

Introduction to Machine Learning

NPFL 054

<http://ufal.mff.cuni.cz/course/npfl054>

Barbora Hladká
hladka@ufal.mff.cuni.cz

Martin Holub
holub@ufal.mff.cuni.cz

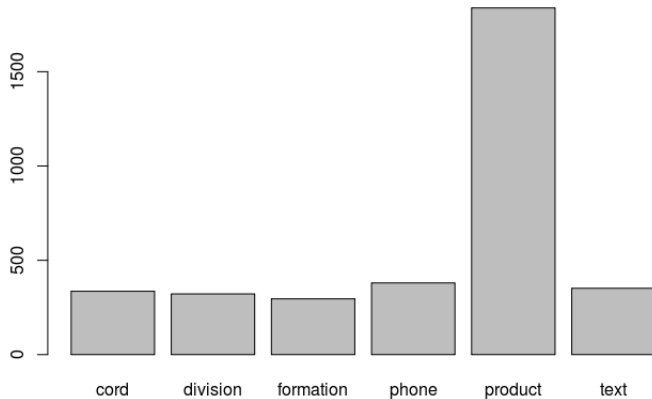
Charles University,
Faculty of Mathematics and Physics,
Institute of Formal and Applied Linguistics

Outline

- **Brief recap of the last lesson**
- **Entropy and conditional entropy**
 - definition, calculation, and meaning
 - application for feature selection
- **Decision Trees**
 - building Decision Trees and using them as prediction function

WSD task — distribution of target class values

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



Amount of information contained in a value?

How much information do you gain when you observe a random event?

According to the **Information Theory**, **amount of information** contained in an event is given by

$$I = \log_2 \frac{1}{p} = -\log_2 p$$

where p is probability of the event occurred.

Thus, the lower probability, the more information you get when you observe an event (e.g. a feature value). If an event is certain ($p = 100\%$), then the amount of information is zero.

Amount of information in SENSE values

```
### probability distribution of SENSE
> round(table(examples$SENSE)/nrow(examples), 3)

      cord  division formation      phone  product      text
0.095    0.091    0.084    0.108    0.522    0.100
>

### amount of information contained in SENSE values
> round(-log2(table(examples$SENSE)/nrow(examples)), 3)

      cord  division formation      phone  product      text
3.391    3.452    3.574    3.213    0.939    3.324
>
```

What is the average amount of information that you get when you observe values of the attribute SENSE?

Entropy

The average amount of information that you get when you observe random values is

$$\sum_{\text{value}} \Pr(\text{value}) \cdot \log_2 \frac{1}{\Pr(\text{value})} = - \sum_{\text{value}} \Pr(\text{value}) \cdot \log_2 \Pr(\text{value})$$

This is what information theory calls *entropy*.

- Entropy of a random variable X is denoted by $H(X)$
 - or, $H(p_1, p_2, \dots, p_n)$ where $\sum_{i=1}^n p_i = 1$
- Entropy is a measure of the uncertainty in a random variable
 - or, measure of the uncertainty in a probability distribution
- The unit of entropy is bit; entropy says how many bits *on average* you necessarily need to encode a value of the given random variable

Properties of entropy

Normality

$$H\left(\frac{1}{2}, \frac{1}{2}\right) = 1$$

Continuity

$H(p, 1 - p)$ is a continuous function

Non negativity and maximality

$$0 \leq H(p_1, p_2, \dots, p_n) \leq H\left(\frac{1}{n}, \frac{1}{n}, \dots, \frac{1}{n}\right)$$

Symmetry

$H(p_1, p_2, \dots, p_n)$ is a symmetric function of its arguments

Recursivity

$$H(p_1, p_2, p_3, \dots, p_n) = H(p_1 + p_2, p_3, \dots, p_n) + (p_1 + p_2)H\left(\frac{p_1}{p_1 + p_2}, \frac{p_2}{p_1 + p_2}\right)$$

Entropy of SENSE

Entropy of SENSE is 2.107129 bits.

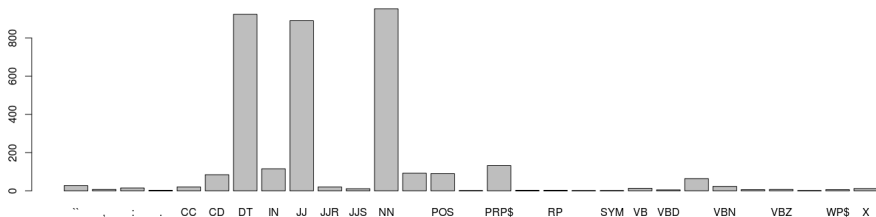
```
### probability distribution of SENSE
> p.sense <- table(examples$SENSE)/nrow(examples)
>
### entropy of SENSE
> H.sense <- - sum( p.sense * log2(p.sense) )
> H.sense
[1] 2.107129
```

The maximum entropy value would be $\log_2(6) = 2.584963$ if and only if the distribution of the 6 senses was uniform.

```
> p.uniform <- rep(1/6, 6)
> p.uniform
[1] 0.1666667 0.1666667 0.1666667 0.1666667 0.1666667 0.1666667
>
### entropy of uniformly distributed 6 senses
> - sum( p.uniform * log2(p.uniform) )
[1] 2.584963
```

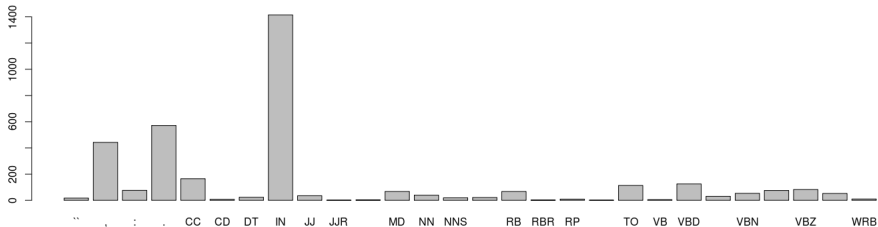

Distribution of feature values – A16

```
> levels(examples$A16)
[1] "``"    ",,"    ":",    ". ."    "CC"    "CD"    "DT"    "IN"    "JJ"
[10] "JJR"   "JJS"   "NN"    "NNS"   "POS"   "PRP"   "PRP$"  "RB"    "RP"
[19] "-RRB-" "SYM"   "VB"    "VBD"   "VBG"   "VBN"   "VBP"   "VBZ"   "WDT"
[28] "WP$"   "X"
> plot(examples$A16)
>
```



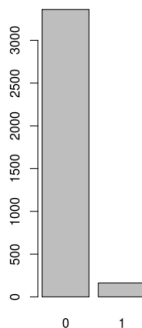
Distribution of feature values – A17

```
> levels(examples$A17)
[1] "``"      ", "      ":"      ". "      "CC"      "CD"      "DT"      "IN"      "JJ"
[10] "JJR"     "-LRB-"   "MD"      "NN"      "NNS"     "PRP"     "RB"      "RBR"     "RP"
[19] "-RRB-"   "TO"      "VB"      "VBD"     "VBG"     "VBN"     "VBP"     "VBZ"     "WDT"
[28] "WRB"
> plot(examples$A17)
>
```



Distribution of feature values – A4

```
> levels(examples$A4)
[1] "0" "1"
>
```



Entropy of features

Entropy of A16 is 2.78 bits.

```
> p.A16 <- table(examples$A16)/nrow(examples)
> H.A16 <- - sum( p.A16 * log2(p.A16) )
> H.A16
[1] 2.777606
```

Entropy of A17 is 3.09 bits.

```
> p.A17 <- table(examples$A17)/nrow(examples)
> H.A17 <- - sum( p.A17 * log2(p.A17) )
> H.A17
[1] 3.093003
```

Entropy of A4 is 0.27 bits.

```
> p.A4 <- table(examples$A4)/nrow(examples)
> H.A4 <- - sum( p.A4 * log2(p.A4) )
> H.A4
[1] 0.270267
```

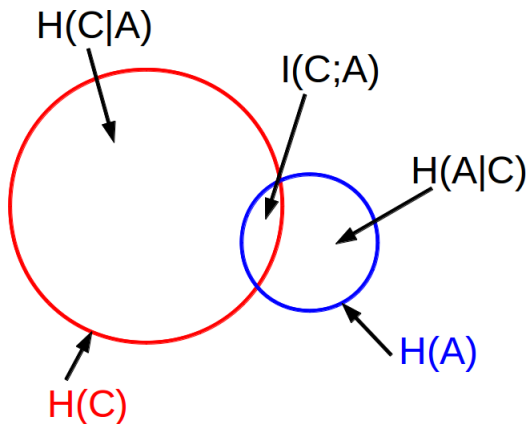
Conditional entropy $H(C | A)$

How much does target class entropy decrease if we have the knowledge of a feature?

The answer is **conditional entropy**:

$$H(C | A) = - \sum_{y \in C, x \in A} \Pr(y, x) \cdot \log_2 \Pr(y | x)$$

Conditional entropy and mutual information



WARNING

There are NO SETS in this picture! Entropy is a quantity, only a number!

Conditional entropy and mutual information

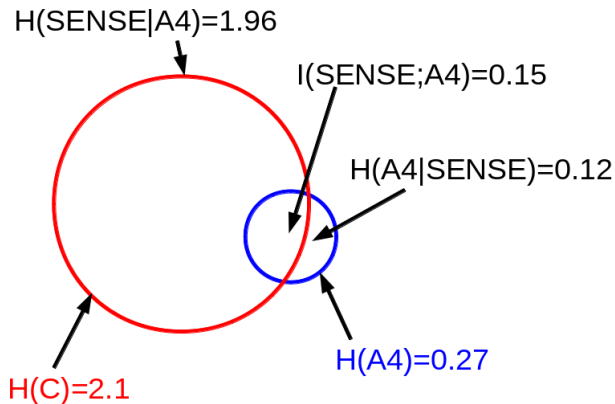
Mutual information measures the amount of information that can be obtained about one random variable by observing another.

Mutual information is a symmetrical quantity.

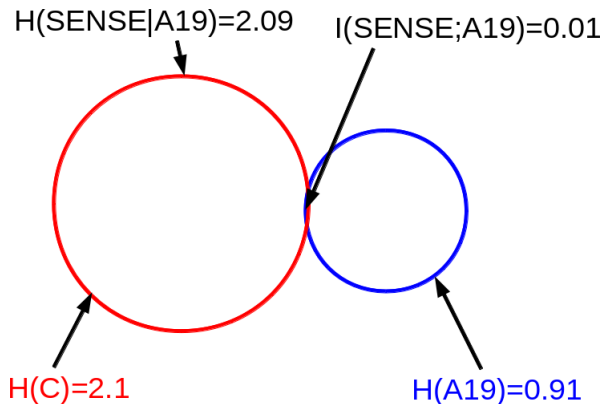
$$H(C) - H(C|A) = I(C;A) = H(A) - H(A|C)$$

Another name for mutual information is **information gain**.

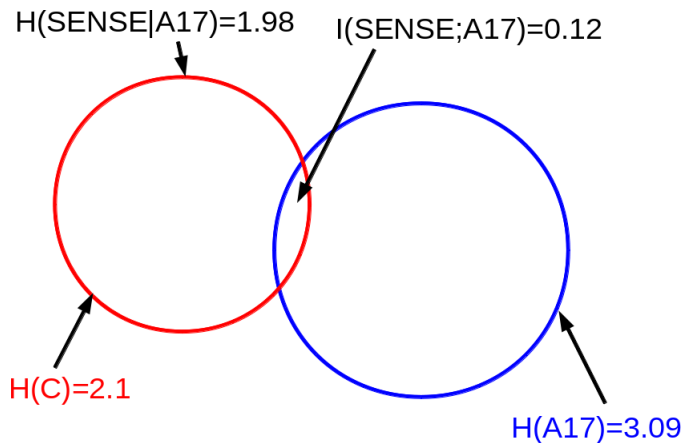
Conditional entropy – feature A4



Conditional entropy – feature A19



Conditional entropy – feature A17



User-defined functions in R

Structure of a user-defined function

```
myfunction <- function(arg1, arg2, ... ){  
  ... statements ...  
  return(object)  
}
```

Objects in a function are local to the function.

Example – a function to calculate entropy

```
> entropy <- function(x){  
+   p <- table(x) / NROW(x)  
+   return( -sum(p * log2(p)) )  
+ }  
>  
  
# invoking the function  
> entropy(examples$SENSE)  
[1] 2.107129
```

Summary

- **Information theory provides a measure** for comparing how the knowledge of features *statistically* contribute to the knowledge about target class.
- The lower conditional entropy $H(C | A)$, the better chance that A is a useful feature.
- However, since features typically interact, conditional entropy $H(C | A)$ should NOT be the only criterion when you do feature selection. You need experiments to see if a feature with high information gain really helps.

Note

Also, decision tree learning algorithm makes use of entropy when it computes purity of training subsets.

Homework

- Write your own function for computing conditional entropy in R. New function `entropy.cond(x,y)` will take two factors of the same length and will compute $H(x|y)$.

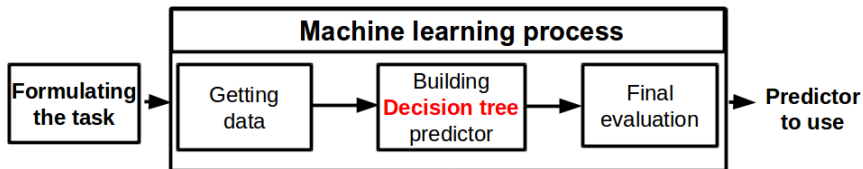
Example use: `entropy.cond(examples$SENSE, examples$A4)`

You should understand and be able to explain and practically use

- entropy
 - motivation
 - definition
 - main properties
 - calculation in R
- conditional entropy
 - definition and meaning
 - relation to mutual information
 - calculation in R
 - information gain – application in feature selection

Decision Tree — a learning method

Decision Tree is a learning method suitable for both classification and regression tasks



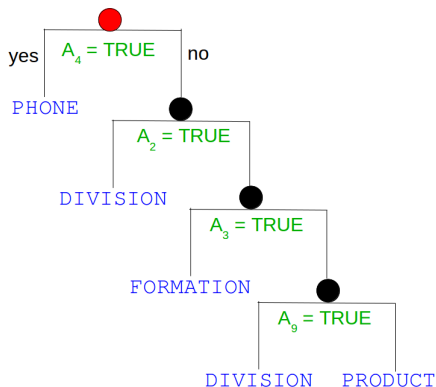
Example classification task: WSD

see the NPFL054 web page → Materials → wsd-attributes.pdf

Decision tree structure

A **decision tree** $T = (V, E)$ is a rooted tree where V is composed of internal **decision nodes** and terminal **leaf nodes**.

- Nodes
 - Root node
 - Internal nodes
 - Leaf nodes with **TARGET OUTPUT VALUES**
- Decisions

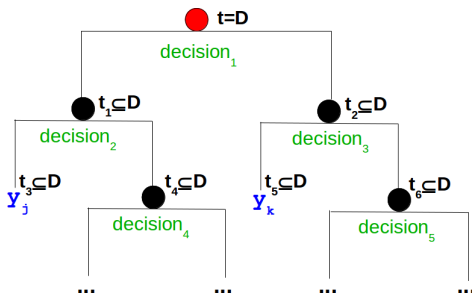


Decision tree learning from training data

Decision tree learning corresponds to building a decision tree $T_D = (V, E)$ based on a training data set $D = \{\langle \mathbf{x}, y \rangle : \mathbf{x} \in X, y \in Y\}$.

When building a tree, each node is associated with a set t , $t \subseteq D$. The root node is associated with $t = D$.

Each leaf node is designated by an output value.



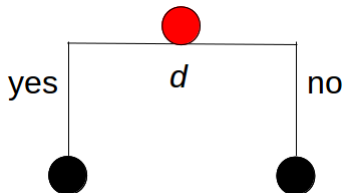
Building a decision tree from training data

A very basic idea: Assume binary decisions

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



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



Building a decision tree from training data

Example

Associate the root node with the training set t .

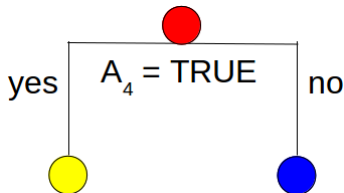
Example

1. Assume decision
if $A_4 = \text{TRUE}$.
2. Split the training set t according
to this decision into two subsets
– "yellow" and "blue".

	SENSE	...	A4	...
t	FORMATION		TRUE	
	FORMATION		FALSE	
	PHONE		TRUE	
	CORD		TRUE	
	DIVISION		FALSE	

Building a decision tree from training data

3. Add two child nodes, "yellow" and "blue", to the root. Associate each of them with the corresponding subset t_L , t_R , resp.



t_L

SENSE	...	A4	...
FORMATION		TRUE	
CORD		TRUE	
PHONE		TRUE	
...	

t_R

SENSE	...	A4	...
FORMATION		FALSE	
DIVISION		FALSE	
...	

Building a decision tree from training data

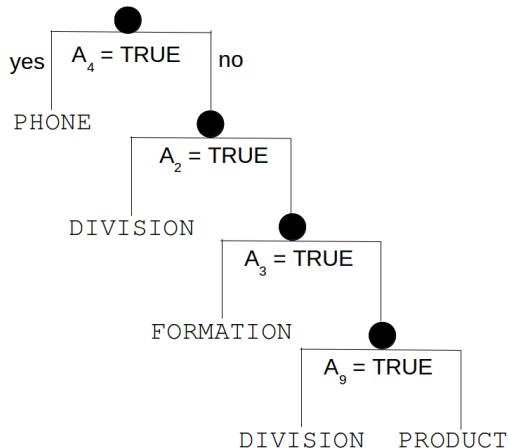
- **Step 4** Repeat recursively steps (2) and (3) for both child nodes and their associated training subsets.
- **Step 5** Stop recursion for a node if a stopping criterion is fulfilled. Create a leaf node with an output value.

Prediction on test data

Once the decision tree predictor is built, an unseen instance is predicted by starting at the root node and moving down the tree branch corresponding to the feature values asked in decisions.

Prediction on test data

Decision tree predictor for the WSD-*line* task



Prediction on test data

Decision tree predictor for the WSD-*line* task

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

True prediction: DIVISION

Prediction on test data

Decision tree predictor for the WSD-*line* task

First, get twenty feature values from the sentence

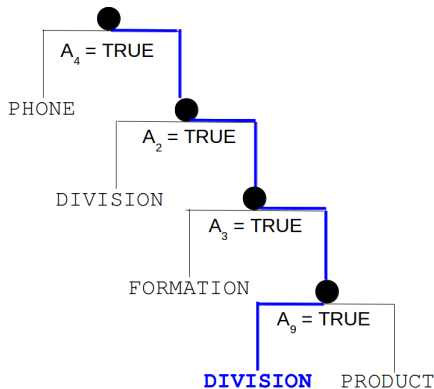
A ₁	A ₂	A ₃	A ₄	A ₅	A ₆	A ₇	A ₈	A ₉	A ₁₀	A ₁₁
0	0	0	0	0	0	0	0	1	0	0

A ₁₂	A ₁₃	A ₁₄	A ₁₅	A ₁₆	A ₁₇	A ₁₈	A ₁₉	A ₂₀
a	draw	X	between	DT	IN	DT	line	dobj

Prediction on test data

Decision tree predictor for the WSD-*line* task

Second, get the classification of the instance using the decision tree



Decision tree predictor for the WSD-*line* task

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

True prediction: DIVISION

Prediction on test data

Decision tree predictor for the WSD-*line* task

First, get twenty feature values from the sentence

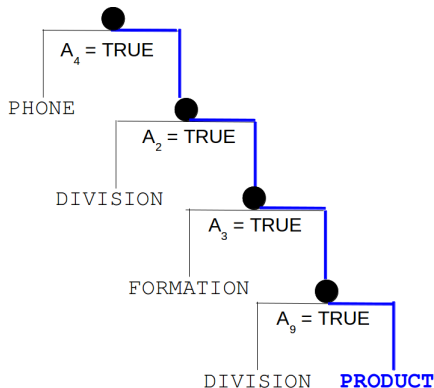
A ₁	A ₂	A ₃	A ₄	A ₅	A ₆	A ₇	A ₈	A ₉	A ₁₀	A ₁₁
0	0	0	0	0	0	0	0	0	0	0

A ₁₂	A ₁₃	A ₁₄	A ₁₅	A ₁₆	A ₁₇	A ₁₈	A ₁₉	A ₂₀
a	draw	X	that	DT	WDT	VB	line	dobj

Prediction on test data

Decision tree predictor for the WSD-*line* task

Second, get the classification of the instance using the decision tree



Decision trees

Classification trees

- Y is a categorical output feature

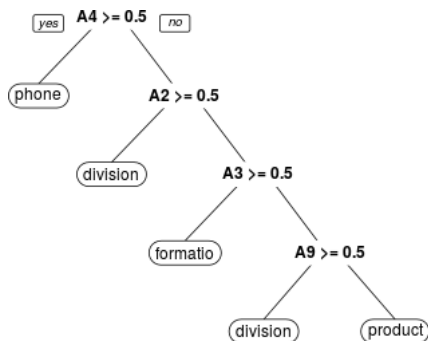


Figure: Tree for predicting the sense of *line* based on binary features.

Regression trees

- Y is a numerical output feature

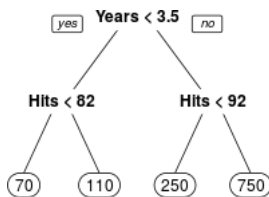


Figure: Tree for predicting the salary of a baseball player based on the number of years that he has played in the major leagues (Year) and the number of hits that he made in the previous year (Hits). See the ISLR Hitters data set.

Historical excursion

Decision trees concept
(Hunt, 1962)



ID3 (Quinlan, 1979)



C4.5 (Quinlan, 1993)

AID (Morgan, 1964)



CART (Breiman, 1984)

- ID3 ~ Iterative Dichotomiser
- AID ~ Automatic Interaction Detection
- CART ~ Classification and Regression Trees

Probably most well-known is the “C5.0” algorithm (Quinlan), which has become the industry standard.

Packages in R: `rpart`

Building a decision tree from training data

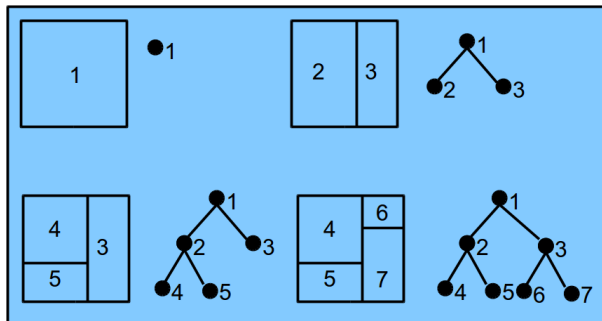
- ① Tree growing
- ② Tree pruning

Basic idea: First, grow a large tree that fits the training data. Second, prune this tree to avoid overfitting.

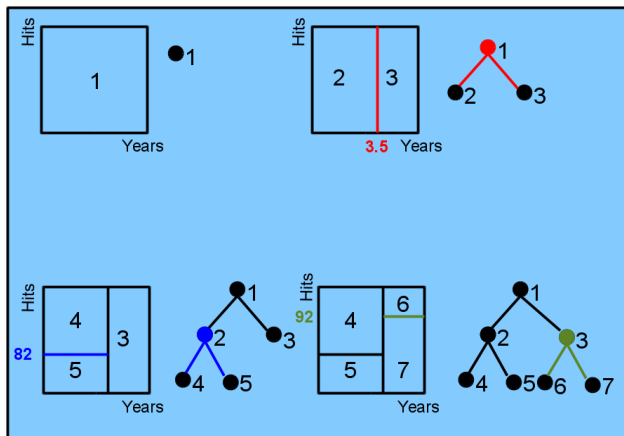
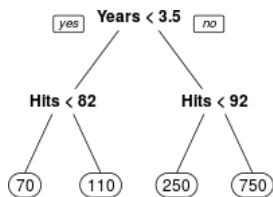
Building a decision tree from training data

- 1 Tree growing
- 2 Tree pruning

The growing process is based on subdividing the feature space recursively into non-overlapping regions.



Building a decision tree from training data



Classification and Regression trees

Each terminal node in the decision tree is associated with one of the regions in the feature space. Then

Classification trees

- **output value:** the most common class in the data associated with the terminal node

Regression trees

- **output value:** the mean output value of the training instances associated with the terminal node

Building a CLASIFICATION tree from training data

Notation

- $Attr = \{A_1, A_2, \dots, A_m\}$,
- $Y = \{y_1, y_2, \dots, y_k\}$
- $Values(A_i)$ is a set of all possible values for feature A_i .
- $D_{i,v} = \{\langle \mathbf{x}, y \rangle \in D \mid x_i = v\}$.

...	...	A_i	...
...	...	v	...
...
...
...	...	v	...
...	...	v	...
...

Building a classification tree from training data

We work with decisions on the value of only a single feature

- For each categorical feature A_j having values $Values(A_j) = \{b_1, b_2, \dots, b_L\}$

is $x_j = b_i?$ as $i = 1, \dots, L$

- For each categorical feature A_j

is $x_j \in$ a subset $\in 2^{Values(A_j)}$?

- For each numerical feature A_j

is $x_j \leq k?$, $k \in (-\infty, +\infty)$

Which decision is the best?

- Focus on a distribution of target class values in associated subsets of training examples.
- Then select the decision that splits training data into subsets as pure as possible.

Building a classification tree from training data

Which decision is the best?

We say a data set is **pure** (or **homogenous**) if it contains only a single class. If a data set contains several classes, then the data set is **impure** (or **heterogenous**).

$\oplus: 5, \ominus: 5$		$\oplus: 9, \ominus: 1$
heterogenous high degree of impurity		almost homogenous low degree of impurity

Which decision is the best?

1. **Define** a candidate set S of splits at each node using possible decisions. $s \in S$ splits t into L subsets t_1, t_2, \dots, t_L .
2. **Define** the node proportions $p(y_j|t), j = 1, \dots, k$, to be the proportion of instances $\langle \mathbf{x}, y_j \rangle$ in t .
3. **Define** an **impurity measure** $i(t)$, i.e. **splitting criterion**, as a nonnegative function Φ of the $p(y_1|t), p(y_2|t), \dots, p(y_k|t)$,

$$i(t) = \Phi(p(y_1|t), p(y_2|t), \dots, p(y_k|t)), \quad (1)$$

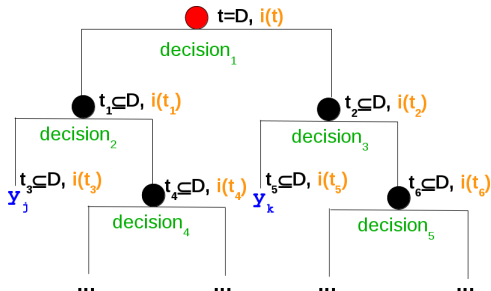
such that

- $\Phi(\frac{1}{k}, \frac{1}{k}, \dots, \frac{1}{k}) = \max$, i.e. the node impurity is largest when all examples are equally mixed together in it.
- $\Phi(1, 0, \dots, 0) = 0, \Phi(0, 1, \dots, 0) = 0, \dots, \Phi(0, 0, \dots, 1) = 0$, i.e. the node impurity is smallest when the node contains instances of only one class

Building a classification tree from training data

Which decision is the best?

4. Define the **goodness of split s** to be the decrease in impurity $\Delta i(s, t) = i(t) - \sum_{l=1}^L p_l * i(t_l)$, where p_l is the proportion of instances in t that go to t_l .
5. Find split s^* with the largest decrease in impurity: $\Delta i(s^*, t) = \max_{s \in S} \Delta i(s, t)$.
6. Use splitting criterion $i(t)$ to compute $\Delta i(s, t)$ and get s^* .



Which decision is the best?

Splitting criteria – examples that are really used

- Misclassification Error – $i(t)_{ME}$
- Information Gain – $i(t)_{IG}$
- Gini Index – $i(t)_{GI}$

Building a classification tree from training data

Which decision is the best?

Splitting criteria

$$i(t)_{ME} = 1 - \max_{j=1,\dots,k} p(y_j|t) \quad (2)$$

	$\oplus: 0, \ominus: 6$	$\oplus: 1, \ominus: 5$	$\oplus: 2, \ominus: 4$	$\oplus: 3, \ominus: 3$
ME	$1 - \frac{6}{6} = 0$	$1 - \frac{5}{6} = 0.17$	$1 - \frac{4}{6} = 0.33$	$1 - \frac{3}{6} = 0.5$

Building a classification tree from training data

Which decision is the best?

Splitting criteria

$$i(t)_{IG} = - \sum_{j=1}^k p(y_j|t) * \log p(y_j|t). \quad (3)$$

Recall the notion of entropy $H(t)$, $i(t)_{IG} = H(t)$.

$$Gain(s, t) = \Delta i(s, t)_{IG} \quad (4)$$

Which decision is the best?

Splitting criteria

$$i(t)_{GI} = 1 - \sum_{j=1}^k p^2(y_j|t) = \sum_{j=1}^k p(y_j|t)(1 - p(y_j|t)). \quad (5)$$

Building a classification tree from training data

Which decision is the best?

Splitting criteria

	\oplus : 0 \ominus : 6	\oplus : 1 \ominus : 5	\oplus : 2 \oplus : 4	\oplus : 3 \oplus : 3
Gini	0	0.278	0.444	0.5
Entropy	0	0.65	0.92	1.0
ME	0	0.17	0.333	0.5

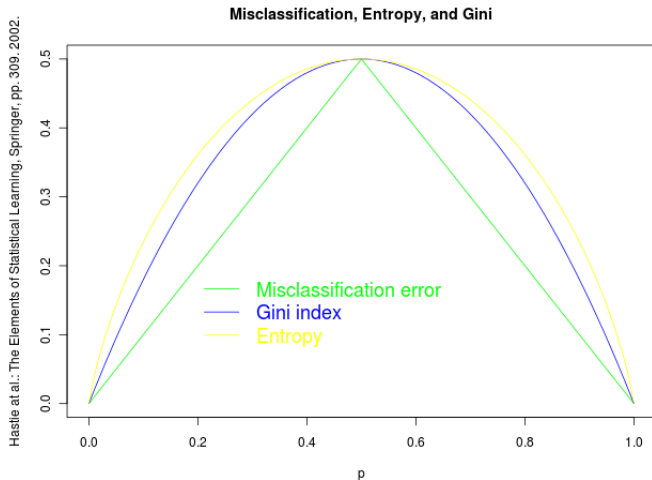
For two classes ($k = 2$), if p is the proportion of the class "1", the measures are:

- Misclassification error: $1 - \max(p, 1 - p)$
- Entropy: $-p * \log p - (1 - p) * \log(1 - p)$
- Gini: $2p * (1 - p)$

Building a classification tree from training data

Which decision is the best?

Splitting criteria



Classification and Regression trees

Each terminal node in the decision tree is associated with one of the regions in the feature space. Then

Classification trees

- Output value: the most common class in the data associated with the terminal node
- A criterion for making splits, e.g.
 - Misclassification error
 - Information gain
 - Gini index

Regression trees

- Output value: the mean output value of the training instances associated with the terminal node

Notation

- $Attr = \{A_1, A_2, \dots, A_m\}$
- $Y = \mathcal{R}$
- $Values(A_i)$ is a set of all possible values for feature A_i

Building a regression tree from training data

Again, we work with decisions on the value of only a single feature

Which decision is the best?

Splitting criterion – usually used

- Squared Error – $i(t)_{SE}$

$$i(t)_{SE} = \frac{1}{|t|} \sum_{\mathbf{x}_i \in t} (y_i - y^t)^2,$$

where $y^t = \frac{1}{|t|} \sum_{\mathbf{x}_i \in t} y_i$.

Classification and Regression trees

Each terminal node in the decision tree is associated with one of the regions in the feature space. Then

Classification trees

- Output value: the most common class in the data associated with the terminal node
- A criterion for making splits, e.g.
 - Misclassification error
 - Information gain
 - Gini index

Regression trees

- Output value: the mean output value of the training instances associated with the terminal node
- A criterion for making splits, e.g. Squared error

Building decision tree from training data

The recursive binary splitting is stopped when a stopping criterion is fulfilled. Then a leaf node is created with an output value.

Stopping criteria, e.g.

- the leaf node is associated with less than five training instances
- the maximum tree depth has been reached
- the best splitting criteria is not greater than a certain threshold

As a splitting criterion, ID3 algorithm uses information gain.

Main idea

- Calculate the entropy of every attribute using the data set S
- Split the set S into subsets using the attribute for which entropy is minimum (or, equivalently, information gain is maximum)
- Make a decision tree node containing that attribute
- Recurse on subsets using remaining attributes

ID3 algorithm is nicely described on the Wikiedia:

— https://en.wikipedia.org/wiki/ID3_algorithm

ID3 → C4.5

ID3 is originally designed with two restrictions:

- 1 classification task
- 2 categorical features used to train a decision tree → Let's extend ID3 for the continuous-valued features

C4.5 algorithm: Incorporating continuous-valued features

For a continuous-valued feature A , define a boolean-valued feature A_c so that if $A(\mathbf{x}) \leq c$ then $A_c(\mathbf{x}) = 1$ else $A_c(\mathbf{x}) = 0$.

C4.5 algorithm: Handling training examples with missing feature values

Consider the situation in which $Gain(t, A)$ is to be calculated at node associated with a training data set t in the decision tree. Suppose that $\langle \mathbf{x}, y \rangle$ is one of the training examples in t and that the value $A(\mathbf{x})$ is unknown.

Possible solutions

- Assign the value that is most common among training instances associated with the node.
- Alternatively, assign the most common value among instances associated with the node t having the classification y .

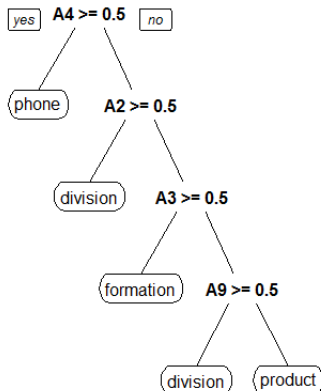
Building a decision tree from training data

- ① Tree growing ✓
- ② Tree pruning

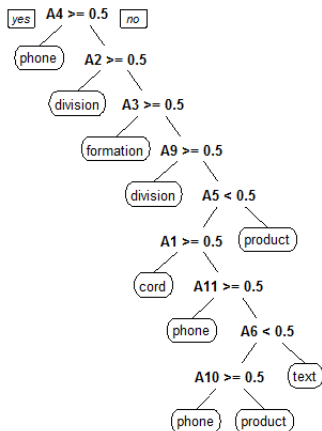
Basic idea: First, grow a large tree that fits the training data. Second, prune this tree to avoid overfitting.

Models built with different cp values

cp=0.04



cp=0.004



Building a decision tree from training data

Overfitting can be avoided by

- applying a stopping criterion that prevents some sets of training instances from being subdivided,
- removing some of the structure of the decision tree after it has been produced.

Preferred strategy

Grow a large tree T_0 , stop the splitting process when only some minimum node size (say 5) is reached. Then prune T_0 using some pruning criteria.

Decision trees — implementation in R

There are two widely used packages in R

- `rpart`
- `tree`

The algorithms used are very similar.

References

- An Introduction to Recursive Partitioning Using the RPART Routines by Terry M. Therneau, Elizabeth J. Atkinson, and Mayo Foundation (available online)
- *An Introduction to Statistical Learning with Application in R* Chapters 8.1, 8.3.1, and 8.3.2 by Gareth James, Daniela Witten, Trevor Hastie and Rob Tibshirani (available online)
- R packages documentation — `rpart`, `tree` (available online)

Decision Trees – weak spots

- **data splitting**
 - deeper nodes can learn only from small data portions
- **sensitivity to training data set (unstable algorithm)**
 - learning algorithm is called unstable if small changes in the training set cause large differences in generated models

References

- Breiman Leo, Friedman Jerome H., Olshen Richard A., Stone Charles J. *Classification and Regression Trees*. Chapman & Hall/CRC, 1984.
- Hunt, E. B. *Concept Learning: An Information Processing Problem*, Wiley. 1962.
- Morgan, J. N., Sonquist, J. A. Problems in the analysis of survey data, and a proposal. *Journal of the American Statistical Association* 58, pp. 415–434. 1963.
- Quinlan, J. R. Discovering rules from large collections of examples: A case study, in D. Michie, ed., *Expert Systems in the Micro Electronic Age*. Edinburgh University Press. 1979.
- Quinlan, J. R. *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Mateo, California. 1993.

Decision trees in R – `rpart()` implementation

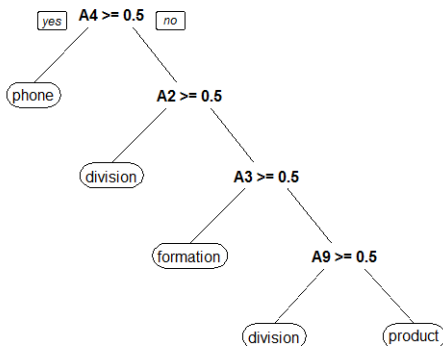
- library **rpart** (but there are also other libraries **tree**, **party**, ...)

```
Model=rpart(formula, data=, method=, control=)
```

- `?rpart`
- formula in the format `TargetClass~Feature1+Feature2+...`
- data specifies the input data frame
- method is "class" for decision trees
- control other optional parameters

Visualisation using `rpart.plot()`

Visualisation of the model with library `rpart.plot`



Decision Trees – parameters

hypothesis parameters - parameters of the prediction function

- output of the learning algorithm, define the structure of the decision tree

learning parameters - parameters of the learning process

- "configuration" of the learning algorithm

Decision trees learning parameters

2 phases of decision tree learning:

- growing
- pruning

Learning parameters are used to control these two phases:

- when to stop growing
- how much to prune the tree

... to avoid overfitting and improve performance

`rpart.control`

minsplit

- the minimum number of observations that must exist in a node in order for a split to be attempted

cp

- complexity parameter, influences the depth of the tree

... and others, see `?rpart.control`

T: try to set different `cp` and `minsplit` values in the M1 model learning and observe the resulting tree

Any split that does not decrease the **relative training error** by a factor of cp is not attempted

⇒ That means, the learning algorithm measures for each split how it improves the tree relative error and if the improvement is too small, the split will not be performed.

Relative error is the error relative to the misclassification error (without any splitting relative error is 100%)

cp parameter

```
> M <- rpart(SENSE ~ A1+A2+A3+A4+A5+A6+A7+A8+A9+A10+A11, data=train,
             method="class", minsplit=5, cp=0.001)
> M$cptable
      CP nsplit rel error   xerror   xstd
1 0.093053735     0 1.0000000 1.0000000 0.01844043
2 0.057667104     1 0.9069463 0.9069463 0.01830335
3 0.048492792     2 0.8492792 0.8591088 0.01817412
4 0.040629096     3 0.8007864 0.8106160 0.01800131
5 0.009174312     4 0.7601573 0.7601573 0.01777550
6 0.003931848     9 0.7064220 0.7070773 0.01748535
7 0.001000000    10 0.7024902 0.7044561 0.01746957
```

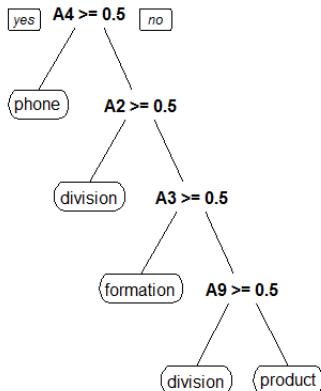
rel error relative error on training data

xerror relative error in x-fold **cross-validation**

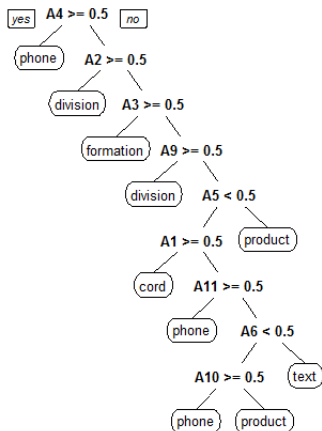
xstd standard deviation of **xerror** on x validation folds

Models built with different cp value

cp=0.04



cp=0.004



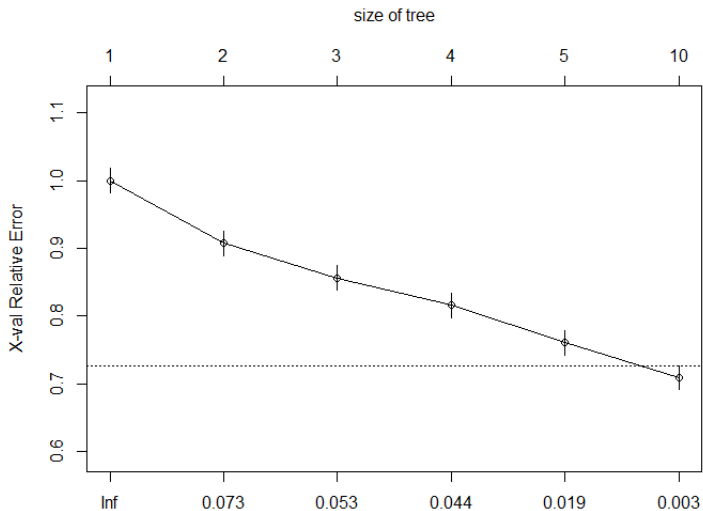
Useful functions

`plotcp(model)` visualisation of the cross-validation error depending on `cp` value

`prune(model, cp=)` prune the model based on `cp` value

```
> M5$cptable[which.min(M5$cptable[, "xerror"]), "CP"]  
[1] 0.001
```

- visualisation of the cross-validation error depending on cp value
- the horizontal line shows the minimal $xerror$ + its standard deviation



How to choose the optimal cp value?

demo code cp-and-pruning.Forbes.R on course page

```
> m = rpart(profits ~ category + sales + assets + marketvalue,
            data=F[data.train, 1:8], cp=0.001)
> m$cptable
```

	CP	nsplit	rel error	xerror	xstd
1	0.543259557	0	1.0000000	1.0482897	0.03178559
2	0.027162978	1	0.4567404	0.4607646	0.02673551
3	0.007042254	3	0.4024145	0.4446680	0.02640028
4	0.006036217	6	0.3762575	0.4507042	0.02652763
5	0.005030181	8	0.3641851	0.4567404	0.02665301
6	0.004024145	15	0.3279678	0.4768612	0.02705703
7	0.003018109	19	0.3118712	0.4688129	0.02689795
8	0.002012072	21	0.3058350	0.4869215	0.02725122
9	0.001006036	23	0.3018109	0.5171026	0.02780383
10	0.001000000	25	0.2997988	0.5412475	0.02821490

How to choose the optimal cp value?

