# Many-Expert Decision Trees

Guillermo Badia[1] Carles Noguera[2] **Alberto Paparella**[3] Guido Sciavicco[3]

October 22, 2024

[1]School of Historical and Philosophical Inquiry, University of Queensland, Australia

[2]Department of Information Engineering and Mathematics, University of Siena, Italy

[3]Department of Mathematics and Computer Science, University of Ferrara, Italy

# Introduction

# Introduction

**Decision Trees** are undoubtedly one of the most traditional, yet powerful algorithms in machine learning, especially when it comes to tabular data.

One of their most prominent features is easily their high **interpretability**, making their behavior always clear and allowing, given a query, to get motivation for the provided outcome.

Taking things a step further, each decision tree has a logical counterpart consisting of a set of **(propositional) logical formulas**. Under this view, one can tie algorithms for decision tree extraction to specific logics.

# Introduction

Decision trees have been enhanced in the past following two directions:

- employing **Fuzzy Logics**, with the main objective of better-treating uncertainty in the data, and
- employing **Modal Logic** [4, 6], also allowing the application of decision trees to non-tabular data.

We propose a novel approach further generalizing the first direction, resorting to **many-valued logics**. Our aim is to model situations in which different experts can have different opinions on the same data.

# Table of contents

# Machine Learning 101

# What is Machine Learning?

Machine Learning is an **optimization** process:

1. define a **model** (decision trees, neural networks, ...) for the task to be solved, depending on a set of **parameters** $\Theta$
2. define a **performance metric** (mean squared errors, cross-entropy, cosine distance, ...), some **error** measure to evaluate the model
3. tune the parameters $\Theta$ to **minimize** the error on a **training set**

Models can be used to **acquire knowledge** on the input data, finding clusters, making correlations, ...

Then, they can be used to **make predictions** on **new data**.

# Why Learning?

Why learning?

- The tuning of parameters is based on observations (**training set**). We learn from our past experiences.
- We use **iterative** techniques to progressively approximate the results, and this can be understood as a form of learning process.

There are different types of learning tasks:

- **supervised learning** (classification, regression)
- **unsupervised learning** (clustering, autoencoding)
- **reinforcement learning** (learning long-term gains through rewards)

# Supervised Learning

Training set: a set of **training examples**

$$< x^{(i)}, y^{(i)} >$$

where

- $x^{(i)} \in X$ (set of inputs)
- $y^{(i)} \in Y$ (set of outputs)
- $i$ is the instance of the training sample

**Problem:** *learn* the function mapping $x^{(i)}$ to $y^{(i)}$.

- $Y$ discrete: **classification** problem (class prediction)
- $Y$ continuous: **regression** problem (value prediction)

# Hypothesis Space

Machine learning techniques require a **commitment** to a given **function space** $H$, inside which we look for the function that provides the **best approximation** for the training set.

$H$ reflects the way we are **modeling** data.

**Example:** Training set $= \{< 2, 3 >, < 3, 4 >\}$

$H =$ linear functions. We have an exact solution: $y = x + 1$.

Adding the instance $< 1, 0 >$, the model is not exact anymore.

Solution:

- we keep the model and content ourselves with approximate answers
- we change the model, for instance taking quadratic functions: $y = -x^2 + 6x - 5$.

# No Free Lunch

**Theorem (No Free Lunch)**
*Even after the observation of the frequent or constant conjunction of objects, we have no reason to draw any inference concerning any object beyond those of which we have had experience.*

**Figure 1:** David Hume.

Supposing that all hypotheses have the **same probability**, there is no reason to prefer one to the other: the choice of the hypothesis (or model) must be driven by **inductive biases**, otherwise, no learning is possible.

The learning biases are the set of assumptions that the model uses to predict outputs for new inputs.

# An Example: Occam's razor

*Pluralitas non est ponenda sine necessitate.*

(Plurality should not be posited without necessity.)

I.e., the **simplest solution** consistent with observation is to be preferred (*lex parsimoniae*).

**Figure 2:** William of Occam.

Occam's razor is used by decision trees.

# Features

Any information relative to a datum that describes some of its relevant properties is called a **feature**.

Features are the **input** of the learning process: learning is highly sensible to the **choice of features**.

Choosing good features is **difficult** (requires good domain knowledge)!

| **Medical diagnosis** | **User profiling** | **Weather forecasting** |
|---|---|---|
| Symptoms | Demographic data | Outlook |
| Patient condition | Personal interests | Temperature |
| Medical record | Social communities | Humidity |
| Result of exams | Lifestyle | Wind |
| ... | ... | ... |

**Table 1:** Examples of features.

# Components trained to learn

- **Machine Learning approach**: compute by hand good features, and apply a simple and well-understood learning algorithm.
- **Deep Learning approach**: supply raw data and let the machine the burden to synthesize good features (internal representations).

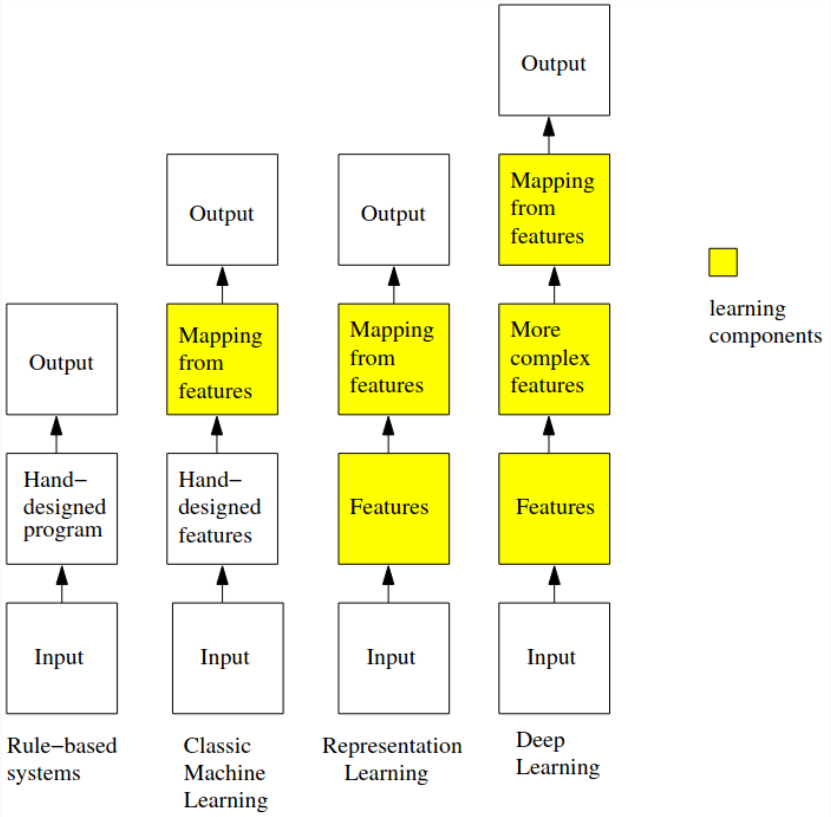



**Figure 3:** Components trained to learn. Image taken from [9].

# Relations between research areas

- **Knowledge-based systems:** take an expert, ask him how he solves a problem, and try to mimic his approach by means of logical rules.
- **Machine Learning**: take an expert, ask him which are the relevant features to solve a given problem, let the machine learn the mapping.
- **Deep Learning**: get rid of the expert.

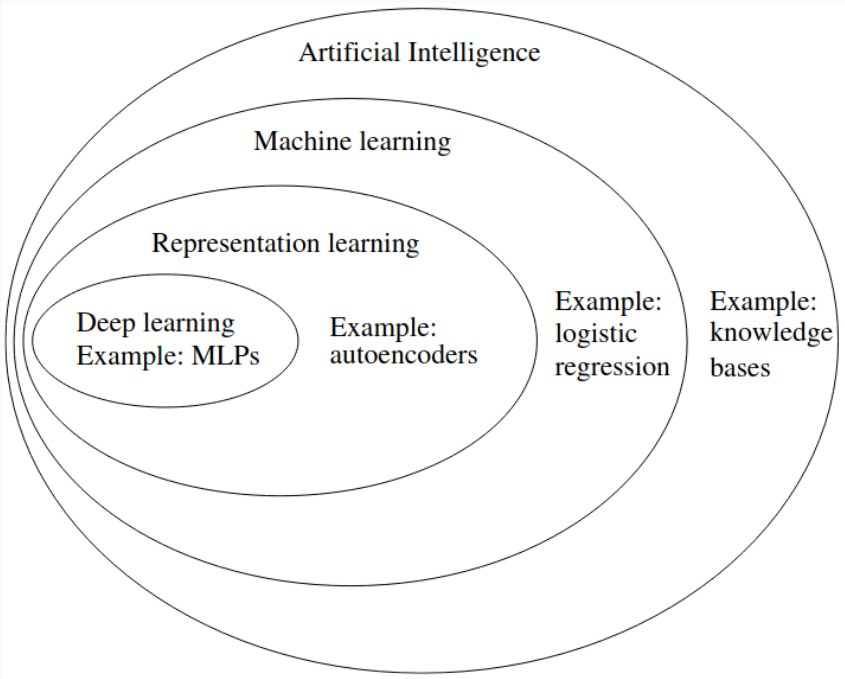



**Figure 4:** Relations between research areas. Image take from [9].

# Recap

Today we will focus on:

- the Machine Learning approach
- Supervised Learning (our training data is labeled)
- Classification Tasks

**Machine Learning approach**: *take an expert, ask him which are the relevant features to solve a given problem, let the machine learn the mapping.*

**Question:** what if we ask more than one expert?

**Problems:** each expert can consider different features to be more relevant than others, interpret numerical values in different ways, or even know more about some features and less about others.

# Decision Trees

# Decision Trees in the literature

The first decision tree algorithm, **ID3**, was first introduced by **Ross Quinlan** in 1979 [1, 11].
It only allowed discrete features.

**Leo Breinman** made it possible to work also with continuous features proposing **CART** in 1984 [3].

Quinlan refined his algorithm as **C4.5** in 1993 [12] introducing pruning and support for missing values.
Its last version **C5.0** is only commercially sold.

In 2001, Breinman proposed an ensemble approach to decision trees, namely **Random Forests**[2].

The problem of learning an optimal decision tree from a given dataset is known to be **NP-hard** [13]

**Figure 5:** Ross Quinlan.

**Figure 6:** Leo Breinman.

# Decision Trees: an example

A good day to play tennis?

$F : Outlook \times Temp \times Humidity \times Wind \rightarrow PlayTennis?$

- Every node tests a feature $X_i$
- Each branch corresponds to one of the possible **discrete** values of $X_i$
- Every leaf predicts the answer $Y$ (or a probability $P(Y|X)$)

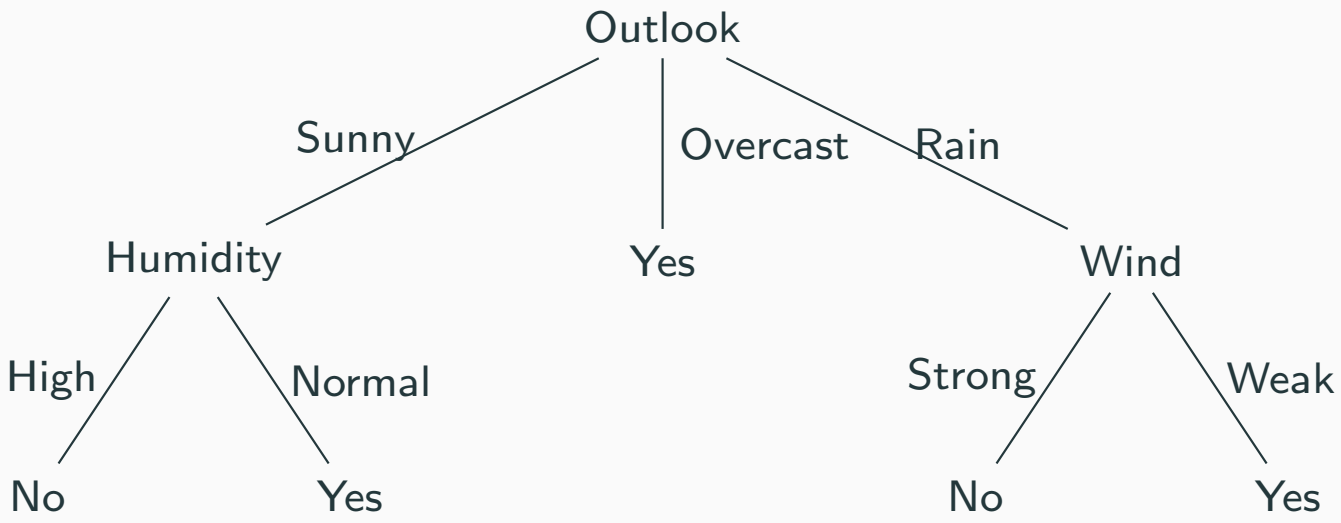



**Figure 7:** An example of a decision tree for classifying whether a person should play tennis based on outlook, humidity, wind, and temperature.

# How to build a Decision Tree

Problem configuration:

- **Input set** $X$
  every instance $x \in X$ is a vector of features of the following kind:

  $$< Outlook = rain, Temp = hot, Humidity = high, Wind = weak >$$

- **Target function** $f : X \rightarrow Y$
  $Y$ takes discrete values (booleans)
- **Hypothesis Space** $H = \{h | h : X \rightarrow Y\}$ (no restriction)
  - we try to model each $h \in H$ with a decision tree
  - each instance $x$ defines a path in the tree leading to a leaf labeled with $y$

# Play-tennis training set

| Outlook | Temp | Humidity | Wind | Play tennis |
|---|---|---|---|---|
| Sunny | Hot | High | Weak | **No** |
| Sunny | Hot | High | Strong | **No** |
| Overcast | Hot | High | Weak | **Yes** |
| Rain | Mild | High | Weak | **Yes** |
| Rain | Cool | Normal | Weak | **Yes** |
| Rain | Cool | Normal | Strong | **No** |
| Overcast | Cool | Normal | Strong | **Yes** |
| Sunny | Mild | High | Weak | **No** |
| Sunny | Cool | Normal | Weak | **Yes** |
| Rain | Mild | Normal | Weak | **Yes** |
| Sunny | Mild | Normal | Strong | **Yes** |
| Overcast | Mild | High | Strong | **Yes** |
| Overcast | Hot | Normal | Weak | **Yes** |
| Rain | Mild | High | Strong | **No** |

**Table 2:** Play-tennis training set.

# Top-down inductive construction

Main loop:

1. assign to the current node the "best" feature $X_i$
2. create a child node for every possible value of $X_i$
3. for every child node, if all the examples in the training set associated with the node have the same label $y$, mark the node (leaf) with label $y$, otherwise iterate from point 1

**Problem:** what is the "best" feature?

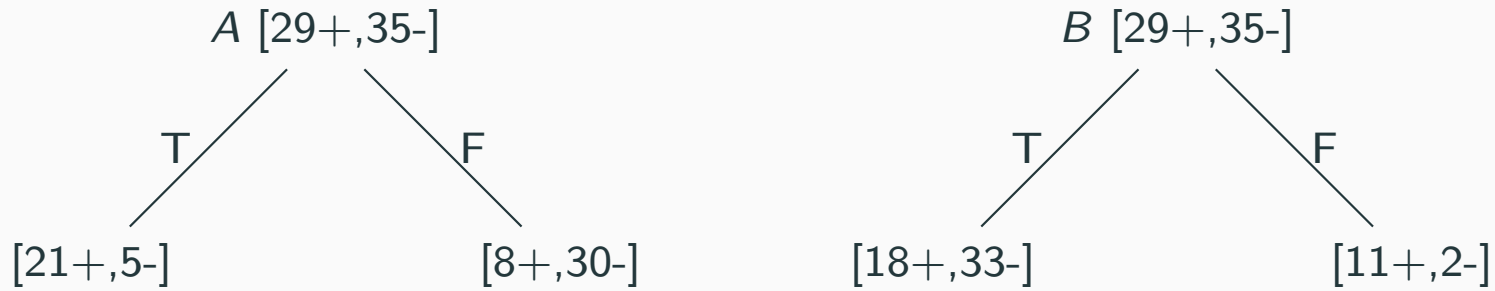



**Figure 8:** Splitting over different features $A$ and $B$. $n+$ represents the number of instances labeled as $y$, $m$- represents the number of instances not labeled as $y$. For simplicity, let's assume a binary classification problem.

# Entropy

The **entropy** $H(X)$ of a random variable $X$ is

$$H(X) = -\sum_{i=i}^{n} P(X = i) log_2 P(X = i)$$

where $n$ is the number of possible values of $X$.

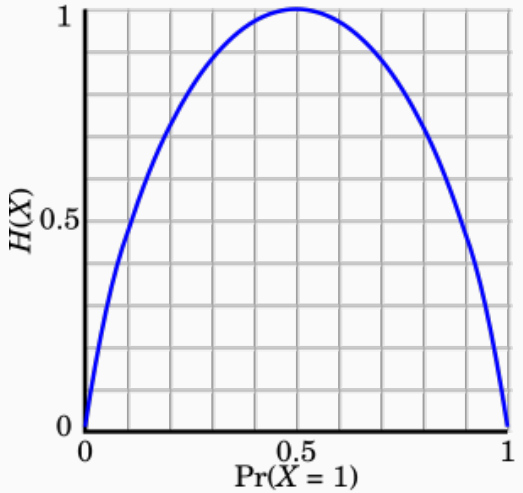



**Figure 9:** Entropy measures the **degree of impurity** of the information. It is maximal when $X$ is uniformly distributed over all values, and minimal (0) when it is concentrated on a single value.

# Entropy

Entropy is the average amount of **information** produced by a stochastic data source.

Information is associated with the **probability** of each data (the "surprise" carried by the event):

- an event with probability 1 carries no information: $I(1) = 0$
- given two independent events with probabilities $p_1$ and $p_2$ their joint probability is $p_1 p_2$ but the information acquired is the sum of the information of the two independent events, so

$$I(p_1 p_2) = I(p_1) + I(p_2)$$

It is hence natural to define

$$I(p) = -log(p)$$

# Entropy

**Example:** suppose to have $n$ events with the **same probability**. What is the entropy (i.e., the amount of information) carried out by each event?

$$
\begin{aligned}
H(X) &= -\sum_{i=1}^{n} P(X=i) log_2 P(X=i) \\
&= -\sum_{i=1}^{n} \frac{1}{n} log_2 \left(\frac{1}{n}\right) \\
&= \mathbf{log(n)}
\end{aligned}
$$

If events are not equiprobable, **we can do better!**

# Information gain

**Entropy of $X$:**

$$H(X) = -\sum_{i=i}^{n} P(X = i) log_2 P(X = i)$$

**Conditional Entropy of $X$ given a specific $Y = v$:**

$$H(X|Y = v) = -\sum_{i=1}^{n} P(X = i|Y = v) log_2 P(X = i|Y = v)$$

**Conditional Entropy of $X$ given $Y$** (weighted average over all $m$ possible values of $Y$):

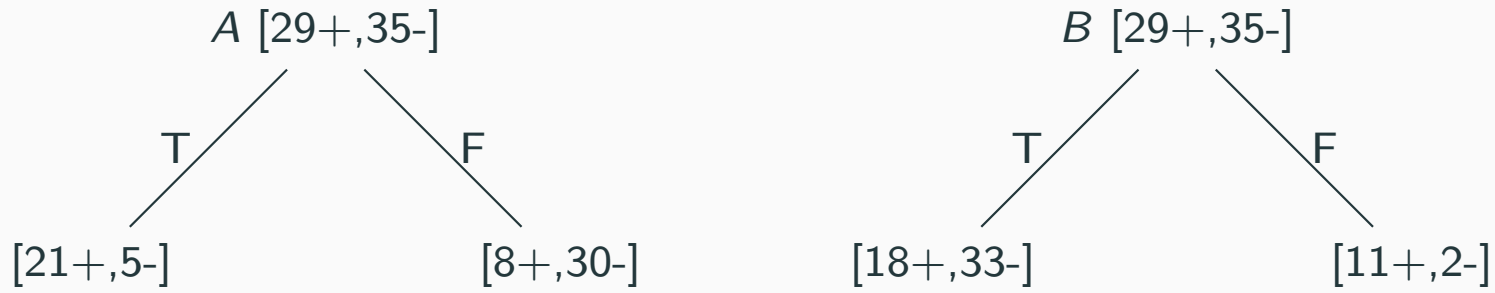
$$H(X|Y) = \sum_{v=1}^{m} P(Y = v) H(X|Y = v)$$

**Information Gain between $X$ and $Y$:**

$$I(X, Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$$

# Back to our example: what is the "best" feature?

Let us measure the entropy reduction of the target variable $Y$ due to some feature $X$, that is, the **information gain** $I(Y, X)$.




$$H(Y) = -(29/64) \cdot log_2(29/64) - (35/64) \cdot log_2(35/64) = .994$$

$$H(Y|A = T) = -(21/26) \cdot log_2(21/26) - (5/26) \cdot log_2(5/26) = .706$$

$$H(Y|A = F) = -(8/38) \cdot log_2(8/38) - (30/38) \cdot log_2(30/38) = .742$$

$$H(Y|A) = .706 \cdot 26/64 + .742 \cdot 38/64 = .726$$

$$I(Y, A) = H(Y) - H(Y|A) = .994 - .726 = .288$$

For $B$, we get $H(Y|B) = .872$ and $I(Y, B) = .122$. Hence, **A is better!**

# Gini's impurity

**Gini's impurity** measures the probability that a generic element gets misclassified according to the current classification.

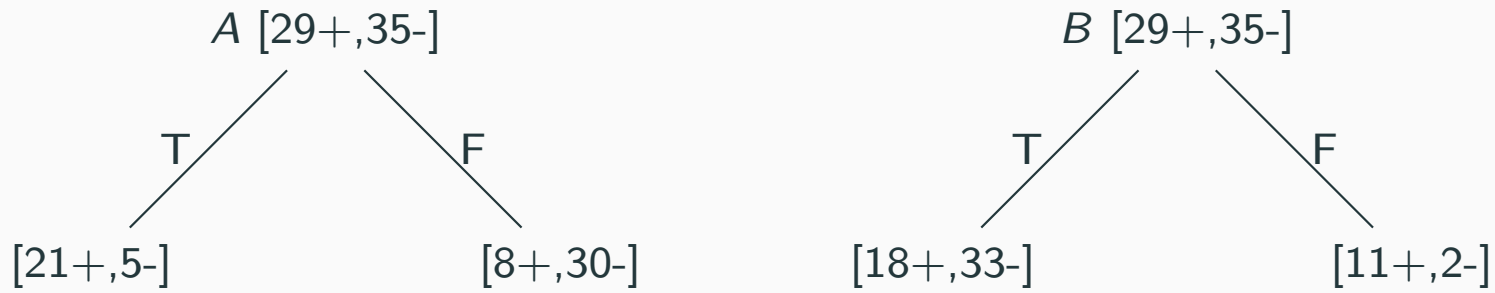
Given $m$ categories, let $f_i$ be the fraction of data with label $i$. This is equal to the probability that an input belongs to the category $i$. The probability of misclassifying it is hence $1 - f_i$, and its weighted average on all categories is just Gini't impurity, that is:

$$I_G(f) = \sum_{i=1}^{m} f_i(1 - f_i) = \sum_{1=1}^{m}(f_i - f_i^2) = \sum_{i=1}^{m} f_i - \sum_{i=1}^{m} f_i^2 = 1 - \sum_{i=1}^{m} f_i^2$$

This metric is applied to every child node, and values are summed in a weighted way (similarly to the definition of information gain) to get a measure of the **quality of a feature**.

# Back to our example: what is the "best" feature?

Let us evaluate the split using Gini's impurity.




For the feature A:

$$I_G(A = T) = 1 - (21/26)^2 - (5/26)^2 = .310$$
$$I_G(A = F) = 1 - (8/38)^2 - (30/38)^2 = .332$$
$$I_G(A) = .310 \cdot 26/64 + .332 \cdot 38/64 = .323$$

For the feature B:

$$I_G(B = T) = 1 - (18/51)^2 - (33/51)^2 = .456$$
$$I_G(B = F) = 1 - (11/13)^2 - (2/13)^2 = .260$$
$$I_G(B) = .456 \cdot 51/64 + .260 \cdot 13/64 = .416$$

Hence, **A is better (lower impurity)!**

# The continuous case

When features are continuous, we make decisions based on **thresholds:**


- we compare thresholds with **information gain**
- how to choose candidate thresholds?
  - **sample** at discrete intervals
  - **order** the test set w.r.t. the given feature and choose threshold at the **average** of two consecutive data

# Overfitting

Let us consider the error of the hypothesis $h$:

- on the training set, $error_{train}(h)$
- on the full data set $\mathcal{D}$, $error_{\mathcal{D}}(h)$

We say that $h$ **overfits** the training set if there exists another hypothesis $h'$ such that

$$error_{train}(h) < error_{train}(h')$$

but

$$error_{\mathcal{D}}(h) > error_{\mathcal{D}}(h')$$

**Problem:** we do not know $\mathcal{D}$!

Divide the available data into two disjoint sets:

- **training set** to be used to choose a candidate $h$
- **validation set** to be used to assess the accuracy of $h$

# Overfitting and model complexity

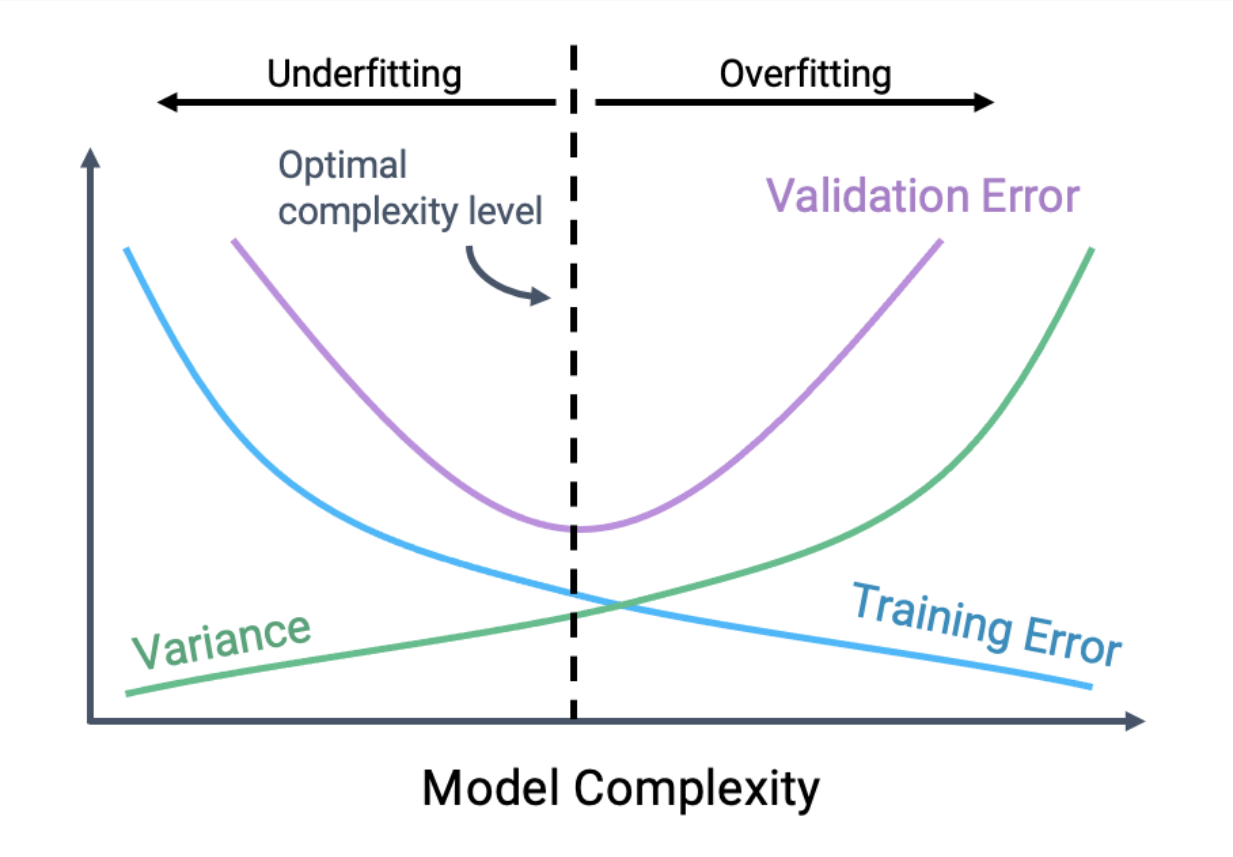



**Figure 13:** A visual representation of the relation between overfitting and model complexity: one should always aim for a model that can effectively approximate the training set while also generalizing to new data.

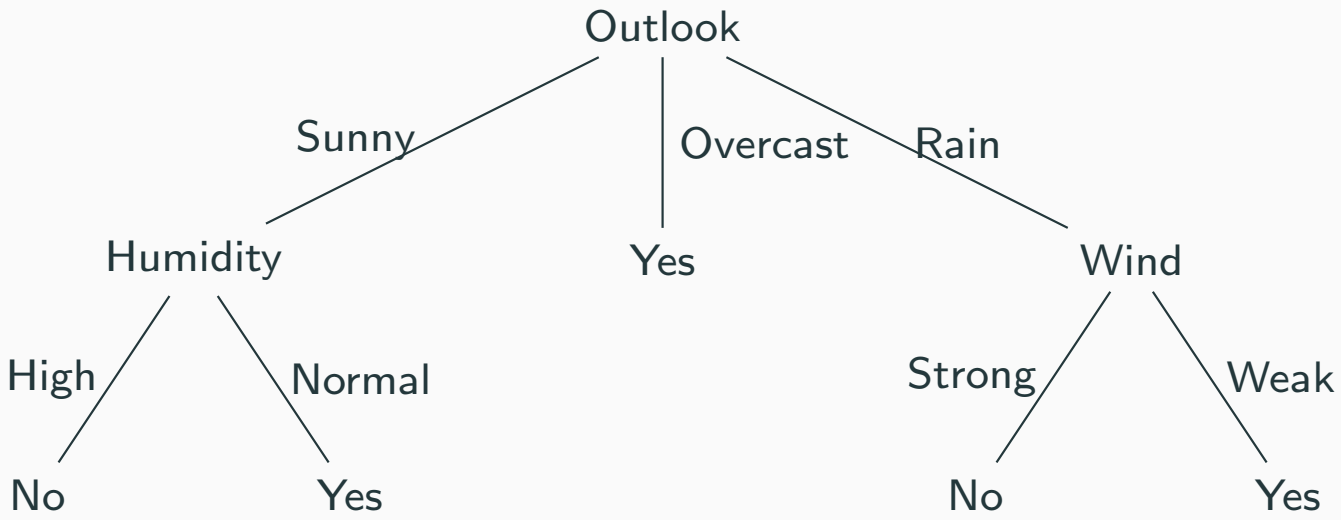
# Avoid overfitting with decision trees

Two approaches:

- **early stopping**: terminate the construction of the tree as soon as the classification improvement is not statistically significative (e.g., the information gain is below some threshold)
- **post-pruning**: develop the full and exact decision tree for the **training set**, and then proceed to backward prune it, repeating the following operation until further pruning does not improve accuracy:
  1. for any subtree, compute the impact of its removal on the classification accuracy on the **validation set**
  2. greedily perform pruning of the subtree that optimizes accuracy

# Decision Trees and Propositional Logic

A decision tree can be easily translated into a set of **logical formulas**!




Let's tie each possible value for each feature to a **propositional variable**:

$$\mathcal{P} = \{O_{Sunny}, O_{Overcast}, O_{Rain}, H_{High}, H_{Normal}, W_{Strong}, W_{Weak}\}$$

The decision tree above corresponds to the following set of formulas $\Gamma$:

$$\varphi_{Yes} := (O_{Sunny} \wedge H_{Normal}) \vee O_{Overcast} \vee (O_{Rain} \wedge W_{Weak})$$

$$\varphi_{No} := (O_{Sunny} \wedge H_{High}) \vee (O_{Rain} \wedge W_{Strong})$$

# Decision Trees and Propositional Logic

$\Gamma$:

$$\varphi_{Yes} := (O_{Sunny} \wedge H_{Normal}) \vee O_{Overcast} \vee (O_{Rain} \wedge W_{Weak})$$
$$\varphi_{No} := (O_{Sunny} \wedge H_{High}) \vee (O_{Rain} \wedge W_{Strong})$$

Each instance $x \in X$ can be translated into a **logical model** $m$ in a similar way, assigning a truth value to each propositional variable accordingly. Then, the classification problem can be solved simply by **checking** which formula $\varphi_y \in \Gamma$ is satisfied by the model $m$, i.e., $m \models \varphi_y$.

**Examples:**

- $x^9 = \{Sunny, Cool, Normal, Weak\} \rightarrow m^9 = \{T, F, F, F, T, F, T\}, m^9 \models \varphi_{Yes}$
- $x^{14} = \{Rain, Mild, High, Strong\} \rightarrow m^{14} = \{F, F, T, T, F, T, F\}, m^{14} \models \varphi_{No}$

**Note:** formulas are **mutually exclusive** (i.e., each model $m$ satisfies one and only one formula $\varphi_y \in \Gamma$).

# Pros and Cons of Decision Trees

Pros:

- **easy to understand**: simple logical rules, trees can be visualized
- little or **no data preprocessing** is required
- very **low prediction cost**
- can be used with both **discrete** and **continuous** features

Cons:

- high risk of **overfitting**
- selection of features quite **unstable**
- easy to build strongly **unbalanced** trees, especially if a class is dominant it can be useful to pre-balance the dataset

# Fuzzy Decision Trees

# Fuzzy Decision Trees in the literature

**Fuzzy decision trees** are an extension of traditional decision trees that incorporate **fuzzy logics** to handle uncertainty and imprecision in data, where decisions must account for various **degrees of truth**.

There are two main strategies for constructing a fuzzy decision tree:

- synthesize a tree from a user-fuzzified dataset (FuzzyID3 [15, 10], FuzzyC4.5 [5])
- fuzzify an already learned decision tree (FuzzyCART [14])

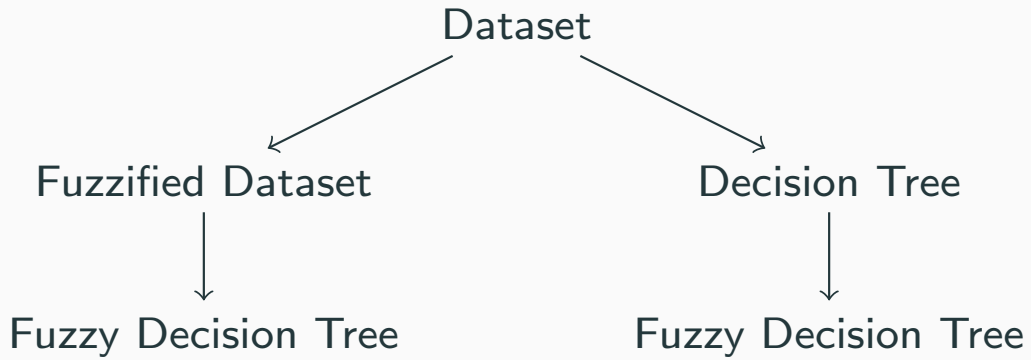



**Figure 14:** Strategies to construct a Fuzzy Decision Tree from the literature.

# Fuzzy Logics and Fuzzy Sets

Unlike classical binary logic where variables are either true or false, **fuzzy logics** allow for **degrees of truth** $t \in \mathbb{R}, t \in [0, 1]$.

Similarly, while crisp sets have clear boundaries (either an element belongs to the set or not), **fuzzy sets** allow **partial membership**.

Each variable is given a **membership value** ranging between 0 and 1, representing the degree to which it belongs to a **fuzzy set**.

**Example:** in a fuzzy set of $Temp_{Hot}$, a temperature of 30°C might have a membership value of 0.7, indicating it's somewhat hot.

# Membership Functions

**Membership functions** define how each point in the input space is mapped to a membership value between 0 and 1.

Common membership functions include:

- **Triangular** $\Delta_{x,y,z}$
- **Trapezoidal** $\mathcal{T}_{x,y,z,w}$
- **Gaussian** $\Phi_{\nu,\sigma}$.

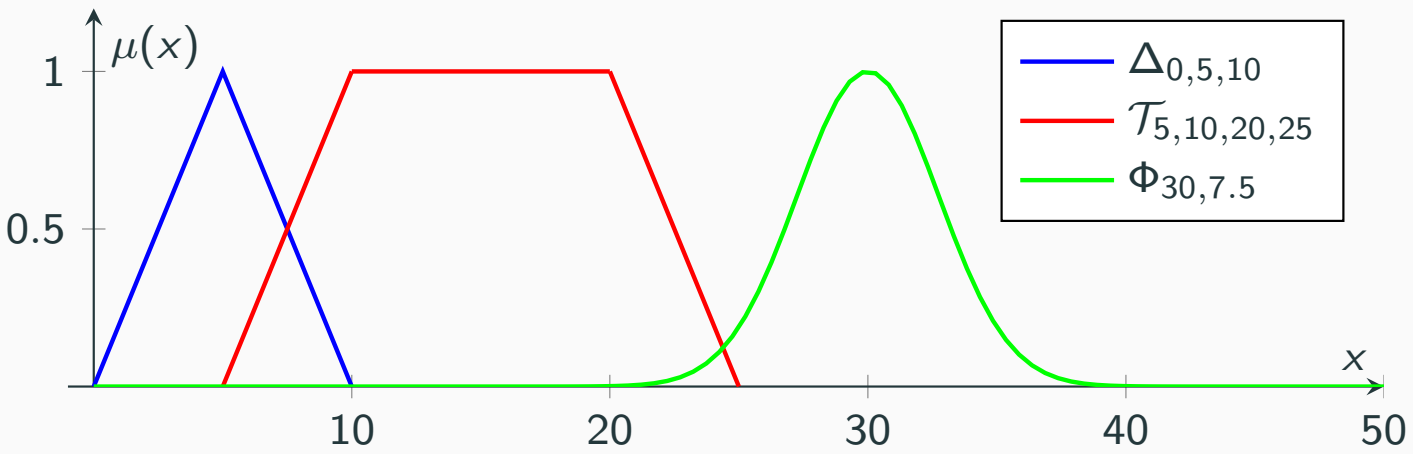



**Figure 15:** Example of different membership functions, normalized in [0,1].

# Play-tennis training set (revised)

| Temp (°C) | Humidity (%) | Wind (km/h) | Play tennis |
|---|---|---|---|
| 21.1 | 65 | 8.0 | Yes |
| 22.2 | 68 | 12.9 | Yes |
| 20.0 | 80 | 19.3 | No |
| 23.9 | 90 | 11.3 | No |
| 26.7 | 85 | 16.1 | No |
| 25.6 | 75 | 22.5 | No |
| 29.4 | 60 | 32.2 | Yes |
| 32.2 | 70 | 16.1 | Yes |

**Table 3:** Play-tennis training set revised (considering continuous rather than discrete features, and dropping the Outlook feature).

# Decision Trees: an example (revised)

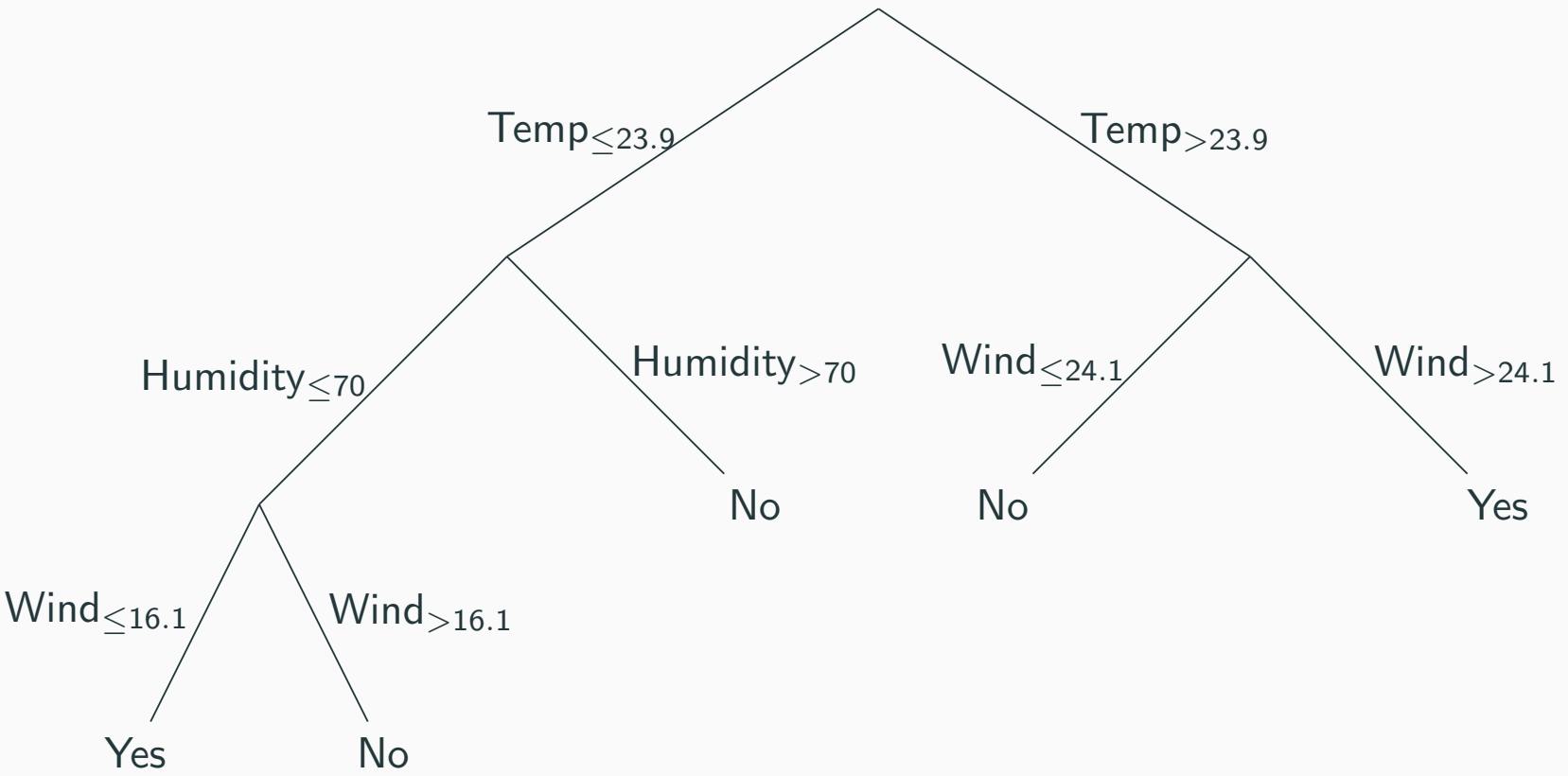



**Figure 16:** An example of a decision tree for classifying whether a person should play tennis based on temperature, humidity, and wind conditions.

# Fuzzy Decision Trees: an example

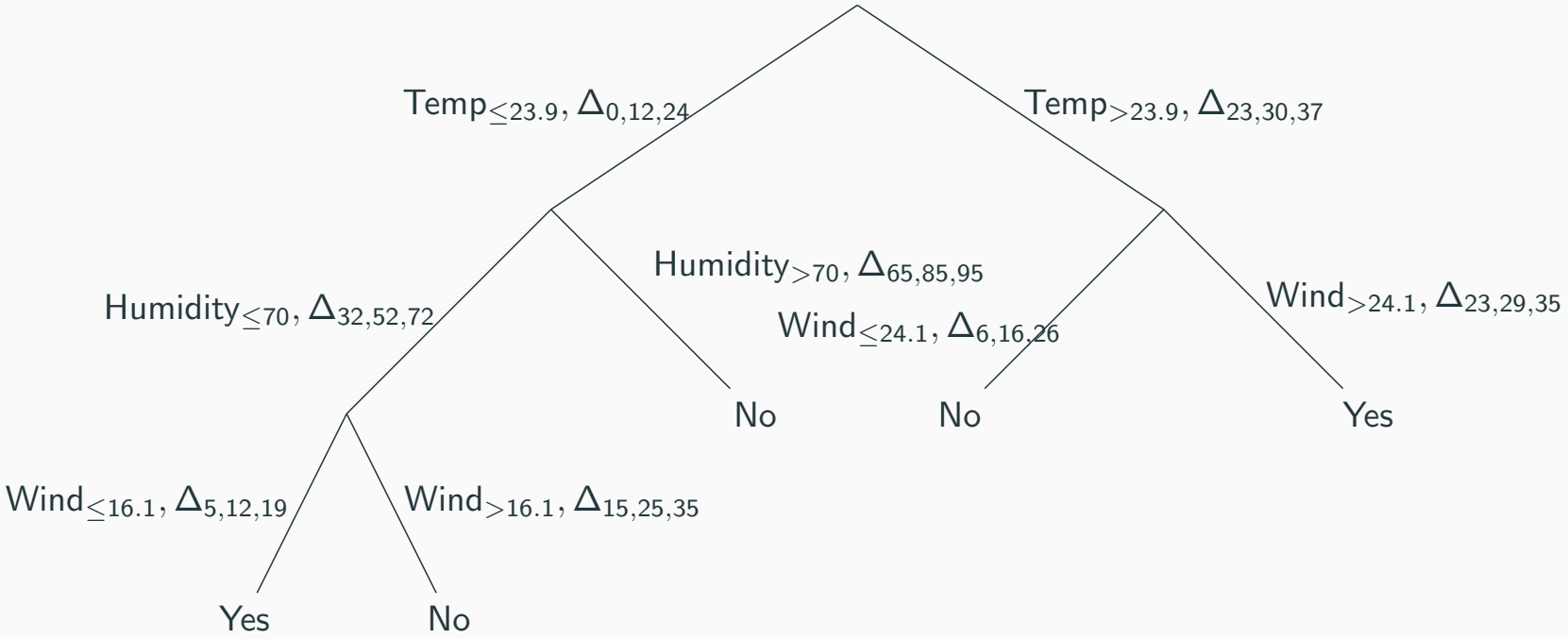



**Figure 17:** An example of a fuzzy decision tree for classifying whether a person should play tennis based on temperature, humidity, and wind conditions. Specifically, the decision tree in Fig. 16 has been fuzzified by learning the parameters for the triangular membership functions.

# How to interpret a Fuzzy Decision Tree

We have two main strategies for interpreting a fuzzy decision tree:

- Take at each step the decision that gives the higher degree of truth
- Fully evaluate all branches using the **t-norm** $\cdot$ associated with the fuzzy logic, and choosing the branch with the higher degree of truth

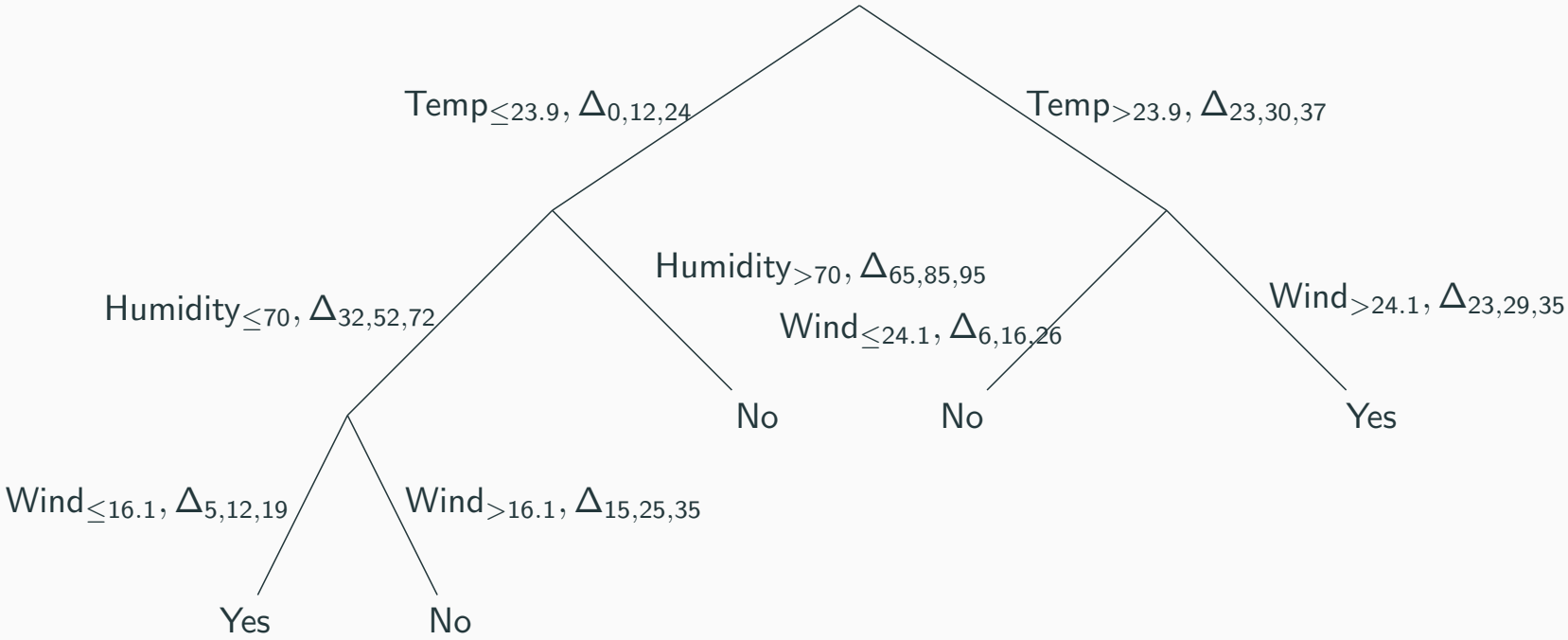
Fuzzy logics offer different t-norms, given two real values $x, y \in [0, 1]$:

- in **Gödel logic**, $x \cdot y = min\{x, y\}$
- in **Łukasiewicz logic**, $x \cdot y = max\{0, x + y - 1\}$
- in **Product logic**, $x \cdot y = xy$

**Note:** one could also think of a third, hybrid strategy, e.g., iteratively evaluating a subtree of a specified height, and then choosing the branch with the higher degree of truth.

# Fuzzy Decision Trees and Fuzzy Propositional Logic

As before, we can translate a fuzzy decision tree into a set of formulas $\Gamma$:




$$\varphi_{Yes}(x) := (Temp_{\leq 23.9} \cdot Humidity_{\leq 70} \cdot Wind_{\leq 16.1}) \vee (Temp_{>23.9} \cdot Wind_{>24.1})$$

$$\varphi_{No}(x) := (Temp_{\leq 23.9} \cdot Humidity_{\leq 70} \cdot Wind_{>16.1}) \vee$$
$$(Temp_{\leq 23.9} \cdot Humidity_{>70}) \vee (Temp_{>23.9} \cdot Wind_{\leq 24.1})$$

# Fuzzy Decision Trees and Fuzzy Propositional Logic

$\Gamma$:

$$\varphi_{Yes}(x) := (Temp_{\leq 23.9} \cdot Humidity_{\leq 70} \cdot Wind_{\leq 16.1}) \vee (Temp_{>23.9} \cdot Wind_{>24.1})$$
$$\varphi_{No}(x) := (Temp_{\leq 23.9} \cdot Humidity_{\leq 70} \cdot Wind_{>16.1}) \vee (Temp_{\leq 23.9} \cdot Humidity_{>70}) \vee (Temp_{>23.9} \cdot Wind_{\leq 24.1})$$

Each instance $x \in X$ can be translated in a logical model $m$ in assigning a truth value to each propositional variable according to the respective membership function. Then, the classification problem can be solved simply by checking which formula $\varphi_y \in \Gamma$ gives the highest truth value.
**Example:**

- $x^7 = \{29.4, 60, 32.2\}$
- $\varphi_{Yes}(x^7) := (0 \cdot 0.4 \cdot 0) \vee (0.9 \cdot 0.25)$
- $\varphi_{No}(x^7) := (0 \cdot 0.4 \cdot 0.1) \vee (0, 0) \vee (0.9, 0)$
- Regardless of which t-norm we are using, $\varphi_{Yes}(x^7) > \varphi_{No}(x^7)$

# Pros and Cons of Fuzzy Decision Trees

Pros:

- better treatment of uncertainty in the data
- better treatment of unclear boundaries between data
- we can postpone judgment on splitting (what if the "best" feature was not the best?)

Cons:

- common membership functions can be too drastic on branches (a membership value of 0 "kills" the branch)
- membership functions give a unified judgment: it is like we are only considering one expert or a decision agreed on by a group of experts
- **what if we want to consider each expert's opinion separately?**

# Many-Expert Decision Trees

# Modeling Many-Expert situations

We would like to:

- represent degrees of truths representing opinions of various experts (i.e., some could be non-comparable $\rightarrow$ we need a **partial order**)
- represent **hierarchies** between experts (i.e., some experts may be specialized in some areas and their opinions should be more valuable)
- find a good way to aggregate values on different features (similarly to the **t-norm** in the fuzzy case)

In his seminal work, **Melvin Fitting** proposed the use of **Heyting Algebras** (i.e., the algebraic counterpart of **Intuitionistic Logic**) to model these kinds of situations [7, 8].

**Problem:** intuitionistic logic does not generalize all fuzzy logics (only Gödel's logic)!

**Figure 18:**
Melvin Fitting

# Partial Taxonomy of well-known Many-Valued Logics

**Note:** fuzzy logics are all many-valued logics in which the corresponding algebra is a chain (i.e., the order is total).

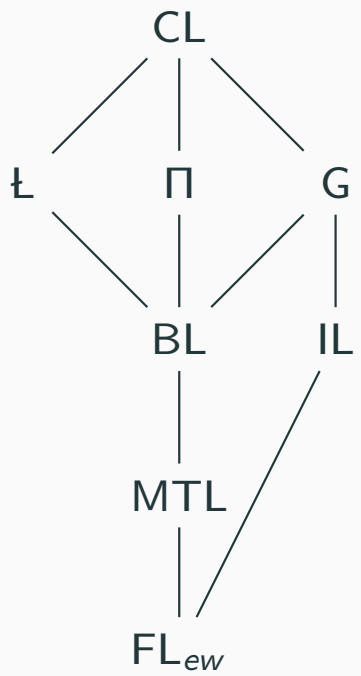



**Figure 19:** Some families of Many-Valued logics: CL is the classical logic; Ł, Π and G are Łukasiewich, Product and Göedel logics all belong to the family of Base Fuzzy Logics (BL); IL is the Intuitionistic Logic; FL$_{ew}$ is the first logic generalizing both BL and IL logics.

# $\mathbf{FL}_{ew}$-Algebra to the rescue

An **$\mathbf{FL}_{ew}$Algebra** is an algebra $\mathfrak{A} = (A, \vee, \wedge, \cdot, \rightarrow, \bot, \top)$, where

- $(A, \vee, \wedge, \bot, \top)$ is a **bounded lattice** with top element $\top$ and bottom element $\bot$ $\rightarrow$ the **partial order**
- $(A, \cdot, \top)$ is a **commutative monoid** $\rightarrow$ the **t-norm**
- The **residuation property** holds: $x \cdot y \preceq z$ iff $x \preceq y \rightarrow z$

**Property**: $x \cdot y \preceq x \wedge y$

If $\mathfrak{A}$ is also a **distributive lattice** (i.e., $\cdot$ and $\wedge$ are the same operation, entailing distributivity of $\wedge$ over $\vee$), $\mathfrak{A}$ is called an **Heyting algebra**.

One can also define another commutative monoid $(A, +, \bot)$ (**co-t-norm**).

# $\mathbf{FL}_{ew}$-Algebras: an example

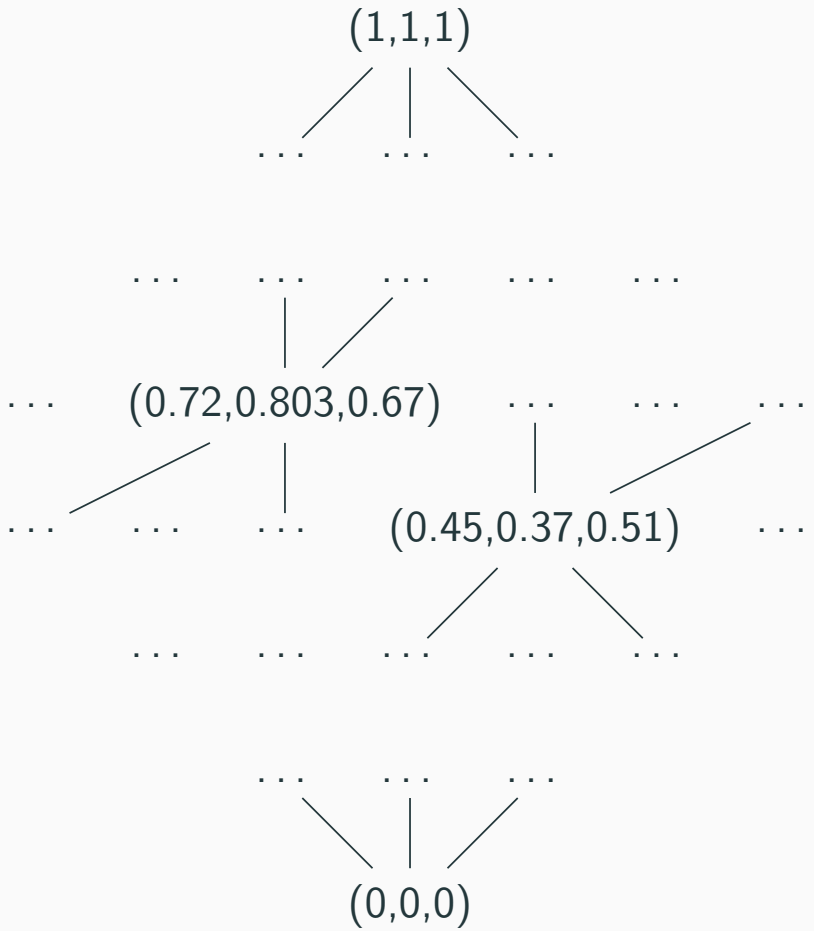



**Figure 20:** Lattice structure for $\mathfrak{A}$ with 3-tuples.

$\mathfrak{A} = (A, \vee, \wedge, \cdot, \rightarrow, (0, \ldots, 0), (1, \ldots, 1))$ is an **$\mathbf{FL}_{ew}$-Algebra** where:

- $(A, \vee, \wedge, (0, \ldots, 0), (1, \ldots, 1))$ is a **bounded lattice** with top element $(1, \ldots, 1)$ and bottom element $(0, \ldots, 0)$ and $\vee$ (resp. $\wedge$) defined as the $\vee$ (resp. $\wedge$) between each member of each tuple.
- $(L, \cdot, (1, \ldots, 1))$ is a **commutative monoid** such that, given two $n$-tuples, returns the **elementwise product** between them.

# How to build a Many-Experts Decision Tree

- Fix a number $n$ of **experts**
- Define an **FL$_{ew}$-Algebra** $\mathfrak{A} = (A, \vee, \wedge, \cdot, \rightarrow, (0, \ldots, 0), (1, \ldots, 1)$ where $A$ is the set of all $n$-tuples, and $\vee$ and $\wedge$ are defined as:

$$(x_1, \ldots, x_n) \vee (y_1, \ldots, y_n) = (x_1 \vee y_1, \ldots, x_1 \vee y_n)$$

$$(x_1, \ldots, x_n) \wedge (y_1, \ldots, y_n) = (x_1 \wedge y_1, \ldots, x_1 \vee y_n)$$

- Define a proper **t-norm** for the algebra (e.g., the elementwise product between two $n$-tuples)
- Divide the available data into $n + 2$ disjoint sets:
  - a **training set** and a **validation set** to build a decision tree
  - $n$ sets to learn the parameters for a specified **membership function** (e.g., Gaussian) for each expert for each decision on each feature

# Different Membership Functions for different Experts

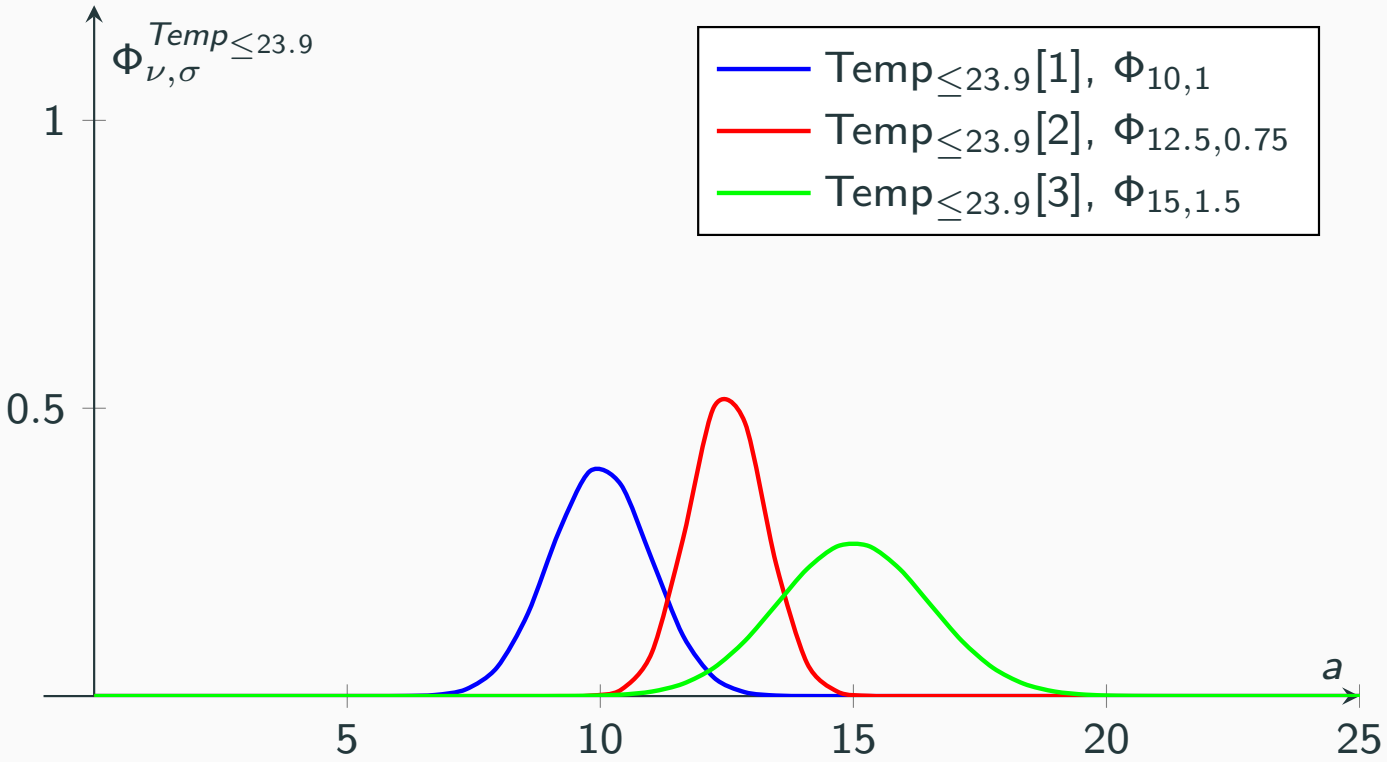



**Figure 21:** Example of different membership functions representing 3 different experts' opinions using the strategy above (i.e., learning the parameters for a Gaussian membership function) for $Temp_{\leq 23.9}$.

# Many-Experts Decision Trees: an example

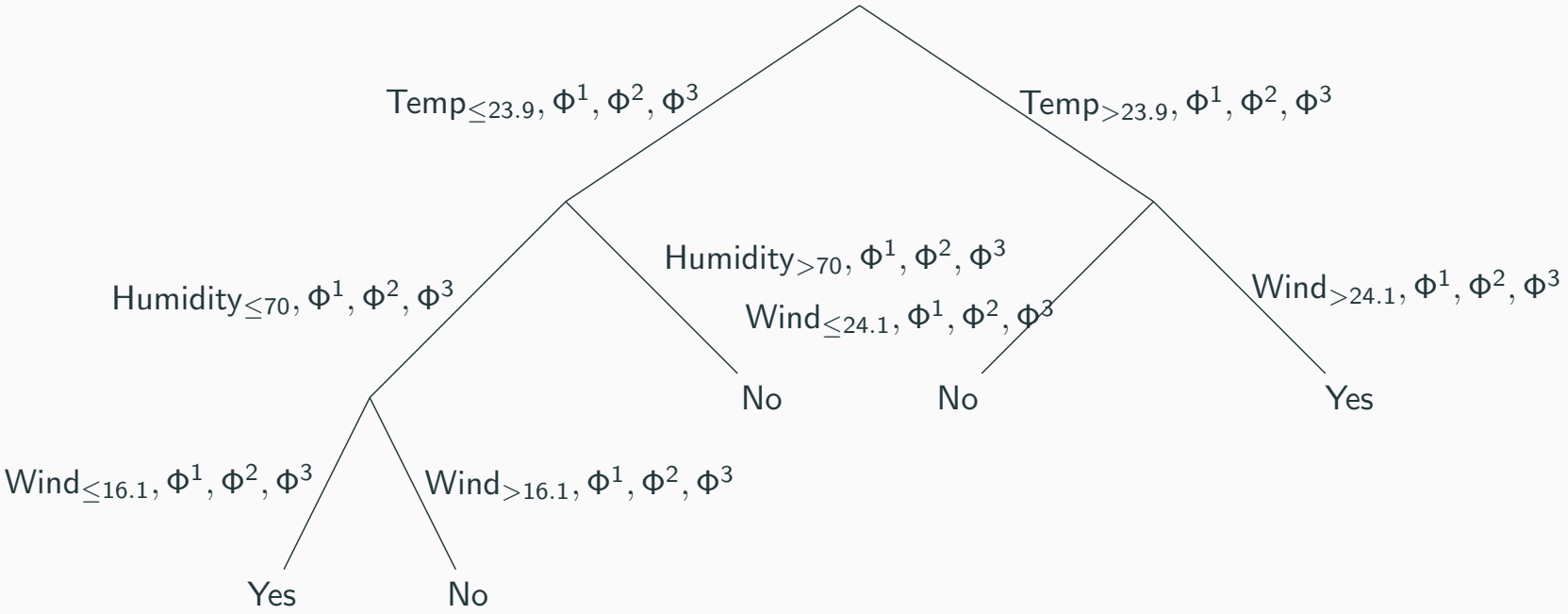



**Figure 22:** An example of a many-expert decision tree for classifying whether a person should play tennis based on temperature, humidity, and wind conditions, obtained from the decision tree in Fig. 7 learning the parameters of a Gaussian membership function for each decision on each feature for each expert.

# How to interpret a Many-Expert Decision Tree

We can translate a many-expert decision tree to a set of formulas:

$$\varphi_{Yes}(x) := (Temp_{\leq 23.9} \cdot Humidity_{\leq 70} \cdot Wind_{\leq 16.1}) \vee (Temp_{>23.9} \cdot Wind_{>24.1})$$

$$\varphi_{No}(x) := (Temp_{\leq 23.9} \cdot Humidity_{\leq 70} \cdot Wind_{>16.1}) \vee (Temp_{\leq 23.9} \cdot Humidity_{>70}) \vee (Temp_{>23.9} \cdot Wind_{\leq 24.1})$$

**t-norm**: given two $n$-tuples, return the **elementwise product** between them, i.e.,

$$\cdot((x_1, \ldots, x_n), (y_1, \ldots, y_n)) = (x_1 \times y_1, \ldots, x_n \times y_n)$$

**Example:**

- $x$ such that $Temp_{\leq 23.9}(x) = (0, 0, 0)$, $Temp_{>23.9}(x) = (0.8, 0.5, 0.6)$, $Wind_{\leq 24.1}(x) = (0.5, 0.5, 0.6)$ and $Wind_{>24.1}(x) = (0.4, 0.3, 0.5)$
- $\varphi_{Yes}(x) := (0.4, 0.3, 0.5)$
- $\varphi_{No}(x) := (0.32, 0.15, 0.3)$
- Since $(0.4, 0.25, 0.36) \succ (0.32, 0.15, 0.3)$, we can play!

# Conclusions and Future Work

# Conclusions and Future Work

Taking inspiration from the literature on fuzzy decision trees, we proposed a novel approach, namely **many-experts decision trees**, to take different experts' opinions on the same features into account.

We think the proposed approach to be quite **general**, as it extends the fuzzy decision tree approach (i.e., one only has to consider one expert and use the t-norm of the chosen fuzzy logic) and highly **parametrizable** (using the elementwise product as a t-norm was merely an example).

On the other hand, we think that a **higher number of data** would be needed for a good fit of these models when compared to their crisp or fuzzy counterparts, and we also expect a **higher complexity** both in the learning and the classification steps.

We think that using a **hybrid technique** when classifying new instances (i.e., **exploring subtrees**) might be a good tradeoff.

# Conclusions and Future Work

Looking at the future of this research, we first and foremost want to **implement** this algorithm to better understand its properties and address its performance when compared to classical and fuzzy approaches.

This will be done as part of an open-source project aimed at representing, reasoning, and learning from structured and unstructured data, namely **Sole.jl** (**S**ymb**o**lic **le**arning), developed in the Julia programming language.

Furthermore, we want to extend this kind of approach to the **modal** case, allowing to work also with non-tabular data (e.g., spatio-temporal data, audio, images, . . . ), **towards many-expert modal decision trees!**

# Questions?

# References i



Discovering rules by induction from large collections of examples.
**Expert systems in the micro electronics age, 1979.**


L. Breiman.
**Random forests.**
*Machine Learning*, 45:5–32, 2001.


L. Breiman.
**Classification and regression trees.**
Routledge, 2017.


A. Brunello, G. Sciavicco, and I. Stan.
**Interval temporal logic decision tree learning.**
In *Proc. of the 16th European Conference on Logics in Artificial Intelligence (JELIA)*, volume 11468 of *LNCS*, pages 778–793. Springer, 2019.

# References ii



M. E. Cintra, M. C. Monard, and H. A. Camargo.
**FuzzyDT—a fuzzy decision tree algorithm based on C4.5.**
In *Proc. of the Brazilian Congress on Fuzzy Systems*, pages 199–211, 2012.


D. Della Monica, G. Pagliarini, G. Sciavicco, and I. Stan.
**Decision trees with a modal flavor.**
In *Proc. of the 21st International Conference of the Italian Association for Artificial Intelligence (AIxIA)*, number 13796 in LNCS, pages 47 – 56. Springer, 2023.


M. Fitting.
**Many-valued modal logics.**
*Fundamenta Informaticae*, 15(3-4):235–254, 1991.

# References iii



M. Fitting.
**Tableaus for many-valued modal logic.**
*Studia Logica*, 55(1):63–87, 1995.


I. Goodfellow.
**Deep learning, 2016.**


C. Z. Janikow.
**Fuzzy decision trees: issues and methods.**
*IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 28(1):1–14, 1998.


J. Quinlan.
**Induction of decision trees.**
*Machine learning*, 1:81–106, 1986.

# References iv


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

R. Rivest.
**Learning Decision Lists.**
*Machine Learning*, 2(3):229–246, 1987.

J.-S. R.Jang.
**Structure determination in fuzzy modeling: a fuzzy CART approach.**
In *Proceedings of 1994 IEEE 3rd international fuzzy systems conference*, pages 480–485. IEEE, 1994.


M. Umanol, H. Okamoto, I. Hatono, H. Tamura, F. Kawachi, S. Umedzu, and J. Kinoshita.
**Fuzzy decision trees by fuzzy ID3 algorithm and its application to diagnosis systems.**
In *Proc. of 3rd IEEE International Fuzzy Systems Conference*, pages 2113–2118. IEEE, 1994.