Classification and regression trees

Pierre Geurts & Louis Wehenkel

Institut Montefiore, University of Liège, Belgium

ELEN062-1
Introduction to Machine Learning
September 2022

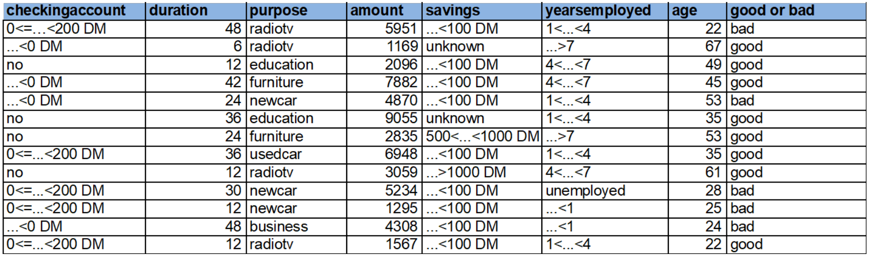
Acknowledgment: These slides have been reformatted by Yann Claes in July 2020.

Outline

Outline

Database/Dataset/Sample

A database/dataset/sample is a **collection of objects/observations** (rows) described by **attributes/features/variables** (columns).



checkingaccount	duration	purpose	amount	savings	yearsemployed	age	good or bad
0<=…<200 DM	48	radiotv	5951	…<100 DM	1<…<4	22	bad
…<0 DM	6	radiotv	1169	unknown	…>7	67	good
no	12	education	2096	…<100 DM	4<…<7	49	good
…<0 DM	42	furniture	7882	…<100 DM	4<…<7	45	good
…<0 DM	24	newcar	4870	…<100 DM	1<…<4	53	bad
no	36	education	9055	unknown	1<…<4	35	good
no	24	furniture	2835	500<…<1000 DM	…>7	53	good
0<=…<200 DM	36	usedcar	6948	…<100 DM	1<…<4	35	good
no	12	radiotv	3059	…>1000 DM	4<…<7	61	good
0<=…<200 DM	30	newcar	5234	…<100 DM	unemployed	28	bad
0<=…<200 DM	12	newcar	1295	…<100 DM	…<1	25	bad
…<0 DM	48	business	4308	…<100 DM	…<1	24	bad
0<=…<200 DM	12	radiotv	1567	…<100 DM	1<…<4	22	good

Supervised learning

Inputs				Output
A1	A2	...	A1000	Y
-0.86	0.17	...	0	C2
-2.3	-1.2	...	-0.42	C1
-0.37	-0.11	...	-0.64	C1
0.41	0.67	...	-0.8	C2
-0.51	-0.59	...	0.98	C2
-0.25	-0.27	...	-0.68	C1
-0.52	0.23	...	0.11	C1
-1.3	-0.2	...	0.14	C1
0.93	-0.78	...	-0.01	C2
-0.25	-0.29	...	0.69	C2
-0.6	0.92	...	-0.64	C1
0.22	-0.8	...	-0.5	C2
-0.62	0.2	...	0.08	C1
-0.3	0.8	...	0.02	C2
-0.91	0.44	...	-0.57	C1
0.76	0.65	...	-0.08	C1

$$\xrightarrow[\text{Automatic learning}]{} \quad \hat{Y} = f\left(A_1, \ldots, A_{1000}\right)$$

Goal: from the database, find a function $f(\cdot)$ of the inputs that approximates at best the output.

2 cases:

- **Discrete** output $\rightarrow$ **classification** problem
- **Continuous** output $\rightarrow$ **regression** problem

Application (i)

- Predict whether or not a bank client will be a good or a bad debtor.

- Image classification:
  - Face recognition
  - Handwritten characters recognition

$\longrightarrow$ 3

$\longrightarrow$ 5

Application (ii)

- Classification of cancer types from gene expression profiles

No. patient	Gene 1	Gene 2	...	Gene 7129	Leucemia
1	-134	28	...	123	AML
2	-123	0	...	17	AML
3	56	-123	...	-23	ALL
...	...	...	...	...	...
72	89	-123	...	12	ALL

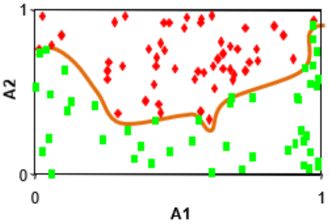
Source: [1]

Learning algorithm

A learning algorithm receives as input a **learning sample** and returns a **function** $h(\cdot)$.

It is defined by:

- A hypothesis space H, which is a set of candidate models.
- A quality measure for a model.
- An optimisation strategy.



A model $h(\cdot) \in H$ obtained by automatic learning.

Outline

Classification trees (aka Decision trees)

A supervised learning algorithm that can handle:

- **Classification problems**, that can be binary or multi-valued.
- **Discrete** (binary or multi-valued) or **continuous** attributes.

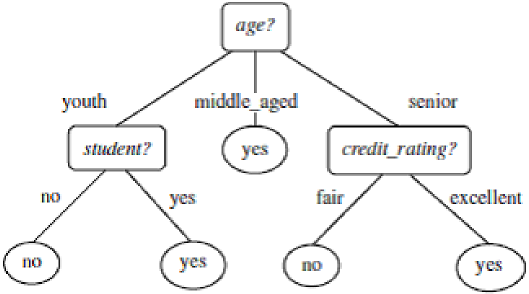
Classification trees were invented several times:

- By statisticians: e.g. CART (Breiman et al.)
- By the AI community: e.g. ID3, C4.5 (Quinlan et al.)

Hypothesis space

A decision tree is a tree where:

- Each interior node tests an attribute
- Each branch corresponds to an attribute value
- Each leaf is labelled with a class



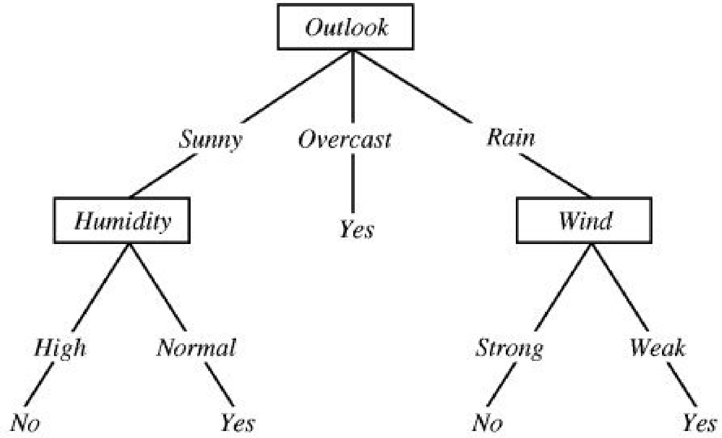
A decision tree allowing one to predict whether a customer will buy a given kind of computer. Source: [2]

Illustration - Should I play tennis? (i)

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	Normal	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	High	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Hot	Normal	Weak	Yes
D10	Rain	Mild	Normal	Strong	Yes
D11	Sunny	Cool	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Adapted from [3]. (NB: first column is 'dummy'; output $Y \equiv$ 'Play Tennis')

Illustration - Should I play tennis? (ii)

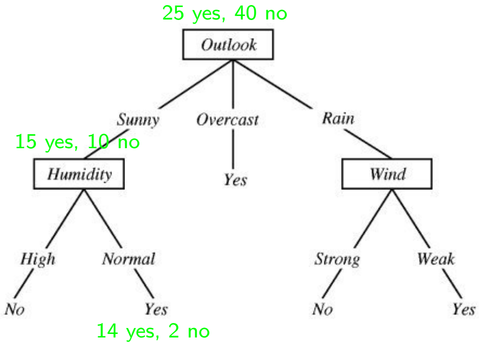


Source: [3]

Tree learning

Tree learning problem consists in choosing the **tree structure** and determining **the predictions** at leaf nodes.

For each leaf, **the prediction (or label)** is chosen such that the misclassification error in the part of the LS reaching that leaf is **minimized**: the **majority class** in the part of the LS reaching the leaf.



How to generate trees? (i)

What properties do we want a decision tree to have?

- It should be consistent with the learning sample (for the moment):
  - Trivial algorithm: construct a decision tree that has one path to a leaf for each example.
    Problem: it does not capture useful information from the database.

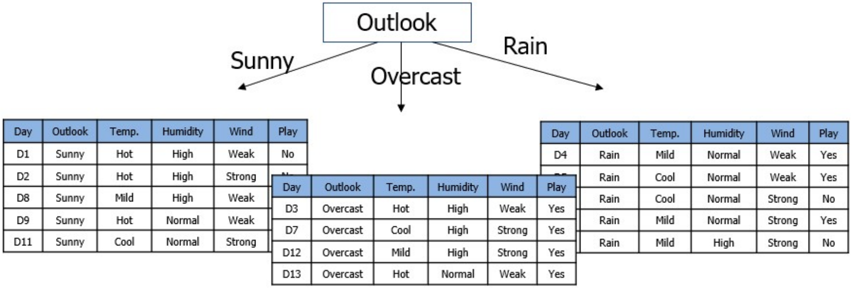
How to generate trees (ii)

What properties do we want decision trees to have?

- It should be at the same time as simple as possible.
  - Trivial algorithm: generate all trees and pick the simplest one that is consistent with the learning sample.
    Problem: there are too many trees.

Top-down induction of decision trees (i)

Idea: Choose the best attribute, split the learning sample accordingly and proceed recursively until each object is correctly classified.



Day	Outlook	Temp.	Humidity	Wind	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D8	Sunny	Mild	High	Weak	
D9	Sunny	Hot	Normal	Weak	
D11	Sunny	Cool	Normal	Strong	

Day	Outlook	Temp.	Humidity	Wind	Play
D3	Overcast	Hot	High	Weak	Yes
D7	Overcast	Cool	High	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes

Day	Outlook	Temp.	Humidity	Wind	Play
D4	Rain	Mild	Normal	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
	Rain	Cool	Normal	Strong	No
	Rain	Mild	Normal	Strong	Yes
	Rain	Mild	High	Strong	No

Top-down induction of decision trees (ii)

```
Algorithm 1: learn_dt($LS$)
if all objects from $LS$ have the same class or if all objects have the
  same values for every attribute then
    Create a leaf with a label corresponding to the majority class of
      the objects of $LS$;
end if
else
    Use $LS$ to find the best splitting attribute $A^*$ ;
    Create a test node for that attribute ;
    forall different values $a$ of $A^*$ do
        Build $LS_a = \{o \in LS \mid A^*(o) \text{ is } a\}$ ;
        Use learn_dt($LS_a$) to grow a subtree from $LS_a$
    end forall
end if
```

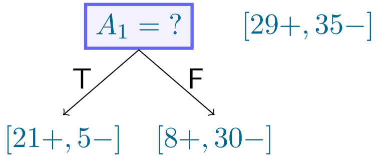
Properties of the top-down approach

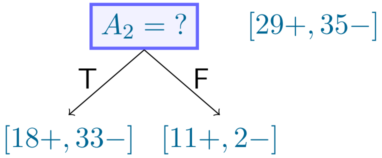
- Hill-climbing algorithm in the space of possible decision trees:
  - It adds a sub-tree to the current tree and continues its search
  - It does not backtrack

- Highly dependent upon the criterion for selecting attributes to test (what we called above the “best splitting attribute A^*”)

- **Sub-optimal (heuristic)** but very fast

Which attribute is the best splitting attribute?





We want a small tree. Therefore, we should **maximize** the class separation at each step, *i.e* make successors as **pure** as possible.
$\Rightarrow$ it will favour short paths in trees.

Impurity measures

Let:

- LS be a sample of objects
- p_j ($\forall j = 1, \ldots, J$) the proportion of objects in the LS belonging to output-class j.

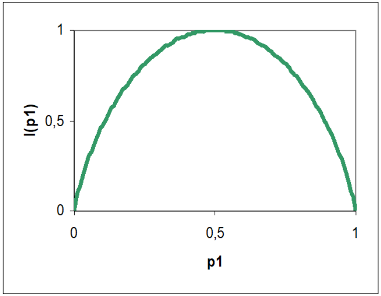
An **impurity** measure $I(LS)$ should satisfy the following props:

- $I(LS)$ is minimum only when $\exists i$ s.t. $p_i = 1$ and $p_j = 0$ for $j \neq i$ (pure sample);
- $I(LS)$ is maximum only when $\forall j : p_j = \frac{1}{J}$ (uniform number of objects among classes);
- $I(LS)$ is a symmetric function of its arguments $p_1, \ldots, p_J$.

Shannon entropy as an impurity measure (i)

Definition of Shannon entropy:

$$H(LS) \triangleq -\sum_{j=1}^{J} p_j \log_2 p_j$$
$$\triangleq I_{\mathsf{Sh}}(LS)$$



If there are only two classes we have (since $p_2 = 1 - p_1$)

$$H(LS) = -p_1 \log_2 p_1 - (1 - p_1) \log_2 (1 - p_1).$$

Notice that I_{Sh} satisfies the above props (see graphic).

(NB: Shannon entropy is the basis of information theory.)

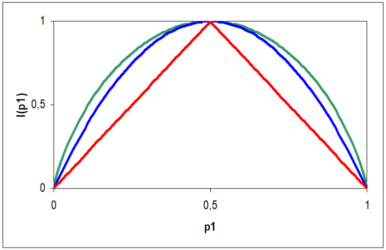
Other examples of impurity measures (ii)

- Gini index:

$$I_{\mathsf{Gi}}(LS) \triangleq \sum_{j=1}^{J} p_j(1-p_j).$$

- (Misclassification) Error rate:

$$I_{\mathsf{ER}}(LS) \triangleq 1 - \max_j p_j.$$



Two-class case. Respectively, the Shannon entropy, the Gini index and the Error rate, normalized between 0 and 1.

Reduction of impurity achieved by a split

For a given impurity measure, the best splitting attribute is the one which **maximizes** the expected **reduction** of impurity defined by

$$\Delta I(LS, A) = I(LS) - \sum_{a \in A(LS)} \frac{|LS_a|}{|LS|} I(LS_a),$$

where LS_a is the subset of objects o from LS such that $A(o) = a$, and where $A(LS)$ is the set of different values of A observed in LS.

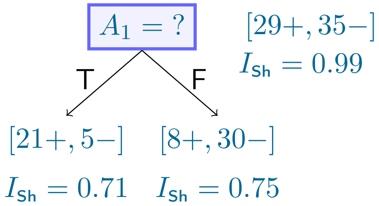
ΔI is also called a **score measure** or a splitting criterion.

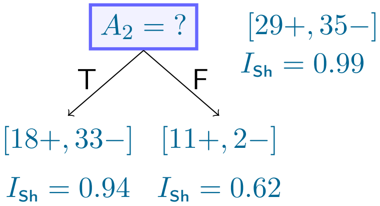
NB: There are other ways to define a splitting criterion that do not rely on an impurity measure.

NB: The reduction of Shannon entropy is called the **information gain**.

Illustration (with Shannon entropy as impurity measure)

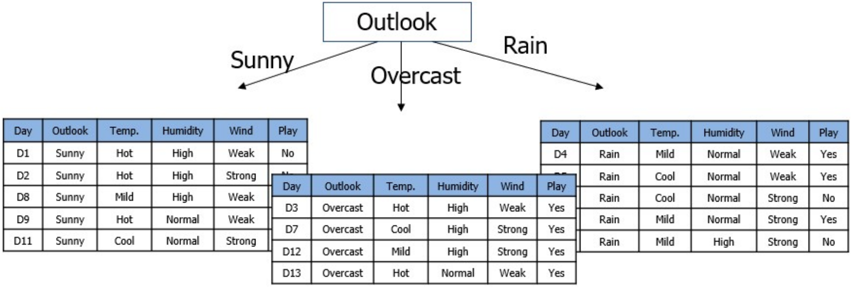
Which attribute is the best one to split?





$$\rightarrow \Delta I_{\mathsf{Sh}}(LS, A_1) = 0.99 - \frac{26}{64} \times 0.71 - \frac{38}{64} \times 0.75 = 0.25$$
$$\rightarrow \Delta I_{\mathsf{Sh}}(LS, A_2) = 0.99 - \frac{51}{64} \times 0.94 - \frac{13}{64} \times 0.62 = 0.12$$

Application to the tennis problem



Day	Outlook	Temp.	Humidity	Wind	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D8	Sunny	Mild	High	Weak	
D9	Sunny	Hot	Normal	Weak	
D11	Sunny	Cool	Normal	Strong	

Day	Outlook	Temp.	Humidity	Wind	Play
D3	Overcast	Hot	High	Weak	Yes
D7	Overcast	Cool	High	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes

Day	Outlook	Temp.	Humidity	Wind	Play
D4	Rain	Mild	Normal	Weak	Yes
	Rain	Cool	Normal	Weak	Yes
	Rain	Cool	Normal	Strong	No
	Rain	Mild	Normal	Strong	Yes
	Rain	Mild	High	Strong	No

Which attribute should be tested here?

- $\Delta I_{\mathsf{Sh}}(LS, \text{Temp.}) = 0.970 - \frac{3}{5} \times 0.918 - \frac{1}{5} \times 0.0 - \frac{1}{5} \times 0.0 = 0.419$
- $\Delta I_{\mathsf{Sh}}(LS, \text{Hum.}) = 0.970 - \frac{3}{5} \times 0.0 - \frac{2}{5} \times 0.0 = 0.970$
- $\Delta I_{\mathsf{Sh}}(LS, \text{Wind}) = 0.970 - \frac{2}{5} \times 1.0 - \frac{3}{5} \times 0.918 = 0.019$

The best attribute is thus *humidity*.

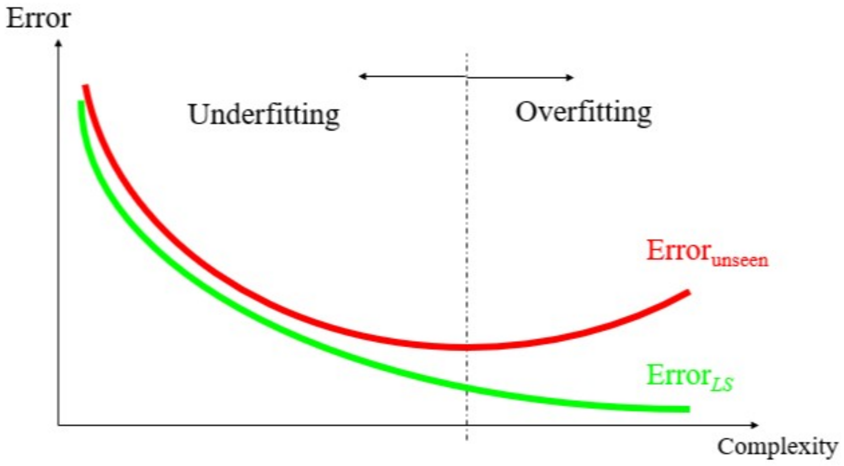
Overfitting (i)

For now, trees are perfectly consistent with the learning sample.
However, often, we would like them to be good at predicting classes of unseen data from the same distribution, which is called **generalization**.

A tree T overfits the learning sample if and only if $\exists T'$ such that:

1. $\mathrm{Error}_{LS}(T) < \mathrm{Error}_{LS}(T')$
2. $\mathrm{Error}_{unseen}(T) > \mathrm{Error}_{unseen}(T')$

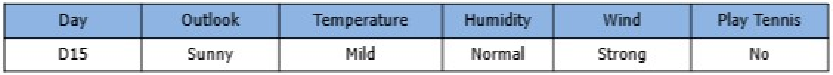
Overfitting (ii)



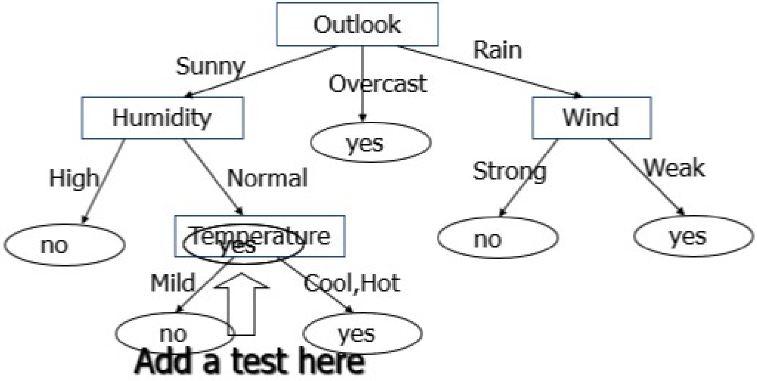
In practice, $\text{Error}_{unseen}(T)$ is estimated from a separate test sample.

Why do trees overfit the learning sample? (i)

Data is noisy or attributes do not completely predict the outcome.

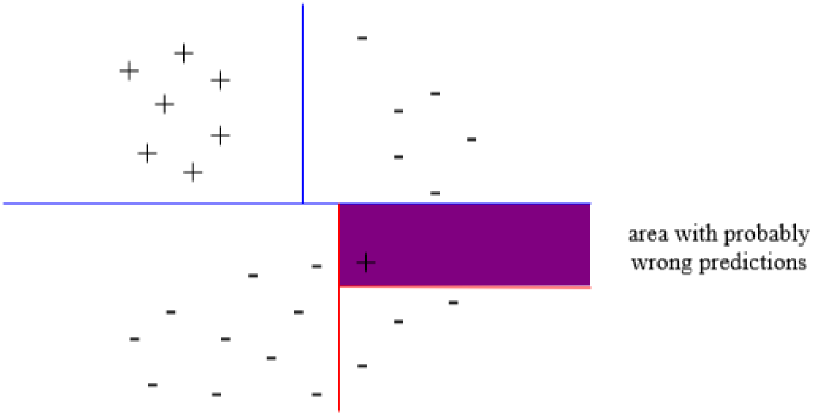


Day	Outlook	Temperature	Humidity	Wind	Play Tennis
D15	Sunny	Mild	Normal	Strong	No



Reasons for overfitting (ii)

Data is incomplete, *i.e.* all cases are not covered.



We do not have enough data in some part of the learning sample to make a good decision.

How to avoid overfitting?

- Pre-pruning: stop growing the tree earlier, before it reaches the point where it perfectly classifies the learning sample.
- Post-pruning: allow the tree to overfit, then post-prune it.
- Ensemble methods: these will be covered later in the course.

Pre-pruning

Idea: stop splitting a node if either:

a. the local sample size is $< N_{\min}$

b. the local sample impurity $< I_{\text{th}}$

c. the impurity reduction ΔI of the best test is not large enough, according to some statistical hypothesis test at level α.

Caveats:

- for criteria [a,b,c] suitable values of the meta-parameters, $N_{\min}$, I_{th}, α, are often problem dependent;
- criterion [c] may recommend stop-splitting too early.

Post-pruning (i)

Idea: split the learning sample into two sets:

- a growing sample GS to build the tree
- a validation sample VS to evaluate its generalization error

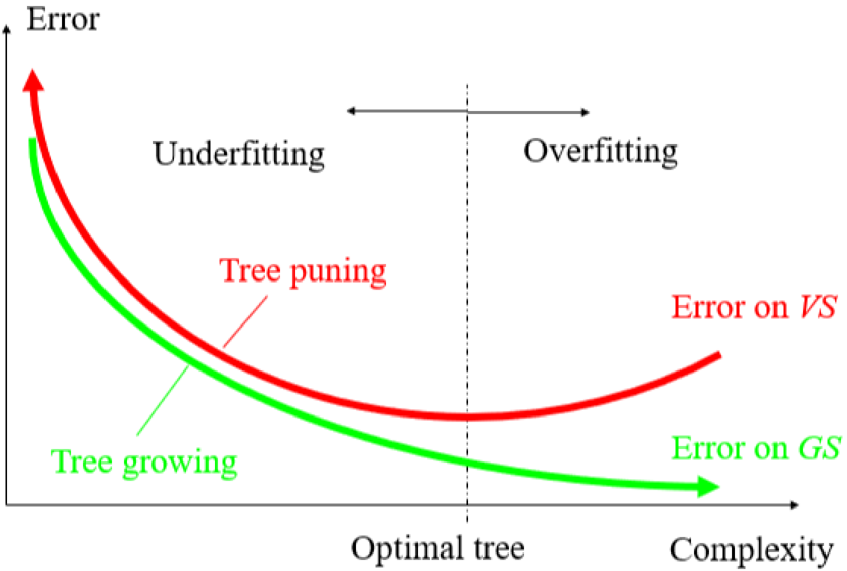
Then, build a complete tree from GS.

Next, compute a sequence of trees $\{T_1, T_2, \ldots\}$ where:

- T_1 is the complete tree
- T_i is obtained by removing some test nodes from T_{i-1}

Finally, select the tree T_i^* from the sequence that **minimizes** the error on the validation sample VS.

Post-pruning (ii)



Post-pruning (iii)

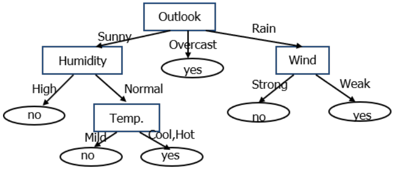
How to build the sequence of trees?

- Reduced error pruning: at each step, remove the node that most decreases the error on the validation sample.
- Cost-complexity pruning: define a cost-complexity criterion

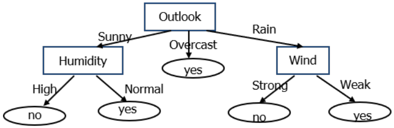
$$\mathrm{Error}_{GS}(T) + \beta \,\mathrm{Complexity}(T)$$

and build the sequence of trees that minimizes this criterion, for increasing values of β.

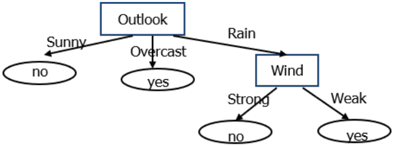
Post-pruning (iv)



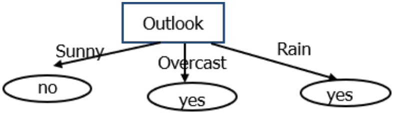
$\text{Error}_{GS} = 0\%, \text{Error}_{VS} = 10\%$



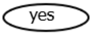
$\text{Error}_{GS} = 6\%, \text{Error}_{VS} = 8\%$



$\text{Error}_{GS} = 13\%, \text{Error}_{VS} = 15\%$



$\text{Error}_{GS} = 27\%, \text{Error}_{VS} = 25\%$



$\text{Error}_{GS} = 33\%, \text{Error}_{VS} = 35\%$

Post-pruning (v)

Problem: it requires to dedicate a part of the learning sample as a validation set, which may be a problem in the case of a small database.

$\Rightarrow$ **Solution:** K-fold cross-validation

- Split the training set into K parts, often 10
- Generate K trees, each one leaving out one part among K
- Make a prediction for each learning object with the only tree built without this case
- Estimate the error of this prediction

K-fold cross-validation may be combined with pruning.

How to use decision trees?

- Large data sets (ideal case):
  - Split the data set into three parts: GS, VS, TS
  - Grow a tree from GS
  - Post-prune it from VS
  - Test it on TS

- Small data sets (often):
  - Grow a tree from the whole database
  - Pre-prune it with default parameters (risky)/post-prune it by 10-fold cross-validation (costly)
  - Estimate its accuracy by 10-fold cross-validation

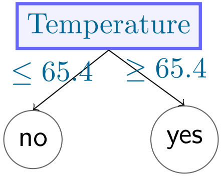
Outline

Continuous attributes (i)

Example: temperatures as a number instead of a discrete value.

There are two solutions:

- **Pre-discretize:** *cold* if the temperature is below 70, *mild* if between 70 and 75, *hot* if above 75.
- **Discretize during tree growing**:



How to find the cut-point?

Continuous attributes (ii)

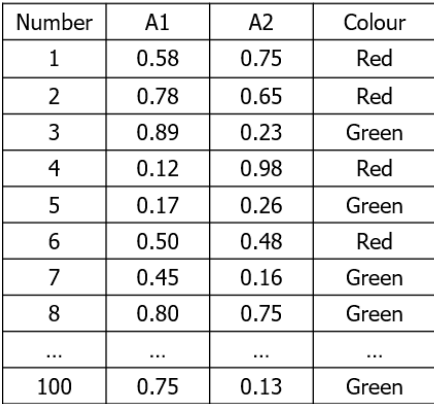
Temp.	Play
80	No
85	No
83	Yes
75	Yes
68	Yes
65	No
64	Yes
72	No
75	Yes
70	Yes
69	Yes
72	Yes
81	Yes
71	No

$\xrightarrow{\text{Sort}}$

Temp.	Play
64	Yes
65	No
68	Yes
69	Yes
70	Yes
71	No
72	No
72	Yes
75	Yes
75	Yes
80	No
81	Yes
83	Yes
85	No

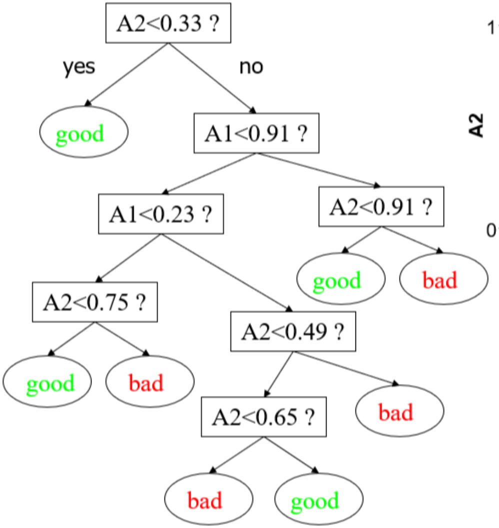
$\text{Temp.} < 64.5$	$\Delta I = 0.048$
$\text{Temp.} < 66.5$	$\Delta I = 0.010$
$\text{Temp.} < 68.5$	$\Delta I = 0.000$
$\text{Temp.} < 69.5$	$\Delta I = 0.015$
$\text{Temp.} < 70.5$	$\Delta I = 0.045$
$\text{Temp.} < 71.5$	$\Delta I = 0.001$
$\text{Temp.} < 73.5$	$\Delta I = 0.001$
$\text{Temp.} < 77.5$	$\Delta I = 0.025$
$\text{Temp.} < 80.5$	$\Delta I = 0.000$
$\text{Temp.} < 82.0$	$\Delta I = 0.010$
$\text{Temp.} < 84.0$	$\Delta I = 0.113$

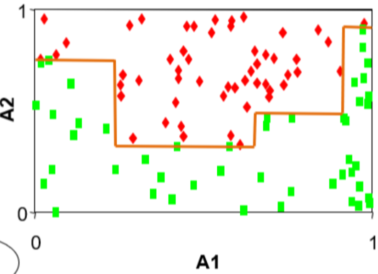
Continuous attributes (iii)



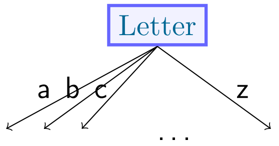
Number	A1	A2	Colour
1	0.58	0.75	Red
2	0.78	0.65	Red
3	0.89	0.23	Green
4	0.12	0.98	Red
5	0.17	0.26	Green
6	0.50	0.48	Red
7	0.45	0.16	Green
8	0.80	0.75	Green
...	...	...	...
100	0.75	0.13	Green

Continuous attributes (iv)





Attributes with many values (i)



Problems:

- Not good splits: they fragment the data too quickly, leaving unsufficient data for the next level.
- The reduction of impurity of such tests is often high (*e.g.* splitting on the object ID)

There are two solutions:

- Change the splitting criterion to penalize attributes with many values.
- Consider only binary splits (preferable)

Attributes with many values (ii)

Modified splitting criteria:

- $\text{GainRatio}(LS, A) = \frac{\Delta I_{\mathsf{Sh}}(LS,A)}{\text{SplitInformation}(LS,A)}$
- $\text{SplitInformation}(LS, A) = -\sum_a \frac{|LS_a|}{|LS|} \log_2 \left(\frac{|LS_a|}{|LS|}\right)$

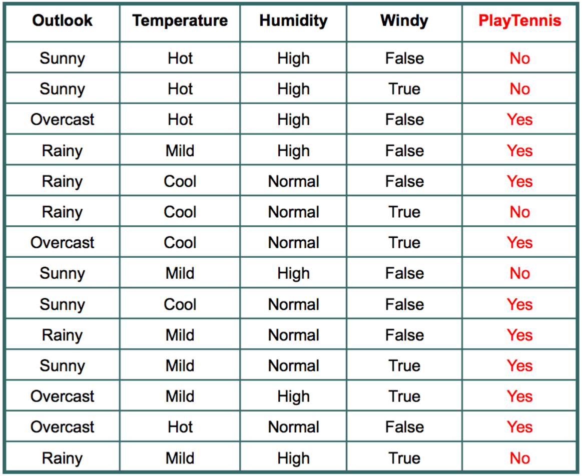
  The split information is high when there are many values.

Example: outlook in the tennis problem.

- $\Delta I_{\mathsf{sh}}(LS, \text{outlook}) = 0.246$
- $\text{SplitInformation}(LS, \text{outlook}) = 1.577$
- $\text{GainRatio}(LS, \text{outlook}) = \frac{0.246}{1.577} = 0.156 < 0.246$

Problem: the gain ratio favours unbalanced tests.

Attributes with many values (iii)

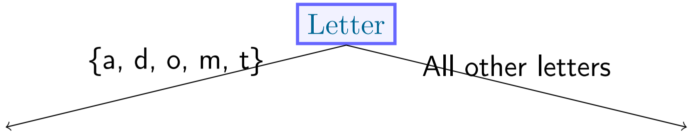


Outlook	Temperature	Humidity	Windy	PlayTennis
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Source: [3]

Attributes with many values (iv)

Allow binary tests only:



There are $2^{N-1} - 1$ non-trivial binary partitions for N values. If N is small, we can use enumeration.
However, if N is large, a **heuristic** is needed.
Example: Greedy approach

Missing attribute values

Not all attribute values are known for every object during learning or testing.

Day	Outlook	Temperature	Humidity	Wind	Play
D15	Sunny	Hot	?	Strong	No

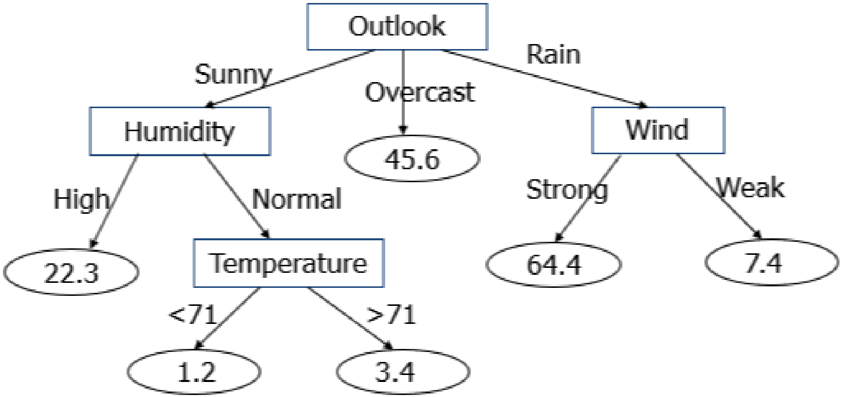
There are three strategies:

- Assign the most common value in the learning sample
- Assgin the most common value in the tree
- Assign a probability to each possible value

Outline

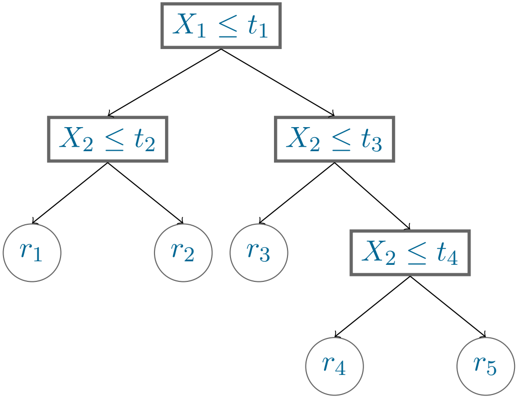
Regression trees (i)

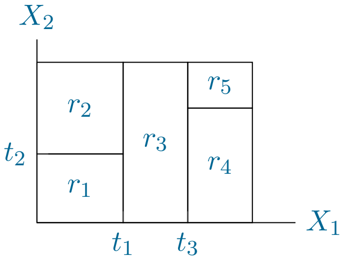
A regression tree is exactly the same model as a decision tree but with a **number** in each **leaf** instead of a class.



Regression trees (ii)

A regression tree is a piecewise constant function of the input attributes.





Regression tree growing

To minimize the square error on the learning sample, the prediction at a leaf is the **average output** of the learning cases reaching that leaf.

The impurity of a sample is defined by the variance of the output in that sample:

$$I(LS) = \mathrm{var}_{y|LS}\{y\} = \mathrm{E}_{y|LS}\left\{\left(y - \mathrm{E}_{y|LS}\{y\}\right)^2\right\}$$

where $\mathrm{E}_{y|LS}\{f(y)\}$ denotes the average of $f(y)$ in the sample LS.

The best split is the one that reduces the most variance:

$$\Delta I(LS, A) = \mathrm{var}_{y|LS}\{y\} - \sum_a \frac{|LS_a|}{|LS|}\mathrm{var}_{y|LS_a}\{y\}$$

Regression tree pruning

The method are exactly the same: pre-pruning and post-pruning.

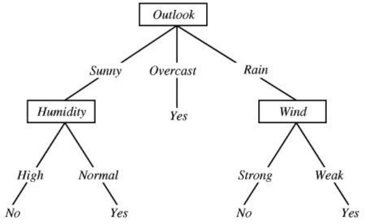
In post-pruning, the tree that minimizes the squared error on VS is selected.

In practice, pruning is more important in regression problems because full trees are much more complex: often, all objects have a different output value and hence the full tree has as many leaves as objects in the learning sample.

Outline

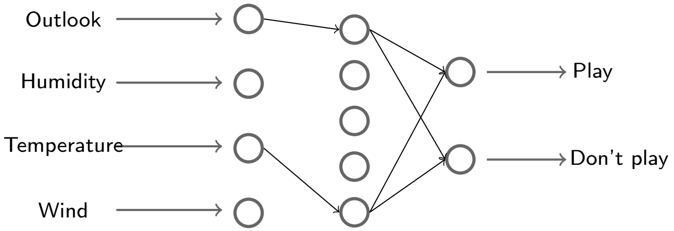
Interpretability (i)

With decision trees, interpretability is obvious.

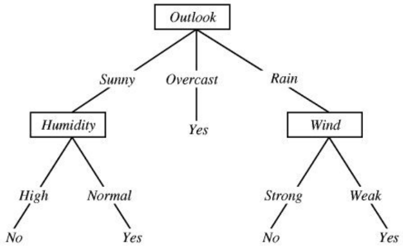


Source: [3]

With neural networks, it is more complex.



Interpretability (ii)



Source: [3]

A tree may be converted into a set of rules:

- **If** $(\text{Outlook} = \textit{Sunny})$ and $(\text{Humidity} = \textit{High})$ **then** $\text{PlayTennis} = \textit{No}$
- **If** $(\text{Outlook} = \textit{Sunny})$ and $(\text{Humidity} = \textit{Normal})$ **then** $\text{PlayTennis} = \textit{Yes}$
- **If** $\text{Outlook} = \textit{Overcast}$ **then** $\text{PlayTennis} = \textit{Yes}$
- ...

Attribute selection

If some attributes are not useful for classification, they will not be selected in the pruned tree.

Attribute selection is of practical importance, if measuring the value of an attribute is costly.
Example: Medical diagnosis

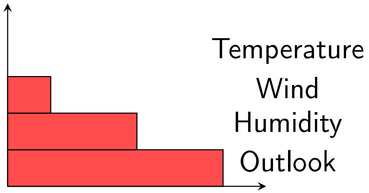
Decision trees are often used as a pre-processing step for other learning algorithms that suffer more when there are irrelevant variables.

Variable importance

In many applications, all variables do not contribute equally in predicting the output.

We can evaluate variable importance by computing the total reduction of impurity brought by each variable:

$$I(A) = \sum_{Nodes\ where\ A\ is\ tested} |LS_{node}| \Delta I(LS_{node}, A)$$



Outline

Decision trees - Conclusions

Advantages:

- They are very fast: they can handle very large datasets with many attributes.
  $\rightarrow$ Complexity: $\mathcal{O}(nN \log N)$ where n is the number of attributes and N the number of samples.
- They are flexible: they can handle several attribute types, classification and regression problems, missing values, ...
- They have a good interpretability: they provide rules and attribute importance.

Disadvantages:

- They are quite unstable, due to their high variance.
- They are not always competitive with other algorithms in terms of accuracy.

Research

- Cost and un-balanced learning sample.
- Oblique trees (test like $\sum \alpha_i A_i < a_{th}$).
- Using predictive models in leaves, *e.g.* using linear regression.
- Induction graphs.
- Fuzzy decision trees (from a crisp partition to a fuzzy partition of the learning sample).

Further reading

- Hastie et al., *The elements of statistical learning: data mining, inference, and prediction*, [4]:
    - chapter 9, Section 9.2
- Louppe Gilles, *Understanding random forests : from theory to practice* [5].
- L. Breiman et al., *Classification and regression trees*, [6]
- J.R. Quinlan, *C4.5: programs for machine learning*, [7]
- D.Zighed and R.Rakotomalala, *Graphes d'induction: apprentissage et data mining*, [8]
- Supplementary slides are also available on the course website.

Software

- `scikit-learn`: http://scikit-learn.org
- `Weka`: https://www.cs.waikato.ac.nz/ml/weka/
- `R`: packages *tree* and *rpart*

References I

Todd R Golub, Donna K Slonim, Pablo Tamayo, Christine Huard, Michelle Gaasenbeek, Jill P Mesirov, Hilary Coller, Mignon L Loh, James R Downing, Mark A Caligiuri, et al.
Molecular classification of cancer: class discovery and class prediction by gene expression monitoring.
science, 286(5439):531–537, 1999.

Decision tree and entropy algorithm.
https://zhengtianyu.wordpress.com/2013/12/13/decision-trees-and-entropy-algorithm/.
Accessed: 2020-07-29.

A tutorial to understand decision tree id3 learning algorithm.
https://nulpointerexception.com/2017/12/16/a-tutorial-to-understand-decision-tree-id3-learning-algorithm/.
Accessed: 2020-07-29.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman.
The elements of statistical learning: data mining, inference, and prediction.
Springer Science & Business Media, 2009.

Gilles Louppe.
Understanding random forests : from theory to practice.
Université de Liège, 2014.

References II

Leo Breiman, Jerome Friedman, Richard Olshen, and Charles Stone.
Classification and regression trees. wadsworth int.
Group, 37(15):237–251, 1984.

J Ross Quinlan.
C4. 5: programs for machine learning.
Elsevier, 2014.

Djamel A Zighed and Ricco Rakotomalala.
Graphes d'induction: apprentissage et data mining.
Hermes Science Publications Paris, 2000.

Frequently asked questions

- What is the computational complexity of the learning algorithm ?
- How do we handle (continuous) numerical input variables ?
- Explain the optimal splitting algo.
- What are the 2-3 main changes to make to implement regression tree learning with respect to decision tree learning ?
- How to adapt this approach to general loss-functions $\ell(\cdot, \cdot)$?
- Discuss theoretical asymptotic ($N \to \infty$) properties ?
- Discuss computational complexity and interpretability.