* * * **Artificial Intelligence and Machine Learning** * * *

Decision Trees and Ensemble Methods

Daniel Wehner, M.Sc.

SAP SE / DHBW Mannheim

Summer term 2025

Find all slides on GitHub (DaWe1992/Applied_ML_Fundamentals)

Lecture Overview

Agenda for this Unit

Section:

Introduction

What are Decision Trees?
An exemplary Decision Tree
An alternative Decision Tree

What are Decision Trees?

- Decision trees are induced in a **supervised fashion**
- The ID3 algorithm was originally proposed by Ross Quinlan in 1986
- Decision trees are grown **recursively** *(divide-and-conquer)*
- Decision trees are **easily interpretable**
  (unlike other methods like e. g. neural networks)
- A decision tree consists of:

Nodes	Each node corresponds to an **attribute test**
Edges	One edge per possible test outcome *(attribute value)*
Leaves	Class label to predict

Portrait: ROSS QUINLAN

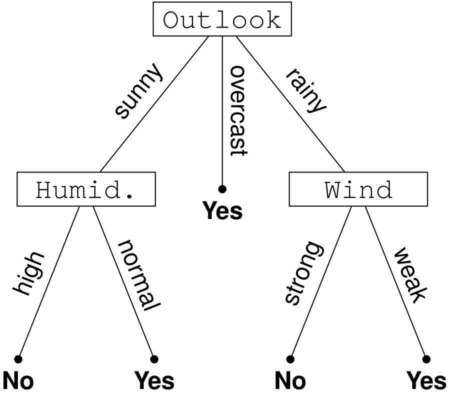
JOHN ROSS QUINLAN is an Australian computer science researcher in data mining and decision theory. He has contributed extensively to the development of decision tree algorithms, including inventing the canonical C4.5 and ID3 algorithms.

He is currently running the company RuleQuest Research which he founded in 1997.

(Wikipedia)

What we want...

Outlook	Temperature	Humidity	Wind	PlayGolf
sunny	hot	high	weak	**no**
sunny	hot	high	strong	**no**
overcast	hot	high	weak	**yes**
rainy	mild	high	weak	**yes**
rainy	cool	normal	weak	**yes**
rainy	cool	normal	strong	**no**
overcast	cool	normal	strong	**yes**
sunny	mild	high	weak	**no**
sunny	cool	normal	weak	**yes**
rainy	mild	normal	weak	**yes**
sunny	mild	normal	strong	**yes**
overcast	mild	high	strong	**yes**
overcast	hot	normal	weak	**yes**
rainy	mild	high	strong	**no**
rainy	mild	normal	strong	**???**



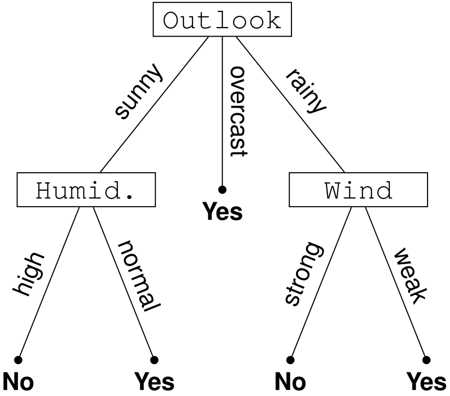
Classification of new Instances

- Suppose we get a new instance:

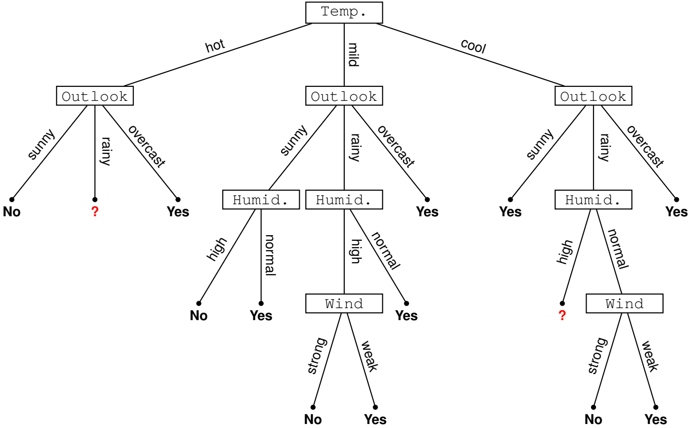
`Outlook`	rainy
`Temperature`	mild
`Humidity`	normal
`Wind`	strong

- **Question:** What is its class?
- **Answer:** No

> **Please note:** Not all attributes must be considered for the classification!



Another Decision Tree...



Questions:

- Is this one better?
- What problems do you see?

Section:

Iterative Dichotomizer (ID3)

- Inductive Bias of Decision Trees
- Split Heuristics: Entropy and Information Gain
- ID3 Algorithm

Inductive Bias of Decision Trees

- Complex models tend to **overfit** the training data and hence **do not generalize well** to unseen data points
- **Therefore:** Prefer the simplest hypothesis that fits the data!
- This leads to:

OCCAM's razor: *'More things should not be used than are necessary'*

Please note: OCCAM's razor is a general methodology in machine learning and not limited to decision trees! Also, it is a guiding principle in other sciences.

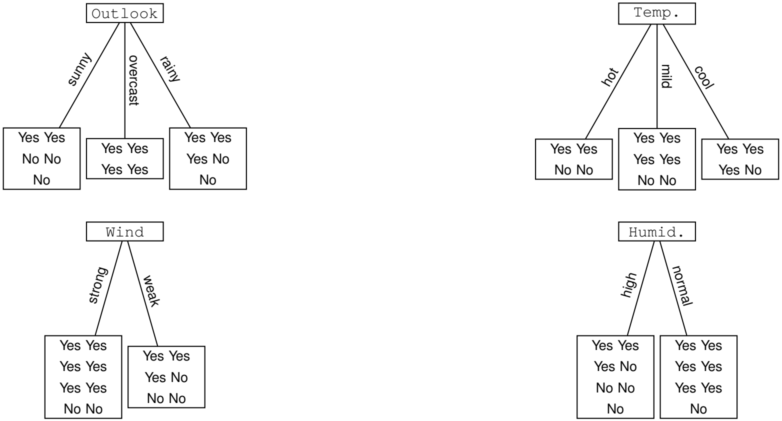
Portrait: William of Ockham

William of Ockham (circa 1287 – 1347) was an English Franciscan friar who is believed to have been born in Ockham, a small village in Surrey. He is considered to be one of the major figures of medieval thought.

He is commonly known for Occam's razor, the methodological principle that bears his name, and also produced significant works on logic, physics and theology.

(Wikipedia)

The Root of all Evil... Which Attribute to choose?



Finding a proper Split Attribute

- Simple and small trees are preferred:
  - Data in successor nodes should be **as pure as possible** *(with respect to the class labels)*
  - This means, nodes containing only one class are preferable
- To learn small trees, we have to split by attributes which **provide the most information** and produce the least successor nodes

Question:

How can we express this thought as a mathematical formula?

Measure of Impurity: Entropy

Answer:

- **Entropy** *H* (greek capital η) introduced by CLAUDE E. SHANNON
- The idea of entropy originates in the field of **information theory**

Properties of entropy:

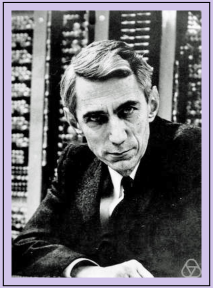
- Entropy reaches its minimum if both classes are equally distributed
- **Pure datasets have minimal entropy**
- Therefore, split by attributes which reduce the entropy the most

Portrait: Claude Shannon

Claude Elwood Shannon (April 30, 1916 – February 24, 2001) was an American mathematician, electrical engineer, computer scientist and cryptographer. He is commonly known as the "father of information theory".

Shannon contributed to the field of cryptanalysis for national defense of the United States during World War II, and his mathematical theory of information became very well cited and laid the foundation for the field of information theory.

(Wikipedia)

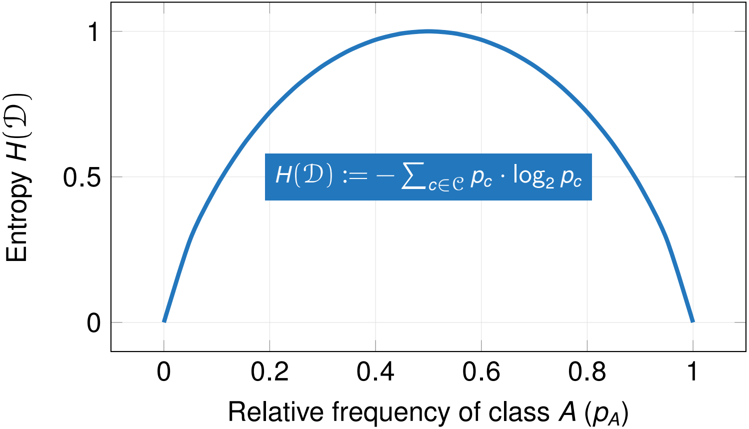


Computation of the Entropy

- The entropy H is a measure of chaos in the data and is measured in bits
- **Example:** Consider the two classes $\mathcal{C}_1 = A$ and $\mathcal{C}_2 = B$

$$
\begin{aligned}
&H\big(\{A, A, A, A, A, A, A, A\}\big) && \rightarrow 0 && \textit{Bits} \\
&H\big(\{A, A, A, A, A, A, B, B\}\big) && \rightarrow 0.81 && \textit{Bits} \\
&H\big(\{A, A, A, A, B, B, B, B\}\big) && \rightarrow 1 && \textit{Bit} \\
&H\big(\{A, A, B, B, B, B, B, B\}\big) && \rightarrow 0.81 && \textit{Bits} \\
&H\big(\{B, B, B, B, B, B, B, B\}\big) && \rightarrow 0 && \textit{Bits}
\end{aligned}
$$

Computation of the Entropy (Ctd.)



Important

Computation of the Entropy (Ctd.)

Entropy formula:

$$H(\mathcal{D}) := -\sum_{c \in \mathcal{C}} p_c \cdot \log_2 p_c \tag{1}$$

(p_c denotes the relative frequency of class $c \in \mathcal{C}$)

Weather data: $\mathcal{C} := \{\text{yes}, \text{no}\} \quad \Rightarrow \quad p_{\text{yes}} = {}^9\!/_{14} \quad \text{and} \quad p_{\text{no}} = {}^5\!/_{14}$

$$H(\mathcal{D}) = -\sum_{c \in \mathcal{C}} p_c \cdot \log_2 p_c = -({}^9\!/_{14} \cdot \log_2 {}^9\!/_{14} + {}^5\!/_{14} \cdot \log_2 {}^5\!/_{14}) = \mathbf{0.9403}$$

Quality of the Split: Average Entropy

- We still do not know which attribute to use for the first split
- **Idea:** Calculate the entropy after each potential split and compare

Average Entropy after splitting by attribute $\mathtt{A}$:

$$H(\mathcal{D}|\mathtt{A}) := \sum_{v \in \mathrm{dom}(\mathtt{A})} \frac{|\mathcal{D}_{\mathtt{A}=v}|}{|\mathcal{D}|} \cdot H(\mathcal{D}_{\mathtt{A}=v}) \tag{2}$$

$\mathtt{A}$	Attribute
$\mathrm{dom}(\mathtt{A})$	Possible values attribute $\mathtt{A}$ can take (domain of $\mathtt{A}$)
$\lvert\mathcal{D}_{\mathtt{A}=v}\rvert$	Number of examples satisfying $\mathtt{A} = v$

Example: Average Entropy

Weather data:

- Potential split by attribute `Outlook`
- We compute the entropy of the resulting subsets:

$$H(\mathcal{D}_{\texttt{Outlook}=\text{sunny}}) = -\left(2/5 \cdot \log_2(2/5) + 3/5 \cdot \log_2(3/5)\right) = 0.9710$$

$$H(\mathcal{D}_{\texttt{Outlook}=\text{rainy}}) = -\left(3/5 \cdot \log_2(3/5) + 2/5 \cdot \log_2(2/5)\right) = 0.9710$$

$$H(\mathcal{D}_{\texttt{Outlook}=\text{overcast}}) = -\left(4/4 \cdot \log_2(4/4) + 0/4 \cdot \log_2(0/4)\right) = 0$$

We directly see that $H(\mathcal{D}_{\texttt{Outlook}=\text{overcast}}) = 0$ *(do not compute it explicitly!)*

Example: Average Entropy (Ctd.)

Weather data:

- Potential split by attribute `Outlook`
- We compute a weighted average:

$$H(\mathcal{D}|\texttt{Outlook}) = \sum_{v \in \text{dom}(\texttt{Outlook})} \frac{|\mathcal{D}_{\texttt{Outlook}=v}|}{|\mathcal{D}|} \cdot H(\mathcal{D}_{\texttt{Outlook}=v})$$

$$= {}^{5}/_{14} \cdot 0.9710 + {}^{5}/_{14} \cdot 0.9710 + {}^{4}/_{14} \cdot 0$$

$$= \mathbf{0.6936}$$

Information Gain

- We have to calculate the average entropy for all attributes
- The difference of entropy before and after split is called the **information gain (IG)**
- Select the attribute with the highest IG

Attribute	H_{before}	H_{after}	IG
`Outlook`	0.9403	0.6936	0.2464
`Temperature`	0.9403	0.9111	0.0292
`Humidity`	0.9403	0.7885	0.1518
`Wind`	0.9403	0.8922	0.0481

- **Here:** Attribute `Outlook` maximizes the information gain

Training Data after the Split by Attribute `Outlook`

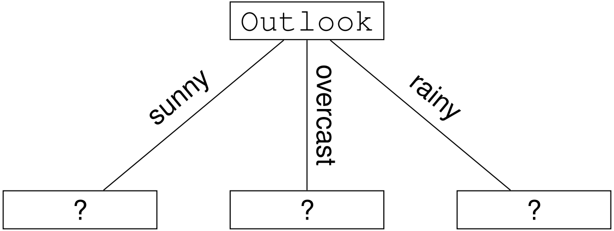
~~`Outlook`~~	`Temperature`	`Humidity`	`Wind`	`PlayGolf`
~~sunny~~	hot	high	weak	no
~~sunny~~	hot	high	strong	no
~~sunny~~	mild	high	weak	no
~~sunny~~	cool	normal	weak	yes
~~sunny~~	mild	normal	strong	yes
~~rainy~~	mild	high	weak	yes
~~rainy~~	cool	normal	weak	yes
~~rainy~~	cool	normal	strong	no
~~rainy~~	mild	normal	weak	yes
~~rainy~~	mild	high	strong	no
~~overcast~~	cool	normal	strong	yes
~~overcast~~	hot	high	weak	yes
~~overcast~~	mild	high	strong	yes
~~overcast~~	hot	normal	weak	yes

- The table on the left displays the dataset $\mathcal{D}$ after the split by attribute `Outlook`
- We obtain three subsets *(one per attribute value)*
- Attribute `Outlook` is removed in the current branch of the tree **(Why?)**

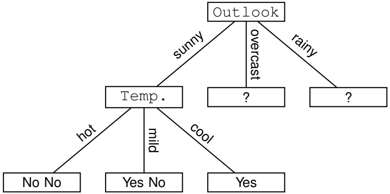
How to proceed?

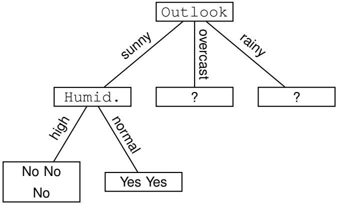
- The algorithm is recursively applied to the resulting subsets:
  1. Calculate entropy before and after each potential split
  2. Calculate the information gain for each attribute
  3. Choose the attribute with maximum information gain for the split
  4. In the current branch: Do not consider the attribute chosen anymore
  5. **Recursion** ↻ *(go to 1)*
- Recursion stops as soon as the subsets are pure $\Rightarrow$ **Danger: ☠ Overfitting ☠**
- In the example above, the subset $\mathcal{D}_{\texttt{Outlook}=\textit{overcast}}$ is already pure
- This algorithm is referred to as **ID3 (Iterative Dichotomizer)**

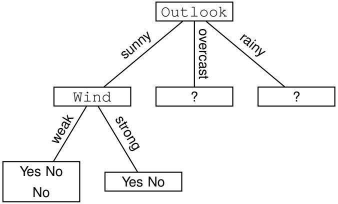
Step by Step: Construction of the Tree



Step by Step: Construction of the Tree (Ctd.)

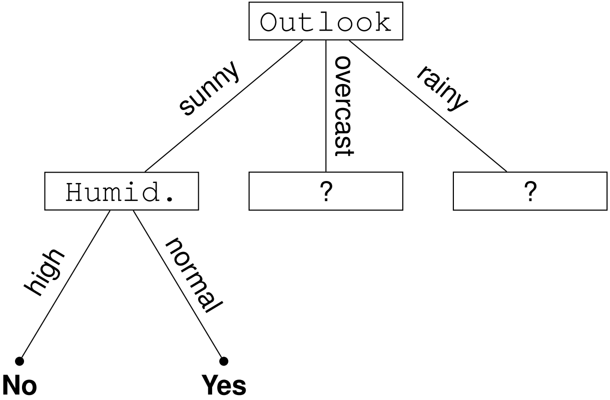




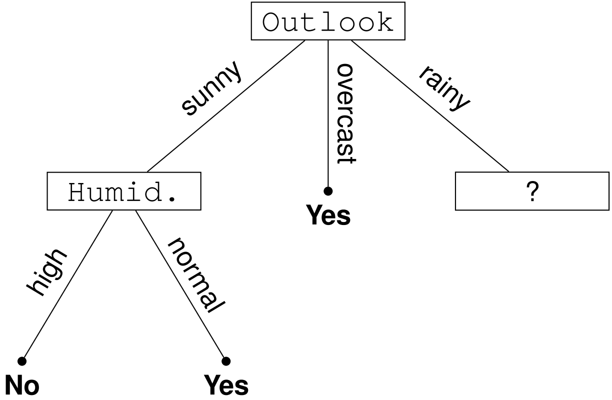


- $\text{IG}(\texttt{Temperature}) = 0.571$
- $\text{IG}(\texttt{Humidity}) = \mathbf{0.971}$
- $\text{IG}(\texttt{Wind}) = 0.020$

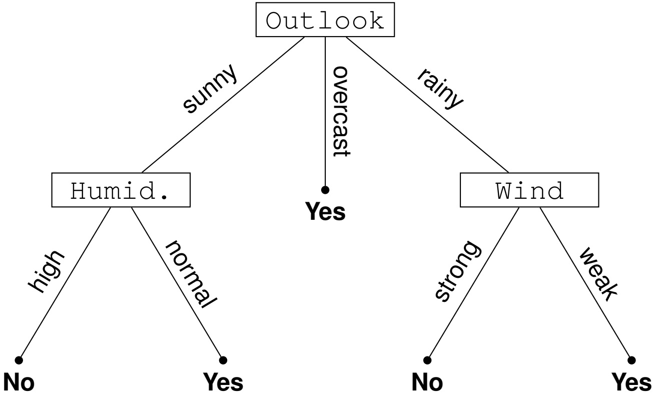
Step by Step: Construction of the Tree (Ctd.)



Step by Step: Construction of the Tree (Ctd.)



Step by Step: Construction of the Tree (Ctd.)



Section:

Extensions and Variants

Other Measures of Impurity
Highly-branching Attributes
Numeric Attributes
Regression Trees

An Alternative to Information Gain: Gini Index

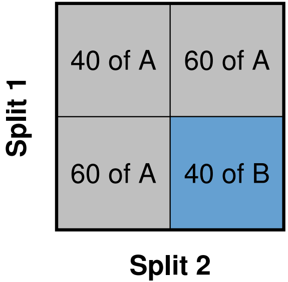
Gini index:

$$\text{Gini}(\mathcal{D}) := \sum_{c \in \mathcal{C}} p_c \cdot (1 - p_c) = 1 - \sum_{c \in \mathcal{C}} p_c^2 \tag{3}$$

- Gini index and entropy always **produce the same decision tree**
- Often used as a default in machine learning libraries **(Why?)**
- Used e. g. in **CART (Classification and Regression Trees)**
- **Gini gain** could be defined analogously to IG *(this is usually not done)*

Why not use the Error as a splitting Criterion?

- The bias towards pure leaves is **not strong enough**
- **Example:**



Error before the split:

20 %

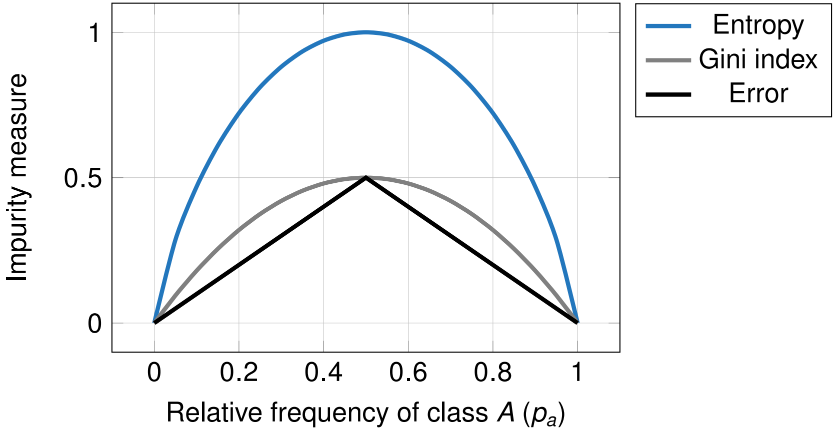
Error after the split:

20 %

Both splits don't improve the error.

But together they give a perfect split!

Summary: Impurity Measures



Highly-branching Attributes

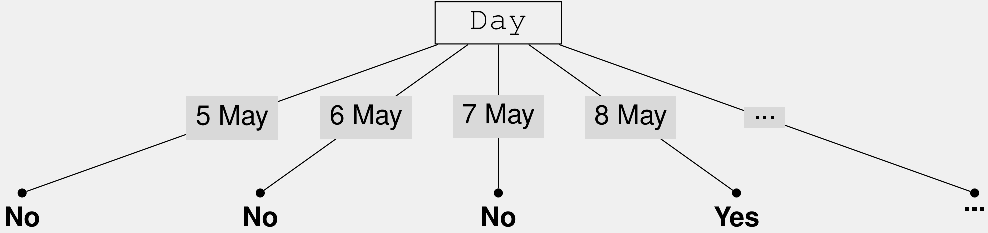
Problem: Attributes with a large number of values are problematic, since the leaves are not 'backed' with sufficient data examples.

In extreme cases there is only one example per node *(e. g. IDs)*

This may lead to:

- **Overfitting** *(selection of attributes which are not optimal for prediction)*
- **Fragmentation** *(data is fragmented into too many small sets)*

Example: Highly-branching Attributes



- Entropy before the split is 0.9403, entropy after the split is 0
- $\mathrm{IG}(\mathcal{D}, \texttt{Day}) = 0.9403$
- Attribute `Day` would be chosen for the split $\Rightarrow$ **Bad for prediction** ☠

Intrinic Information

Solution: Calculate the **intrinsic information (IntI)**:

$$\mathrm{IntI}(\mathcal{D}, \mathtt{A}) := -\sum_{v \in \mathrm{dom}(\mathtt{A})} \frac{|\mathcal{D}_{\mathtt{A}=v}|}{|\mathcal{D}|} \cdot \log_2 \frac{|\mathcal{D}_{\mathtt{A}=v}|}{|\mathcal{D}|} \tag{4}$$

- Attributes with high *IntI* are **less useful** as they result in high fragmentation
- New splitting heuristic: **Gain ratio (GR)**

$$\mathrm{GR}(\mathcal{D}, \mathtt{A}) := \frac{\mathrm{IG}(\mathcal{D}, \mathtt{A})}{\mathrm{IntI}(\mathcal{D}, \mathtt{A})} \tag{5}$$

Example: Intrinsic Information

- Intrinsic information for attribute `Day`:

$$\mathrm{IntI}(\mathcal{D}, \texttt{Day}) = 14 \cdot \left(-{}^1\!/_{14} \cdot \log_2({}^1\!/_{14})\right) = \mathbf{3.807} \tag{6}$$

- Gain ratio for attribute `Day`:

$$\mathrm{GR}(\mathcal{D}, \texttt{Day}) = \frac{0.9403}{3.807} = \mathbf{0.246} \tag{7}$$

Attention: In this case, attribute `Day` would still be chosen. Be careful which features you include in the training dataset! **(Feature engineering is important!)**

Handling of numeric Attributes

- Usually, only **binary splits** are considered, e. g.:
  - $\texttt{Temperature} < 48$
  - **Not:** $24 \leqslant \texttt{Temperature} \leqslant 31$ *(produces three subsets)*
- To support non-binary splits, the attribute is **not removed** and can be chosen again

Problem: There is an **infinite number** of possible splits! **Splitting on numeric attributes is computationally more demanding!**

Solution: Discretize the range using a fixed step size

Example I: Handling numeric Attributes

- Consider the attribute `Temperature`
- Unlike before, the attribute takes **numerical values** instead of discrete ones:

64	65	68	69	70	71	72	72	75	75	80	81	83	85
Yes	**No**	**Yes**	**Yes**	**Yes**	**No**	**No**	**Yes**	**Yes**	**Yes**	**No**	**Yes**	**Yes**	**No**

- $\texttt{Temperature} < 71.5$

  # yes: 4 | # no: 2

- $\texttt{Temperature} \geqslant 71.5$

  # yes: 5 | # no: 3

$$H(\mathcal{D}|\texttt{Temp.}) = {}^{6}/_{14} \cdot H(\texttt{Temp.} < 71.5) + {}^{8}/_{14} \cdot H(\texttt{Temp.} \geqslant 71.5) = \mathbf{0.939}$$

Example II: Handling numeric Attributes

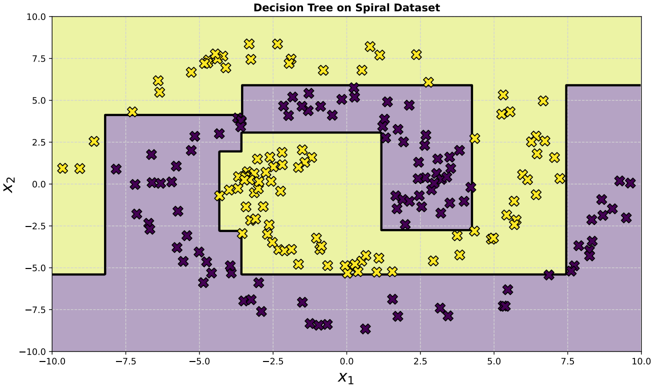
Dataset:

Taxable income	60	70	75	85	90	95	100	120	125	220
Class label	No	No	No	Yes	Yes	Yes	No	No	No	No

Evaluation of splits:

Split point	55		65		72		80		87		92		97		110		122		172		230	
	$\leqslant$	$>$	$\leqslant$	$>$	$\leqslant$	$>$	$\leqslant$	$>$	$\leqslant$	$>$	$\leqslant$	$>$	$\leqslant$	$>$	$\leqslant$	$>$	$\leqslant$	$>$	$\leqslant$	$>$	$\leqslant$	$>$
Yes	0	3	0	3	0	3	0	3	1	2	2	1	3	0	3	0	3	0	3	0	3	0
No	0	7	1	6	2	5	3	4	3	4	3	4	3	4	4	3	5	2	6	1	7	0
Gini index	0.420		0.400		0.375		0.343		0.417		0.400		**0.300**		0.343		0.375		0.400		0.420	

Decision Tree on a Spiral Dataset



Regression Trees

- We can use decision trees to predict **continuous target variables**
- Predict the average value of all examples in the leaf
- Split the data such that the variance in the leaves is minimized

Standard deviation reduction (SDR):

$$\mathrm{SDR}(\mathcal{D}, \mathtt{A}) := \mathrm{SD}(\mathcal{D}) - \sum_{v \in \mathrm{dom}(\mathtt{A})} \frac{|\mathcal{D}_{\mathtt{A}=v}|}{|\mathcal{D}|} \cdot \mathrm{SD}(\mathcal{D}_{\mathtt{A}=v}) \tag{8}$$

Regression Trees (Ctd.)

Caveat: It is important to specify a termination criterion when using decision trees in regression tasks, otherwise we get a single data point per leaf!

For example: Continue splitting the dataset until the standard deviation falls below a specified threshold for the first time

Section:

Ensemble Methods

- Introduction to Ensembles
- Bootstrap Aggregating (Bagging)
- Randomization
- Random Forests
- ExtraTrees

Introduction Ensemble Methods

Key Idea: Do not learn a single classifier, but a **set of classifiers**. Then combine the predictions of the base classifiers to obtain the ensemble prediction

Problem: How can we induce multiple classifiers from a single dataset without getting the same classifier over and over again? **We want to have diverse classifiers, otherwise the ensemble is useless!**

- Basic techniques are: **Bagging**, **boosting**, and **stacking**
- We shall have a closer look into the bagging technique

What is the Advantage of an Ensemble?

- Let 25 **independent** base classifiers be given
- **Independence assumption:** The probability of a single classifier misclassifying an example **does not** depend on other classifiers in the ensemble
- This condition is usually not fully satisified in practice **(Why?)**
- Each individual classifier in the ensemble is assumed to have an error rate of $\varepsilon := 0.35$

Question: What is the error rate of the ensemble?

What is the Advantage of an Ensemble? (Ctd.)

- The prediction of the ensemble is given by the **majority vote**, i. e. the class with the most votes is predicted
- The ensemble makes a wrong prediction **if the majority is wrong**, i. e. at least 13 base classifiers misclassify the example
- This probability is computed according to the **binomial distribution**

$$\varepsilon_{\text{ensemble}} := \sum_{k=13}^{25} \binom{25}{k} \cdot \varepsilon^k \cdot (1-\varepsilon)^{25-k} \approx 0.06 \tag{9}$$

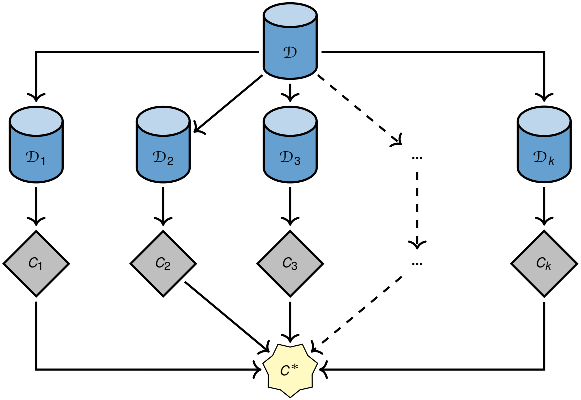
- We see: $\varepsilon_{\text{ensemble}} \approx 0.06 \ll 0.35 = \varepsilon$

Bootstrap Aggregating (Bagging)

Create
multiple datasets

Learn
multiple classifiers

Combine decisions



Creating the Bootstrap Samples

- **Quesion:** How to generate multiple datasets which are different?
- **Solution:** Use sampling with replacement

Original Data	1	2	3	4	5	6	7	8	9	10
Bagging (Round 1)	7	8	10	8	2	5	10	10	5	9
Bagging (Round 2)	1	4	9	1	2	3	2	7	3	2
Bagging (Round 3)	1	8	5	10	5	5	9	6	3	7

- Some examples may appear **in more than one set**
- Some examples may appear **more than once** in one set
- Some examples may **not appear at all** in a set

Bagging Algorithm

Input: Training set $\mathcal{D}$, number of base classifiers K

Training phase:

forall $k \in \{1, 2, \ldots, K\}$ **do**

  Draw a bootstrap sample $\mathcal{D}_k$ with replacement from $\mathcal{D}$

  Learn a base classifier C_k (e. g. a decision tree) from $\mathcal{D}_k$

  Add the classifier C_k to the ensemble

end

Prediction phase:

forall *unlabeled instances* **do**

  Get predictions from all classifiers C_k $(1 \leqslant k \leqslant K)$

end

return *Class which receives the majority of votes (combined classifier C^*)*

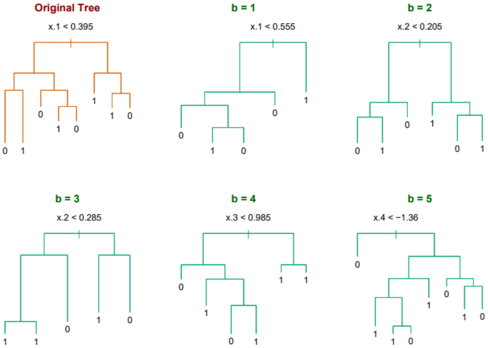
Bagging Variations

We have considered bootstrap samples of equal size, drawn with replacement

Also conceivable are the following scenarios:

1. **Varying the size** of the bootstrap samples
2. Sampling **without replacement** **(Pasting)**
3. **Sampling of features**, not instances *(not all features are available)*
4. Creating **heterogeneous ensembles** comprising different types of base classifiers *(e. g. neural networks, decision trees, support vector machines)*

Bagged Decision Trees



cf. [HASTIE.2008], page 284

Randomization

- Why not **randomizing the algorithm** instead of the data?
- Some algorithms already do that: For example neural networks randomly initialize the weights before training
- Especially greedy algorithms can be randomized:
  - Pick from the options **randomly**, instead of picking the best one
  - E. g. decision trees: Do not choose the attribute with the highest information gain, but select a split attribute randomly

A random forest combines randomization and bagging

Random Forest Algorithm

- A **random forest** is an ensemble of decision trees
- It combines **bagging** and **random attribute subset selection**
- We grow a decision tree from each bootstrap sample
- We select the best splitting attribute among a random subset of attributes

At each step a random forest selects the best splitting attribute from a randomly chosen subset of features, but the globally best feature **may not** be available.

Random Forest Algorithm

Input: Training set $\mathcal{D}$, number of base classifiers K

Training phase:

for $k \in \{1, 2, \ldots, K\}$ **do**

  Create a bootstrap sample from $\mathcal{D}$ (e. g. with replacement) $\Rightarrow$ **Bagging**

  begin

    Grow the tree

    At every node: Randomly choose a subset of attributes to be considered for the split

    $\Rightarrow$ **Randomization**

  end

  Add tree C_k to the ensemble

end

Prediction phase:

forall *unlabeled instances* **do**

  Get predictions from all classifiers C_k $(1 \leqslant k \leqslant K)$

end

return *Class which receives the majority of votes (combined classifier C^*)*

ExtraTrees (Randomization 2.0)

- One more step of randomization ⇒ **Ext**remely **Ra**ndomized **Trees** (ExtraTrees)
- The general approach is the same as for random forests, **but:**
  - Instead of choosing the optimal split point...
  - ...it is selected randomly
  - Each decision tree is grown without having to calculate entropy
- It is **much faster** due to less computation

The large number of classifiers compensates for suboptimal splits

Section:

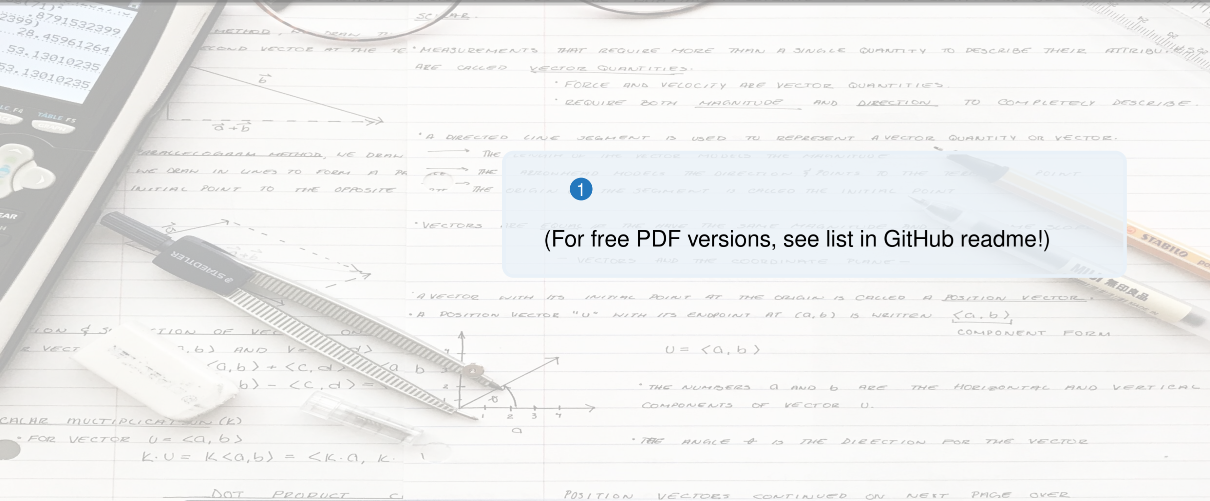
Wrap-Up

- Summary
- Recommended Literature
- Self-Test Questions
- Lecture Outlook

Summary

1. **Decision trees:**
    - The construction of decision trees is guided by an **impurity measure**, e. g. the entropy measure H or the Gini index
    - Recursively select features which **maximize the information gain**
    - Decision trees can handle **numeric attributes** and **continuous output**
    - Be careful with highly-branching attributes
2. **Ensembles:**
    - Usually, ensembles allow for a significant error reduction
    - **Bagging:** Sample diverse datasets from an underlying dataset
    - **Random forests** combine bagging and randomization

Recommended Literature



1

(For free PDF versions, see list in GitHub readme!)

Self-Test Questions

1. What is the inductive bias? What is the inductive bias of decision trees?
2. Explain what OCCAM's razor is.
3. What does entropy measure? How do you compute the information gain?
4. True or false? *'Pure datasets have maximal entropy.'*
5. What is the advantage of ensemble methods?
6. What is bagging?
7. Explain how a random forest works.

What's next...?

- **I** Machine Learning Introduction
- **II** Optimization Techniques
- **III** Bayesian Decision Theory
- **IV** Non-parametric Density Estimation
- **V** Probabilistic Graphical Models
- **VI** Linear Regression
- **VII** Logistic Regression
- **VIII** Deep Learning
- **IX** Evaluation
- **X** Decision Trees
- ● **XI** Support Vector Machines
- **XII** Clustering
- **XIII** Principal Component Analysis
- **XIV** Reinforcement Learning
- **XV** Advanced Regression

Thank you very much for the attention!

* * * **Artificial Intelligence and Machine Learning** * * *

Topic: Decision Trees and Ensemble Methods

Term: Summer term 2025

Contact:

Daniel Wehner, M.Sc.

SAP SE / DHBW Mannheim

daniel.wehner@sap.com

Do you have any questions?