Deriving Rules: In general, we can derive several classification rules r_i from a leaf of a decision tree, namely one for each possible value v_i, $1 \leq i \leq n$, of the target attribute Z:

$$r_i \;=\; A_1 = w_1 \wedge \ldots \wedge A_m = w_m \overset{s_i, p_i}{\Longrightarrow} Z = v_i,$$

where s_i is the support and p_i is the confidence of the rule r_i.

Then, $\sum_{i=1}^{n} p_i = 1$, and the confidences $p_i = s_i/s$ correlate with the supports s_i, where $s = \sum_{i=1}^{n} s_i$.

Of course, if there are only two possible values v_1 and v_2 for Z, i.e., $n = 2$, then the two rules r_1 and r_2 are equivalent, and we only list the rule with the higher confidence.

In general, we would like to measure the quality of a leaf of a decision tree – or the collection of the classification rules r_i from the corresponding branch – depending on the confidences p_i using a function $\varepsilon(p_1, \ldots, p_n)$.

We can then define the entropy of the whole decision tree as the weighted average entropy of the tables T_i, $1 \leq i \leq n$, corresponding to the leaves:

$$\mathit{entropy}_Z(T_1, \ldots, T_n) \;=\; \sum_{i=1}^{n} \frac{|T_i|}{|T|} \cdot \mathit{entropy}_Z(T_i).$$

In our example with four sub–tables, and the target attribute $Z = \mathit{Def}$:

$$T_1 = \sigma_{\mathit{Married}=\mathit{yes} \wedge \mathit{PrevDef}=\mathit{no}}(T), \;\; T_2 = \sigma_{\mathit{Married}=\mathit{no} \wedge \mathit{PrevDef}=\mathit{no}}(T),$$
$$T_3 = \sigma_{\mathit{Married}=\mathit{yes} \wedge \mathit{PrevDef}=\mathit{yes}}(T), \;\; T_4 = \sigma_{\mathit{Married}=\mathit{no} \wedge \mathit{PrevDef}=\mathit{yes}}(T),$$

$$\begin{aligned} \mathit{entropy}_Z(T_1, \ldots, T_4) \;=\; & \\ & \tfrac{11}{20} \cdot \mathit{entropy}_Z(T_1) + \tfrac{3}{20} \cdot \mathit{entropy}_Z(T_2) + \\ & \tfrac{3}{20} \cdot \mathit{entropy}_Z(T_3) + \tfrac{3}{20} \cdot \mathit{entropy}_Z(T_4). \end{aligned}$$

Since the tables T_2, T_3, and T_4 are homogeneous in Z, their entropy is 0. Below, we will explore some suitable entropy functions, which satisfy the permutation, merging, and averaging conditions.

Standard Entropy Function from Information Theory

We define

$$\varepsilon(p) \;=\; \begin{cases} -p \cdot \log_2 p, & \text{for } 0 < p \leq 1, \\ 0, & \text{for } p = 0. \end{cases}$$

The function is continuous in the closed interval $[0, 1]$, since

$$\lim_{p \to 0+0} -p \cdot \log_2 p \;=\; 0.$$

Notice, that $-\log_2 p \;=\; \log_2 1/p$.

We use the following function from information theory:

$$\varepsilon(p_1, \ldots, p_n) \;=\; \sum_{i=1}^{n} \varepsilon(p_i).$$

where $0 \leq p_i \leq 1$, for $1 \leq i \leq n$.

We will see, that ε satisfies the merging and averaging conditions.

For a table T with the attribute set U and an attribute $A \in U$,
let $T_A = \Pi_A(T)$ be the domain of A in T, and let $v \in T_A$ be a value.

- Let $|\sigma_{A=v}(T)|$ be the number of tuples from T with the value v for A.
- Let $p_{A=v}(T)$ be the relative frequency of the value v for A:

  $$p_{A=v}(T) = \frac{|\sigma_{A=v}(T)|}{|T|}.$$

If $T_A = \{ v_1, \ldots, v_n \}$ and $p_i = p_{A=v_i}(T)$, then

$$\sum_{i=1}^{n} p_i = 1,$$

and we define the entropy of A in T:

$$entropy_A(T) = \varepsilon(p_1, \ldots, p_n).$$

If all p_i are positive, then $\varepsilon(p_1, \ldots, p_n) = \sum_{1 \leq i \leq n} -p_i \cdot \log_2 p_i$.

Example (Creditworthiness)

- For $A = \mathit{PrevDef}$ and $A = \mathit{Def}$, we get $|\sigma_{A=no}(T)| = 14$ and $|\sigma_{A=yes}(T)| = 6$, and thus:

  $$p_{A=no}(T) = \tfrac{14}{20} = 70\%, \quad p_{A=yes}(T) = \tfrac{6}{20} = 30\%,$$
  $$\mathit{entropy}_A(T) = \varepsilon(\tfrac{14}{20}, \tfrac{6}{20}) = -\tfrac{14}{20} \cdot \log_2 \tfrac{14}{20} - \tfrac{6}{20} \cdot \log_2 \tfrac{6}{20} = 0.881.$$

- For $A = \mathit{Married}$, we get

  $$p_{A=no}(T) = \tfrac{6}{20} = 30\%, \quad p_{A=yes}(T) = \tfrac{14}{20} = 70\%,$$
  $$\mathit{entropy}_A(T) = \varepsilon(\tfrac{6}{20}, \tfrac{14}{20}) = 0.881.$$

- For $A = \mathit{Income}$, we get

  $$p_{A<30}(T) = \tfrac{5}{20} = 25\%, \quad p_{A\geq 30}(T) = \tfrac{15}{20} = 75\%,$$
  $$\mathit{entropy}_A(T) = \varepsilon(\tfrac{5}{20}, \tfrac{15}{20}) = -\tfrac{5}{20} \cdot \log_2 \tfrac{5}{20} - \tfrac{15}{20} \cdot \log_2 \tfrac{15}{20} = 0.811.$$

- We observe, that the entropy $\varepsilon(\tfrac{5}{20}, \tfrac{15}{20})$ is slightly lower than $\varepsilon(\tfrac{14}{20}, \tfrac{6}{20}) = \varepsilon(\tfrac{6}{20}, \tfrac{14}{20})$.

- Moreover, in the sub–tables we get:

  $$\begin{aligned} \mathit{entropy}_{\mathit{Def}}(\sigma_{\mathit{PrevDef=yes}}(T)) &= \varepsilon(\tfrac{3}{6}, \tfrac{3}{6}) \;=\; 1, \\ \mathit{entropy}_{\mathit{Def}}(\sigma_{\mathit{PrevDef=no}}(T)) &= \varepsilon(\tfrac{3}{14}, \tfrac{11}{14}) \;=\; 0.75. \end{aligned}$$

  I.e., the first one is higher than $\mathit{entropy}_{\mathit{Def}}(T) = 0.881$, while the second one is lower.

  The weighted average entropy of the sub–tables is

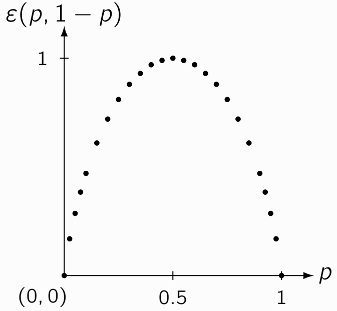
  $$\mathit{entropy}_{\mathit{Def}}(\sigma_{\mathit{PrevDef=yes}}(T), \sigma_{\mathit{PrevDef=no}}(T)) = \\ 6/20 \cdot 1 + 14/20 \cdot 0.75 = 0.825,$$

  which is lower than the entropy 0.881 of T.

Later, we will show that the weighted average entropy of the sub–tables is always lower than or equal to the entropy of a table.

Excursus: Analysis of Entropy Function ε

For $n = 2$, and $p_1 = p$ and $p_2 = 1 - p$, the curve $f(p) = \varepsilon(p, 1 - p)$ is concave with its maximum 1 in $p = 0.5$:



The maximal entropy $entropy_A(T)$ is obtained, if the two tables $\sigma_{A=v_i}(T)$, $i = 1, 2$, have the same size; the maximal entropy is 1, here.

For arbitrary n, if all values in $v \in T_A$ have the same relative frequency $1/n$, then

$$entropy_A(T) = n \cdot (1/n \cdot \log_2 n) = \log_2 n.$$

n	$p_1, \ldots, p_n$	$entropy_A(T)$
1	1	0
2	0.5, 0.5	1
3	0.33, 0.33, 0.33	1.58
4	0.25, . . . , 0.25	2
8	0.125, . . . , 0.125	3

Thus, doubling n adds 1 to the entropy here.

Splitting p into $p/2$ and $p/2$ adds p to the entropy:

$$\varepsilon(p/2, p/2) = 2 \cdot \left(p/2 \cdot \log_2 \frac{1}{p/2} \right) = p \cdot (1 + \log_2 1/p) = \varepsilon(p) + p.$$

Let $p_\oplus = sum_{1 \le i \le n}\, p_i = \sum_{1 \le i \le n} p_i$ and $p_\varnothing = avg_{1 \le i \le n}\, p_i = 1/n \cdot p_\oplus$.

The sum of the function values $\varepsilon(p_i),\ 1 \le i \le n$, is greater than or equal to the function value $\varepsilon(p_\oplus)$ of the sum $p_\oplus$ of the arguments p_i:
$sum_{1 \le i \le n}\, \varepsilon(p_i) \ \ge\ \varepsilon(sum_{1 \le i \le n}\, p_i)$, i.e.

$$\begin{aligned} \varepsilon(p_1, \dots, p_n) &= \sum_{1 \le i \le n} \varepsilon(p_i) \ =\ \sum_{1 \le i \le n} -p_i \cdot \log_2 p_i \\ &\ge -p_\oplus \cdot \log_2(p_\oplus) \ =\ \varepsilon(p_\oplus). \end{aligned}$$

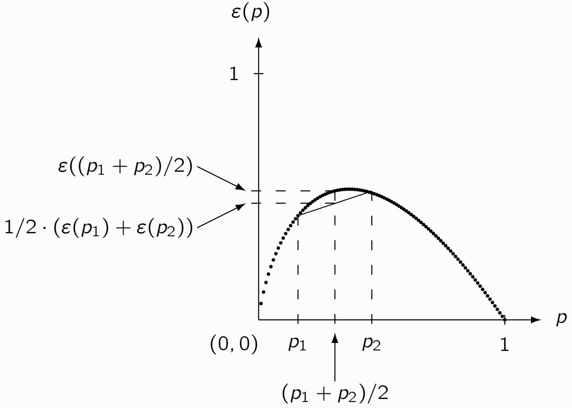
ε is a *concave* function:
The average of the function values $\varepsilon(p_i),\ 1 \le i \le n$, is smaller than or equal to the function value $\varepsilon(p_\varnothing)$ of the average $p_\varnothing$ of the arguments p_i:
$avg_{1 \le i \le n}\, \varepsilon(p_i) \ \le\ \varepsilon(avg_{1 \le i \le n}\, p_i)$, i.e.

$$1/n \cdot \varepsilon(p_1, \dots, p_n) \ =\ 1/n \cdot \sum_{1 \le i \le n} \varepsilon(p_i) \ \le\ \varepsilon(p_\varnothing).$$

Concavity

For $\varepsilon(p) = -p \cdot \log_2 p$ and $n = 2$, we get

$$avg_{1 \le i \le 2}\, \varepsilon(p_i) \;=\; 1/2 \cdot (\varepsilon(p_1) + \varepsilon(p_2)) \;\le\; \varepsilon((p_1 + p_2)/2) \;=\; \varepsilon(avg_{1 \le i \le 2}\, p_i).$$



Analysis of Alternative Entropy Functions

$$\varepsilon'(p_1, \dots, p_n) \;=\; \prod_{i=1}^{n} p_i \;=\; p_1 \cdot \dots \cdot p_n,$$
$$\varepsilon''(p_1, \dots, p_n) \;=\; 1 - \sum_{i=1}^{n} p_i^2 \;=\; 1 - (p_1^2 + \dots + p_n^2) \quad \text{(Breiman).}$$

1. Maximum: For $\sum_{i=1}^{n} p_i = 1$, all three functions ε, ε', and ε'' become maximal for identical values $p_i = 1/n$.

2. The Breiman function is also *concave:*

   $$avg_{1 \le i \le n}\, \varepsilon''(p_i) \;\le\; \varepsilon''(avg_{1 \le i \le n}\, p_i).$$

   For $p_{\emptyset} \;=\; avg_{1 \le i \le n}\, p_i \;=\; 1/n \cdot \sum_{1 \le i \le n} p_i,$ we get

   $$1/n \cdot \sum_{1 \le i \le n} \varepsilon''(p_i) \;\le\; \varepsilon''(p_{\emptyset}),$$

   since $1/n \cdot \sum_{1 \le i \le n} p_i^2 \;\ge\; p_{\emptyset}^2.$

3. Arithmetic mean value – comparing (p_1, p_2) with $(\frac{p_1+p_2}{2}, \frac{p_1+p_2}{2})$:

   From $p_1^2 + p_2^2 - 2 \cdot p_1 \cdot p_2 = (p_1 - p_2)^2 \geq 0$, we get

$$\begin{aligned} \varepsilon'(p_1, p_2) &= p_1 \cdot p_2 \\ &\leq \frac{p_1^2+p_2^2+2 \cdot p_1 \cdot p_2}{4} \\ &= \varepsilon'(\frac{p_1+p_2}{2}, \frac{p_1+p_2}{2}) \quad \text{and} \end{aligned}$$

$$\begin{aligned} \varepsilon''(p_1, p_2) &= 1 - (p_1^2 + p_2^2) \\ &\leq 1 - 2 \cdot \frac{p_1^2+p_2^2+2 \cdot p_1 \cdot p_2}{4} \\ &= \varepsilon''(\frac{p_1+p_2}{2}, \frac{p_1+p_2}{2}). \end{aligned}$$

   We have already shown $\varepsilon(p_1, p_2) \leq 2 \cdot \varepsilon(\frac{p_1+p_2}{2}) = \varepsilon(\frac{p_1+p_2}{2}, \frac{p_1+p_2}{2})$.

4. Sum – comparing (p_1, p_2) with $(p_1 + p_2)$:

   From $p_1^2 + p_2^2 \leq (p_1 + p_2)^2$, we get

$$\begin{aligned} \varepsilon''(p_1, p_2) &= 1 - (p_1^2 + p_2^2) \\ &\geq 1 - (p_1 + p_2)^2 \\ &= \varepsilon''(p_1 + p_2). \end{aligned}$$

   However,

$$\begin{aligned} \varepsilon'(p_1, p_2) &= p_1 \cdot p_2 \\ &\leq p_1 + p_2 \\ &= \varepsilon'(p_1 + p_2). \end{aligned}$$

   We have already shown $\varepsilon(p_1, p_2) \geq \varepsilon(p_1 + p_2)$.

5. Summarizing, the functions ε and ε'' are suitable for defining an entropy, while ε' is not, since it violates the merging condition.

Horizontal Decomposition: $T \mapsto (T_1, \ldots, T_n)$

$$\begin{tikzcd}
 & T \arrow[dl] \arrow[dr] & \\
T_1 & \cdots & T_n
\end{tikzcd}$$

Let $T_A = \Pi_A(T)$ be the domain of A in T.

1. If T_A is small, then we can split T into one table $T_i = \sigma_{A=v_i}(T)$ for each value $v_i \in T_A$, for $1 \leq i \leq n = |T_A|$.
2. If T_A is bigger, then we can split T_A into disjoint subsets S_i, and we can split T into one table $T_i = \sigma_{A \in S_i}(T)$ for each subset $S_i \subset T_A$, for $1 \leq i \leq n < |T_A|$.

Then, the weighted average entropy of $(T_1, \ldots, T_n)$ is given by

$$entropy_Z(T_1, \ldots, T_n) = \sum_{i=1}^{n} \frac{|T_i|}{|T|} \cdot entropy_Z(T_i).$$

Successive splits of a table T w.r.t. two attributes A, B with

$$T_A = \{ v_1, \ldots, v_n \}, \; T_B = \{ w_1, \ldots, w_m \},$$

lead to the tables $T_\ominus = \sigma_\ominus(T)$.

$$\begin{tikzcd}
& & & T \arrow[dll] \arrow[drr] & & & \\
& T_{A=v_1} \arrow[dl] \arrow[dr] & & \cdots & & T_{A=v_n} \arrow[dl] \arrow[dr] & \\
T_{A=v_1 \wedge B=w_1} & \cdots & T_{A=v_1 \wedge B=w_m} & & T_{A=v_n \wedge B=w_1} & \cdots & T_{A=v_n \wedge B=w_m}
\end{tikzcd}$$

For $A = \mathit{PrevDef}$ and $B = \mathit{Married}$, the entropy decreases, when the decision tree becomes deeper:

$$\begin{aligned}
&\mathit{entropy}_{\mathit{Def}}(T) = 0.881, \\
&\mathit{entropy}_{\mathit{Def}}(T_{A=yes}, T_{A=no}) = 0.825, \\
&\mathit{entropy}_{\mathit{Def}}(T_{A=yes \wedge B=yes}, T_{A=yes \wedge B=no}, \\
&\quad T_{A=no \wedge B=yes}, T_{A=no \wedge B=no}) = 0.465.
\end{aligned}$$

If we further split $T_{A=no \wedge B=yes}$ w.r.t. $I = \mathit{Income}$, then the entropy further decreases:

$$\begin{aligned}
&\mathit{entropy}_{\mathit{Def}}(T_{A=yes \wedge B=yes}, T_{A=yes \wedge B=no}, \\
&\quad T_{A=no \wedge B=yes \wedge I<30}, T_{A=no \wedge B=yes \wedge I\geq 30}, T_{A=no \wedge B=no}) = 0.226.
\end{aligned}$$

Later, we will show that every split of a sub–table of a decomposition further decreases the entropy (or leaves it unchanged).

Let $Z \in U$ be the target attribute.

1. If a table T with $T_A = \{ v_1, \ldots, v_n \}$ is split w.r.t. $A \in U$, then the weighted average entropy of the sub–tables $T_i = \sigma_{A=v_i}(T)$ is given by

$$\mathit{entropy}_Z^A(T) \;=\; \mathit{entropy}_Z(T_1, \ldots, T_n).$$

   It always holds $\mathit{entropy}_Z^Z(T) = 0$.

2. The **information gain** of the split is

$$\mathit{gain}_Z^A(T) \;=\; \mathit{entropy}_Z(T) - \mathit{entropy}_Z^A(T).$$

In the example, we gain, since $\mathit{entropy}_{Def}(T) = 0.881$, and

$$\mathit{entropy}_{Def}^{PrevDef}(T) \;=\; \frac{6}{20} \cdot 1 + \frac{14}{20} \cdot 0.75 \;=\; 0.825,$$

i.e., $\mathit{gain}_{Def}^{PrevDef}(T) \;=\; 0.881 - 0.825 \;=\; 0.056.$

Example (Creditworthiness)

The following spreadsheet shows – among others – the values

- $|\sigma_{A=v}(T)|$ and
- $|\sigma_{A=v \wedge Z=w}(T)|$

for $A =$ PrevDef and $Z =$ Def:

	PrevDef = yes	PrevDef = no	*Sum*
Def = yes	3	3	6
Def = no	3	11	14
Sum	6	14	20

$$entropy_{Def}^{PrevDef}(T) = \frac{6}{20} \cdot \varepsilon(\frac{3}{6}, \frac{3}{6}) + \frac{14}{20} \cdot \varepsilon(\frac{3}{14}, \frac{11}{14}) = 0.825,$$
$$entropy_{Def}(T) = \varepsilon(\frac{6}{20}, \frac{14}{20}) = 0.881.$$

Excursus: Theorem (Entropy of a Decomposition)

The weighted average entropy of a decomposition $(T_1, \ldots, T_n)$ is smaller than or equal to the entropy of the table T itself:

$$entropy_Z^A(T) \;=\; entropy_Z(T_1, \ldots, T_n) \;\leq\; entropy_Z(T).$$

Proof:

- Let $T_A = \{\, v_1, \ldots, v_n \,\}$ and $T_Z = \{\, w_1, \ldots, w_m \,\}$.
- Let

$$\begin{aligned}
N_{i,j} &= |\sigma_{Z=w_i \wedge A=v_j}(T)|, \\
N_{i,\oplus} &= |\sigma_{Z=w_i}(T)| = \textstyle\sum_{1\leq j\leq n} N_{i,j}, \\
N_{\oplus,j} &= |\sigma_{A=v_j}(T)| = \textstyle\sum_{1\leq i\leq m} N_{i,j}, \\
N_{\oplus,\oplus} &= |T| = \textstyle\sum_{1\leq i\leq m} N_{i,\oplus} = \sum_{1\leq j\leq n} N_{\oplus,j}, \\
N_{\oplus,\odot} &= \Pi_{1\leq j\leq n} N_{\oplus,j}.
\end{aligned}$$

	$A = v_1$	$\dots$	$A = v_j$	$\dots$	$A = v_n$	*Sum*
$Z = w_1$	$N_{1,1}$	$\dots$	$N_{1,j}$	$\dots$	$N_{1,n}$	$N_{1,\oplus}$
			$\vdots$			
$Z = w_i$	$N_{i,1}$	$\dots$	$N_{i,j}$	$\dots$	$N_{i,n}$	$N_{i,\oplus}$
			$\vdots$			
$Z = w_m$	$N_{m,1}$	$\dots$	$N_{m,j}$	$\dots$	$N_{m,n}$	$N_{m,\oplus}$
Sum	$N_{\oplus,1}$	$\dots$	$N_{\oplus,j}$	$\dots$	$N_{\oplus,n}$	$N_{\oplus,\oplus}$

Then, we get for $1 \leq i \leq m$:

$$\begin{aligned} p_{i,j} &= p_{Z=w_i}(\sigma_{A=v_j}(T)) = N_{i,j}/N_{\oplus,j}, \text{ for } 1 \leq j \leq n, \\ p_i &= p_{Z=w_i}(T) = N_{i,\oplus}/N_{\oplus,\oplus} = 1/N_{\oplus,\oplus} \cdot \sum_{1 \leq j \leq n} p_{i,j} \cdot N_{\oplus,j}. \end{aligned}$$

The following operations do not change the entropies $entropy_Z^A(T)$ and $entropy_Z(T)$:

- Multiplying all entries of the spreadsheet by $N_{\oplus,\odot}$:
  the new entries are
  $$N'_{\alpha,\beta} = N_{\alpha,\beta} \cdot N_{\oplus,\odot},$$
  where $1 \leq \alpha \leq m$ or $\alpha = \oplus$, and $1 \leq \beta \leq n$ or $\beta = \oplus$.

- Successively, splitting each column j, where $1 \leq j \leq n$, into $N_{\oplus,j}$ identical columns with the entries
  $$N''_{\alpha,j} = N'_{\alpha,j}/N_{\oplus,j} = N_{\alpha,j} \cdot N_{\oplus,\odot}/N_{\oplus,j} \in \mathbb{N},$$
  where $1 \leq \alpha \leq m$ or $\alpha = \oplus$.

The resulting spreadsheet consists of natural numbers, and the entries $N''_{\oplus,j}$, for $1 \leq j \leq n$, of the new summary row are all identical: $N''_{\oplus,j} = N_{\oplus,\odot}$.

In our example about creditworthiness, we could achieve the desired effect simply as follows:

- we multiply all entries of the spreadsheet by 7;
- we split the column for ”PrevDef=yes” into 3 identical columns and the column for ”PrevDef=no” into 7 identical columns.

	PrevDef = yes			PrevDef = no			*Sum*
Def = yes	7	7	7	3	...	3	42
Def = no	7	7	7	11	...	11	98
Sum	14	14	14	14	...	14	140

Then, the 10 regular entries of the new summary row are all identical, namely 14.

Thus, w.l.o.g., we can assume that all column sums $N_{\oplus,j}$ are identical:

$$N_{\oplus,j} = 1/n \cdot N_{\oplus,\oplus}, \text{ for all } 1 \leq j \leq n.$$

Then, $p_i = 1/N_{\oplus,\oplus} \cdot \sum_{1 \leq j \leq n} p_{i,j} \cdot N_{\oplus,j} = 1/n \cdot \sum_{1 \leq j \leq n} p_{i,j}$, and we get:

$$\begin{aligned}
entropy_Z^A(T) &= entropy_Z(T_1, \ldots, T_n) \\
&= \sum_{1 \leq j \leq n} N_{\oplus,j}/N_{\oplus,\oplus} \cdot \varepsilon(p_{1,j}, \ldots, p_{m,j}) \\
&= \sum_{1 \leq j \leq n} 1/n \cdot \sum_{1 \leq i \leq m} \varepsilon(p_{i,j}) \\
&= \sum_{1 \leq i \leq m} 1/n \cdot \sum_{1 \leq j \leq n} \varepsilon(p_{i,j}) \\
&\leq \sum_{1 \leq i \leq m} \varepsilon(p_i) = entropy_Z(T).
\end{aligned}$$

□

Analogously, we can show the following for the Breiman function:

$$
\begin{aligned}
& \mathit{entropy}''_Z(T_1, \ldots, T_n) \\
&\quad = \sum_{1 \le j \le n} N_{\oplus,j}/N_{\oplus,\oplus} \cdot \varepsilon''(p_{1,j}, \ldots, p_{m,j}) \\
&\quad = \sum_{1 \le j \le n} N_{\oplus,j}/N_{\oplus,\oplus} - \sum_{1 \le j \le n} N_{\oplus,j}/N_{\oplus,\oplus} \cdot \sum_{1 \le i \le m} p_{i,j}^2 \\
&\quad = 1 - \sum_{1 \le j \le n} 1/n \cdot \sum_{1 \le i \le m} p_{i,j}^2 \\
&\quad = 1 - \sum_{1 \le i \le m} 1/n \cdot \sum_{1 \le j \le n} p_{i,j}^2 \\
&\quad \le 1 - \sum_{1 \le i \le m} p_i^2 \; = \; \mathit{entropy}''_Z(T).
\end{aligned}
$$

Theorem (Entropy of an Extended Decomposition)

Given a decomposition $(T_1, \dots, T_n)$. If we split one of the tables T_k into sub–tables $(T_{k,1}, \dots, T_{k,m})$, then the weighted average entropy of the extended decomposition becomes smaller (or stays the same):

$$\mathit{entropy}_Z(T_1, \dots, T_{k,1}, \dots, T_{k,m}, \dots, T_n) \le \mathit{entropy}_Z(T_1, \dots, T_n).$$

Proof: From the previous theorem we know

$$\mathit{entropy}_Z(T_{k,1}, \dots, T_{k,m}) \le \mathit{entropy}_Z(T_k).$$

Thus, it follows

$$\mathit{entropy}_Z(T_1, \dots, T_n) = \sum_{i=1}^{n} \frac{|T_i|}{|T|} \cdot \mathit{entropy}_Z(T_i) \ge \sum_{i=1, i \neq k}^{n} \frac{|T_i|}{|T|} \cdot \mathit{entropy}_Z(T_i) + \frac{|T_k|}{|T|} \cdot \sum_{j=1}^{m} \frac{|T_{k,j}|}{|T_k|} \cdot \mathit{entropy}_Z(T_{k,j}) = \mathit{entropy}_Z(T_1, \dots, T_{k,1}, \dots, T_{k,m}, \dots, T_n).$$

$\square$

The previous theorem is a special case ($n = 1$) of the current theorem.

Example (Information Gain)

For $Z = Def$, we get $entropy_Z(T) = 0.881$.

A	$entropy_Z^A(T)$
$Married$	0.825
$PrevDef$	0.825
$Income_{\geq 30}$	0.870
$Income_{\geq 80}$	0.784
$Income_{\geq 135}$	0.688
Def	0

The attribute $A = Income_{\geq 135}$ would have the largest information gain $gain_Z^A(T)$, except, of course, the target attribute $A = Z$ itself.

Double split w.r.t. *PrevDef* and *Married* leads to more homogeneous tables with

$$entropy_Z(T_{yes,no}, \ldots, T_{no,yes}) = \frac{11}{20} \cdot \varepsilon(\tfrac{3}{11}, \tfrac{8}{11}) + 0 + 0 + 0 = 0.465.$$

$T_{yes,no}$

Id	Married	PrevDef	Income	Def
C1	yes	no	50	no
C2	yes	no	100	no
C4	yes	no	125	no
C5	yes	no	50	no
C8	yes	no	10	yes
C9	yes	no	75	no
C11	yes	no	60	yes
C16	yes	no	15	yes
C17	yes	no	35	no
C19	yes	no	40	no
C20	yes	no	30	no

$T_{yes,yes}$

Id	Married	PrevDef	Income	Def
C7	yes	yes	10	no
C10	yes	yes	45	no
C13	yes	yes	20	no

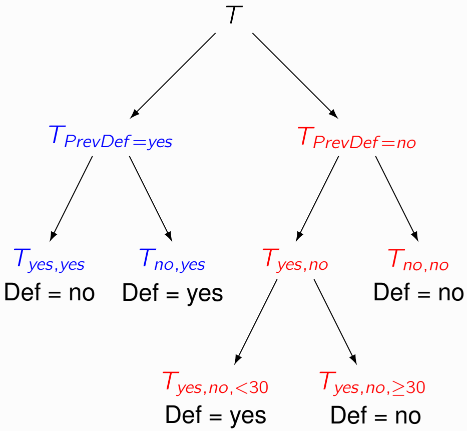
$T_{no,no}$

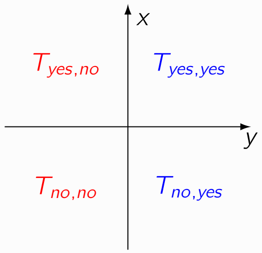
Id	Married	PrevDef	Income	Def
C6	no	no	30	no
C14	no	no	15	no
C15	no	no	60	no

$T_{no,yes}$

Id	Married	PrevDef	Income	Def
C3	no	yes	135	yes
C12	no	yes	125	yes
C18	no	yes	160	yes

We define $T_{x,y} = T_{Married=x \wedge PrevDef=y}$, where $T_\Theta = \sigma_\Theta(T)$.





$$entropy_Z(T_{yes,yes}, T_{no,yes}, T_{yes,no,<30}, T_{yes,no,\geq 30}, T_{no,no}) = 0.226$$

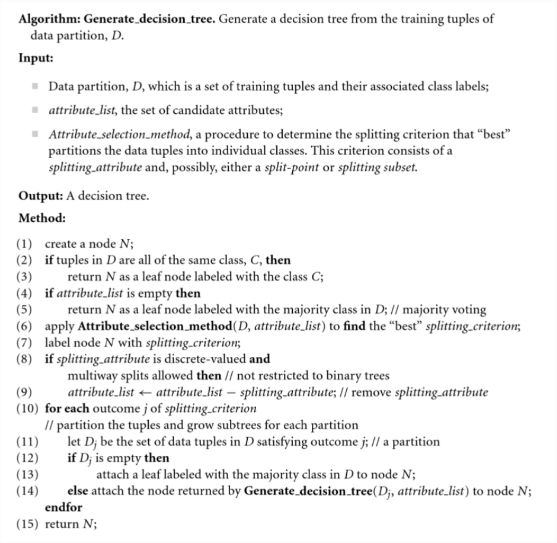
For the following table T (left) and the target variable Z, an unbalanced split w.r.t. A is obviously better than the balanced split w.r.t. B:

B	A	Z
2	1	1
2	0	0
2	0	0
3	0	0
3	0	0
3	0	0

	$B = 2$	$B = 3$	Sum
$Z = 0$	2	3	5
$Z = 1$	1	0	1
Sum	3	3	6

The spreadsheet on the right shows the joint distribution of the values for the attributes B and Z – as used in the proof before.

We get $entropy_Z(T) = 0.65$ and $entropy_Z^A(T) = 0$, $entropy_Z^B(T) = 0.46$.



Algorithm: Generate_decision_tree. Generate a decision tree from the training tuples of data partition, D.

Input:

- Data partition, D, which is a set of training tuples and their associated class labels;
- *attribute_list*, the set of candidate attributes;
- *Attribute_selection_method*, a procedure to determine the splitting criterion that "best" partitions the data tuples into individual classes. This criterion consists of a *splitting_attribute* and, possibly, either a *split-point* or *splitting subset*.

Output: A decision tree.

Method:

```
(1)  create a node $N$;
(2)  if tuples in $D$ are all of the same class, $C$, then
(3)      return $N$ as a leaf node labeled with the class $C$;
(4)  if attribute_list is empty then
(5)      return $N$ as a leaf node labeled with the majority class in $D$; // majority voting
(6)  apply Attribute_selection_method($D$, attribute_list) to find the "best" splitting_criterion;
(7)  label node $N$ with splitting_criterion;
(8)  if splitting_attribute is discrete-valued and
         multiway splits allowed then // not restricted to binary trees
(9)      attribute_list $\leftarrow$ attribute_list $-$ splitting_attribute; // remove splitting_attribute
(10) for each outcome $j$ of splitting_criterion
     // partition the tuples and grow subtrees for each partition
(11)     let $D_j$ be the set of data tuples in $D$ satisfying outcome $j$; // a partition
(12)     if $D_j$ is empty then
(13)         attach a leaf labeled with the majority class in $D$ to node $N$;
(14)     else attach the node returned by Generate_decision_tree($D_j$, attribute_list) to node $N$;
     endfor
(15) return $N$;
```

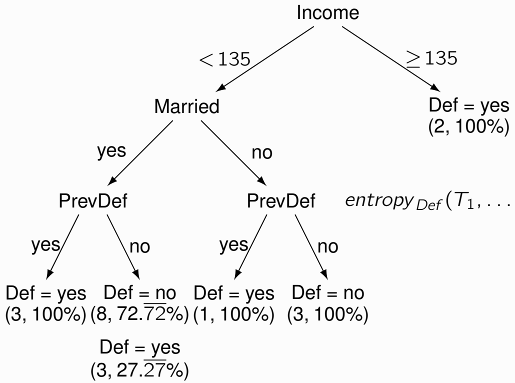
The ID3–Algorithm of Quinlan, 1986

1. If $entropy_Z(T) = 0$, then there is exactly one value v for Z, and T is not split. T is marked by v.
2. If $entropy_Z(T) \neq 0$ and T has no more attributes, then T is not split either. T is marked by the most frequent value v for Z.
3. Otherwise: Determine the attribute A with the largest information gain and split T into the sub–tables $\sigma_{A=v}(T)$, for $v \in T_A$. Recursively apply this procedure to the sub–tables.

There is a variant C4.5 with discretization of continuous domains.

The applied *greedy heuristics* does not always produce the optimal result.

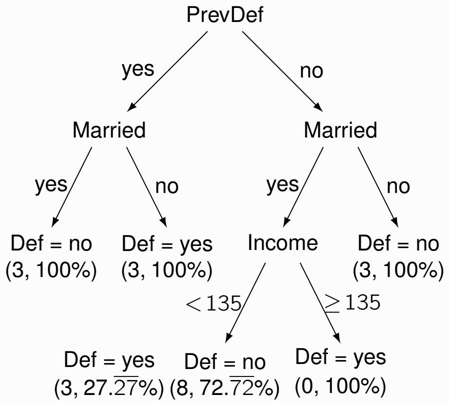
Example (Decision Trees Generated by ID3, Creditworthiness)



$entropy_{Def}(T_1, \ldots, T_5) = 0.465$

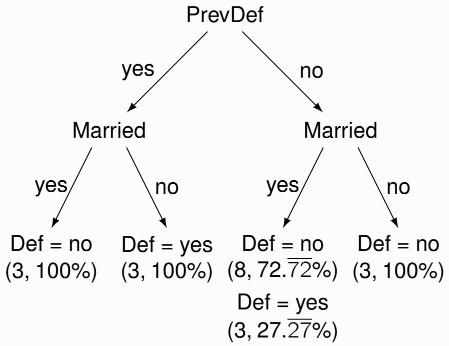
This tree is not optimal. We have already seen a better tree with the entropy 0.226, where Income was split w.r.t. the value 30.

Splitting w.r.t. the same attributes and values, but in another order, gives a decision tree with the same entropy:



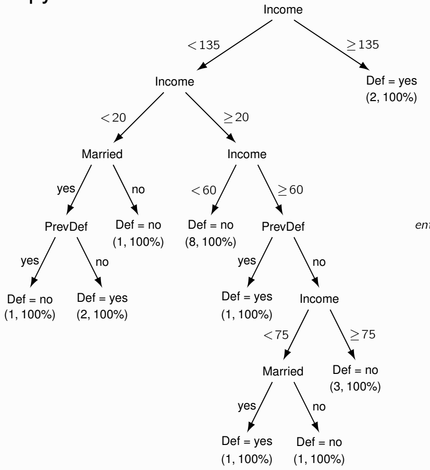
PrevDef=yes/no, Married=yes/no, Income$\geq$135/$<$135 $\mapsto$ entropy 0.465

However, the leaves 3 and 4 of the previous decision tree can be cut without changing the entropy 0.465, since leaf 4 does not refer to any tuples:



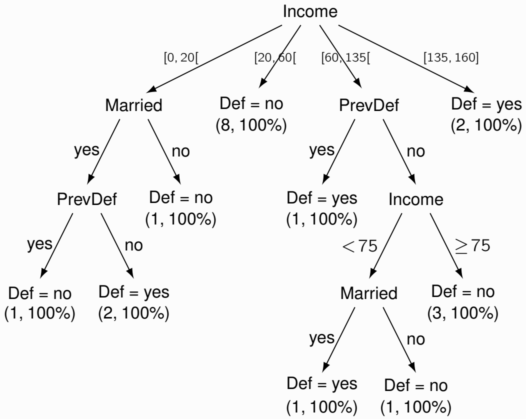
Thus, we obtain a smaller tree, which we would prefer.

In our small example, *repeated choice* of the same attribute Income can reduce the entropy to 0:



$entropy_{Def}(T_1, \ldots, T_9) = 0$

Equivalently, an immediate split into 4 tables w.r.t. *intervals* $S_i \subset T_{Income}$ instead of successive splits into 2 tables gives entropy 0:



Example (Classification Rules, Creditworthiness)

very low:	$0 \leq \mathit{Income} < 20$
low:	$20 \leq \mathit{Income} < 60$
middle:	$60 \leq \mathit{Income} < 75$
high:	$75 \leq \mathit{Income} < 135$
very high:	$135 \leq \mathit{Income} \leq 160$

1. Married people with very low income default,
   if and only if they did not default previously:

   $(\mathit{Income} < 20) \wedge (\mathit{Married} = \mathit{yes}) \wedge (\mathit{PrevDef} = \mathit{yes}) \Rightarrow (\mathit{Def} = \mathit{no})$

   $(\mathit{Income} < 20) \wedge (\mathit{Married} = \mathit{yes}) \wedge (\mathit{PrevDef} = \mathit{no}) \Rightarrow (\mathit{Def} = \mathit{yes})$

   Un–married people with very low income don't default:

   $(\mathit{Income} < 20) \wedge (\mathit{Married} = \mathit{no}) \Rightarrow (\mathit{Def} = \mathit{no})$

2. People with low income don't default (8 cases in our database):

   $(20 \leq \mathit{Income} < 60) \Rightarrow (\mathit{Def} = \mathit{no})$

3. People with middle or high income default, if they have defaulted before:

   $(60 \leq \mathit{Income} < 135) \wedge (\mathit{PrevDef} = \mathit{yes}) \Rightarrow (\mathit{Def} = \mathit{yes})$

4. People with middle income, who did not default before, default if and only if they are married:

   $(60 \leq \mathit{Income} < 75) \wedge (\mathit{PrevDef} = \mathit{no}) \wedge (\mathit{Married} = \mathit{yes}) \Rightarrow (\mathit{Def} = \mathit{yes})$

   $(60 \leq \mathit{Income} < 75) \wedge (\mathit{PrevDef} = \mathit{no}) \wedge (\mathit{Married} = \mathit{no}) \Rightarrow (\mathit{Def} = \mathit{no})$

5. People with high income don't default, if they did not default before:

   $(75 \leq \mathit{Income} < 135) \wedge (\mathit{PrevDef} = \mathit{no}) \Rightarrow (\mathit{Def} = \mathit{no})$

6. People with very high income default (in our database):

   $(\mathit{Income} \geq 135) \Rightarrow (\mathit{Def} = \mathit{yes})$

Of course, these classification rules only hold in our toy database.

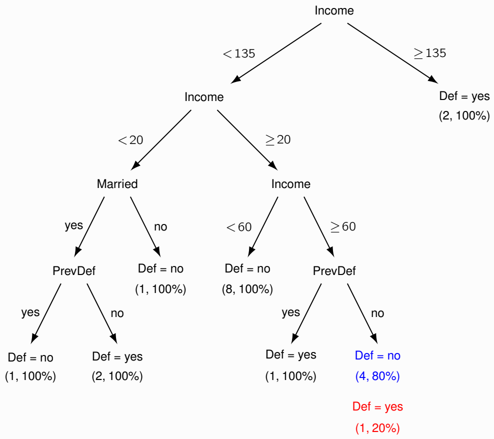
Machine Learning Process/Refinements

- Obvious: Training/Validation/Test sets - also for pruning
  - Training Set: computation of the decision tree
  - Validation Set: cutting of the decision tree at the leaves
  - Test Set: test of the decision tree
- Split functions:
  - the *gain ratio* (Quinlan, 1986) can be used to compensate for the fact that the information gain favours attributes A with a large number of values:
    $gain\_ratio_Z^A(T) = gain_Z^A(T)/entropy_A(T)$
  - ***gini index*** (Breiman et al., 1984):
    $gini_Z(T) = 1 - \sum_{v \in T_Z}(p_{Z=v}(T))^2$

Gini Index (CART Algorithm)

- If a dataset D contains examples from n classes $T_Z = \{v_1, \ldots, v_n\}$ for the target Z, the *gini index* is defined as: $1 - \sum_{v \in T_Z} (p_{Z=v}(T))^2$
- If dataset D is split on attribute A into two subsets D_1 and D_2, then the gini index is given as: $gini_A = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)$
- The gini index considers all (possible) binary splits for an attribute
- For numeric attributes, all possible split points are considered (considering all possible splits given the sorted list of occurring values)
- Reduction in impurity: $gini(A) = gini(D) - gini_A(D)$
- The attribute A with the largest $gini(A)$ is chosen for splitting a node

Cutting of the decision tree at the leaves:



Overfitting and Pruning

- Overfitting: A learned decision tree may overfit the training data
  - Too many branches; some may reflect anomalies due to noise or outliers
  - Poor accuracy for unseen samples
- Two approaches to avoid overfitting
  - Prepruning: Stop tree construction early - do not split if a goodness measure is below threshold (problem: choosing the threshold !)
  - Postpruning: Remove branches from a “fully grown tree” – get a sequence of progressively pruned trees (problem: how to select the “best pruned tree”)

Pruning: Reduced Error Pruning

Basic Approach:

- Split data into training I and test set T
- Construct decision tree D using the training set I
- Prune D using T:
  - Identify the subtree S of D, for which its removal reduces the (classification) error on D the most
  - Remove S
  - Algorithm terminates, if no such subtree S exists

Pruning: Other Options

- Error-complexity pruning: Balancing number of (classification) errors and size of the tree
- Critical value pruning: Based on threshold (goodness-of-split) assessed for each split attribute when building the tree
- Minimum error pruning: Similar to reduced error pruning, identify exact minimal error for equally distributed classes

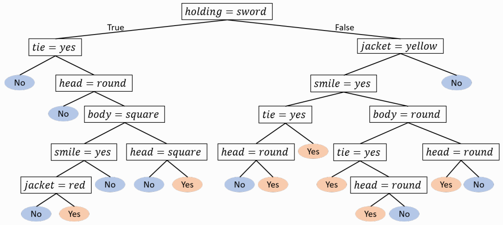
Enhancements to Basic Decision Tree Learning

- Allow for continuous-valued attributes: Dynamically define new discrete-valued attributes that partition the continuous attribute value into a discrete set of intervals
- Handle missing attribute values
  - Assign the most common value of the attribute
  - Assign probability to each of the possible values
- Attribute construction
  - Create new attributes based on existing ones that are sparsely represented
  - Reduces fragmentation, repetition, and replication

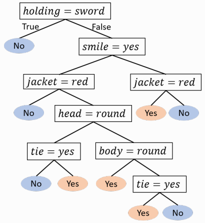
Optimal Decision Trees

- So far: heuristic methods
- However: Often (simple) decision trees have lower performance than black-box models (e. g., deep learning)
- BUT: See random forests (below)
- ALSO: With increase in computational power, fully optimize trees (!)
  $\rightsquigarrow$ Huge search problem / optimization problem
- Approaches for fully optimized trees
  - Mixed integer programming
  - SAT solvers
  - Stochastic search through the space of trees
  - Customized dynamic programming with branch-and-bound

Example: Optimal Decision Tree



(a) training accuracy: 75.74%; test accuracy: 69.44%



(b) training accuracy: 81.07%; test accuracy: 73.15%

(a) 16-leaf decision tree learned by CART (Breiman et al., 1984) and (b) 9-leaf decision tree generated by GOSDT (Lin et al., 2020) for the classic Monk 2 dataset (Dua and Graff, 2017). The GOSDT tree is optimal with respect to a balance between accuracy and sparsity. (Rudin et al. 2021)

“Let us provide some background on decision trees. Since Morgan and Sonquist (1963) developed the first decision tree algorithm, many works have been proposed to build decision trees and improve their performance. However, learning decision trees with high performance and sparsity is not easy. **Full decision tree optimization is known to be an NP-complete problem** (Laurent and Rivest, 1976), and heuristic greedy splitting and pruning procedures have been the major type of approach since the 1980s to grow decision trees (Breiman et al., 1984; Quinlan, 1993; Loh and Shih, 1997; Mehta et al., 1996). **These greedy methods for building decision trees create trees from the top down and prune them back afterwards. They do not go back to fix a bad split if one was made. Consequently, the trees created from these greedy methods tend to be both less accurate and less interpretable than necessary.** That is, greedy induction algorithms are not designed to optimize any particular performance metric, leaving a gap between the performance that a decision tree might obtain and the performance that the algorithms decision tree actually attains, with no way to determine how large the gap is [...] **This gap can cause a problem in practice because one does not know whether poor performance is due to the choice of model form (the choice to use a decision tree of a specific size) or poor optimization (not fully optimizing over the set of decision trees of that size). When fully optimized, single trees can be as accurate as ensembles of trees, or neural networks, for many problems.** Thus, it is worthwhile to think carefully about how to optimize them." (Rudin et al. 2021 – highlighting added).

GOSDT - Generalized and Scalable Optimal Sparse Decision Trees

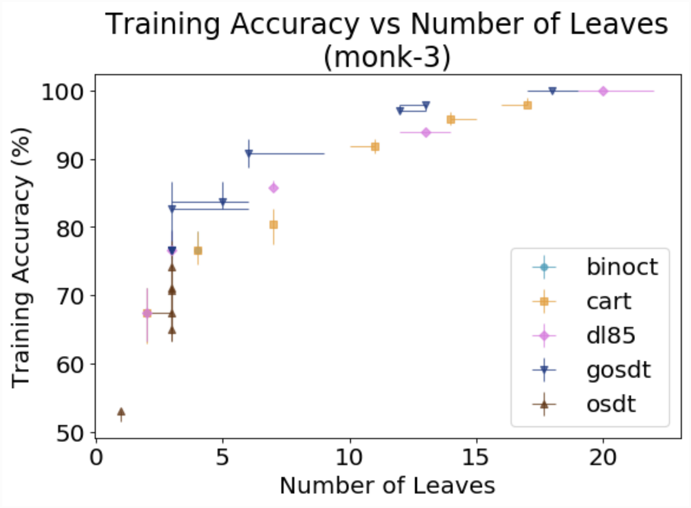
(Lin et al. ICML 2020)

- Branch-and-Bound: Fast analytical bounds
- Smart implementation (bitsets)
- Caching of (redundant) intermediate results when exploring search space

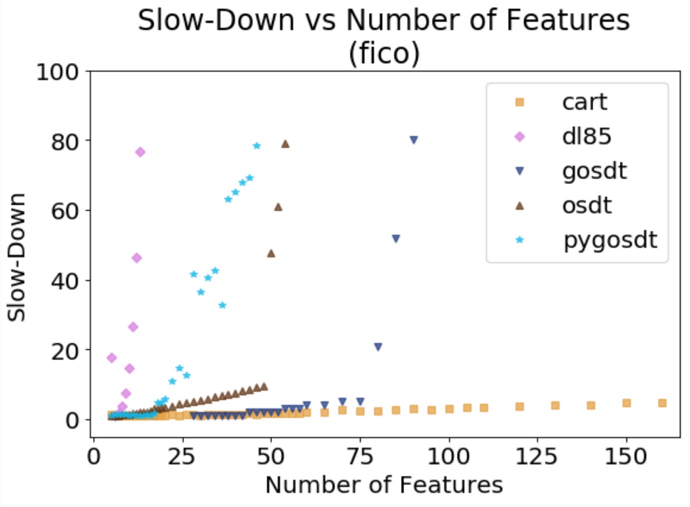
Main experimental results:

- Similar classification error to black box methods
- Sparser than all heuristic methods
- Orders of magnitude faster than the next best method

Example: Optimal Decision Tree (GOSDT) – Accuracy (Lin et al. ICML 2020)



Example: Optimal Decision Tree (GOSDT) – Runtime (Lin et al. ICML 2020)



Towards Random Forest: Ensemble methods

- Starting from *simple* decision trees, we can build more “advanced” structures, i. e., sets of trees $\leadsto$ ensembles
- General method: Building ensemble classifiers
- In general, ensemble methods feature the following two things:
  - They construct multiple, diverse predictive models from adapted versions of the training data (most often reweighted or resampled);
  - They combine the predictions of these models in some way, often by simple averaging or voting (possibly weighted).

Bagging

Algorithm Bagging($D, T, \mathscr{A}$) – train an ensemble of models from bootstrap samples.

Input : data set D; ensemble size T; learning algorithm $\mathscr{A}$.

Output : ensemble of models whose predictions are to be combined by voting or averaging.

```
1 for $t = 1$ to $T$ do
2     build a bootstrap sample $D_t$ from $D$ by sampling $|D|$ data points
        with replacement;
3     run $\mathscr{A}$ on $D_t$ to produce a model $M_t$;
4 end
5 return $\{M_t | 1 \leq t \leq T\}$
```

Random forests

Algorithm RandomForest(D, T, d) – train an ensemble of tree models from bootstrap samples and random subspaces.

Input : data set D; ensemble size T; subspace dimension d.
Output : ensemble of tree models whose predictions are to be combined by voting or averaging.

```
1 for $t = 1$ to $T$ do
2     build a bootstrap sample $D_t$ from $D$ by sampling $|D|$ data points
        with replacement;
3     select $d$ features at random and reduce dimensionality of $D_t$
        accordingly;
4     train a tree model $M_t$ on $D_t$ without pruning;
5 end
6 return $\{M_t|1 \le t \le T\}$
```