“ for splitting?

(2) exact specification of a stop criterion

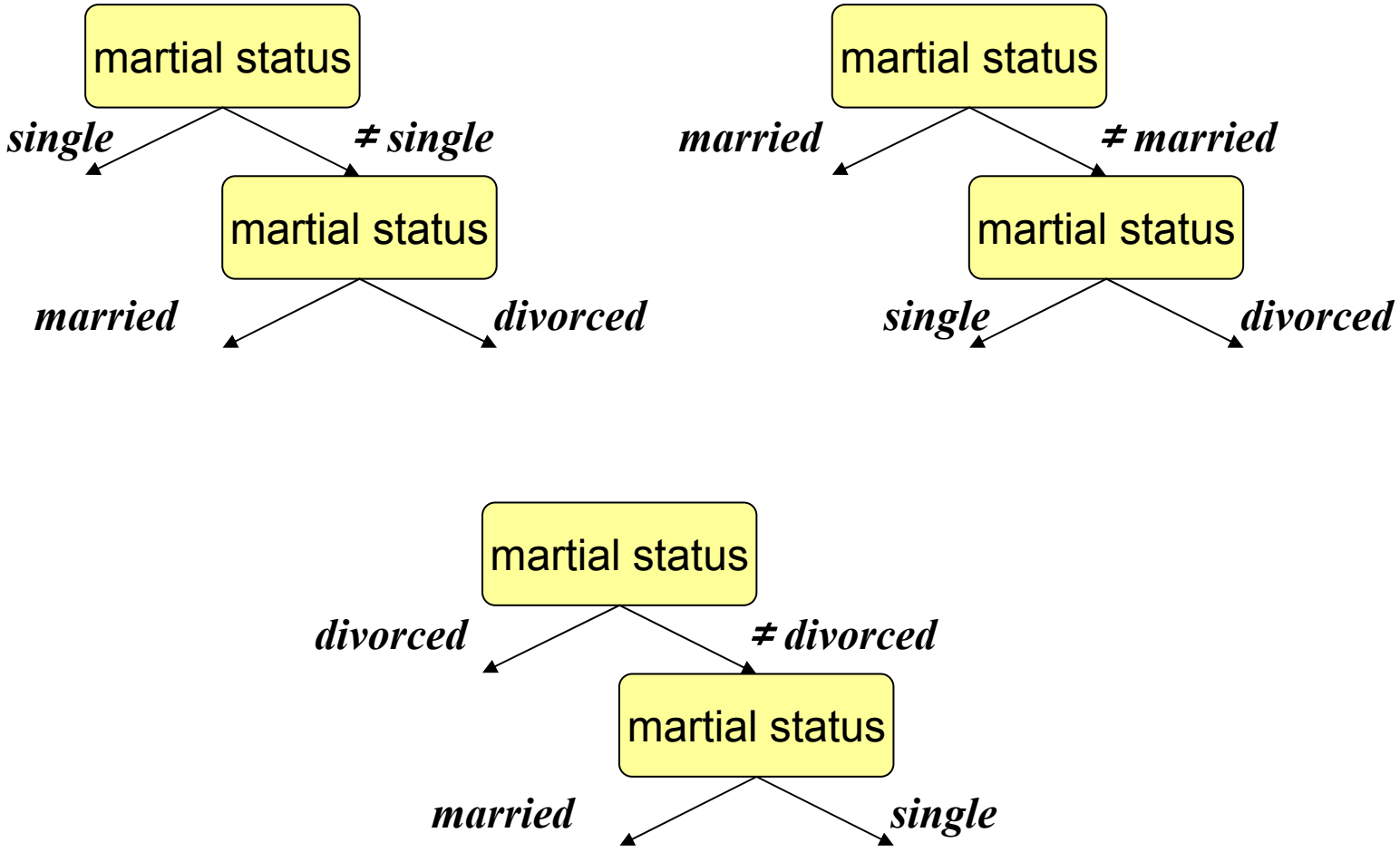
- If all DO of a group belong to the same class?
- If all DO have the same value for the chosen attribute?
- any other criterion ?

(1 a) How to determine the attribute values at the edges?

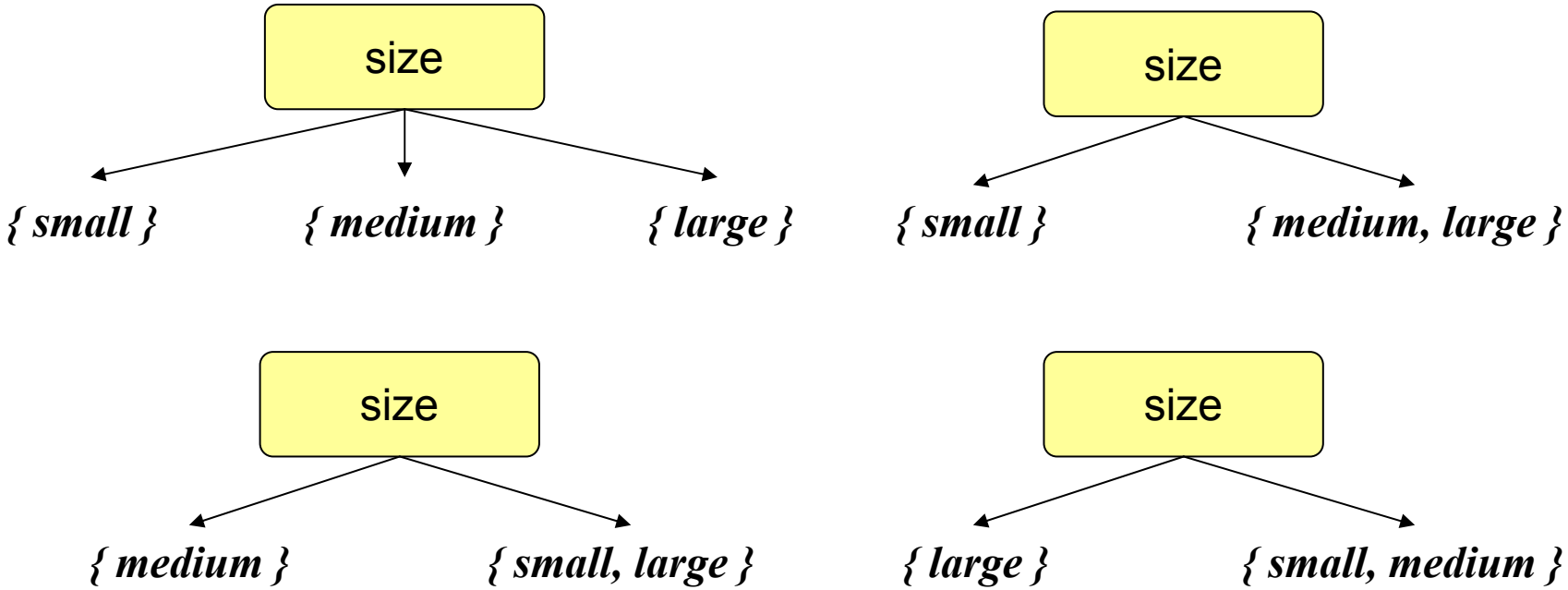
- **binary attributes** each of the two values defines an edge
- **nominal attributes** (without ordering relation)
  - Each splitting into non-empty value sets is possible



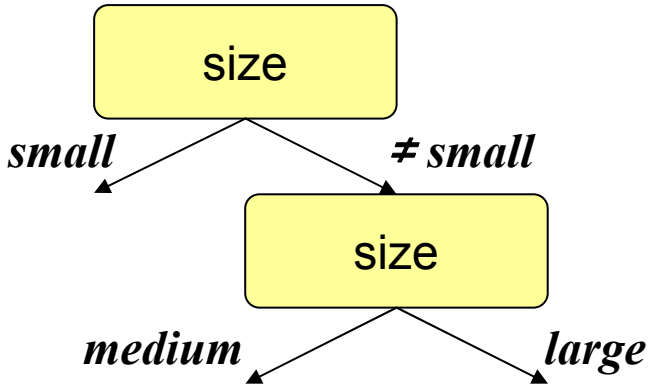
- **nominal attributes** (without ordering relation)
  - Also, each cascading of the k different values towards $2^{k-1}-1$ binary decisions may be possible

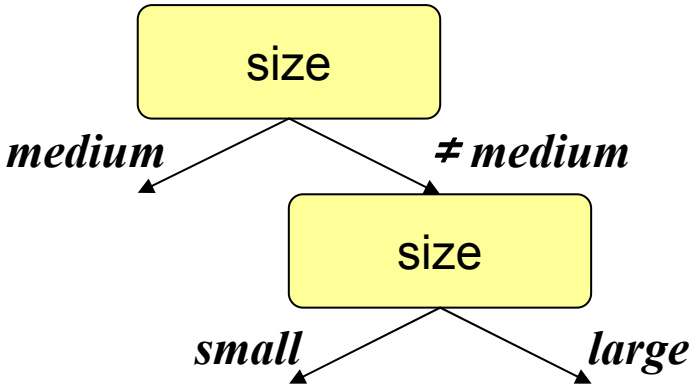


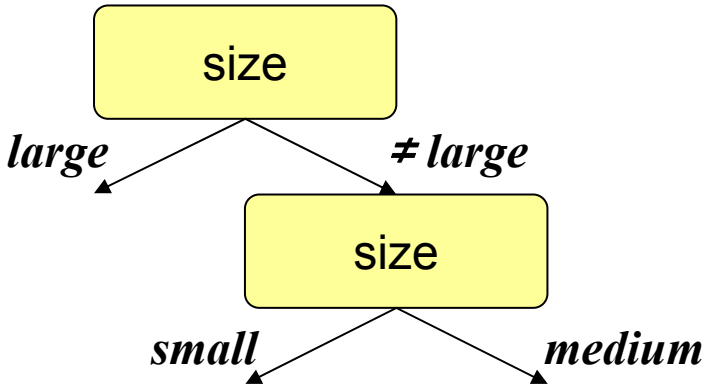
- **ordinal attributes** (with ordering relation)
  - Again, each splitting into non-empty value sets is possible.



- **ordinal attributes** (with ordering relation)
  - Again, also each cascading of the k different values towards $2^{k-1}-1$ binary decisions is possible



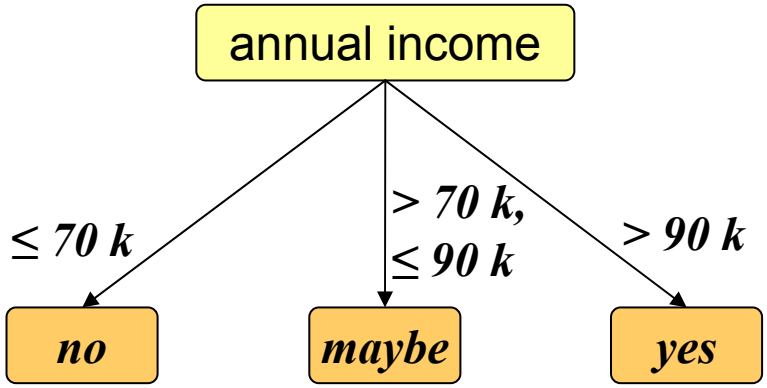




- **continuous attributes**

  Value range has to be discretized

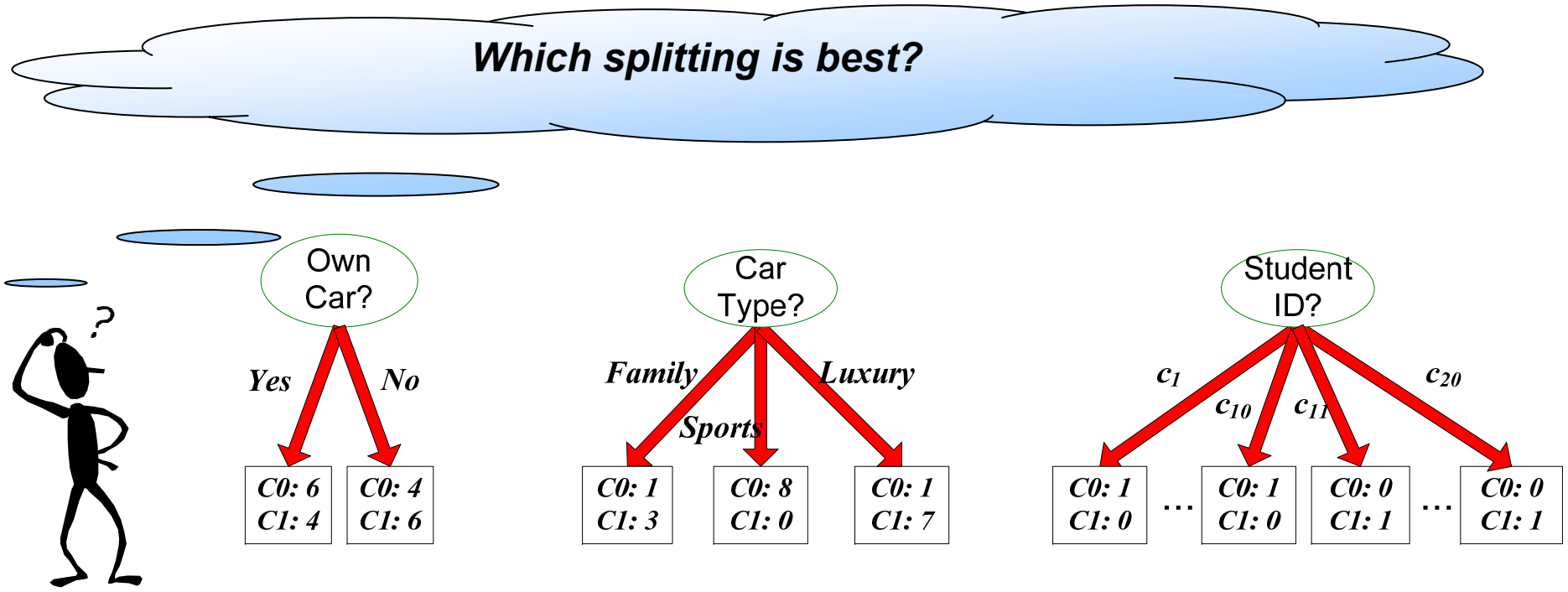
  - Establish a number n of discrete values
  - Establish $n-1$ „split points“ $x_1, \ldots, x_{n-1}$ in-between these values
  - Map each value to a group according to the intervals
    - $\{ (x_0 , x_1] , (x_1 , x_2] , \ldots, (x_{n-1} , x_n) \}$ respectively
    - $x_0 < x \leq x_1, x_1 < x \leq x_2 , \ldots, x_{n-1} < x \leq x_n$
  - x_0 and x_n may also be $-\infty$ or $+\infty$



(1 b) How to find a “best“ attribute for splitting?

example:

- *Before splitting, there are 5 DO of class 0 (**C0**) and 5 DO of class 1 (**C1**)*
- *available attributes are*
  - *a binary attribute „**own a car**“, value set: **{ yes, no }***
  - *a nominal attribute „**preferred car type**“, value set: **{ family, sports, luxory }***
  - *a nominal attribute „**student ID**“, value set: **{ c_1, c_2, …, c_{20}}***



(1 b) How to find a “best“ attribute for splitting?

- optimal: nodes with consistent classes, all DO hold the same class

⇨ A measure of impurity is needed

C0: 5 ***C1: 5***	inhomogeneous high impurity
C0: 9 ***C1: 1***	homogeneous low impurity

- In the example,
  - ***C0: 10, C1: 0*** and ***C0: 0, C1: 10*** would be optimal
  - ***C0: 5, C1: 5*** would be the worst case

Frequently used measures of impurity

- Entropy
- Gini – Index
- Classification Error

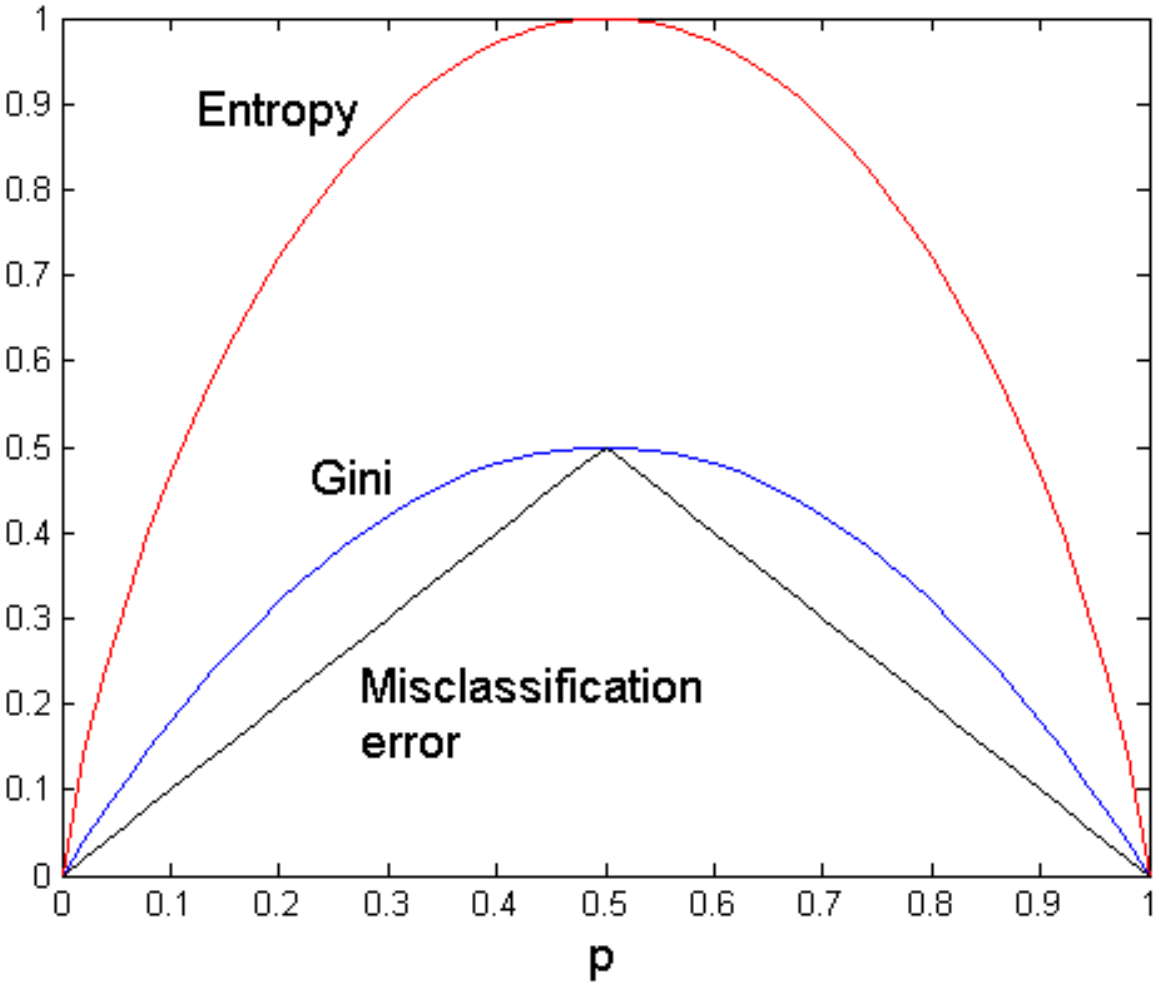
- Let $p_i = p(i|a)$ be the fraction of DOs behind a node with the attribute a, which belong to class i
  - In the binary case, there is p_0 and p_1 only and $p_1 = 1 - p_0$
- Measures of impurity are

  - **Entropy** $$H(a) = -\sum_{i=1}^{m} p(i \mid a) * ld\ p(i \mid a)$$
  - **Gini Index** $$Gini(a) = 1 - \sum_{i=1}^{m} p(i \mid a)^2$$
  - **Classification Error** $$F(a) = 1 - \max_i p(i \mid a)$$

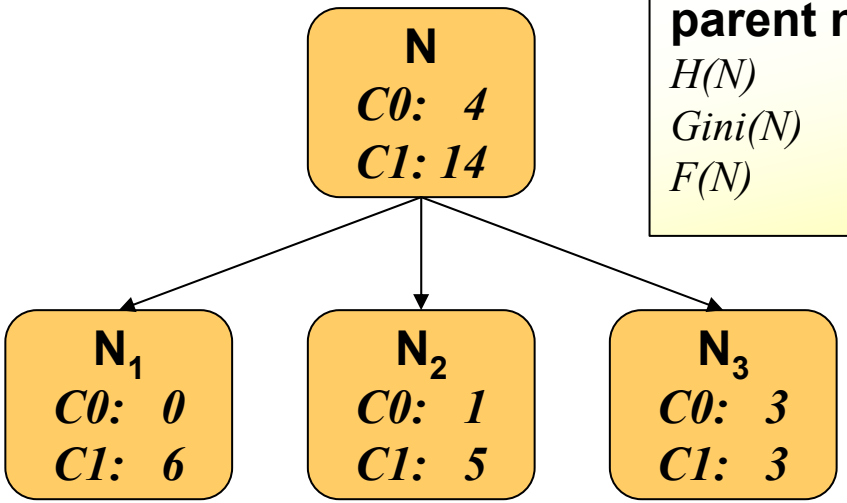
  with

  - m being the number of classes
  - $0\ ld\ 0 = 0$ in the Entropy calculation (because $\lim_{p \to 0}(p \cdot ld(p)) = 0$)
  - These measures are calculated for each successor node
  - the „quality“ of an attribute is defined by the degree of improving impurity: parent node vs. all kid nodes (weighted average)
  - Difference of impurities between parent – and kid nodes is called **Information Gain** of an attribute a: $IG(a)$

Entropy, **Gini** and **Classification Errors** for binary classification



Entropy, **Gini** and **Classifikation Error** - an Example



parent node

$H(N) = -4/18*ld(4/18) - 14/18*ld(14/18) = 0.764$

$Gini(N) = 1 - (4/18)^2 - (14/18)^2 = 0.346$

$F(N) = 1 - max(4/18, 14/18) = 0.222$

Weighted average of kids

$$H(N_{all}) = \frac{6}{18}H(N_1) + \frac{6}{18}H(N_2) + \frac{6}{18}H(N_3)$$

$= 0.550$ (better than parent node)

$$Gini(N_{all}) = \frac{6}{18}Gini(N_1) + \frac{6}{18}Gini(N_2) + \frac{6}{18}Gini(N_3)$$

$= 0.259$ (better than parent node)

$$F(N_{all}) = \frac{6}{18}F(N_1) + \frac{6}{18}F(N_2) + \frac{6}{18}F(N_3)$$

$= 0.222$ (equal to parent node)

kid nodes

$H(N_1) = -0/6*ld(0/6) - 6/6*ld(6/6) = 0$

$Gini(N_1) = 1 - 0/6^2 - 6/6^2 = 0$

$F(N_1) = 1 - max(0/6, 6/6) = 0$

$H(N_2) = -1/6*ld(1/6) - 5/6*ld(5/6) = 0.650$

$Gini(N_2) = 1 - 1/6^2 - 5/6^2 = 0.278$

$F(N_2) = 1 - max(1/6, 5/6) = 0.167$

$H(N_3) = -3/6*ld(3/6) - 3/6*ld(3/6) = 1$

$Gini(N_3) = 1 - 3/6^2 - 3/6^2 = 0.5$

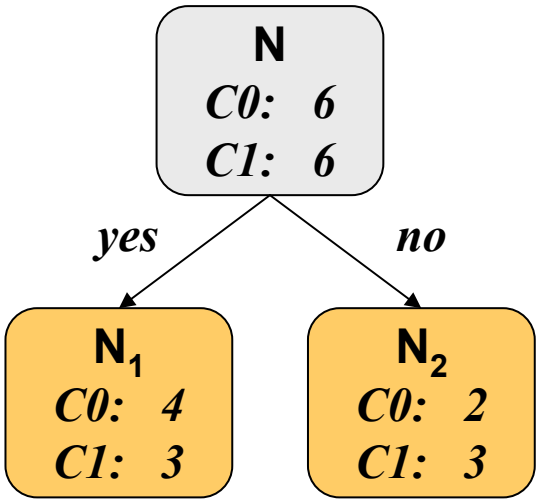
$F(N_3) = 1 - max(3/6, 3/6) = 0.5$

Information Gain

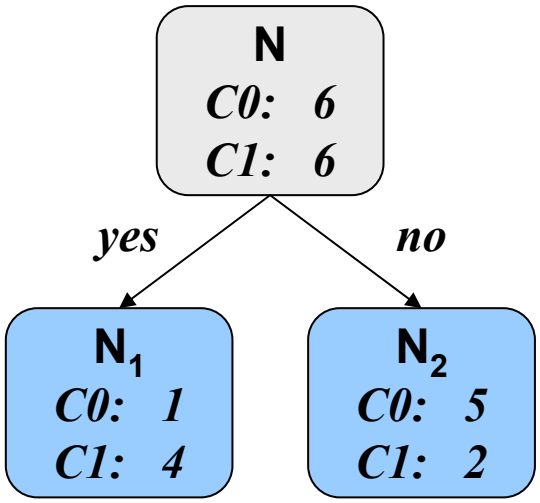
- $IG_H = H(N) - H(N_{all}) = 0.214$
- $IG_{Gini} = Gini(N) - Gini(N_{all}) = 0.087$
- $IG_F = F(N) - F(N_{all}) = 0$

Splitting of various attribute types
Binary attributes

Split with binary attribute ***a***



Split with binary attribute ***b***



	attribute ***a***	attribute ***b***
Gini(N)	***0.5***	
Gini(N_1)	***0.4898***	***0.3200***
Gini(N_2)	***0.4800***	***0.4082***
Gini(N_{all})	***0.4857***	***0.3714***
IG	***0.0143***	***0.1286***

to prefer

Splitting of various attribute types

Nominal attributes

comparison of all groupings (partitionings)

value range: ***{ married, single, divorced }***

N
C0: 10
C1: 10

	N_1	N_2	N_1	N_2	N_1	N_2	N_1	N_2	N_3
Partition	*{ s }*	*{ m, d }*	*{ m }*	*{ s, d }*	*{ d }*	*{ s, m }*	*{ m }*	*{ s }*	*{ d }*
C0 C1	*8* *0*	*2* *10*	*1* *3*	*9* *7*	*1* *7*	*9* *3*	*1* *3*	*8* *0*	*1* *7*
Gini(N)	*0.5*								
Gini(N_1)	*0*		*0.3750*		*0.2188*		*0.3750*		
Gini(N_2)		*0.2778*		*0.4922*		*0.3750*		*0*	
Gini(N_3)									*0.2188*
Gini(N_{all})	*0.1667*		*0.4688*		*0.3124*		*0.1625*		
IG	*0.3333*		*0.0312*		*0.1876*		*0.3375*		

To prefer

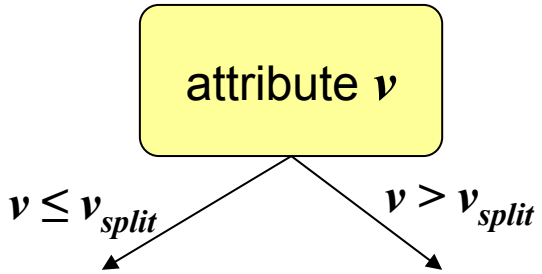
result is not surprising

- All other groupings are derived from the „winner“ partition by merging two groups
- Merging increases impurity

Splitting of various attribute types

continuous attributes

Finding appropriate split values



Variant # 1

- determining **exactly one** split value v_{split}
- (Gini-) optimum among the means between the DO

example

class	No	No	No	Yes	Yes	Yes	No	No	No	No
annual income	60	70	75	85	90	95	100	120	125	220

v_{split}	65		72.5		80		87.5		92.5		97.5		110		122.5		172.5	
v	≤	>	≤	>	≤	>	≤	>	≤	>	≤	>	≤	>	≤	>	≤	>
No	1	6	2	5	3	4	3	4	3	4	3	4	4	3	5	2	6	1
Yes	0	3	0	3	0	3	1	2	2	1	3	0	3	0	3	0	3	0
Gini	0.400		0.375		0.343		0.417		0.400		0.300		0.343		0.375		0.400	

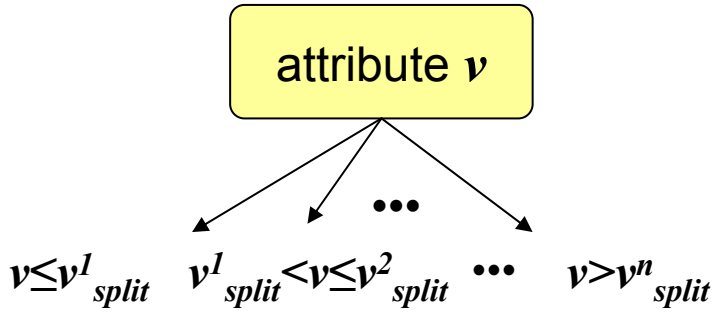
Opti-mum

⇨ $v_{split} = 97.5$

Splitting of various attribute types

continuous attributes

Finding appropriate split values



Variant # 2

- determining **several** split values $v^1_{split}, v^2_{split}, \ldots, v^n_{split}$
- means between neighboring DO belonging to different classes

(same) example

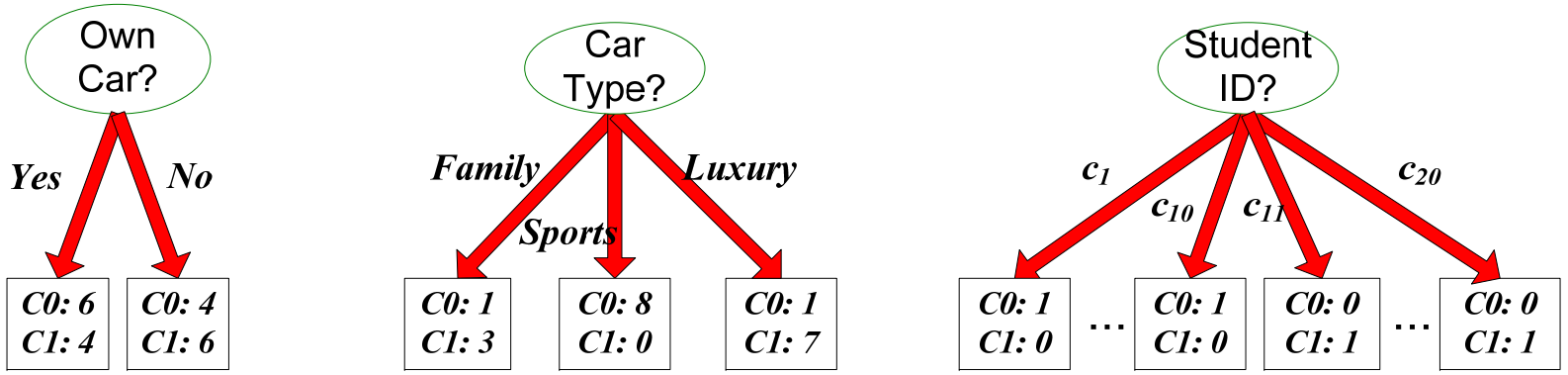
class	No	No	No	Yes	Yes	Yes	No	No	No	No
annual income	60	70	75	85	90	95	100	120	125	220

$$\Rightarrow v^1_{split} = 80 \qquad v^2_{split} = 97.5$$

v_{split}	80		97.5
v	$v \le 80$	$80 < v \le 97.5$	$v > 97.5$
No	3	0	4
Yes	0	3	0
$Gini(N_i)$	0	0	0
$Gini(v)$	0		

Gain Ratio

- Impurity measures like Gini or Entropy favor attributes, which have many values before those having a smaller number of values
- the number of DO per sub-tree, however, is smaller than with attributes having less values
- extreme case: just one DO (see split on the right) – perfect information gain, but useless as a general model



Too less DO don‘t say anything statistically relevant about the universe of all objects

a solution to this problem

- create binary splits only
- nominal und continuous attributes can be cascaded, if needed

another solution to this problem

Relating the Information Gain to the number of DO, that remain in the sub trees (as practiced in decision tree algorithm C4.5):

$$\text{Gain - Ratio} = \frac{IG(a)}{\text{Split - Info}}$$

with

$$\text{Split - Info} = -\sum_{i=1}^{k} p(i) * ld(p(i)) \qquad p = \frac{n_i}{n}$$

- Here, the set of $\boldsymbol{n}$ DO is partitioned into k subsets having $\boldsymbol{n_i}$ DO each.
- The Information Gain is adjusted by the Entropy of the partitioning (Split-Info).
- Higher entropy partitioning (large number of small partitions) will by penalized.

An algorithm to construct a decision tree

input

- Set of DO with known class membership (examples) ***E***
- Set of attributes (features) ***F***

output

- pointer to the decision tree ***root***

```
TreeGrowth(E,F)
if stopping_cond(E,F) then
        leaf = create_node()
        leaf.label = classify(E)
        return leaf
else
        root = create_node()
        root.test_cond = find_best_split(E,F)
        Values := {value: value is possible value of root.test_cond}
        for each v ∈ Values do
          E_v := {example: example ∈ E, root.test_cond(example) = v}
          child_v := TreeGrowth(E_v, F)
          add child_v as descendent of root and label the edge root→child_v as v
        end for
end if
return root
```

An algorithm to construct a decision tree

- `create_node()` extends the tree by a new node, which is either an attribute (*node.test_condition)* or a class, i.e. a leaf node *(node.label)*
- `find_best_split(`E,F`)` determines the best attribute (and, if applicable, the best split values of this attribute) based on an impurity measure (Gini, Entropie), extended by the Gain Ratio, if appropriate
- `classify(`E`)` determines the class, which is attached to a leaf node t of the decision tree, reasonably the class with the largest number of DO:
  $leaf\_label := \{l: l \in E_t, \neg \exists p(i|t): p(i|t) > p(l|t)\}$.

  Additionally, an estimated probability for correctness $p(l|t)$ could be added.
- `stopping_cond(`E,F`)` terminates the recursive construction of the decision tree.

  Reasonable stopping conditions may be:
  - A large fraction of DO (all, in the “perfect“ case) belong to the same class.
  - The number of DO went below a certain minimum.

Algorithms with this basic structure can be found at
http://www.kdnuggets.com/software/classification-decision-tree.html

Model Overfitting

Sources of error

1. **Training errors**
   - wrong classes over training data
2. **Generalization errors**
   - Wrong generalization from training data to new cases

- Training errors can be reduced as far as desired (even down to zero, if there are enough attributes)
- However, the “price“ of error reduction is increasing model complexity
- If a model fits too perfect with the training data, it may even become worse, when applying it to test data.

This effect is called **Model Overfitting**.

- If a model is too simple, both error rates, the trainings- and the generalization error, are high

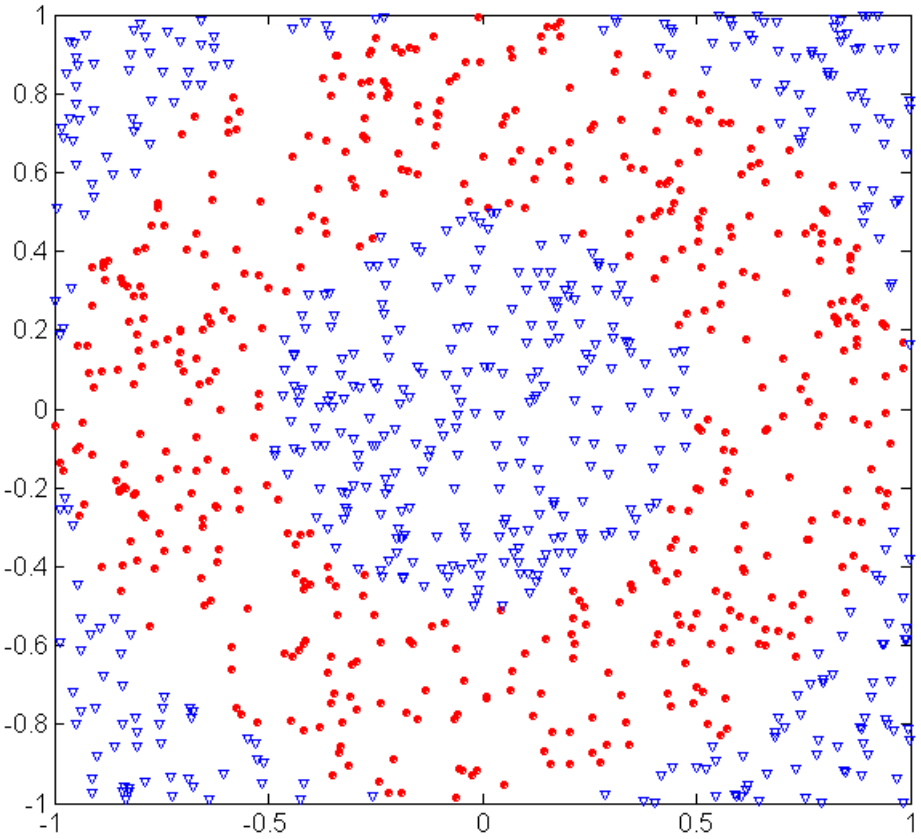
This effect is called **Model Underfitting**.

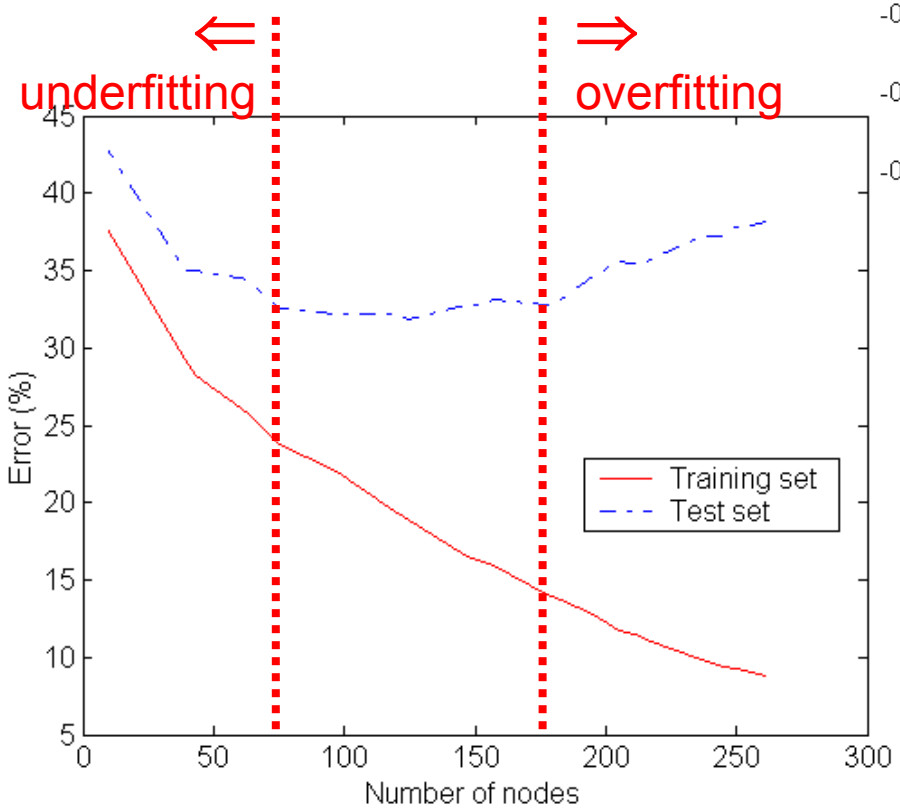
Model Overfitting

example
2 classes

Red circles:
$0.5 \leq sqrt(x_1^2+x_2^2) \leq 1$

blaue triangles:
$sqrt(x_1^2+x_2^2) > 0.5$ oder
$sqrt(x_1^2+x_2^2) < 1$





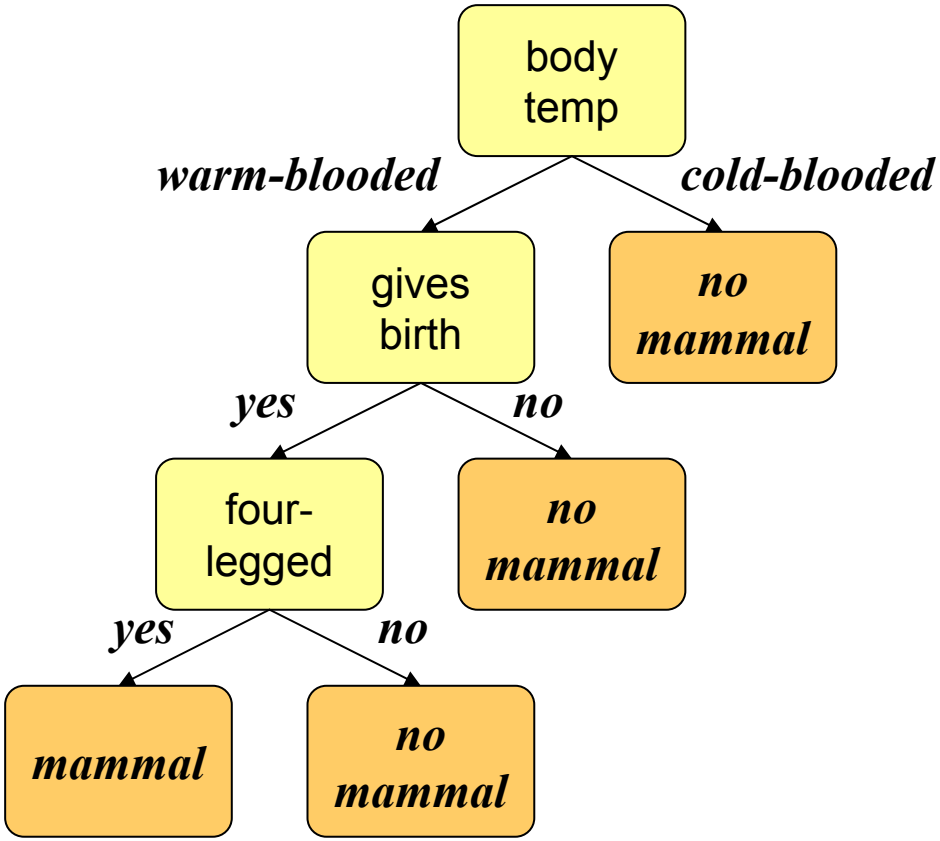
Causes of model overfitting

1. Overfitting due to noisy data

noisy data

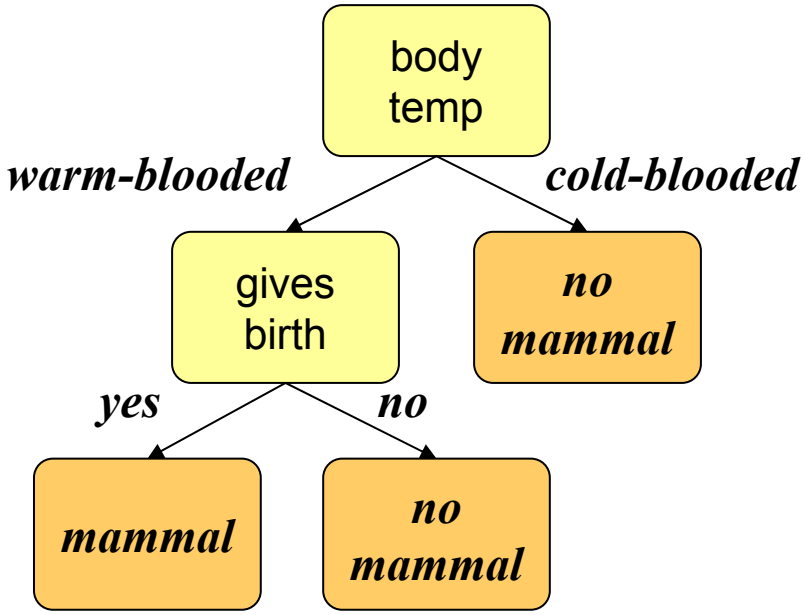
= training data with **wrong class membership**, e.g.

Species	body temperature	gives birth	four-legged	hiber-nates	mam-mal?
porcubine	*warm-blooded*	*yes*	*yes*	*yes*	*yes*
cat	*warm-blooded*	*yes*	*yes*	*no*	*yes*
bat	*warm-blooded*	*yes*	*no*	*yes*	***no***
whale	*cold-blooded*	*yes*	*no*	*no*	***no***
salamander	*cold-blooded*	*no*	*yes*	*yes*	*no*
komodo dragon	*cold-blooded*	*no*	*yes*	*no*	*no*
python	*cold-blooded*	*no*	*no*	*yes*	*no*
salmon	*cold-blooded*	*no*	*no*	*no*	*no*
eagle	*warm-blooded*	*no*	*no*	*no*	*no*
guppy	*cold-blooded*	*yes*	*no*	*no*	*no*



“perfect“ model

0 misclassifications in training data



imperfect model

10 % misclassifications in training data

Effect of both models on test data

test data

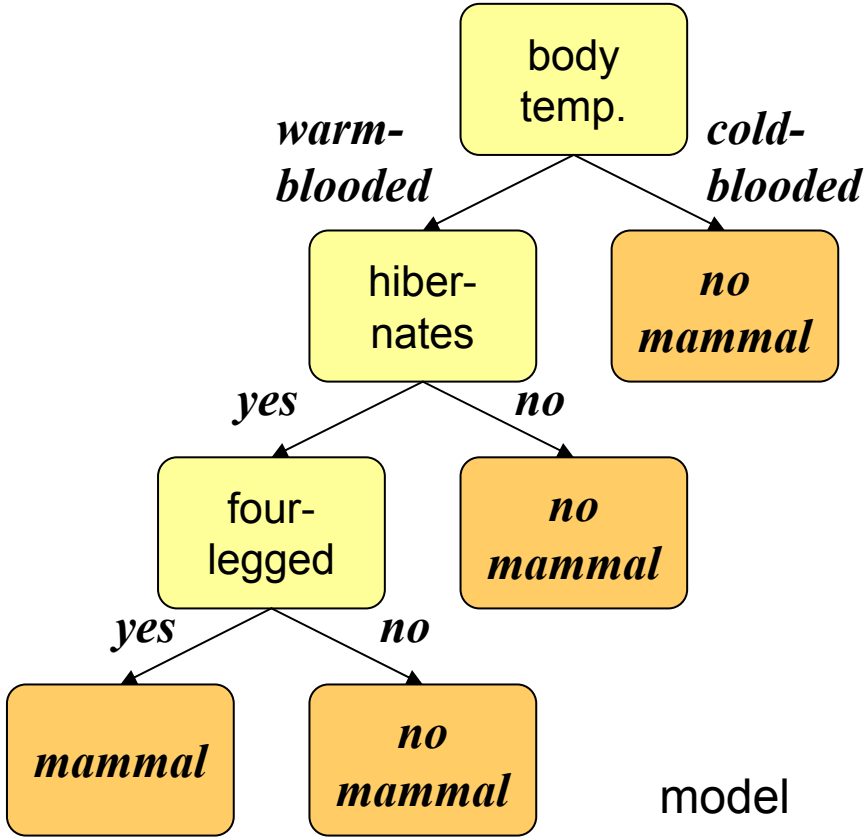
Species	body temp.	gives birth	four-legged	hibernates	mam-mal?
human	*warm-blooded*	*yes*	*no*	*no*	***yes***
pigeon	*warm-blooded*	*no*	*no*	*no*	***no***
elephant	*warm-blooded*	*yes*	*yes*	*no*	***yes***
leopard shark	*cold-blooded*	*yes*	*no*	*no*	***no***
turtle	*cold-blooded*	*no*	*yes*	*no*	***no***
penguin	*cold-blooded*	*no*	*no*	*no*	***no***
eel	*cold-blooded*	*no*	*no*	*no*	***no***
dolphin	*warm-blooded*	*yes*	*no*	*no*	***yes***
spiny anteater	*warm-blooded*	*no*	*yes*	*yes*	***yes***
gila monster	*cold-blooded*	*no*	*yes*	*yes*	***no***

	training errors	performance of the model error rate on test data
“perfect“ model	**0 %**	**30 %**
imperfect model	**10 %**	**20 %**

2. Overfitting due to lack of representative data

too less training data make the model fragile for overfitting, e.g.

Species	body temp.	gives birth	four-legged	hiber-nates	mam-mal?
sala-mander	*cold-blooded*	*no*	*yes*	*yes*	*no*
guppy	*cold - blooded*	*yes*	*no*	*no*	*no*
eagle	*warm - blooded*	*no*	*no*	*no*	*no*
poorwill	*warm - blooded*	*no*	*no*	*no*	*no*
platypus	*warm-blooded*	*no*	*yes*	*yes*	*yes*



model

training errors	performance of the model error rate on test data
0 %	**30 %**

Estimation of generalization errors

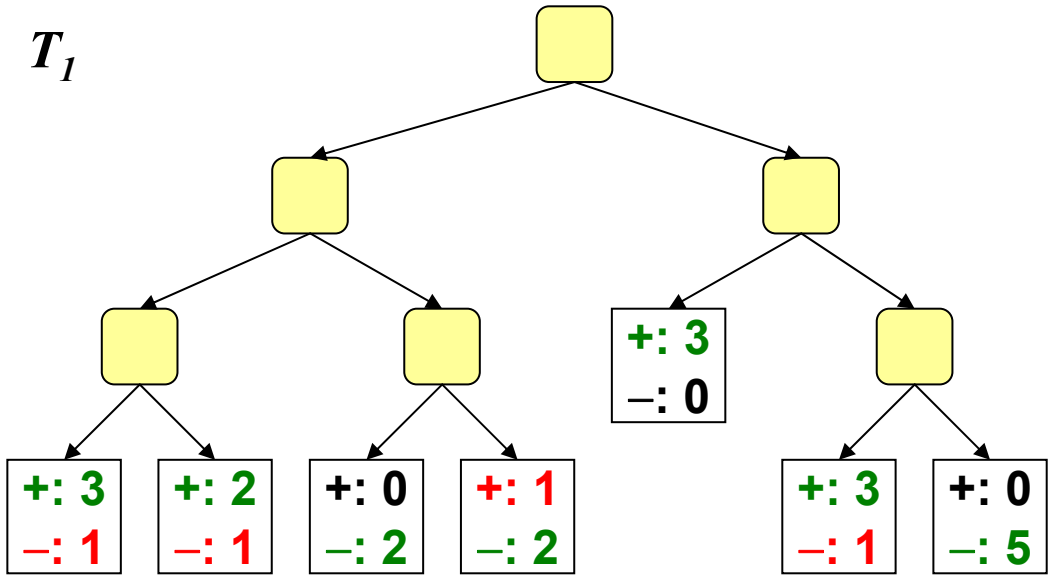
1. Evaluation with training data
(Resubstitution Estimate, Optimistic Approach)

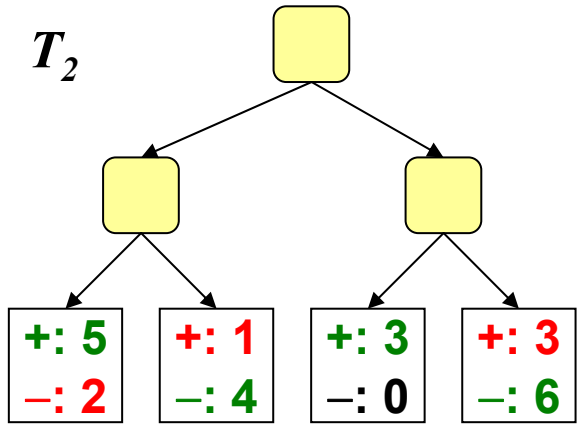
assumption: training data represents **all** (possible) data well

an example

error rates e of two models T_1 and T_2 of different complexity

- generated from the same training data
- leaf nodes contain the **class of the majority of DO**





$e(T_2) = 6 / 24 = 0.25$

$e(T_1) = 4 / 24 = 0.167$

2. Incorporating model complexity

2.1 Occam‘s Razor (“principle of parsimony“)

If there are two models with the same generalization error, prefer the one of less complexity (i.e. with less nodes).

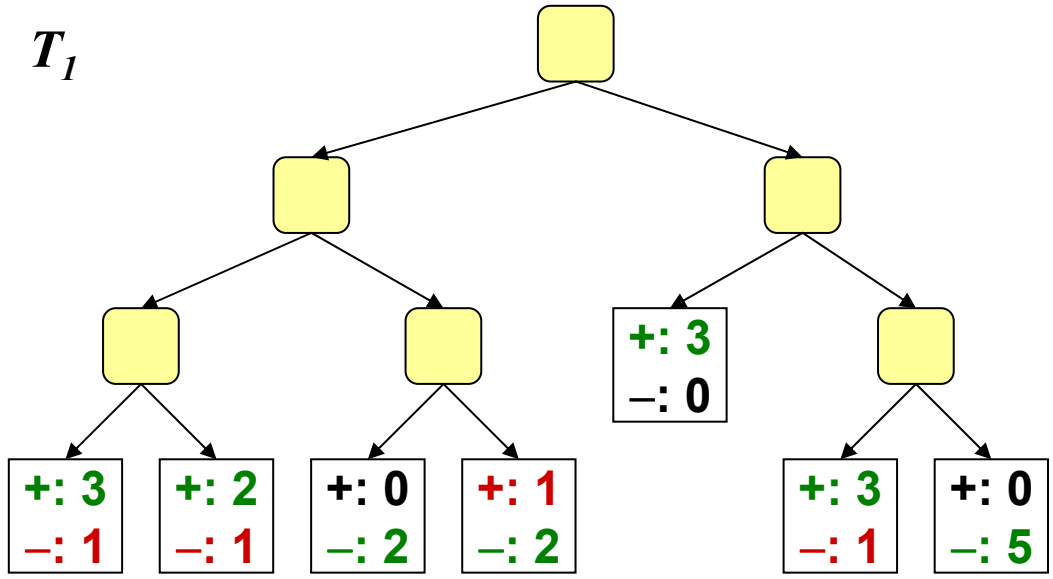
2.2 Pessimistic error estimate $e_g(T)$

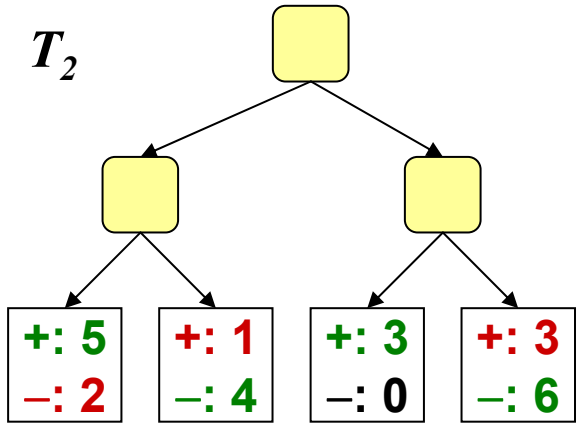
To the sum of all misclassifications $e(t_i)$ at the leaf nodes on the training data, a fine (a malus, a “punishment“) $\Omega(t_i)$ is added for each leaf node t_i in the tree and the result is related to the number of DO in the training data:

$$e_g(T) = \frac{\sum_{i=1}^{k} [e(t_i) + \Omega(t_i)]}{\sum_{i=1}^{k} n(t_i)} = \frac{e(T) + \Omega(T)}{N_t}$$

Estimation of generalization errors

examples





with $\Omega(t_i) = 0.5$ is $e_g(T_1) = \frac{4 + 7 * 0.5}{24} = \frac{7.5}{24} = 0.3125$ and $e_g(T_2) = \frac{6 + 4 * 0.5}{24} = \frac{8}{24} = 0.3333$

with $\Omega(t_i) = 1$ is $e_g(T_1) = \frac{4 + 7 * 1}{24} = \frac{11}{24} = 0.458$ and $e_g(T_2) = \frac{6 + 4 * 1}{24} = \frac{10}{24} = 0.417$

2.3 Description Complexity of the tree and its misclassifications (Minimum Description Length Principle)

Both

- a measure for the binary coding of the misclassified examples $cost(\ data \mid tree\ )$ (e.g., for each misclassification the length of the code of the example-ID) $\lceil ld(\ number\ of\ classes\ )\ \rceil$
  e.g. for 16 examples and 3 misclassifications $cost(data|tree) = 3 * 4 = 12$
- And - to "punish" the tree complexity – a measure for the code length of a binary coding of the tree's nodes $cost(\ tree\ )$, e.g.
  - for leaf nodes the code length of the node-associated class $\lceil ld(\ number\ of\ classes)\ \rceil$
  - for all other nodes the code length of the node-associated attribute $\lceil ld(\ number\ of\ attribute)\ \rceil$

  e.g. for 5 leafs denoting 2 classes, 7 non-leafs denoting 4 attributes
  $cost(tree) = 5 * 1 + 7 * 2 = 19$

are added: $cost(\ tree\ ,\ data\ ) = cost(\ data \mid tree\ ) + cost(\ tree\ )$

2.4 ... by statistically correcting the training error

Typically, generalization errors are larger than training errors.

⇨ can be estimated from the training error by assuming a likelihood of correctness of each split (each “fork“ in the tree) and computing its upper limit for the entire tree (as a function of its depth).

2.5 ... by using a validation set

- Trainings set is divided into two sub-sets,
  - one for building the model
  - one for testing the model
- typically: 2/3 of DO for training, 1/3 for validation

Avoiding Overfitting in decision trees

1. Pre-Pruning (Early Stopping Rule)

Approach

A more restrictive stopping condition `stopping_cond`(E,F)

Stopping tree development before it becomes 100% correct (w.r.t. training data)

typical stop conditions:

- All DO belong to the same class
- All attribute values are identical

$\Rightarrow$ **more restrictive stop condition**

- Number of DO is lower than a certain minimum
- Class distribution does not depend any more from attribute values (correlation between class and attribute values is lower than a minimum)
- Purity gain of splitting (e.g. Gini, Information Gain) is below a minimum

Problem

- Even a slight purity gain of a splitting can result in high purity gains of subsequent splits.

2. Post-Pruning

- Develop decision tree as far as possible (until same class or same attribute values)
- Cut the tree from bottom to top, i.e. replace sub-trees by …
  - a leaf containing the majority class of the DO (***Subtree Replacement***)
  - its (by training DO) most frequent used sub-sub-tree in it (***Subtree Raising***)

  ..., in case the generalization error decreases by doing so.

Advantage

- Generally better than pre-pruning, because it starts with a completely developed tree and avoids an unfair early stop of further splitting

Drawback

- worse (time-, space-) complexity, because the complete tree needs to be built first

Performance estimation of a method to construct a classifier
(not of a tree itself!)
(derived from all available classified DO)

1. Holdout – Method

- partition all classified DO into a trainings- and a test set (e.g. half each or 2/3 : 1/3)
- tree construction done by trainings set, performance estimation done by test set

Drawbacks

- less DO for tree construction ⇒ worse tree performance
- partition sizes are subject to a trade-off between a good model and a reliable performance statement
- interdependence between both sets: a class that is over-represented in one set, will be under-represented in the other

2. Arbitrary Partitioning (Random Sub-Sampling)

- repeated Holdout with different training / test set distributions
- computing the average of the resulting performance values

Advantage

- several models are built, from which the best can be chosen

Drawback

- same as with simple Holdout
- no control on the issue, how often a DO is used for training and for test
  - ⇨ some DO may be used much more frequent for training than for test and vice versa

3. Cross Validation

- each DO is
  - equally frequent used for training and
  - exactly once used for testing
- partition the set of classified DO into k equally sized sub-sets
- in k cycles, each partition is used exactly once for test, whereas the union of the other $k-1$ sub-sets is used for tree construction
- compute the average of the k resulting performance values

Special case

- $k = |\ example\ set\ |$, called “leave one out“ approach

Advantages

- uses as many as possible DO for model construction
- test sets are disjunct

Drawbacks

- high complexity, $|\ example\ set\ |$ cycles
- reliability of the performance statement becomes weaker by the fact, that it was derived from one DO only

4. Bootstrap

- all previous methods do not use a DO multiple (in the same cycle)
- here, the training set is formed by randomly choosing a number of training DO from always the entire set of classified DO (sampling with replacement)
- i.e., there is a certain likelihood that a chosen DO occurs multiple times in the training set
- by choosing N DO out of a set that (also) contains N DO with this method (sampling with replacement), the training set finally will contain 63,2 % of the original DO:
  - probability to be chosen for training is $1-(1-1/N)^N$
  - for $N \rightarrow \infty$, this value converges towards $1-1/e \approx 0.632$.
- DO, which don‘t end up in the training set by doing so, are used for test
- by repeating this method b times and deriving a correctness rate ϵ_i $(i=1...b)$ each time, the total correctness rate acc_{boot} can be estimated (with the real correctness rate acc_s over all DO, which only can be estimated as being 1) as follows:

$$acc_{boot} = \frac{1}{b}\sum_{i=1}^{b}(0.632 * \epsilon_i + 0.368 * acc_s)$$

2.2 Rule Based Classifiers

Given: Data Set

DO	Species	Body Temp.	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hiber-nates	Class
1	human	*warm*	*hair*	*yes*	*no*	*no*	*yes*	*no*	***Mammal***
2	python	*cold*	*scales*	*no*	*no*	*no*	*no*	*yes*	***Reptile***
3	salmon	*cold*	*scales*	*no*	*yes*	*no*	*no*	*no*	***Fish***
4	whale	*warm*	*hair*	*yes*	*yes*	*no*	*no*	*no*	***Mammal***
5	frog	*cold*	*none*	*no*	*semi*	*no*	*yes*	*yes*	***Amphibian***
6	komodo dragon	*cold*	*scales*	*no*	*no*	*no*	*yes*	*no*	***Reptile***
7	bat	*warm*	*heir*	*yes*	*no*	*yes*	*yes*	*yes*	***Mammal***
8	pigeon	*warm*	*feathers*	*no*	*no*	*yes*	*yes*	*no*	***Bird***
9	cat	*warm*	*fur*	*yes*	*no*	*no*	*yes*	*no*	***Mammal***
10	guppy	*cold*	*scales*	*yes*	*yes*	*no*	*no*	*no*	***Fish***
11	alligator	*cold*	*scales*	*no*	*semi*	*no*	*yes*	*no*	***Reptile***
12	Penguin	*warm*	*feathers*	*no*	*semi*	*no*	*yes*	*no*	***Bird***
13	porcupine	*warm*	*quills*	*yes*	*no*	*no*	*yes*	*yes*	***Mammal***
14	eel	*cold*	*scales*	*no*	*yes*	*no*	*no*	*no*	***Fish***
15	salamander	*cold*	*none*	*no*	*semi*	*no*	*yes*	*yes*	***Amphibian***

looked for:
rule set, which classifies the data as correct as possible

R1: (Gives Birth = no) $\wedge$ (Aerial Creature = yes) $\rightarrow$ Bird
R2: (Gives Birth = no) $\wedge$ (Aquatic Creature = yes) $\rightarrow$ Fish
R3: (Gives Birth = yes) $\wedge$ (Body Temp = warm) $\rightarrow$ Mammal
R4: (Gives Birth = nein) $\wedge$ (Aerial Creature = nein) $\rightarrow$ Reptile
R5: (Aquatic Creature = semi) $\rightarrow$ Amphibian

Rule r_i: $Premise_i \rightarrow Konklusion_i$

- *Coverage(r_i)*
  = *number of DO with true premise / number of all DO*
- *Accuracy(r_i)*
  = *number of DO with true premise / number of DO with true premise and true conclusion*

- A rule set is **mutually exclusive**, if for each DO there is at most one rule with a true premise.

  Mutually non-exclusive rule sets may result in conflicting classes. There are two ways to overcome this problem:

  - **Ordered Rules**: Rules are ordered according to their priority (*Accuracy* or *Coverage* e.g.). The highest priority decides, which rule (out of the conflict set) is applied.
  - **Unordered Rules**: All rules of the conflict set are applied. The frequency of derived classes (weighted by the rules' *Accuracies*, if applicable) decides the class membership.

- **Rule Ordering Schemes** for ordered rules
  - **Rule-Based Ordering Scheme**:

    Order according a rule quality measure (e.g. *Accuracy*)
  - **Class-Based Ordering Scheme**:
    - Rules are grouped according to their conclusion (class)
    - Groups are ordered according to their classes
    - Within a group, the sequence doesn‘t matter, because the conclusion (class) is same anyway

- A rule set is **complete**, if there is at least one true rule premise for each combination of attribute values.

  Incomplete rule sets may be completed by a Default Rule with an empty (true) premise and a default conclusion (class).

Most rule generation algorithms use class based ordering scheme.

Rule Extraction: **direct / indirect methods**

Direct Rule Extraction

- Order of handling the classes may depend on
  - Number of DO belonging to it
  - Costs of miss-classification for each class
  - Frequency of occurrence (independently from its frequency in the training set)
- For each class
  - DO of this class = positive examples
  - DO of other classes: = negative examples

Algorithm

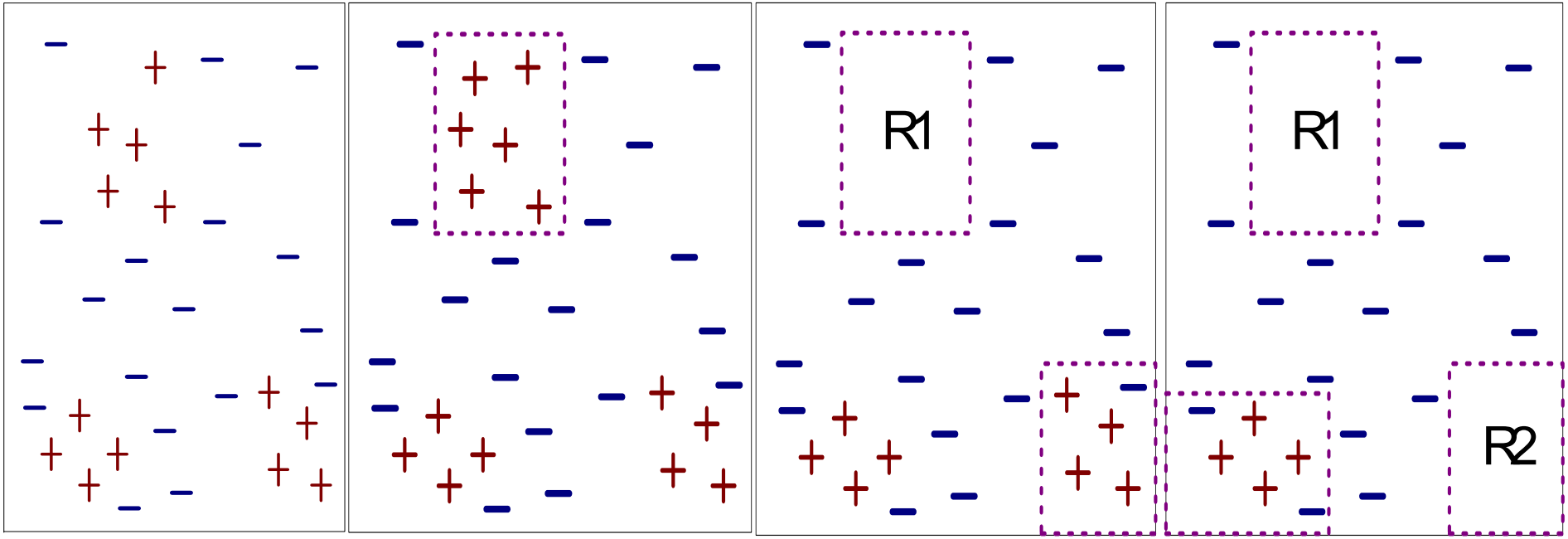
E: training set, A: set of all attribute-value-pairs
$Y = \{ y_1 , y_2 , \ldots , y_k \}$:ordered set (list) of all classes
$R :=$ ∅ (initial) list of rules
for $y = y_1, \ldots, y_{k-1}$ **do**
  while stop condition is not met **do**
    $r \leftarrow$ LearnOneRule(E , A , y)
    remove DO from E, for which r 's premise is true
    add r to the (end of) rule set: $R \leftarrow R \cup \{ r \}$
  end while
end for
Add default rule $true \rightarrow y_k$ to the (end of) the rule set

A rule is good for a considered class,

- all (or most) positive and
- (almost) no negative

examples are covered by it.

After finding such a rule, the examples that are covered by it, are removed from the training set:



Rule Growing Strategies

- **top down** (conclusion driven, general-to-specific)
- **bottom up** (premise driven, specific-to-general)

top down rule growing

- initial rule for a class y_i: empty list of (with „and“ connected) premises
- new premises are systematically added to the list
- next premise should be an attribute, which, when having a certain value, covers
  - a max. number of DO of the considered class and
  - a min. number of DO of other classes

bottom up rule growing

- initial rule for a class y_i is recruited from a DO of this class
- rule is systematically generalized by removing premises
- such premises are preferred, by which‘s removal
  - a max. number of DO of the considered class and
  - a min. number of DO of other classes

  are covered additionally.

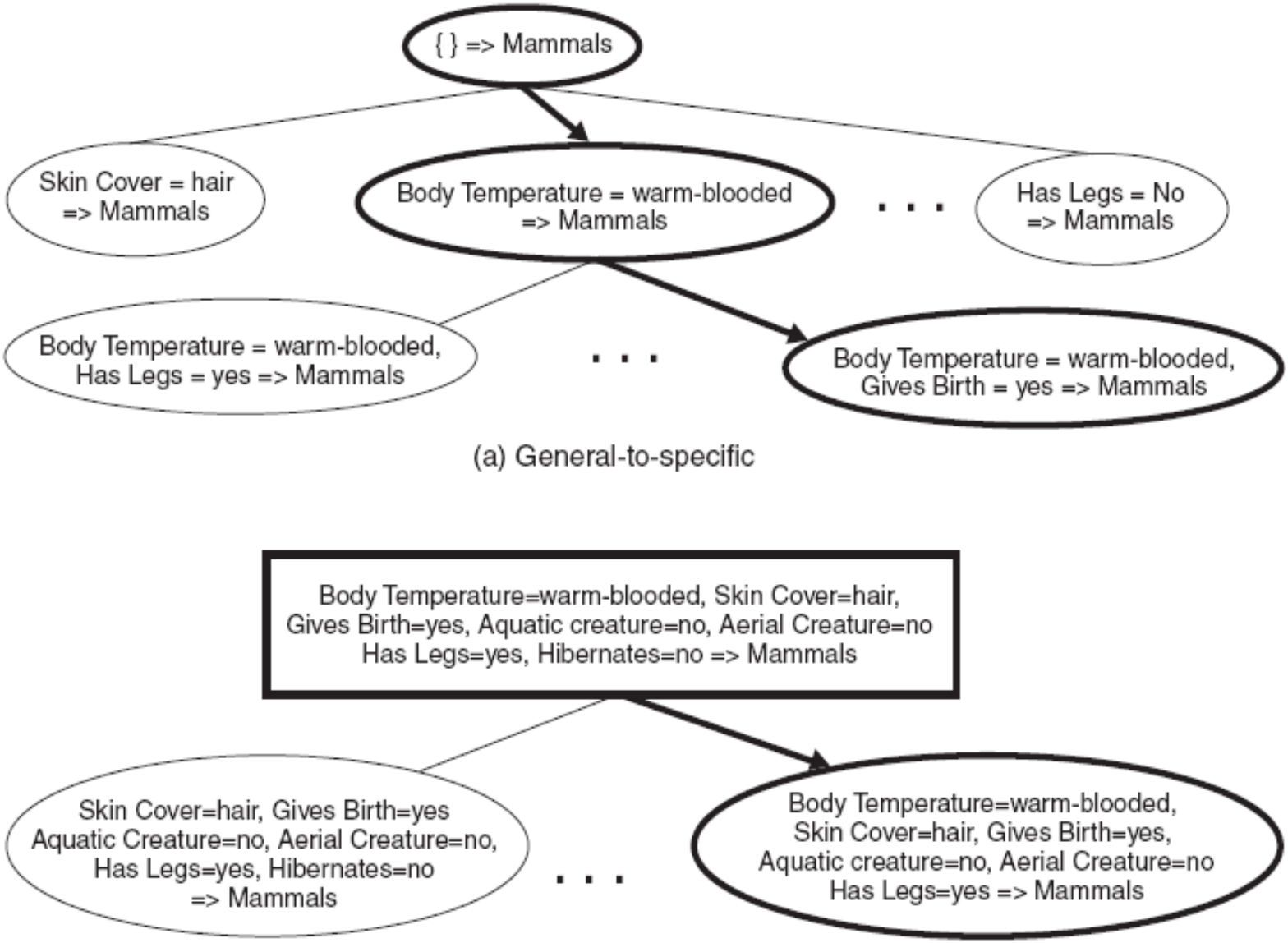


Figure 5.3. General-to-specific and specific-to-general rule-growing strategies.

Rule Evaluation
for selecting an appropriate attribute to add / remove

using *Accuracy(r)* ...

... is not a good idea, because *Coverage(r)* stays unconsidered:

Imagine a data set with 60 positive and 100 negative DO.

- Let r_1 be a rule, which covers 50 positive and 5 negative DO.
- Let r_2 be a rule, which covers 2 positive and no negative DO.

r_2 has a better *Accuracy*, but r_1 is for sure a better rule.

Rule Evaluation

for selecting an appropriate attribute to add / remove

1. Statistical Test

Computing the likelihood ratio statistic:

$$R = 2 \sum_{i=1}^{k} f_i \, ld\left(\frac{f_i}{e_i}\right)$$

k number of classes

f_i number of DO in class i, which are covered by the rule

e_i expected frequency of a rule that makes random predictions
= estimated number of correct classifications out of the covered DO, when "guessing" the class

Consider the example with 60 positive and 100 negative DO

- Let r_1 be a rule, which covers 50 positive and 5 negative DO
- Let r_2 be a rule, which covers 2 positive and no negative DO

$r_1 : f_+ = 50 \;\; f_- = 5$

$$e_+ = 55 * \frac{60}{160} = 20.625 \;\; e_- = 55 * \frac{100}{160} = 34.375$$

$$R(r_1) = 2\left(50 * ld\left(\frac{50}{20.625}\right) + 5 * ld\left(\frac{5}{34.375}\right)\right) = 99.9$$

$r_2 : f_+ = 2 \;\; f_- = 0$

$$e_+ = 2 * \frac{60}{160} = 0.750 \;\; e_- = 2 * \frac{100}{160} = 1.250$$

$$R(r_2) = 2\left(2 * ld\left(\frac{2}{0.750}\right) + 0 * ld\left(\frac{0}{1.250}\right)\right) = 5.66$$

2. LAPLACE, m-estimate

take the *Coverage* of a rule into account

$$Laplace(r) = \frac{f_+ + 1}{n + k} \qquad m - estimate(r) = \frac{f_+ + k * p_+}{n + k}$$

n	number of DO, which are covered by the rule
f_+	number of pos. DO, which are covered by the rule
k	number of classes
p_+	fraction of pos. DO in the entire training set

For $p_+ = 1/k$ is $Laplace(r) = m\text{-}estimate(r)$.

Consider the example with 60 positive and 100 negative DO

- Let r_1 be a rule, which covers 50 positive and 5 negative DO
- Let r_2 be a rule, which covers 2 positive and no negative DO

$$Laplace(r_1) = \frac{50 + 1}{55 + 2} = 0.8947 \qquad m - estimate(r_1) = \frac{50 + 2 * \frac{60}{160}}{55 + 2} = 0.8904$$

$$Laplace(r_2) = \frac{2 + 1}{2 + 2} = 0.75 \qquad m - estimate(r_2) = \frac{2 + 2 * \frac{60}{160}}{2 + 2} = 0.6875$$

3. FOIL's information gain

combines the information gain with the rule's *Coverage* after its modifikation

$$\text{FOIL's information gain} = p_1\left(ld\left(\frac{p_1}{p_1+n_1}\right) - ld\left(\frac{p_0}{p_0+n_0}\right)\right)$$

When expanding a rule $r\colon A \rightarrow +$ to $r'\colon A \wedge B \rightarrow +$,

p_0 is the number of pos. DO covered by r

n_0 is the number of neg. DO covered by r

p_1 is the number of pos DO covered by r'

n_1 is the number of neg. DO covered by r'

Consider the example with 60 positive and 100 negative DO

- Let r_1 be a rule, which covers 50 positive and 5 negative DO
- Let r_2 be a rule, which covers 2 positive and no negative DO

$$FOIL(r_1) = 50\left(ld\left(\frac{50}{55}\right) - ld\left(\frac{60}{160}\right)\right) = 63.87$$

$$FOIL(r_2) = 2\left(ld\left(\frac{2}{2}\right) - ld\left(\frac{60}{160}\right)\right) = 2.83$$

- **Stop Condition of the Rule Extraction Algorithm** (page 65)
  - Compute the gain of a new rule (statistically, Laplace, m-estimate, FOIL's information gain).
  - Is the gain not significant, refuse the new rule and stop the algorithm

- **Rule Pruning**
  - like decision tree generation methods, rule generation methods tend towards overfitting (see page 45 ff) and need similar Pruning Methods (see page 51 ff).

- **Algorithmic Aspects**
  - The Rule Extraction Algorithm (see page 65) contains the removal of those DO, which are covered by the generated rule.
  - However, rules' coverages may overlap.
  - Thus, the estimation of the incremental improvement of a new rule is realistically only with a training data set after removing DO, which are covered already.

RIPPER Algorithm

- widespread
- appropriate for unbalanced class distribution of DO
- appropriate for noisy data
- overfitting is avoided by a Validation Set
- **two class problems**
  - majority class is set to be the default class
  - rules for the minority class are extracted
- **multi class problems**
  - classes are sorted according to their frequencies: $[y_1, y_2, ..., y_c]$
  - RIPPER starts with the least frequent class y_1
  - in the i-th cycle, rules for y_i are generated, whereas
    - DO for y_i are considered as positive DO and
    - all others ($y_{i+1} \cup y_{i+2} \cup ... \cup y_c$) are considered as negative DO of a two class problem
  - there are $c-1$ cycles and finally, y_c is the default class

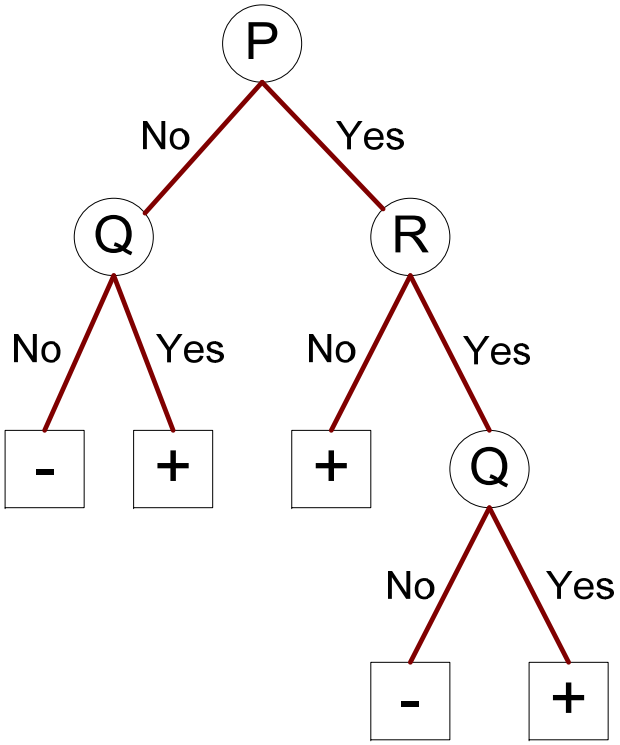
RIPPER Algorithmus

- top-down rule extraction (see page 67)
- FOIL's information gain is used to select the best premise for further rule refinement
- stops, if a rule no longer covers negative examples
- after extracting a rule, all DO covered by it will be removed from the data set and the rule will be added
- rule extraction stops, if
  1. a new rule increases the ***Description Length*** (definition similar than with decision trees, see page 54) of the rule set increases by d bits or more (default: $d = 64$) or
  2. the misclassification rate of a new rule exceeds 50 %
- post-pruning of each rule $r \in R$ by a validation set
  - metric: $(p-n)/(p+n)$ with p = number of pos. DO and n = number of neg. DO
  - if this metric improves (increases) by temporally removing a premise, this premise will be finally removed
  - pruning starts with the last premise and checks premises from the last to the first in this sequence
  - before pruning, r covers pos. DO only
  - after pruning, r covers both pos. and neg. DO of the validation set

Indirect Rule Extraction

First, a Decision Tree is constructed, which is transformed into a Rule Set.

Simple approach: making a rule out of each path



Rule Set

r1: (P=No,Q=No) ==> -
r2: (P=No,Q=Yes) ==> +
r3: (P=Yes,R=No) ==> +
r4: (P=Yes,R=Yes,Q=No) ==> -
r5: (P=Yes,R=Yes,Q=Yes) ==> +

Indirect Rule Extraction

This approach can be optimized, e.g.

r_2 $(P=no) \wedge (Q=yes) \rightarrow +$
r_3 $(P=yes) \wedge (R=no) \rightarrow +$
r_5 $(P=yes) \wedge (R=yes) \wedge (Q=yes) \rightarrow +$

towards

r_2' $(Q=yes) \rightarrow +$
r_3 $(P=yes) \wedge (R=no) \rightarrow +$

both rule sets describe the same boolean function:

P	Q	R	
no	no	no	
no	no	yes	
no	yes	no	+
no	yes	yes	+
yes	no	no	+
yes	no	yes	
yes	yes	no	+
yes	yes	yes	+

Approach of the C4.5 Rule Extraction Algorithm

Rule Generation

- First, a rule is made out of each path of the decision tree
- For the removal of each conjunction, the ***Pessimistic Error Rate*** (see page 52/53) is computed
- The rule reduction with the smallest *Pessimistic Error Rate* is performed, if it results in a better (smaller) Error Rate compared to the original rule
- The resulting reduced (shortened) rule is subject to the same procedure for the remaining conjunctions etc. until there is no more improvement for any conjunction
- By doing that, identical rules may come up. Therefore, finally duplicates will be removed

Rule Sorting

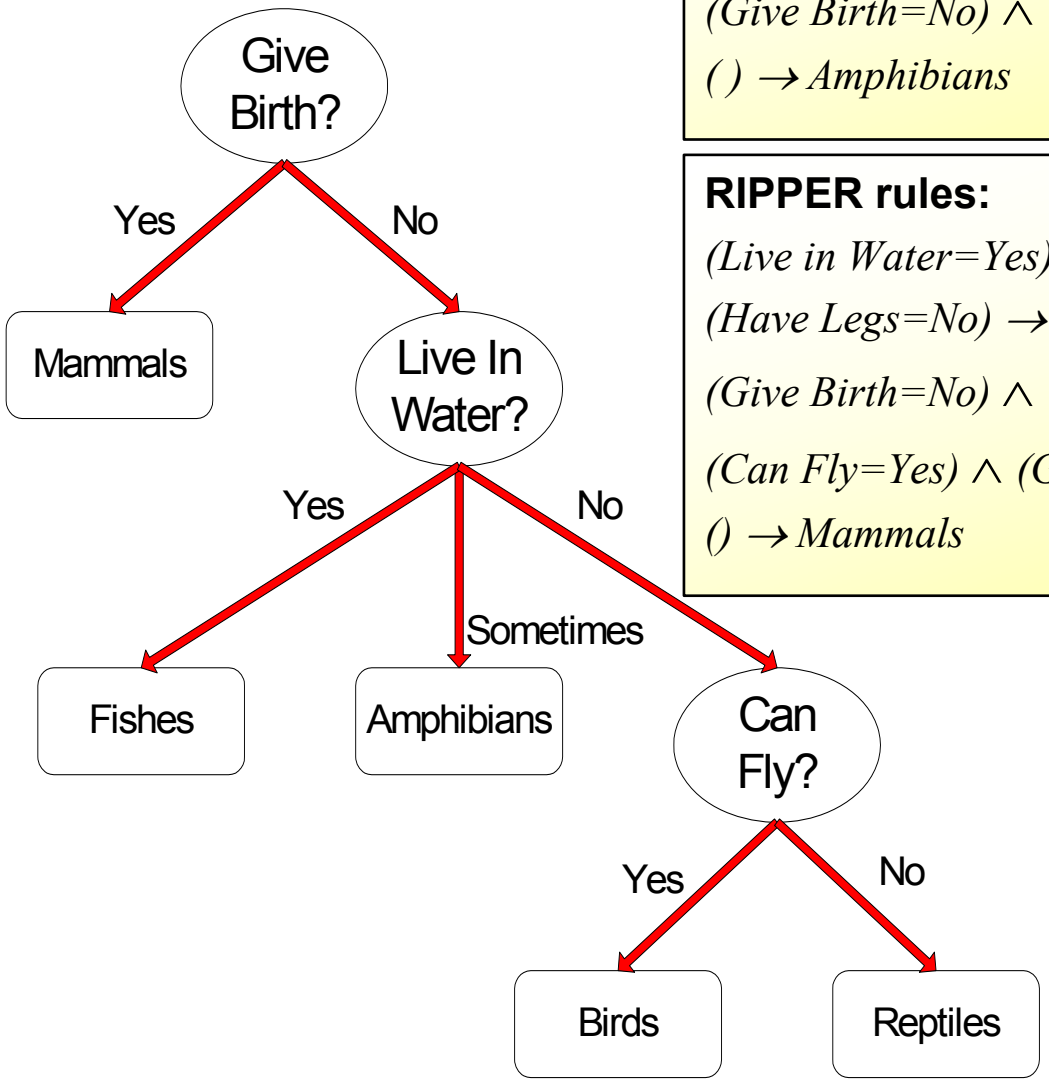
- Class based, rules of the same class are grouped together
- Groups are sorted according increasing ***Total Description Length*** (see page 52)
- $Description\ Length = L_{exception} + g * L_{model}$ with
  - $L_{exception}$ number of bits, which are needed to code the misclassified DO
  - L_{model} number of bits, which are needed to code the model (the rules)
  - g Tuning-Parameter, which is the smaller, the more redundant attributes are in the model, default value: 0.5

Example to compare indirect (C 4.5) and direct (RIPPER) methods

Name	Give Birth	Lay Eggs	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	no	yes	mammals
python	no	yes	no	no	no	reptiles
salmon	no	yes	no	yes	no	fishes
whale	yes	no	no	yes	no	mammals
frog	no	yes	no	sometimes	yes	amphibians
komodo	no	yes	no	no	yes	reptiles
bat	yes	no	yes	no	yes	mammals
pigeon	no	yes	yes	no	yes	birds
cat	yes	no	no	no	yes	mammals
leopard shark	yes	no	no	yes	no	fishes
turtle	no	yes	no	sometimes	yes	reptiles
penguin	no	yes	no	sometimes	yes	birds
porcupine	yes	no	no	no	yes	mammals
eel	no	yes	no	yes	no	fishes
salamander	no	yes	no	sometimes	yes	amphibians
gila monster	no	yes	no	no	yes	reptiles
platypus	no	yes	no	no	yes	mammals
owl	no	yes	yes	no	yes	birds
dolphin	yes	no	no	yes	no	mammals
eagle	no	yes	yes	no	yes	birds

Results

decision tree



C4.5 rules:

(Give Birth=No) ∧ (Can Fly=Yes) → Birds

(Give Birth=No) ∧ (Live in Water=Yes) → Fishes

(Give Birth=Yes) → Mammals

(Give Birth=No) ∧ (Can Fly=No) ∧ (Live in Water=No) → Reptiles

( ) → Amphibians

RIPPER rules:

(Live in Water=Yes) → Fishes

(Have Legs=No) → Reptiles

(Give Birth=No) ∧ (Can Fly=No) ∧ (Live In Water=No) → Reptiles

(Can Fly=Yes) ∧ (Give Birth=No) → Birds

() → Mammals

C4.5 rules versus RIPPER rules

correct classes vs. misclassifications

C4.5 rules:

			PREDICTED CLASS			
		Amphibians	**Fishes**	**Reptiles**	**Birds**	**Mammals**
ACTUAL	**Amphibians**	**2**	0	0	0	0
CLASS	**Fishes**	0	**2**	0	0	**1**
	Reptiles	**1**	0	**3**	0	0
	Birds	**1**	0	0	**3**	0
	Mammals	0	0	**1**	0	**6**

RIPPER rules:

			PREDICTED CLASS			
		Amphibians	**Fishes**	**Reptiles**	**Birds**	**Mammals**
ACTUAL	**Amphibians**	0	0	0	0	**2**
CLASS	**Fishes**	0	**3**	0	0	0
	Reptiles	0	0	**3**	0	**1**
	Birds	0	0	**1**	**2**	**1**
	Mammals	0	**2**	**1**	0	**4**

Advantages of Rule-Based Classifiers

- ✓ As highly expressive as decision trees
- ✓ Easy to interpret
- ✓ Easy to generate
- ✓ Can classify new instances rapidly
- ✓ Performance comparable to decision trees

2.3 Nearest Neighbor Classification (kNN)

- "eager" learners learn a model (a decision tree, a rule set, …) as soon as trainings DO are available
- "lazy" learners do not model the training data until a a classification request for a new object comes up – so does kNN, too

Idea

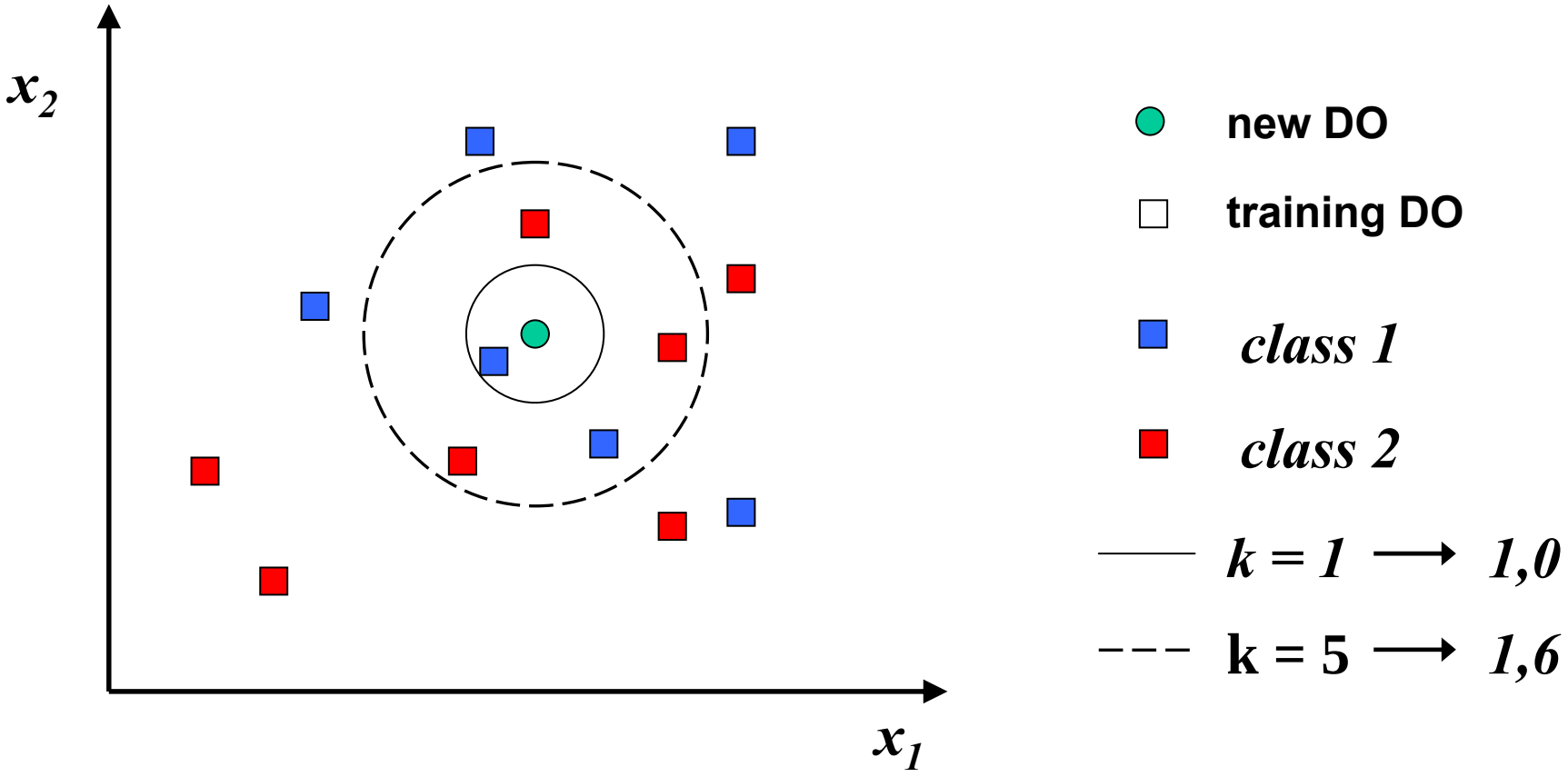
- Class is derived from the classes of the nearest neighbors in the training data, e.g.
  - Class of the majority of neighbors
  - (weighted, if applicable) average class

Approach

- Each training DO represents a dot in a d-dimensional space (d: number of attributes)
- For each new DO, the similarity with all training DOs is computed
- the k most similar training DO determine the class of the new DO
- **Similarity measures** (see page 9 ff):
  - Distance measures in the Euclidean space (Manhattan city block distance L_1, Euclidian distance L_2, Supremum L_∞)
  - Equivalence degrees for binary attributes (Simple Matching Coefficient, Jaccard Coefficient)
  - Direction measures in the Euclidean space (Cosine, Dice, Tanimoto, or Overlap Coefficient)
  - Linear interdependency between attribute values of two DO (Correlation)

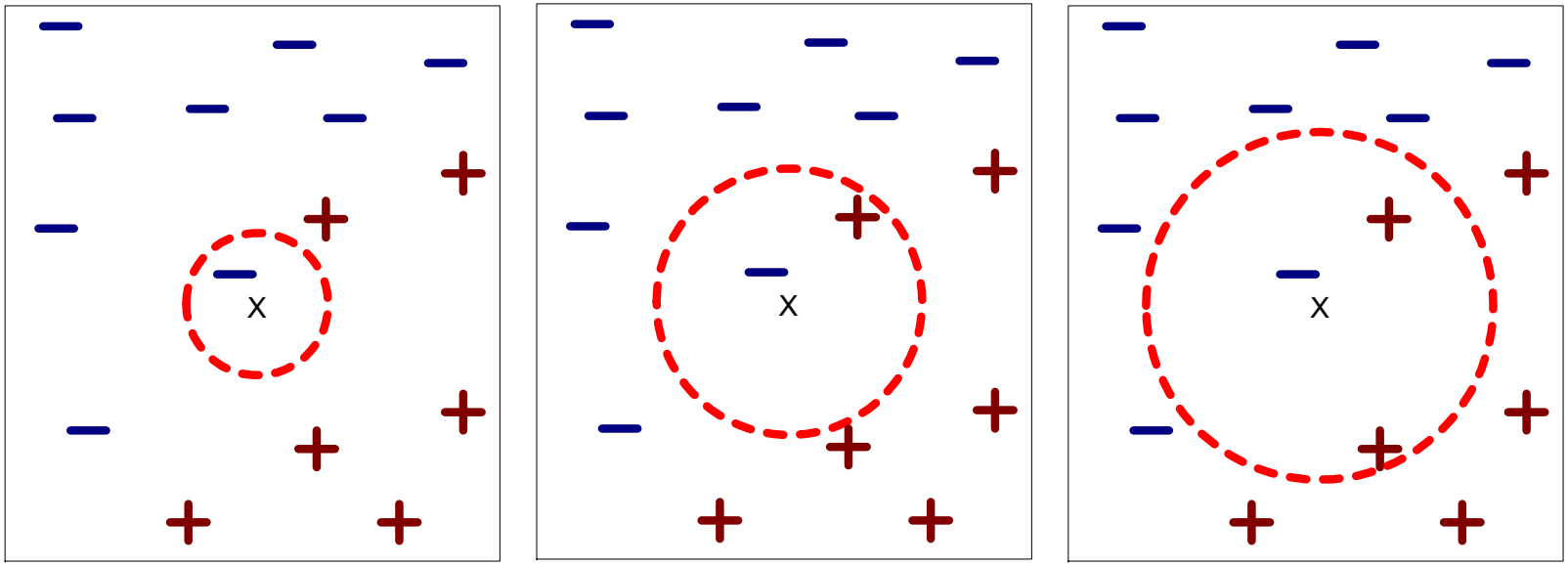
example

- 2 real valued attributes x_1, x_2
- 2 rational valued classes *class 1* and *class 2*
- Similarity measure: Euklidian distance (L_2 – norm)



another example

- 2 real valued attributes x_1, x_2
- 2 binary classes + und -
- Similarity measure: Euklidian distance (L_2 – norm)



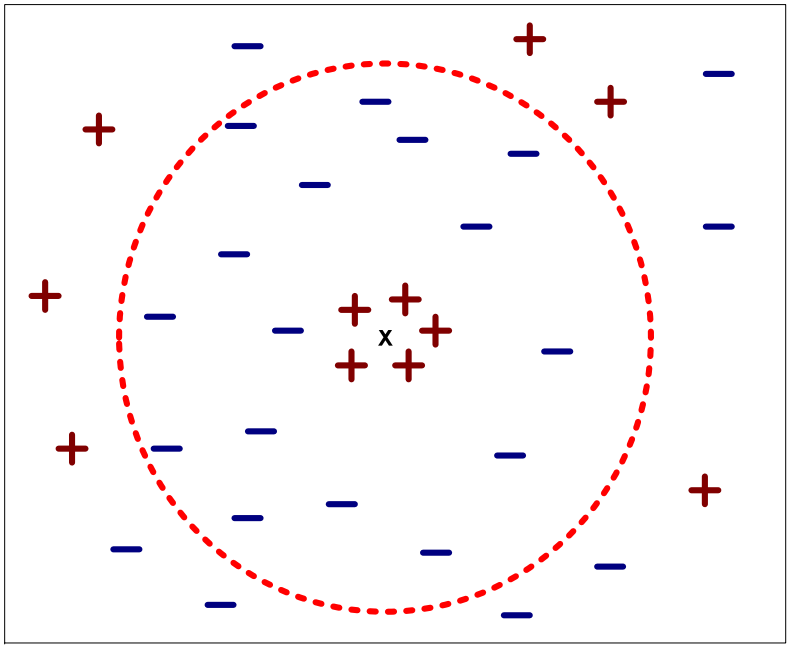
(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

How to come up with an appropriate value of k ?

- k too small ⇒ high risk of overfitting

  $k = 1$, e.g., may cover exactly an "outlier", despite all other training DO in the neighborhood hold a different class membership

- k too big ⇒ high risk of underfitting:



- in practice: $2 \leq k \leq 6$

- Each element of the set ***kNN(x')*** that contains the ***k*** nearest neighbors of a DO $z = [x', y']$ to be classified has the same degree of impact on the derived class membership y'

⇒ High sensibility in terms of ***k***

⇒ Decrement this sensibility by weighting each of the ***k*** neighbors x_i with its inverse quadratic distance to (similarity with) the DO to be classified x' :

$$y' = \arg\max_{v} \sum_{[x_i, y_i] \in kNN(x')} \frac{1}{d(x', x_i)^2} * ord(v = y_i)$$

with

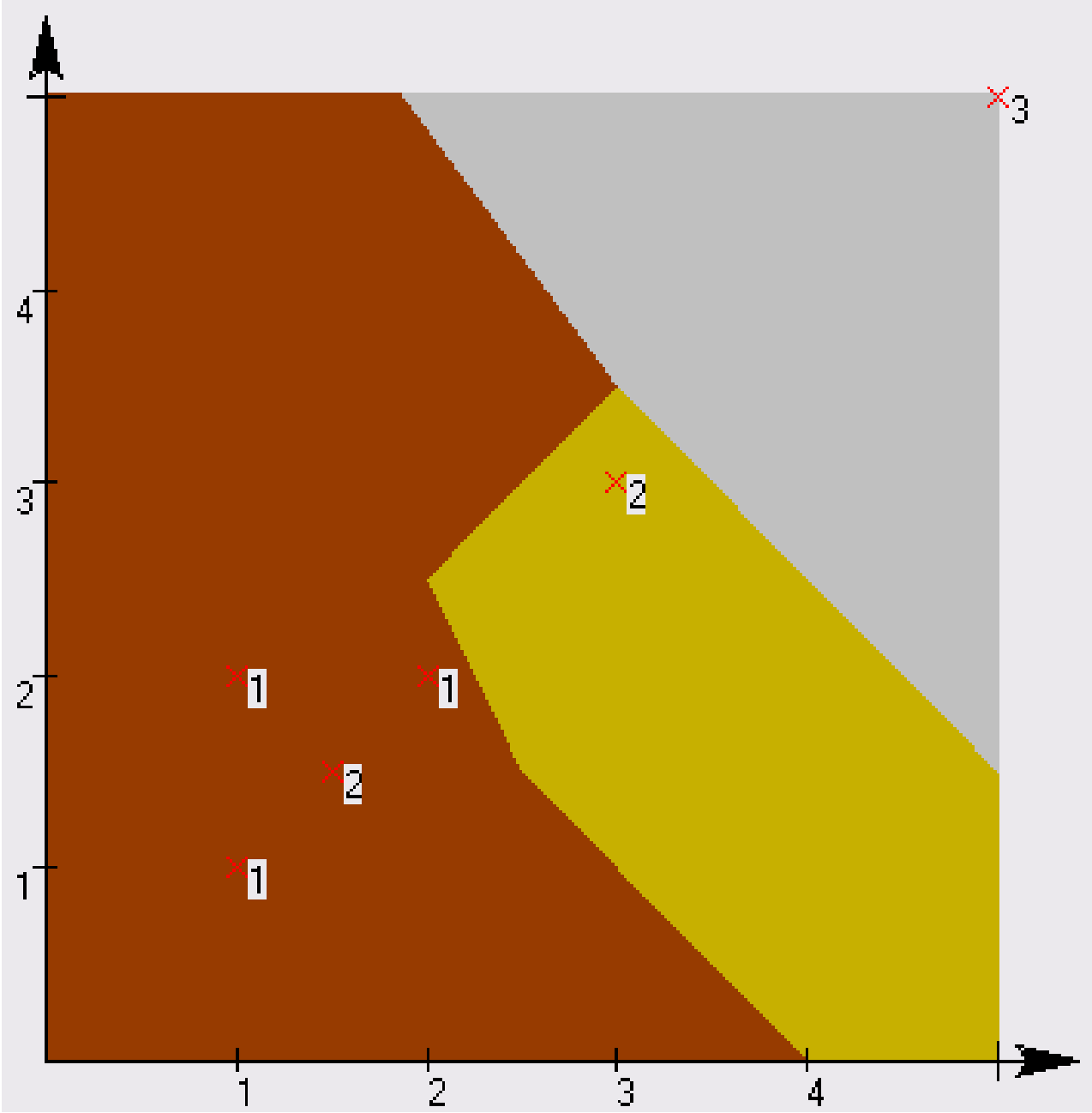
$$ord(prop) = \begin{cases} 1 & , \text{if } prop = true \\ 0 & , \text{if } prop = false \end{cases}$$

Representation of the class assignment in a d-dimensional space

Voronoi-Diagram

example

- $k = 3$
- three classes: $1, 2, 3$
- Similarity measure: Euklidian distance (L_2 norm)
- Simple majority decides class membership
- No weighting of the k most similar DO
- Grey area: no majority

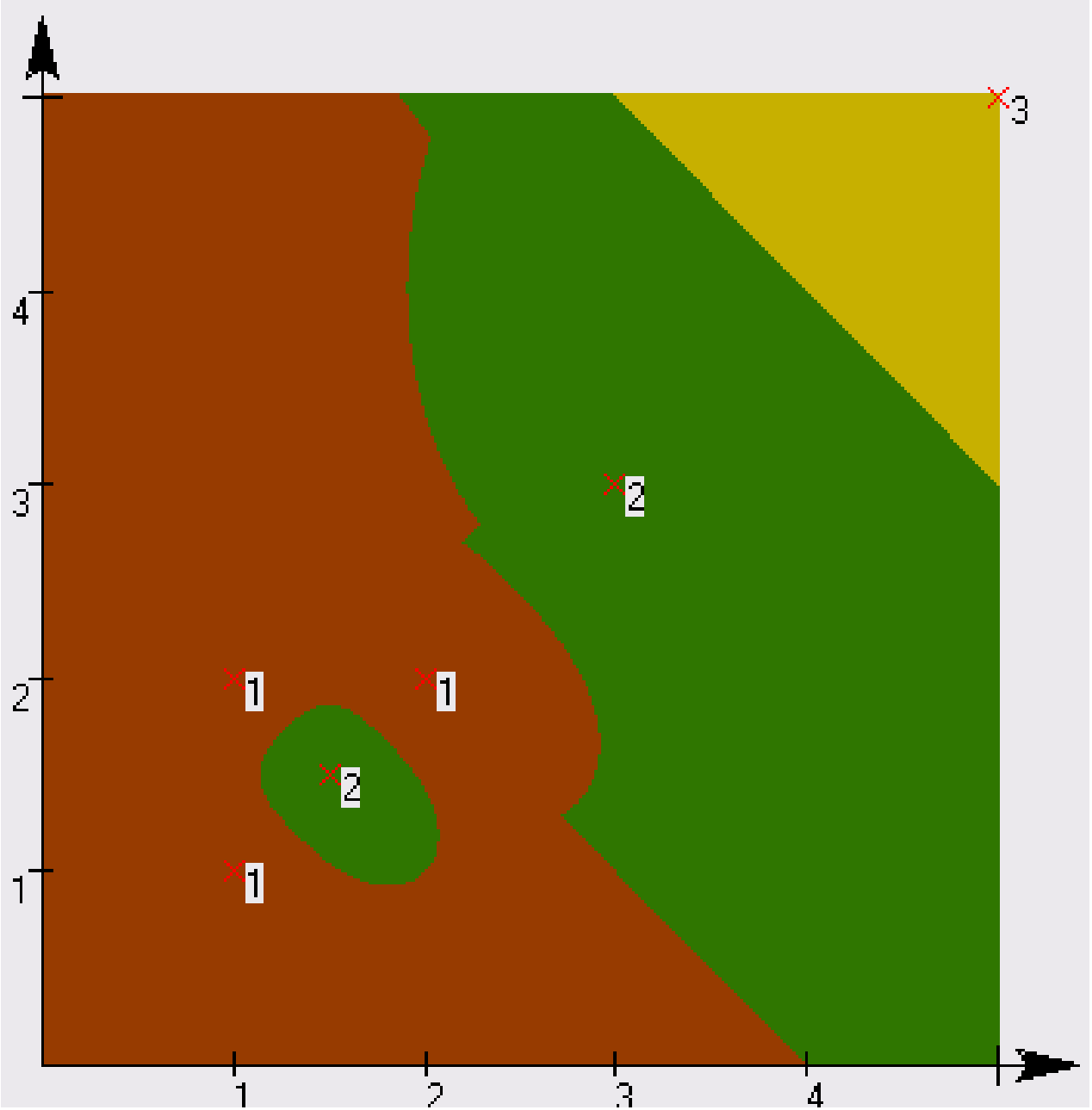


Representation of the class assignment in a d-dimensional space

Voronoi-Diagram

one more example
(same training DO as before)

- $k = 3$
- three classes: $1, 2, 3$
- Similarity measure: Euklidian distance (L_2 Norm)
- Simple majority decides class membership
- Weighting of the k most similar DO with their inverse quadratic distance



2.4 Bayesian Classification

Given
- set of DO I
- k-dimensional attribute space, in which a dot can be assigned to each DO $i \in I$
- a partitioning of I into disjunct object classes $C = \{ c_1, \ldots, c_n \}$
- a training set $E \subset I$: objects with known class membership

looked for
- Most likely class membership of a new object (without building a model)

➢ Bayesian Classification is also a “lazy“ learning method (like kNN)

Parallels to Case Based Reasoning (CBR)

➢ CBR is based on a case library, in which (a) a “most similar“ case is retrieved and (b) its solution is adapted towards a proper solution of the current case

➢ Here, (a) is degenerated to the most likely case and (b) degenerates to a copy of its solution

Example

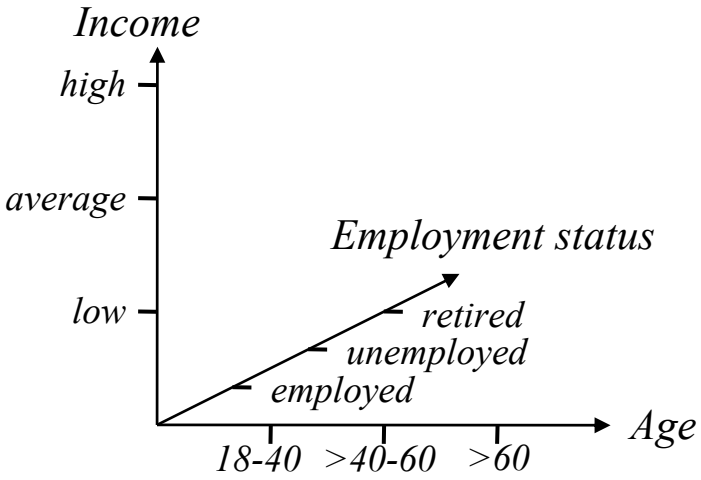
Given

- Set of objects: *residents of the city Ilmenau*
- Three-dimensional attributes space: *see figure below*
- Four disjunct classes: *residents with a daily TV consumption of*
  - *(1) less than 1 hour*
  - *(2) >1 – 2 hours*
  - *(3) >2 – 3 hours*
  - *(4) more than 3 hours*
- Trainings set with 20 examples: *see table at next page*

looked for

- Most likely class membership of a 25 year old employed resident with an average income

Attribute space:



Training Data (example set)

Age	Income	Employment status	class
22	*average*	*employed*	*>1 – 2 h*
43	*high*	*employed*	*>1 – 2 h*
45	*low*	*employed*	*>1 – 2 h*
76	*average*	*retired*	*>3 h*
34	*low*	*unemployed*	*>1 – 2 h*
34	*average*	*employed*	*>2 – 3 h*
56	*high*	*employed*	*>3 h*
32	*average*	*employed*	*>1 – 2 h*
23	*low*	*unemployed*	*>2 – 3 h*
65	*average*	*retired*	*>2 – 3 h*

Age	income	Employment status	class
33	*low*	*unemployed*	*>3 h*
36	*high*	*employed*	*>1 – 2 h*
25	*low*	*unemployed*	*≤ 1 h*
47	*average*	*employed*	*≤ 1 h*
45	*high*	*employed*	*>2 – 3 h*
63	*average*	*retired*	*>3 h*
51	*high*	*employed*	*>1 - 2 h*
60	*average*	*retired*	*>1 – 2 h*
31	*low*	*unemployed*	*>1 – 2 h*
39	*average*	*employed*	*>2 – 3 h*

BAYES' Formula *(here: interpreted by logic, in literature: a set theory)*

1. Let A and B be propositions about events and $P(A)$ and $P(B)$ likelihoods that A and B take place, A and B are not impossible: $0 < P(A),P(B) \leq 1$.
2. The conditional likelihood of A under the condition of B is $P(A \mid B) = \frac{P(A \wedge B)}{P(B)}$
3. Let $\Omega = B_1 \vee \ldots \vee B_n$ be a sure event consisting of pair-wise contradicting events B_1, ..., B_n: $P(B_1 \vee \ldots \vee B_n) = \sum_{i=1}^{n} P(B_i) = 1$, $P(B_i \wedge B_j) = 0$ für $i \neq j$
4. An accidental event A holds: $A = A \wedge \Omega = A \wedge (B_1 \vee B_2 \vee \ldots \vee B_n)$
5. Thus, $P(A) = P(A \wedge \Omega) = P(A) * \sum_{i=1}^{n} P(B_i) = \sum_{i=1}^{n} P(A \wedge B_i) = \sum_{i=1}^{n} P(B_i) * P(A \mid B_i)$

Vice versa, the likelihood of B_i under the condition of A (*a posteriori likelihood*) is

$$P(B_i \mid A) = \frac{P(B_i \wedge A)}{P(A)} = \frac{P(B_i) * P(A \mid B_i)}{P(A)} = \frac{P(B_i) * P(A \mid B_i)}{\sum_{i=1}^{n} P(B_i) * P(A \mid B_i)}$$

BAYES' Formula

2.4.1 Naïve BAYES Classification

given

- examples, i.e. DO with known values for all attributes A_i $(i=1,..,n)$ and known class membership B_j $(j=1,..,k)$

looked for

- A class $C \in \{B_1,...,B_k\}$, which is the most likely class for a new object with known attribute values for all A_i

$$C = B_j : \max\{P(B_j \mid A_1,..., A_n)\}$$

$$= B_j : \max\{\frac{P(A_1,..., A_n \mid B_j) * P(B_j)}{P(A_1,..., A_n)}\}$$ *according to BAYES' Formula*

$$= B_j : \max\{P(A_1,..., A_n \mid B_j) * P(B_j)\}$$ *ignoring division due to identical denominators*

$$= B_j : \max\{\prod_{i=1}^{n} P(A_i \mid B_j) * P(B_j)\}$$ *assuming independency of the attributes from each other, i.e. their conjunction multiplies their likelihoods*

Example revisited

Attributes

- *Age* $\in$ { *18-40*, *>40-60* , *>60* }
- *income* $\in$ { *low* , *average*, *high* }
- *Employment status* $\in$ { *employed*, *retired* , *unemployed* }

Classes

- B_1: $\leq$ *1 h daily TV consumption*
- B_2: *>1-2 h daily TV consumption*
- B_3: *>2-3 h daily TV consumption*
- B_4: *> 3 h daily TV consumption*

Examples *see* page 93

Looked for most likely class membership of a resident that holds the attribute values

A_1: *Age = 25* | A_2: *Income = average* | A_3: *Employment status = employed*

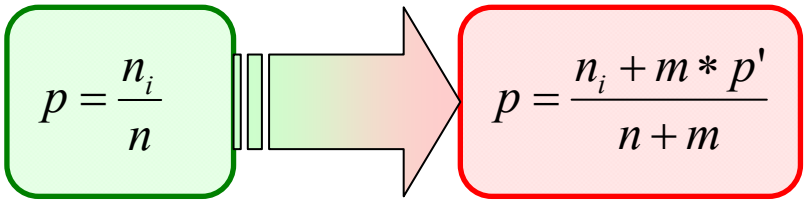
likelihoods, estimated by the relative frequency in the training set

				product of line
$P(A_1\|B_1)$: 1/2	$P(A_2\|B_1)$: 1/2	$P(A_3\|B_1)$: 1/2	$P(B_1)$: 2/20	**0,01250**
$P(A_1\|B_2)$: 5/9	$P(A_2\|B_2)$: 3/9	$P(A_3\|B_2)$: 6/9	$P(B_2)$: 9/20	**0,05556**
$P(A_1\|B_3)$: 3/5	$P(A_2\|B_3)$: 3/5	$P(A_3\|B_3)$: 3/5	$P(B_3)$: 5/20	**0,05400**
$P(A_1\|B_4)$: 1/4	$P(A_2\|B_4)$: 2/4	$P(A_3\|B_4)$: 1/4	$P(B_4)$: 4/20	**0,00625**
			Maximum of line products	**0,05556 (from B_2)**

The resident belongs to class B_2 *(>1-2 h daily TV consumption)* with the highest likelihood.

Refinement for training sets that are too small: m-estimate instead frequency in the training set

- Estimated likelihoods for belonging to a class B_i , for which there are n_i (out of n) DO in the training set, is supplemented by a weighted a priori estimated likelihood p' (e.g., from former observations)
- As weight factor, a so called equivalent sample size m is used
- In the extreme case of an empty training set, $n_i = 0$ and $n = 0$ and thus, $p = p'$



- n_i number of DO belonging to class i
- n total number of DO
- m equivalent sample size, weight of p'
- p' a priori estimated probability (e.g., from former observations)
  - could be estimated as equally class distribution, if there is no former data set)

Problem with Naïve Bayes: Correlating Attributes

Let's consider a 2-class problem (+ and -) with 2 binary attributes A and B with

- $p(A=0|-) = 0.4$, $p(A=1|-) = 0.6$,
- $p(A=0|+) = 0.6$, $p(A=1|+) = 0.4$
- Let B perfectly correlate with A in class – ($B=A$), *but independent in class* +
- for simplicity, let's assume
  - $p(+) = p(-)$
  - B has the same conditional probabilities as with A (see above)

According naïve Bayes

- $p(-|A=0,B=0) = \mathbf{p(A=0|-)\ p(B=0|-)} * p(-) / p(A=0, B=0) = \mathbf{0.16} * p(-) / p(A=0, B=0)$
- $p(+|A=0,B=0) = \mathbf{p(A=0|+)\ p(B=0|+)} * p(+) / p(A=0, B=0) = \mathbf{0.36} * p(+) / p(A=0, B=0)$

and + would be determined as the most likely class.

However, the truth is

- $p(A=0,B=0|-) = p(A=0|-) = 0.4$, because B is always the same as A in class –

Thus,

- $p(-|A=0,B=0) = p(A=0,B=0|-)\ p(-) / p(A=0,B=0) = \mathbf{0.4} * p(-) / p(A=0, B=0)$

(not ***0.16*** as calculated with naïve Bayes) and – should be the most likely class.

solution to this problems:
analyzing dependencies between attributes $\Rightarrow$ **Bayesian Belief Networks**

2.4.2 Bayesian Belief Networks