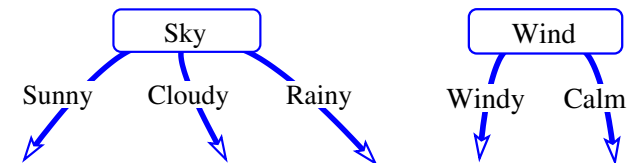


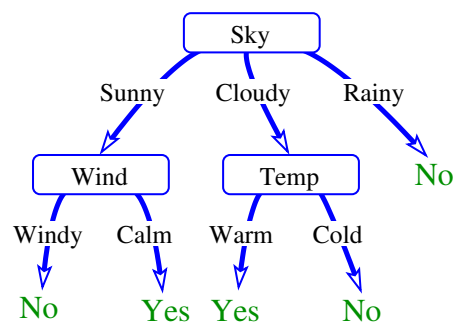
Decision Trees

Using Trees

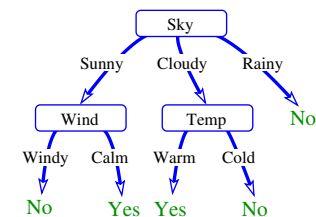
Basic Idea: Test the attributes sequentially



The whole analysis strategy can be seen as a tree.



The results (classifications) are coded by the *leaves*



What does the tree encode?

$$(\text{Sunny} \wedge \text{Calm}) \vee (\text{Cloudy} \wedge \text{Warm})$$

Works as a *disjunction of conjunctions*

Normal Form for boolean functions

Arbitrary boolean functions can be represented!

How can a decision tree be constructed automatically?

- ① Choose an attribute to test
- ② Branches with a unique class become leaves
- ③ Other branches are extended recursively

Remaining question: how do we choose attributes?

Greedy approach:

Choose the attribute which *tells us most* about the answer

Entropy

Entropy — measure of **unpredictability**

$$\text{Entropy} = \sum_i -p_i \log_2 p_i$$

p_i probability for event i

Entropy

Example: tossing a coin

$$p_{\text{head}} = 0.5; \quad p_{\text{tail}} = 0.5$$

$$\begin{aligned} \text{Entropy} &= \sum_i -p_i \log_2 p_i = \\ &= -0.5 \log_2 0.5 + -0.5 \log_2 0.5 = -0.5 \underbrace{\log_2 0.5}_{-1} + -0.5 \underbrace{\log_2 0.5}_{-1} = \\ &= 1 \end{aligned}$$

The result of a coin-toss has **1 bit** of information

Entropy

Example: rolling a dice

$$p_1 = \frac{1}{6}; \quad p_2 = \frac{1}{6}; \dots \quad p_6 = \frac{1}{6}$$

$$\begin{aligned} \text{Entropy} &= \sum_i -p_i \log_2 p_i = \\ &= 6 \times -\frac{1}{6} \log_2 \frac{1}{6} = \\ &= -\log_2 \frac{1}{6} = \log_2 6 \approx 2.58 \end{aligned}$$

The result of a dice-roll has **2.58 bit** of information

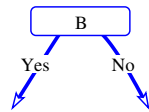
Entropy

Example: rolling a **fake dice**

$$p_1 = 0.1; \dots \quad p_5 = 0.1; \quad p_6 = 0.5$$

$$\begin{aligned} \text{Entropy} &= \sum_i -p_i \log_2 p_i = \\ &= -5 \cdot 0.1 \log_2 0.1 - 0.5 \log_2 0.5 = \\ &\approx 2.16 \end{aligned}$$

A real dice is **more unpredictable** (2.58 bit) than a fake (2.16 bit)

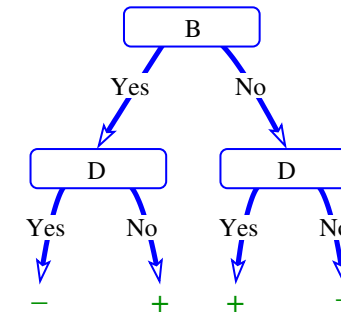


Examples where $B = \bullet$

A	B	C	D	
•	•	○	○	+
○	•	•	○	+
○	•	○	○	+
•	•	○	○	+
•	•	•	○	+
•	•	•	○	+
○	•	○	•	
•	•	○	•	
○	•	○	○	+
○	•	○	○	+
○	•	•	○	+

Examples where $B = \circ$

A	B	C	D	
○	○	○	○	
•	○	○	•	+
•	○	•	○	
○	○	○	○	
○	○	•	○	
○	○	○	•	+
•	○	○	○	
○	○	○	○	
•	○	○	○	
○	○	•	○	
•	○	○	○	
○	○	○	•	+
○	○	○	○	
•	○	○	○	



Which Bias does this learning algorithm have?

- **Restriction Bias?**
No, all hypotheses can be represented
- **Preference Bias?**
Yes, some trees are found before others

Which hypotheses (here: trees) are preferred?

- Shallow trees
- "Important attributes" early

Which hypothesis should be preferred when several are compatible with the data?

Occam's principle (*Occam's razor*, "Occam's rakkniv")

William from Ockham, Theologian and Philosopher (1288–1348)

"Entia non sunt multiplicanda praeter necessitatem"



translated:

"Entities should not be multiplied beyond necessity"

Overfitting (*överträning*)

When the hypotheses are overly specialized for the available training examples.

Good results on training data, but generalizes badly

When does this occur?

- Non-representative sample
- Noisy examples

What can be done about it?

Choose a simpler hypothesis and accept some errors for the training examples

Why are simple hypotheses more likely to be correct?

Philosophical argument:

It is more likely that the reality from which the examples come have a simple generating mechanism.

Pragmatic argument:

Simple hypotheses tends to generalize better.

Possible ways of improving the decision trees

- Avoid overfitting
 - Limit the tree's height
 - Pruning (*Beskärning*)
- Attributes with graded values
- Missing attribute values
- Variable cost for different attributes