• 2.1 A few necessary R functions
• 2.2 Mathematics
• 2.3 Decision tree learning – Theory
• 2.4 Decision tree learning – Practice
• Summary
We already know from yesterday

- `<-` ... assignment operator
- `+ - * / ( )` ... basic arithmetics is applicable also to vectors, BUT works with vector elements!
- `c()` ... combines its arguments to form a vector
- `str()` ... structure of an object
- `length()` ... length of a vector
- `1:15` ... vector containing the given sequence of integers
- `x[5:7]; y[c(1,2,10)]` ... selecting elements from a vector
- `sample(x)` ... random permutation of a vector
- `help(), ?` ... built-in help
Working with external files

- `getwd()` ... to print the working directory
- `setwd()` ... to set your working directory
- `list.files()` ... to list existing files in your working directory
- `read.table()` ... to load data from a .csv file
  - This function is the principal means of reading tabular data into R.
Your objects in the R environment

- `ls()` ... to get the list of your existing objects
- `rm()` ... to delete an object
- `rm(list=ls())` ... to delete all your existing objects

```r
> ls()
[1] "c"  "data"  "g"  "i"  "index"
[6] "k"  "m"  "n"  "nn"  "prediction"

> rm(list=ls())
> ls()
character(0)
```

Exiting R

```r
> q()
```
Vector types

Vector elements can be numerical, logical, or string values
You cannot combine different types within a vector

```r
> x <- c(3,6,5,3,2,7,5)
> x
[1] 3 6 5 3 2 7 5
> y <- 3:9
> y
[1] 3 4 5 6 7 8 9
> x == y
[1] TRUE FALSE TRUE FALSE FALSE FALSE FALSE FALSE
> x < y
[1] FALSE FALSE FALSE TRUE TRUE TRUE TRUE TRUE TRUE
```
> z <- as.logical(c(T,T,F))
> z
[1] TRUE TRUE FALSE

> str(z)
 logi [1:3] TRUE TRUE FALSE

> sum(z)
[1] 2

Note: When you calculate the sum of a logical vector, logical true values are regarded as one, false values as zero.

# Does y have any elements bigger than x?
> sum(y > x)
[1] 4

>
In R, “vectors” of categorical values are called factors.

```r
examples <- read.table("wsd.development.csv", header=T)
> str(examples$SENSE)
   Factor w/ 6 levels "cord","division",..: 1 1 1 1 1 1 1 1 ... 

> levels(examples$SENSE)
[1] "cord" "division" "formation" "phone" "product" "text"
```

A factor stores both values and possible levels of a categorical variable. Levels are "names" of categorical values.
Examples: creating factors

```r
> word.forms <-
as.factor(c("lines", "line", "line", "line", "lines", "lines"))

> str(word.forms)
Factor w/ 2 levels "line","lines": 2 1 1 1 2 2

> table(word.forms)
word.forms
       line lines
     3     3

> people <- factor( c(1,1,1,0,1,0,0,0,1,0,1,1,1,1),
            labels=c("male", "female"))

> table(people)
people
          male female
      5         9
```
Looking at data in a data frame – head()

```r
> examples <- read.table("wsd.development.csv", header=T)

> head(examples)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>cord</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>safety</td>
<td>special</td>
<td>install</td>
<td>inside</td>
<td>NN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>cord</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>wash</td>
<td>a</td>
<td>and</td>
<td></td>
<td>NN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>cord</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>moor</td>
<td>steel</td>
<td>by</td>
<td></td>
<td>VBG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>cord</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>frozen</td>
<td>the</td>
<td>thaw</td>
<td>at</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>cord</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>dock</td>
<td>a</td>
<td>throw</td>
<td>to</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>cord</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>green</td>
<td>the</td>
<td>come</td>
<td>at</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A17 A18 A19 A20
1 IN  DT  lines  dobj
2 .   X    line  conj_and
3 ,   DT  lines  agent
4 IN  DT  lines  dobj
5 TO  DT  line  dobj
6 IN  DT  line  nsubj
>
```
Looking at data in a data frame – `table()`

```r
> str(examples$SENSE)
  Factor w/ 6 levels "cord","division",..: 1 1 1 1 1 1 ...
> table(examples$SENSE)
    cord division formation phone product text
      336     322      296     380    1838     352
```

Mathematicians call it contingency table (first used by K. Pearson, 1904).
Looking at data in a data frame – `table()`

```r
> str(examples$SENSE)
  Factor w/ 6 levels "cord","division","formation",..: 1 1 1 1 1 1 1 ...
> table(examples$SENSE)
   cord division formation phone product text
     336     322      296    380   1838    352
```

2-dimensional `table()`

```r
> table(examples$SENSE, examples$A19)

   line lined lines
  cord   226     0   110
division 250     0    72
   formation 189    2 105
   phone    201    0 179
product 1319    0 519
   text    207    0 145
```

Mathematicians call it **contingency table** (first used by K. Pearson, 1904).
Getting probability of factor levels using `table()`

```r
> table(examples$SENSE)/sum(table(examples$SENSE))

cord division formation phone product text
0.09534620 0.09137344 0.08399546 0.10783201 0.52156640 0.09988649
```
Getting probability of factor levels using `table()`

\[
> \text{table}(\text{examples}\$\text{SENSE})/\text{sum}(\text{table}(\text{examples}\$\text{SENSE}))
\]

<table>
<thead>
<tr>
<th></th>
<th>cord</th>
<th>division</th>
<th>formation</th>
<th>phone</th>
<th>product</th>
<th>text</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.09534620</td>
<td>0.09137344</td>
<td>0.08399546</td>
<td>0.10783201</td>
<td>0.52156640</td>
<td>0.09988649</td>
</tr>
</tbody>
</table>

The same using `nrow()`, and with rounded numbers

\[
> \text{round}(\text{table}(\text{examples}\$\text{SENSE})/\text{nrow}(\text{examples}), 3)
\]

<table>
<thead>
<tr>
<th></th>
<th>cord</th>
<th>division</th>
<th>formation</th>
<th>phone</th>
<th>product</th>
<th>text</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.095</td>
<td>0.091</td>
<td>0.084</td>
<td>0.108</td>
<td>0.522</td>
<td>0.100</td>
</tr>
</tbody>
</table>
>
Getting subsets from data frames

Getting a subset of observations

```r
> examples.only_lines <- subset(examples, A19=='lines')

> str(examples.only_lines)
'data.frame': 1130 obs. of 21 variables:
$ SENSE: Factor w/ 6 levels "cord","division",..: 1 1 1 1 1 1 1 1 1 1 ...
$ A1 : int 1 0 0 0 1 1 0 0 0 0 ... 
$ A2 : int 0 0 0 0 0 0 0 0 0 0 ... 
$ A3 : int 0 0 0 0 0 0 0 0 0 0 ... 
>
```
Getting subsets from data frames

Getting a subset of observations

> examples.only_lines <- subset(examples, A19=='lines')

> str(examples.only_lines)
'data.frame': 1130 obs. of 21 variables:
$ SENSE: Factor w/ 6 levels "cord","division",...: 1 1 1 1 1 1 1 1 ...
$ A1 : int 1 0 0 0 1 1 0 0 0 0 ...
$ A2 : int 0 0 0 0 0 0 0 0 0 0 ...
$ A3 : int 0 0 0 0 0 0 0 0 0 0 ...

Getting selected variables only

> examples[1:20, c('SENSE', 'A4', 'A19')]

– Will retrieve first 20 observations and select only the 3 given variables.
Machine learning requires some mathematical knowledge, especially

- statistics
- probability theory
- information theory
- algebra (vector spaces)
Why statistics and probability theory?

Motivation

• In machine learning, models come from data and provide insights for understanding data or making prediction.
• A good model is often a model which not only fits the data but gives good predictions, even if it is not interpretable.

Statistics

• is the science of the collection, organization, and interpretation of data
• uses the probability theory
Two purposes of statistical analysis

**Statistics** is the study of the collection, organization, analysis, and interpretation of data. It deals with all aspects of this, including the planning of data collection in terms of the design of surveys and experiments.

**Description**
- describing what was observed in sample data numerically or graphically

**Inference**
- drawing inferences about the population represented by the sample data
A random variable (or sometimes stochastic variable) is, roughly speaking, a variable whose value results from a measurement/observation on some type of random process. Intuitively, a random variable is a numerical or categorical description of the outcome of a random experiment (or a random event).

Random variables can be classified as either

- **discrete**
  - a random variable that may assume either a finite number of values or an infinite sequence of values (countably infinite)

- **continuous**
  - a variable that may assume any numerical value in an interval or collection of intervals.
Features as random variables

In machine learning theory we take features as random variables.

Target class is a random variable as well.

Data instance is considered as a vector of random values.
Probability theory – basic terms

Formal definitions

- random experiment
- elementary outcomes $\omega_i$
- sample space $\Omega = \bigcup \{\omega_i\}$
- event $A \subseteq \Omega$
- complement of an event $A^c = \Omega \setminus A$
- probability of any event is a non-negative value $P(A) \geq 0$
- total probability of all elementary outcomes is one

$$\sum_{\omega \in \Omega} P(\omega) = 1$$

- if two events $A, B$ are *mutually exclusive* (i.e. $A \cap B = \emptyset$), then
  $P(A \cup B) = P(A) + P(B)$
Basic formulas to calculate probabilities

Generally, probability of an event $A$ is

$$P(A) = \sum_{\omega \in A} P(\omega)$$

Probability of a complement event is

$$P(A^c) = 1 - P(A)$$
Calculating probability by relative frequency

IF all elementary outcomes have the same probability, THEN probability of an event is given by the proportion of

\[
\frac{\text{number of desired outcomes}}{\text{total number of outcomes possible}}
\]
What is $P(A \text{ or } B)$?

\[ P(A \text{ or } B) = P(A \cup B) \]

For \textit{mutually exclusive} events:

\[ P(A \text{ or } B) = P(A) + P(B) \]

otherwise (generally):

\[ P(A \text{ or } B) = P(A) + P(B) - P(A \cap B) \]
What is $P(A \text{ and } B)$?

$$P(A, B) = P(A \text{ and } B) = P(A \cap B)$$

If events $A$ and $B$ come from two different random processes, $P(A, B)$ is called joint probability.

Two events $A$ and $B$ are **independent** of each other if the occurrence of one has no influence on the probability of the other.

**For independent events:** $P(A \text{ and } B) = P(A) \cdot P(B)$.

otherwise (generally):

$$P(A \text{ and } B) = P(A | B) \cdot P(B) = P(B | A) \cdot P(A)$$
Warming exercises
If you want to make sure that you understand well basic probability computing

Rolling two dice, observing the sum. What is likelier?

a) the sum is even
b) the sum is greater than 8
c) the sum is 5 or 7

What is likelier:

a) rolling at least one six in four throws of a single die, OR
b) rolling at least one double six in 24 throws of a pair of dice?
Conditional probability of the event $A$ given the event $B$ is

$$P(A | B) = \frac{P(A \cap B)}{P(B)} = \frac{P(A, B)}{P(B)}$$

Or, in other words,

$$P(A, B) = P(A | B)P(B)$$
**Definition:** The random event $B$ is *independent* of the random event $A$, if the following holds true at the same time:

$$P(B) = P(B | A), \quad P(B) = P(B | A^c).$$

An equivalent definition is that $B$ is independent of $A$ if

$$P(A) \cdot P(B) = P(A \cap B).$$
Exercise

The probability that it is Friday and that a student is absent is 3%. Since there are 5 school days in a week, the probability that it is Friday is 20%. What is the probability that a student is absent given that today is Friday?
Random experiment:
At a random moment we observe the day in working week and the fact if a student is absent.

Events:
- \( A \) ... it is Friday
- \( B \) ... a student is absent

Probabilities:
- \( P(A, B) = 0.03 \)
- \( P(A) = 0.2 \)
- \( P(B | A) = \frac{P(A, B)}{P(A)} = 0.15 \)

Correct answer: The probability that a student is absent given that today is Friday is 15\%. 
Example – probability of target class

Look at the wsd.development data. There are 3524 examples in total. Each example can be considered as a random observation, i.e. as an outcome of a random experiment.

Occurrence of a particular value of the target class can be taken as an event, similarly for other attributes.

Assume that

- event $A$ stands for $\text{SENSE} = \text{‘PRODUCT’}$
- event $B$ stands for $A19 = \text{‘lines’}$

Then unconditioned probabilities $\Pr(A)$ and $\Pr(B)$ are

\[
\Pr(A) = \frac{\text{number of observations with } \text{SENSE}=\text{‘PRODUCT’}}{\text{number of all observations}} = \frac{1838}{3524} = 52.16\% 
\]

\[
\Pr(B) = \frac{\text{number of observations with } A19=\text{‘lines’}}{\text{number of all observations}} = \frac{1130}{3524} = 32.07\% 
\]
Example – conditional probability of target class

To compute conditional probability $\text{Pr}(A \mid B)$ you need to know joint probability $\text{Pr}(A, B)$

$$\text{Pr}(A, B) = \frac{\text{number of observations with SENSE=‘PRODUCT’ and A19=‘lines’}}{\text{number of all observations}}$$

$$\text{Pr}(A, B) = \frac{519}{3524} = 14.73\%$$

$$\text{Pr}(A \mid B) = \frac{\text{Pr}(A, B)}{\text{Pr}(B)} = \frac{14.73\%}{32.07\%} = 45.93\%$$

Or, equivalently

$$\text{Pr}(A \mid B) = \frac{\text{number of observations with SENSE=‘PRODUCT’ and A19=‘lines’}}{\text{number of observations with A19=‘lines’}}$$

$$\text{Pr}(A \mid B) = \frac{519}{1130} = 45.93\%$$
Bayes rule

Because of the symmetry $P(A, B) = P(B, A)$, we have

$$P(A, B) = P(A | B)P(B) = P(B | A)P(A) = P(B, A)$$

And thus

$$P(B | A) = \frac{P(A | B)P(B)}{P(A)}$$
Using Bayes rule

Exercise

One coin in a collection of 65 has two heads. The rest are fair.

If a coin, chosen at random from the lot and then tossed, turns up heads 6 times in a row, what is the probability that it is the two-headed coin?
Random experiment and considered events
We observe if a chosen coin is two-headed (event $A$), and if all 6 random tosses result in heads (event $B$). So, we want to know $P(A \mid B)$.

Probabilities

- $P(A \mid B)$ is the probability that we are looking for
  $= P(B \mid A)P(A)/P(B)$ (application of Bayes rule)
- $P(B \mid A) = 1$ (two-headed coin cannot give any other result)
- $P(A) = 1/65$; $P(A^c) = 64/65$
- $P(B) = P(B, A) + P(B, A^c)$ (two mutually exclusive events)
  $= P(A)P(B \mid A) + P(A^c)P(B \mid A^c)$ (by definition)
- $P(B \mid A^c) = 1/2^6 = 1/64$ (six independent events)
- $P(B) = 1/65 + (64/65)(1/64) = 2/65$
- $P(A \mid B) = (1/65)/(2/65) = 50\%$ (the correct answer)
Homework 2.1

1. **Practise using R!**
   Go thoroughly through all examples in our presentation and try it on your own
   - using your computer, your hands, and your brain :-)

2. **Study the Homework 1.1 Solution.**
   Understand it. Especially the conditional probability computing.
Machine learning process - five basic steps

1. Formulating the task
2. Getting classified data, i.e. training and test data
3. Learning from training data: Decision tree learning
4. Testing the learned knowledge on test data
5. Evaluation
Decision tree for the task of WDS of line

Example

yes $A_4 = \text{TRUE}$

no

PHONE

$A_2 = \text{TRUE}$

DIVISION

$A_3 = \text{TRUE}$

FORMATION

$A_9 = \text{TRUE}$

DIVISION PRODUCT
Using the decision tree for classification

Example

Assign the correct sense of *line* in the sentence "Draw a line between the points P and Q."

First, get twenty feature values from the sentence

<table>
<thead>
<tr>
<th>A₁</th>
<th>A₂</th>
<th>A₃</th>
<th>A₄</th>
<th>A₅</th>
<th>A₆</th>
<th>A₇</th>
<th>A₈</th>
<th>A₉</th>
<th>A₁₀</th>
<th>A₁₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A₁₂</th>
<th>A₁₃</th>
<th>A₁₄</th>
<th>A₁₅</th>
<th>A₁₆</th>
<th>A₁₇</th>
<th>A₁₈</th>
<th>A₁₉</th>
<th>A₂₀</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>draw</td>
<td>X</td>
<td>between</td>
<td>DT</td>
<td>IN</td>
<td>DT</td>
<td>line</td>
<td>dobj</td>
</tr>
</tbody>
</table>
Second, get the classification of the instance using the decision tree.
Using the decision tree for classification

Example

Assign the correct sense of *line* in the sentence "Draw a line that passes through the points P and Q."

First, get twenty feature values from the sentence

<table>
<thead>
<tr>
<th>A₁</th>
<th>A₂</th>
<th>A₃</th>
<th>A₄</th>
<th>A₅</th>
<th>A₆</th>
<th>A₇</th>
<th>A₈</th>
<th>A₉</th>
<th>A₁₀</th>
<th>A₁₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A₁₂</th>
<th>A₁₃</th>
<th>A₁₄</th>
<th>A₁₅</th>
<th>A₁₆</th>
<th>A₁₇</th>
<th>A₁₈</th>
<th>A₁₉</th>
<th>A₂₀</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>draw</td>
<td>X</td>
<td>that</td>
<td>DT</td>
<td>WDT</td>
<td>VB</td>
<td>line</td>
<td>dobj</td>
</tr>
</tbody>
</table>
Using the decision tree for classification

Second, get the classification of the instance using the decision tree.
Building a decision tree from training data

Tree structure description

- **Nodes**
  - Root node
  - Internal nodes
  - Leaf nodes with **TARGET CLASS VALUES**

- **Decisions**
  - Binary questions on a single feature, i.e. each internal node has two child nodes
Start building a decision tree

- **Step 1** Create a root node.

- **Step 2** Select decision \( d \) and add two child nodes to an existing node.
How to select decision $d$?

Associate the root node with the training set $t$.

Example

1. Assume decision if $A_4 = TRUE$.
2. Split the training set $t$ according to this decision into two subsets – "pink" and "blue".
3. Add two child nodes, "pink" and "blue", to the root. Associate each of them with the corresponding subset $t_L, t_R$, resp.

$$A_4 = \text{TRUE}$$
Building a decision tree from training data

How to select decision $d$?

Working with more than one feature, more than one decision can be formulated.

Which decision is the best?

Focus on a distribution of target class values in associated subsets of training examples.
Example

- Assume a set of 120 training examples from the task of WSD.
- Some decision splits them into two sets (1) and (2) with the following target class value distribution:

<table>
<thead>
<tr>
<th></th>
<th>CORD</th>
<th>DIVISION</th>
<th>FORMATION</th>
<th>PHONE</th>
<th>PRODUCT</th>
<th>TEXT</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>120</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(2)</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
</tbody>
</table>

A "pure" training subset contains mostly examples of a single target class value.
Which decision is the best?

Decision that splits training data into subsets as pure as possible.
Building a decision tree from training data

Decision tree learning algorithm – a very basic formulation

- **Step 1** Create a root node.

- **Step 2** Select decision $d$ and add two child nodes to an existing node.

- **Step 3** Split the training examples associated with the parent node $t$ according to $d$ into $t_L$ and $t_R$.

- **Step 4** Repeat recursively steps (2) and (3) for both child nodes and their associated training subsets.

- **Step 5** Stop recursion for a node if all associated training examples have the same target class value. Create a leaf node with this value.
Task
Assign the correct sense to the target word “line” (“lines”, “lined”)

Objects
Sentences containing the target word (“line”, “lines”, “lined”)

Target class
SENSE = \{CORD, DIVISION, FORMATION, PHONE, PRODUCT, TEXT\}

Features
Binary features $A_1, A_2, \ldots, A_{11}$
Subtasks

2. Build a classifier trained on eleven binary features A1, A2, ..., A11.
Getting classified data

First, get examples into R

```r
## Read the file with examples
> examples <- read.table("wsd.development.csv", header=T)

## Review the data
> str(examples)
'data.frame': 3524 obs. of 21 variables:
$ SENSE: Factor w/ 6 levels "cord","division",...: 1 1 1 1 1 ...
$ A1   : logi  TRUE FALSE FALSE FALSE FALSE FALSE ...
$ A2   : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
...
$ A8   : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
...
$ A12  : Factor w/ 920 levels ".",","","","-",...: 667 862 512 ...
$ A19  : Factor w/ 3 levels "line","lined",...: 3 1 3 3 1 ...
$ A20  : Factor w/ 80 levels "advcl","agent",...: 12 6 2 12 ...
```
Second, split them into the training and test sets

```r
## Get the number of input examples
> num.examples <- nrow(examples)

## Set the number of training examples = 90% of examples
> num.train <- round(0.9 * num.examples)

## Set the number of test examples = 10% of examples
> num.test <- num.examples - num.train

## Check the numbers
> num.examples
[1] 3524
> num.train
[1] 3172
> num.test
[1] 352
```
## Randomly split examples into training and test data

## Use set.seed() to be able to reconstruct the experiment

## with the SAME training and test sets

```r
> sample(10)
[1]  8  7 10  3  1  4  2  6  5  9
> sample(10)
[1]  9  8  5 10  7  6  3  2  4  1
> sample(10)
[1]  7  4  6 10  1  9  5  2  3  8
> sample(10)
[1]  9 10  4  5  1  6  8  2  3  7
> set.seed(123)
> sample(10)
[1]  3  8  4  7  6 10  9  2  5
> set.seed(123)
> sample(10)
[1]  3  8  4  7  6 10  9  2  5
```
Splitting classified data into training and test data

## Randomly split examples into training and test data

## Use set.seed() to be able to reconstruct the experiment
## with the SAME training and test sets

```r
> set.seed(123)
> s <- sample(num.examples)
```
### Get the training set

# First, generate indices of training examples ("blue" ones)
\[
> \text{indices.train} \leftarrow s[1: \text{num.train}]
\]

# Second, get the training examples
\[
> \text{train} \leftarrow \text{examples}[\text{indices.train},]
\]

### Get the test set (see "pink" indeces)
\[
> \text{indices.test} \leftarrow s[(\text{num.train}+1): \text{num.examples}]
> \text{test} \leftarrow \text{examples}[\text{indices.test},]
\]

## Check the results
\[
> \text{str(train)}; \text{str(test)}
\]
Load the package `rpart`

```r
## Use the "rpart" package
## ! Run install.packages("rpart"), ***if not installed***.

# Check if the package is installed
> library()

## Load the package
> library(rpart)

# to get help info
> help(rpart)
```
Subtask 1 Build a dependency tree classifier using only one feature, namely A4

Train decision tree model M1

```r
## Run the learning process using function "rpart"
M1 <- rpart(SENSE ~ A4, data=train, method="class")
> 
```
rpart documentation

rpart(formula, data= , method= , ... )

• formula is $y \sim \text{model}$ where
  • $y$ is a target class
  • $\sim$ stands for 'is modeled as'
  • model is a combination of features (model by statisticians).

• data specifies the training set,

• method="class" for classification,
Display the trained tree

```r
## Draw tree on screen
> plot(M1); text(M1)

## Draw tree to a file
> png("../img/dtM1.png", width=4.8, height=4.8, units="in",
     res=600, bg="transparent")
> plot(M1, margin=0.05)
> text(M1)
> title(main = "Decision tree trained on feature A4")
> dev.off()
```
WSD task: decision tree trained on feature A4

```
+------------------+
| A4 ≥ 0.5         |
|                  |
| phone            |
|                  |
| product          |
```

Trained decision tree
Trained decision tree

Display the model M1

```r
## Display the model
> M1
n= 3172

node), split, n, loss, yval, (yprob)
   * denotes terminal node

1) root 3172 1526 product (0.096 0.093 0.084 0.11 0.52 0.099)
   2) A4>=0.5 150  8 phone (0 0 0 0.95 0 0.053) *
   3) A4< 0.5 3022 1376 product (0.1 0.097 0.089 0.068 0.54 0.1) *
```
Trained decision tree

How to read the model

\(n = 3172\)

node), split, n, loss, yval, (yprob)

* denotes terminal node

\n\n\begin{align*}
n &= 3172 \\
node) &= \text{A node number.} \\
split &= \text{Decision.} \\
n &= \text{The number of training examples associated to the given node.} \\
loss &= \text{The number of examples incorrectly classified with the majority class value yval.} \\
yval &= \text{The default classification for the node by the majority class value.} \\
yprob &= \text{The distribution of class values at the associated training subset.}
\end{align*}
### Test the trained model M1 on test examples

## Use the function predict()

```r
> ?predict()
predict package:stats R Documentation
```

Model Predictions

Description:

> predict’ is a generic function for predictions
> the results of various model ...

```r
> P11 <- predict(M1, test, type="class")
```
Comparing the predicted values with the true senses

\[
\begin{align*}
\text{P11} & \quad \text{cord} & \text{division} & \text{formation} & \text{phone} & \text{product} & \text{text} \\
\text{cord} & 0 & 0 & 0 & 0 & 33 & 0 \\
\text{division} & 0 & 0 & 0 & 0 & 28 & 0 \\
\text{formation} & 0 & 0 & 0 & 0 & 28 & 0 \\
\text{phone} & 0 & 0 & 0 & 12 & 21 & 0 \\
\text{product} & 0 & 0 & 0 & 0 & 192 & 0 \\
\text{text} & 0 & 0 & 0 & 1 & 37 & 0 
\end{align*}
\]

57.95 % of test examples are predicted correctly

\[
\text{round}(100*\text{sum(P11 == test$SENSE)}/\text{num.test},2)
\]

\[1\] 57.95
Prediction on training data

```r
### Test the trained model M1 on training examples.

> P12 <- predict(M1, train, type="class")
> 
```
Comparing the predicted values with the true senses

```r
> print(table(train$SENSE, P12))

P12

<table>
<thead>
<tr>
<th></th>
<th>cord</th>
<th>division</th>
<th>formation</th>
<th>phone</th>
<th>product</th>
<th>text</th>
</tr>
</thead>
<tbody>
<tr>
<td>cord</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>303</td>
<td>0</td>
</tr>
<tr>
<td>division</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>294</td>
<td>0</td>
</tr>
<tr>
<td>formation</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>268</td>
<td>0</td>
</tr>
<tr>
<td>phone</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>142</td>
<td>205</td>
<td>0</td>
</tr>
<tr>
<td>product</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1646</td>
<td>0</td>
</tr>
<tr>
<td>text</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>306</td>
<td>0</td>
</tr>
</tbody>
</table>
```

56.37% of training examples are predicted correctly

```r
> message(round(100*sum(P12 == train$SENSE)/num.train, 2), "\%")

[1] 56.37
```
Subtask 2 Build a dependency tree classifier using all binary features, namely A1, ..., A11

Train decision tree model M2

```r
## Run the learning process using function "rpart"
M2 <- rpart(SENSE ~ A1+A2+A3+A4+A5+A6+A7+A8+A9+A10+A11,
             data=train, method="class")
> 
```
Display the trained tree

```r
## Draw tree on screen
> plot(M2); text(M2)

## Draw tree to a file
> png("../img/dtM2.png", width=4.8, height=4.8, units="in",
     res=600, bg="transparent")
> plot(M2, margin=0.05)
> text(M2)
> title(main = "Decision tree trained on all binary features")
> dev.off()
```
Trained decision tree

WSD task: decision tree trained on A1,...,A11

```
    A4 >= 0.5
     /  \
    /    /
phone  A2 >= 0.5
     /  \
  /    /
division A3 >= 0.5
     /  \
   /    /
 formation A9 >= 0.5
     /  \
  /    /
division product
```
Trained decision tree

Display the trained model M2

```r
> ## Display the model
> M2
n= 3172

node), split, n, loss, yval, (yprob)
  * denotes terminal node

1) root 3172 1526 product (0.096 0.093 0.084 0.11 0.52 0.099)
  2) A4>=0.5 150 8 phone (0 0 0 0.95 0 0.053) *
  3) A4< 0.5 3022 1376 product (0.1 0.097 0.089 0.068 0.54 0.1)
    6) A2>=0.5 88 0 division (0 1 0 0 0 0) *
    7) A2< 0.5 2934 1288 product (0.1 0.07 0.091 0.07 0.56 0.1)
      14) A3>=0.5 79 5 formation (0.063 0 0.94 0 0 0) *
      15) A3< 0.5 2855 1209 product (0.1 0.072 0.068 ...)
        30) A9>=0.5 66 3 division (0.015 0.95 0 ...) *
        31) A9< 0.5 2789 1144 product (0.11 0.051 0.07 ...) *
```
### Test the trained model on test examples.

> P21 <- predict(M2, test, type="class")
### Evaluation

#### Comparing the predicted values with the true senses

```r
> print(table(test$SENSE, P21))

          P21
         cord division formation phone product text
cord       0       0       0   0      33     0
division   0      15       0   0      13     0
formation  0       0       6   0      22     0
phone      0       0      12   0      21     0
product    0       1       0   0     191     0
text       0       0       1   1      36     0
```

63.64 % of test examples are predicted correctly

```r
> round(100*sum(P21 == test$SENSE)/num.test,2)
[1] 63.64
```
Testing trained decision tree on training data

Prediction on training data

```r
### Test the trained model on training examples.

> P22 <- predict(M2, train, type="class")
```
**Evaluation**

**Comparing the predicted values with the true senses**

```r
> print(table(train$SENSE, P22))

     P22
     cord division formation phone product text
    cord  0      1     5    0   297     0
   division  0   151    0    0   143     0
 formation  0    0    74    0   194     0
    phone  0     1     0   142  204     0
   product  0     1     0    0  1645    0
     text  0     0     0    8  306     0
```

**63.43 % of training examples are predicted correctly**

```r
> round(100*sum(P22 == train$SENSE)/num.train,2)
[1] 63.64
```
Run the script in R

The R script `DT-WSD.R`

- builds the classifier M1 using the feature A4, classifies training and test data using M1 and computes the performance of M1.
- builds the classifier M2 using binary features $A_1, ..., A_{11}$, classifies training and test data using M2 and computes the performance of M2.

Download the script from the course page and run in R

```r
> source("DT-WSD.R")
> 
```
Generate the same training and test sets as we did in practice above. Assume the following feature groups:

1. $A_2, A_3, A_4, A_9$
2. $A_1, A_6, A_7$
3. $A_1, A_{11}$

For each of them, build a decision tree classifier and list its percentage of correctly classified training and test examples.
Theory

- Decision tree structure: nodes, decisions
- A basic formulation of decision tree learning algorithm
We built two decision tree classifiers (M1, M2) on two different sets of features and we tested them on both training and test sets.

<table>
<thead>
<tr>
<th>features used</th>
<th>trained model</th>
<th>data set</th>
<th>performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_4$</td>
<td>M1</td>
<td>train</td>
<td>57.37</td>
</tr>
<tr>
<td></td>
<td></td>
<td>test</td>
<td>57.95</td>
</tr>
<tr>
<td>$A_1, \ldots, A_{11}$</td>
<td>M2</td>
<td>train</td>
<td>63.43</td>
</tr>
<tr>
<td></td>
<td></td>
<td>test</td>
<td>63.64</td>
</tr>
</tbody>
</table>

!!! You know how to build a decision tree classifier from training examples in R. Performance is not important right now. !!!