Grundlagen der Künstlichen Intelligenz I

Decision Tree Learning, Linear Regression

Tamas Horvath

University of Bonn and Fraunhofer IAIS

horvath@cs.uni-bonn.de

Outline

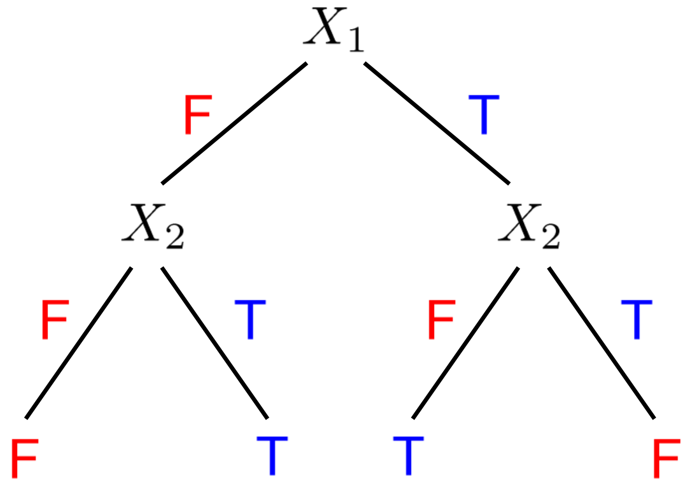
- complexity of constructing optimal decision trees
- TDIDT algorithm: example (Quinlan, 1986)
- information gain
- expressivity of decision trees
- overfitting

Expressivity of Decision Trees

high expressive power, e.g., any function $f : X_1 \times \ldots \times X_m \to \mathcal{Y}$ can trivially be expressed by a finite decision tree if $|X_i| < \infty$ for all i

- e.g., Boolean functions: rows in the truth table $\to$ paths

X_1	X_2	Y
F	F	F
F	T	T
T	F	T
T	T	F



Boolean decision tree: boolescher Entscheidungsbaum

Enormous Number of (Boolean) Decision Trees

the number of decision trees over n attributes is **enormous**

- even for **Boolean decision trees** over n attributes, as each Boolean function $f : \{0,1\}^n \to \{0,1\}$ can be represented by a Boolean decision tree

  number of pairwise non-equivalent decision trees with n Boolean attributes

  = number of Boolean functions over n variables

  = number of distinct truth tables with 2^n rows

  = 2^{2^n}

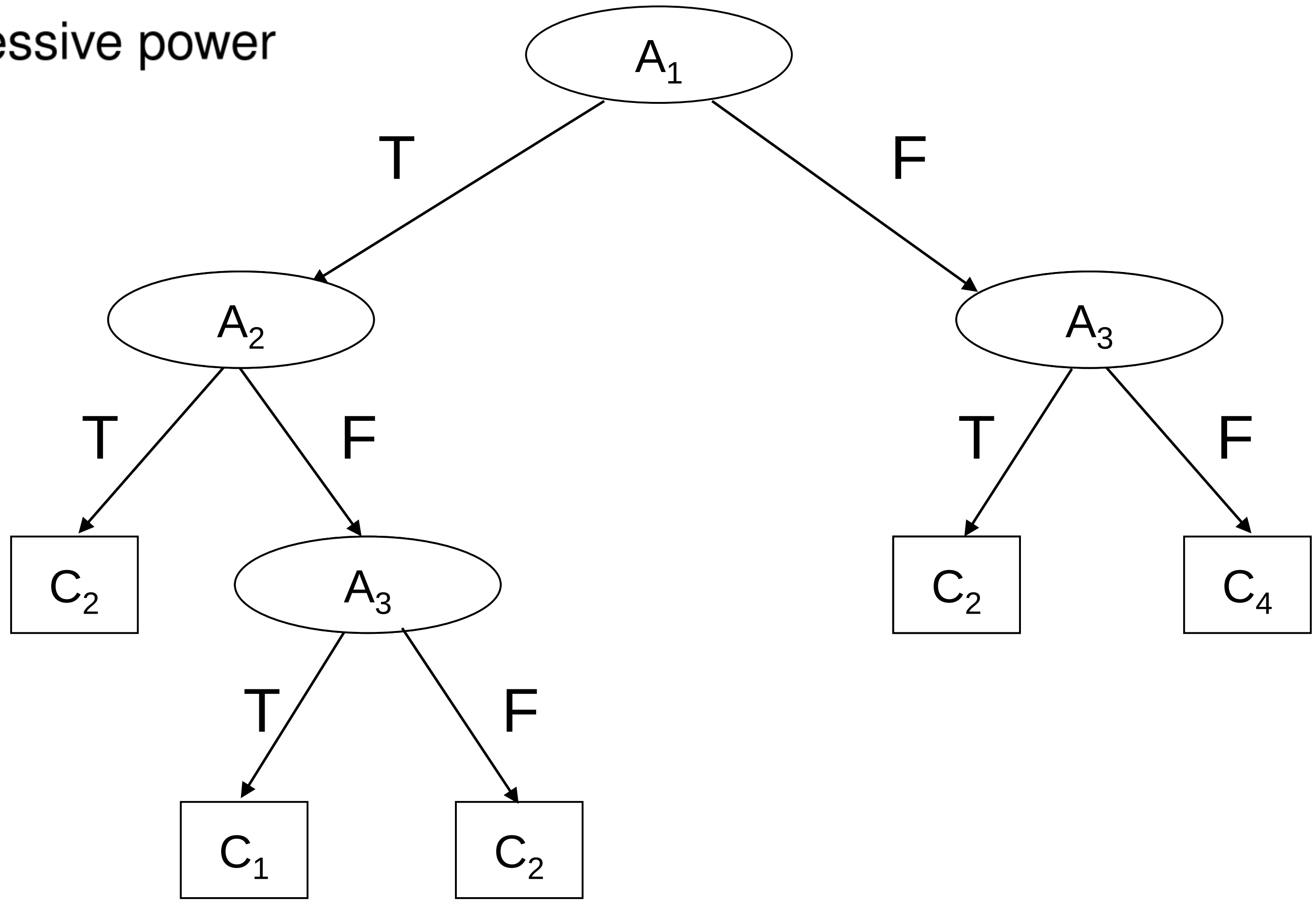
  - e.g., for $n = 6$ we have

$$2^{2^6} = 2^{64} = 18,446,744,073,709,551,616 \approx 2 \cdot 10^{20}$$

Hypothesis Language: Binary Decision Trees

most often: only **binary** splits are allowed

- no loss in expressive power



Binary Decision Trees: Continuous Valued Attributes

search at split time: dynamic construction of binary test $A \leq c$ for a continuous attribute A

algorithm:

1: sort examples covered by the current node according to the values of A
2: **forall** adjacent examples with different classes **do**
3: take the mean m of the values of A for the two examples
4: calculate the information gain for the split $A \leq m$ for the current node
5: **return** $A \leq c$ that maximizes the information gain

Binary Decision Trees: Continuous Valued Attributes

remarks:

- there is no c' such that $A \leq c'$ has strictly larger information gain than $A \leq c$ on the set of examples covered by the current node
  - (Fayyad & Irani, 1992)
- for attribute A, the information gain of $A \leq c$ compete with the splits for the other attributes

Binary Decision Trees: Continuous Valued Attributes

example:

Temperature:	40	48	60	72	80	90
PlayTennis:	No	No	Yes	Yes	Yes	No

two tests must be processed:

- Temperature $\leq \frac{48+60}{2}$
- Temperature $\leq \frac{80+90}{2}$

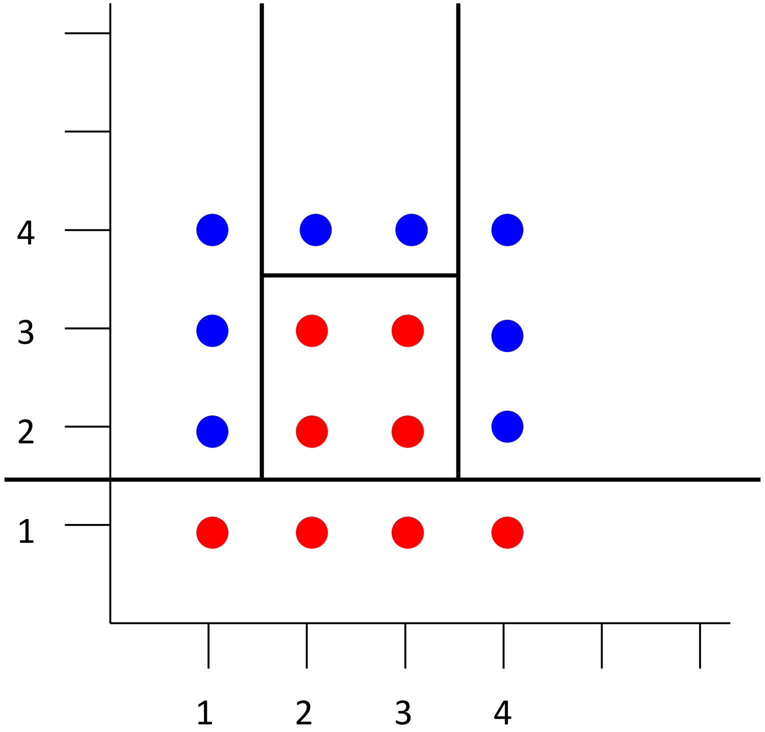
Binary Decision Trees: Categorical Attributes

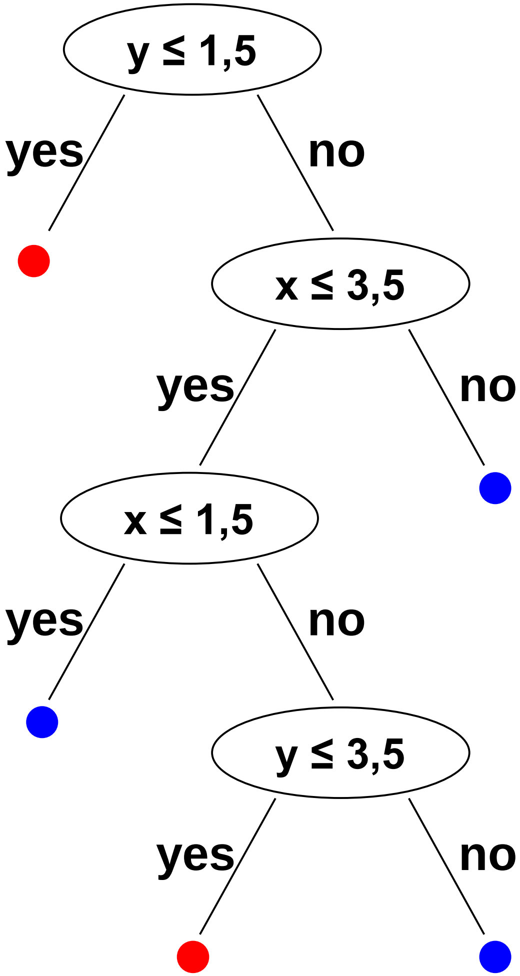
need further split operator: **subsetting**

- use value subsets for categorical attributes
- (binary) TEST: "Is value of attribute A in set S?"
- use greedy algorithm for finding S, i.e.,

  1. **let** $S = \emptyset$
  2. add that attribute value of A to S which gives the best split according to information gain
  3. repeat step 2 until there is no improvement in the splits

Binary Decision Trees: Decision Boundaries





- input space is divided into axis-parallel rectangles and each rectangle is associated with one of the target classes

Expressive Power of Axis-Parallel Rectangles

decision trees are

$\Rightarrow$ **good** at problems in which the class label is constant in large, connected, axis-orthogonal regions of the input space

$\Rightarrow$ **bad** e.g. for parity/XOR type problems, or for problem like below

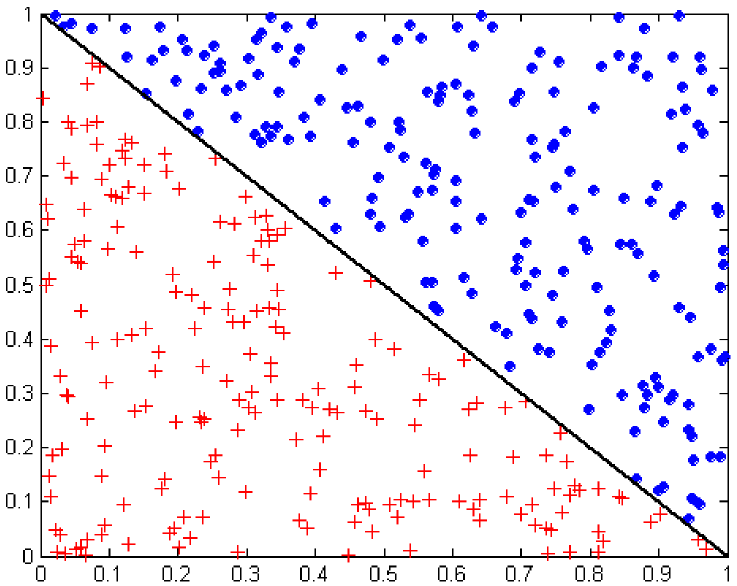


Figure: © P. Kärger

Outline

- complexity of constructing optimal decision trees
- TDIDT algorithm: example (Quinlan, 1986)
- information gain
- expressivity of decision trees
- overfitting

Overfitting

consider error of hypothesis h over

- training error (empirical risk) on training data $\mathcal{D}$: $\text{error}_{\mathcal{D}}(h)$
- true error (expected error): $\text{error}(h)$

Def. (overfitting): hypothesis $h \in \mathcal{H}$ **overfits** training data if there is an alternative hypothesis $h' \in \mathcal{H}$ such that

$$\text{error}_{\mathcal{D}}(h) < \text{error}_{\mathcal{D}}(h')$$

and

$$\text{error}(h) > \text{error}(h')$$

Overfitting: Causes

overfitting: decision trees are more complex than necessary due to

- noisy data
- lack of representative instances
- lack of samples

pruning: Beschneiden

Avoiding Overfitting

two approaches to avoid overfitting:

1. **prepruning:** stop growing when data split not statistically significant
2. **postpruning:** grow full tree, then post-prune

we focus on postpruning

- preferred in practice (prepruning can “stop too early”) statistical noise: statistisches Rauschen

Postpruning

two-step approach:

step 1: tree building

repeatedly partition the training data until all the examples in each partition belong to one class or the partition is sufficiently small

step 2: tree pruning

remove dependency e.g. on statistical noise that may be particular only to the training set

Reduced Error Pruning (Quinlan, 1987)

split the training data into training set $\mathcal{D}_1$ and validation set $\mathcal{D}_2$

build decision tree T on training set $\mathcal{D}_1$

forall internal nodes v of T in a post-order traversal **do**

    evaluate the accuracy A_{T_v} of the subtree of T rooted at v on $\mathcal{D}_2$

    evaluate the accuracy A_v of pruning the subtree rooted at v in $\mathcal{D}_2$

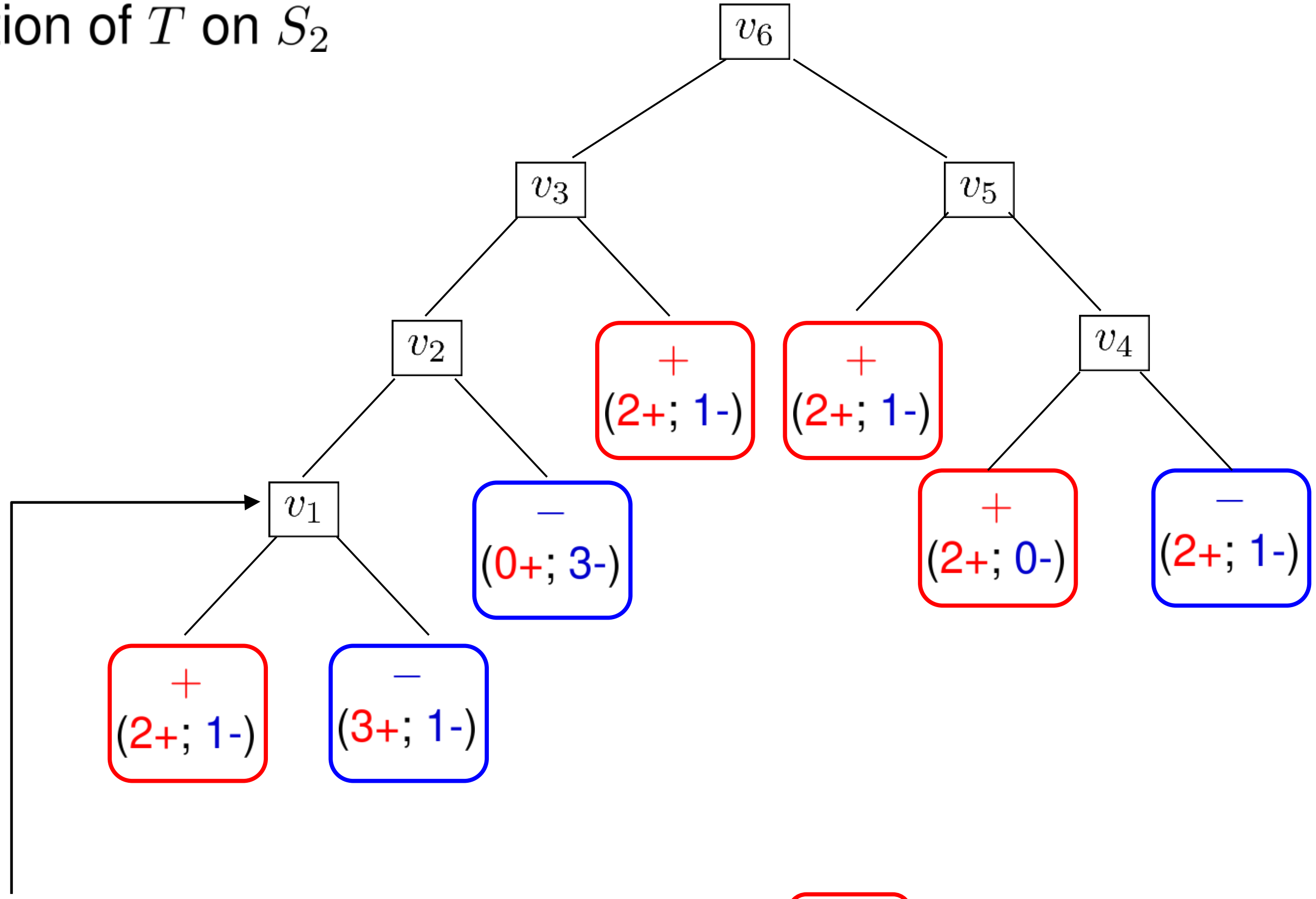
    // see example on the next slides

    if $A_{T_v} \leq A_v$ **then**

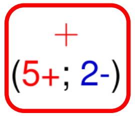
        replace T_v in T by a leaf with the majority class corresponding to the examples of $\mathcal{D}_2$ in T_v

Reduced Error Pruning: Example

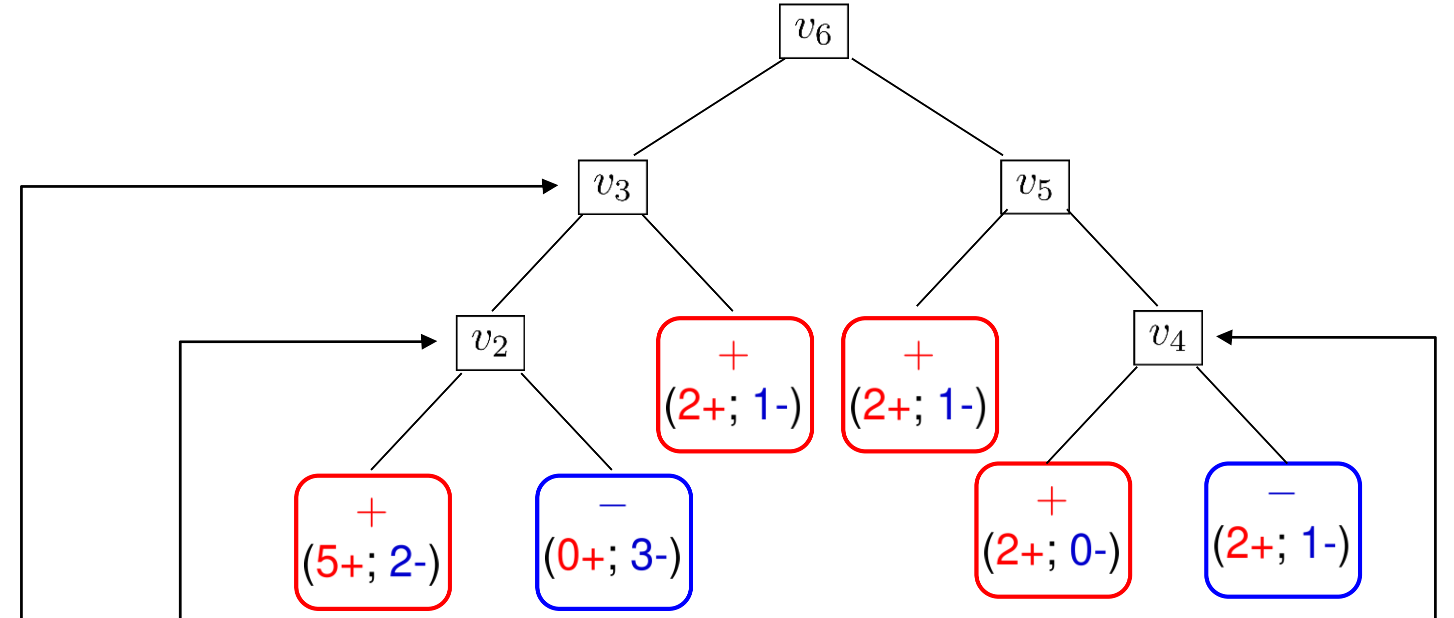
evaluation of T on S_2



$A_{T_{v_1}} = 3/7 \leq 5/7 = A_{v_1} \implies$ replace T_{v_1} by



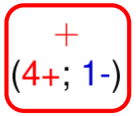
Reduced Error Pruning: Example



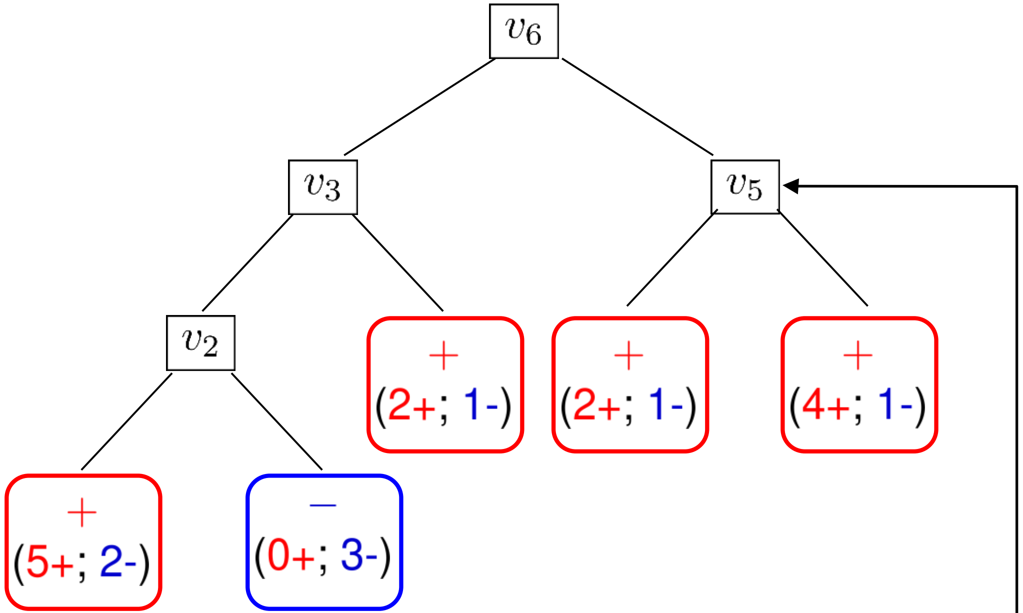
$$A_{T_{v_2}} = 8/10 > 5/10 = A_{v_2} \implies \text{no pruning}$$

$$A_{T_{v_3}} = 10/13 > 7/13 = A_{v_3} \implies \text{no pruning}$$

$$A_{T_{v_4}} = 3/5 \leq 4/5 = A_{v_4} \implies \text{replace } T_{v_4} \text{ by}$$



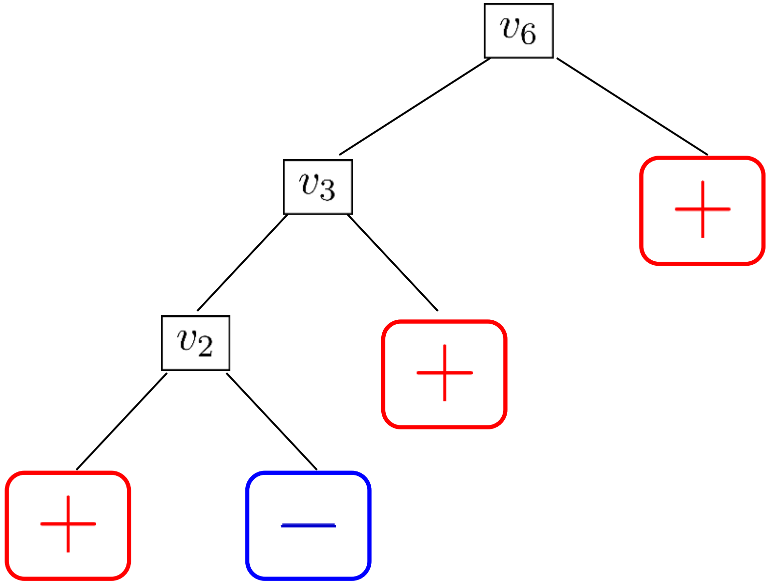
Reduced Error Pruning: Example



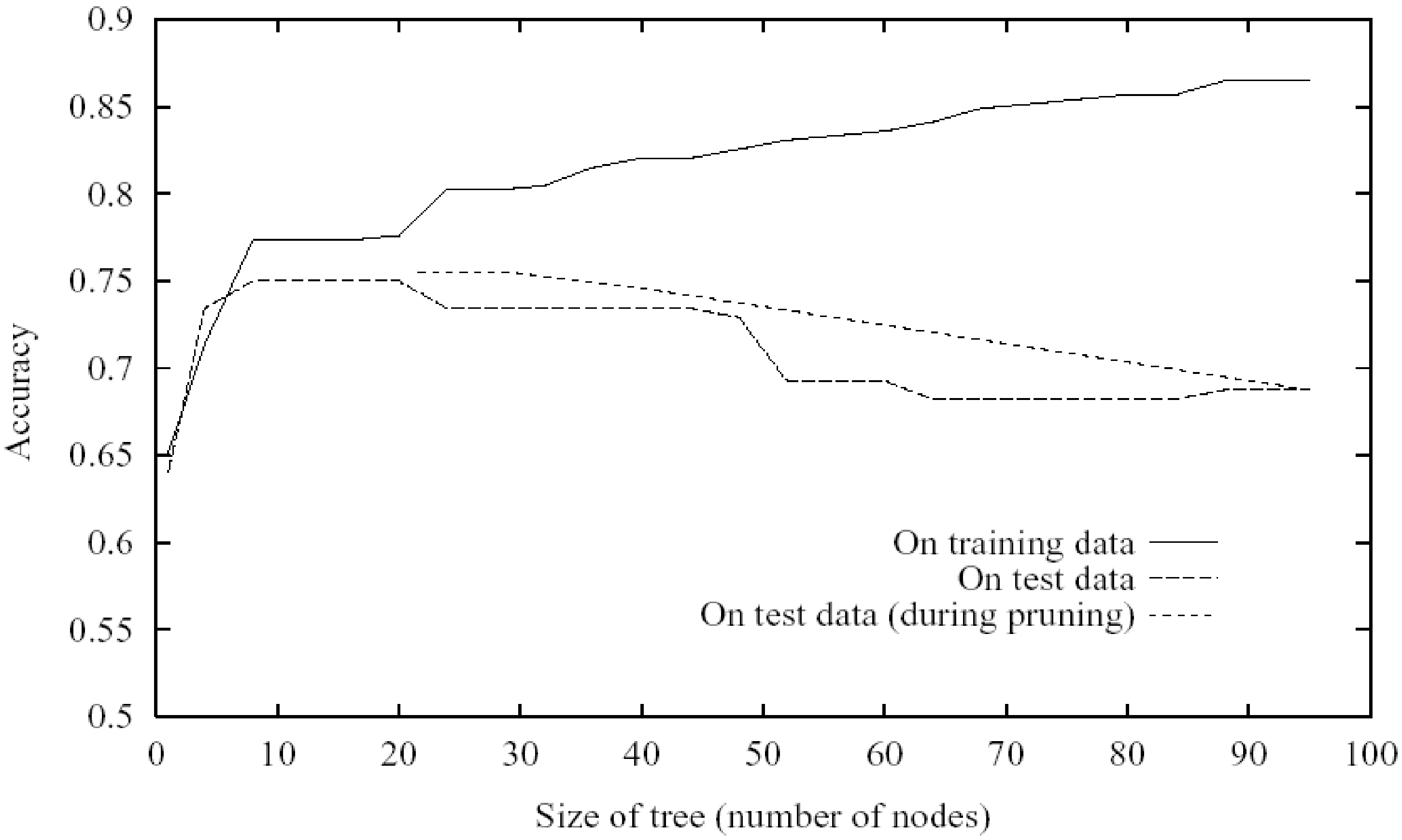
$A_{T_{v_5}} = 6/8 \leq 6/8 = A_{v_5} \implies$ replace T_{v_5} by $\boxed{\begin{array}{c}+\\(6+;\ 2-)\end{array}}$

Reduced Error Pruning: Example

final tree



Effect Of Reduced-Error Pruning



[Foll. Mitchell/97]

Reduced Error Pruning (Quinlan, 1987)

remarks:

- use around $1/3$ of the training data for the validation set (i.e., $\mathcal{D}_2$)
- produces minimally most accurate subtree // not globally smallest!
- drawback: when data available is limited
- C4.5: more advanced method based on *pessimistic error estimates*
  - (Quinlan, 1993)
  - no validation set

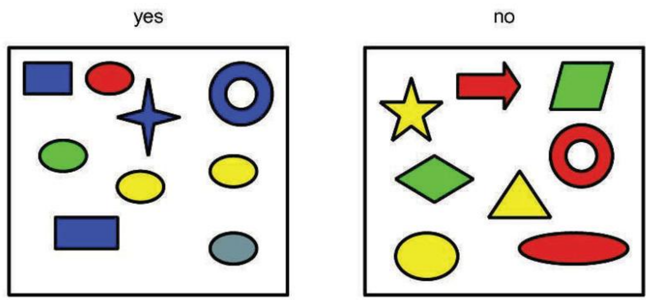
Decision Trees: Summary

- decision trees a popular method for predictive learning from examples
- high expressive power
- “optimal” decision tree: computationally intractable
- trees induced top-down (depth-first) or breadth-first
- key ingredients:
    - split selection relies on heuristics:
        - information gain
        - another common heuristic: Gini gain
- binary decision trees: handling numerical and categorical values

Uncertainty (Unsicherheit)

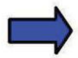
epistemic: epistemische Unsicherheit (hier: Modellunsicherheit)
aleatoric: aleatorische Unsicherheit (hier: Datenunsicheheit)
noisy data: verrauschte Daten

Uncertainty



© K.P. Murphy, 2022

classification task with two classes, say $\{\textsc{Yes}, \textsc{No}\}$

- what is the class of  ?  ?

two types of uncertainty that result in imperfect prediction of the target label:

1. **epistemic** (**model**) **uncertainty** (the model does not know enough)
   - **cause:** lack of knowledge or limited training data

     $\Rightarrow$ **can** be reduced by more training data
2. **aleatoric** (**data**) **uncertainty** (the data is noisy)
   - **cause:** inherent noise or randomness in the data

     $\Rightarrow$ **cannot** be reduced by more training data

Classification Problems: Uncertainty

capture uncertainty using the **conditional probability distribution**:

$$\Pr(y = c | \mathbf{x}; \boldsymbol{\theta}) = f_c(\mathbf{x}; \boldsymbol{\theta})$$

where $f : \mathcal{X} \rightarrow [0,1]^C$ maps inputs to a probability distribution over the C possible output labels

- i.e., $f(\mathbf{x}; \boldsymbol{\theta}) \in \mathbb{R}^C$, $f_c(\mathbf{x}; \boldsymbol{\theta})$ is the c-th entry of $f(\mathbf{x}; \boldsymbol{\theta})$
- for classification problems with $\mathcal{Y} = \{1, \ldots, C\}$

- $f_c(\mathbf{x}; \boldsymbol{\theta})$ returns the probability of class label c

$\Rightarrow$ we require $f_c(\mathbf{x}; \boldsymbol{\theta}) \in [0,1]$ for all $c \in \{1, \ldots, C\}$ and $\sum_{c=1}^{C} f_c(\mathbf{x}; \boldsymbol{\theta}) = 1$

Supervised Learning: Uncertainty

technical remark: it is common to instead require the model to return **unnormalized** values

- can be converted to probabilities using the softmax function defined by

$$\mathrm{softmax}(a) = \mathrm{softmax}((a_1,\ldots,a_C)) \triangleq \left[\frac{e^{a_1}}{\sum_{c'=1}^{C} e^{a_{c'}}},\ldots,\frac{e^{a_C}}{\sum_{c'=1}^{C} e^{a_{c'}}}\right]$$

where $a_c = f_c(\mathbf{x};\boldsymbol{\theta})$ for all $c \in \{1,\ldots,C\}$

$\Rightarrow$ overall model for this case is defined by

$$\Pr(y = c|\mathbf{x};\boldsymbol{\theta}) = \mathrm{softmax}_c(f(\mathbf{x};\boldsymbol{\theta}))$$

Empirical Risk Minimization

Empirical Risk Minimization

goal of supervised learning: train (**compute**) a classification model capable of reliably predicting labels for unseen inputs

- this is **not** yet a formal definition
- performance measure on $\mathcal{D}$: e.g. **misclassification rate** on $\mathcal{D}$ defined by

$$\mathcal{L}(\boldsymbol{\theta}) \triangleq \frac{1}{N} \sum_{n=1}^{N} \mathbb{I}(y_n \neq f(\mathbf{x}_n; \boldsymbol{\theta}))$$

where

$$\mathbb{I}(e) = \begin{cases} 1 & \text{if } e \text{ is true} \\ 0 & \text{if } e \text{ is false} \end{cases}$$

is the binary **indicator function**

remark: assumes all errors are equal

empirical risk: empirisches Risiko
0-1 loss function: 0-1-Verlustfunktion

Empirical Risk Minimization

previous slide: empirical risk is defined by the **zero-one loss function**

$$\ell_{01}(y, \hat{y}) = \mathbb{I}(y \neq \hat{y})$$

	predicted class ($\hat{y}$)	
true class (y)	real chanterelle	false chanterelle
real chanterelle	0	1
false chanterelle	1	0

$\mathbb{I}(y \neq \hat{y})$

harmless

dangerous

assymetric loss function makes more sense, e.g.,

	predicted class ($\hat{y}$)	
true class (y)	real chanterelle	false chanterelle
real chanterelle	0	1
false chanterelle	**100**	0

Empirical Risk Minimization

empirical risk for the case that some errors are more costly than others:

$$\mathcal{L}(\boldsymbol{\theta}) \triangleq \frac{1}{N} \sum_{n=1}^{N} \ell(y_n, f(\mathbf{x}_n; \boldsymbol{\theta}))$$

where ℓ is the **loss function**

- i.e., the **"price"** you need to pay for the prediction of the true label y_n with $\hat{y}_n = f(\mathbf{x}_n; \boldsymbol{\theta})$

$$\mathcal{L}(\boldsymbol{\theta}) \triangleq \frac{1}{N}\sum_{n=1}^{N} \ell(y_n, f(\mathbf{x}_n; \boldsymbol{\theta}))$$

Empirical Risk Minimization

empirical risk minimization: the problem of **model fitting** or **training** is to find the parameters that **minimize** the empirical risk on the training set, i.e., solve

$$\begin{aligned}\hat{\boldsymbol{\theta}} &= \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, \mathcal{L}(\boldsymbol{\theta}) \\ &= \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \frac{1}{N}\sum_{n=1}^{N} \ell(y_n, f(\mathbf{x}_n; \boldsymbol{\theta}))\end{aligned}$$

true goal of supervised learning: minimize the expected loss on **future data** (i.e., data not yet observed)

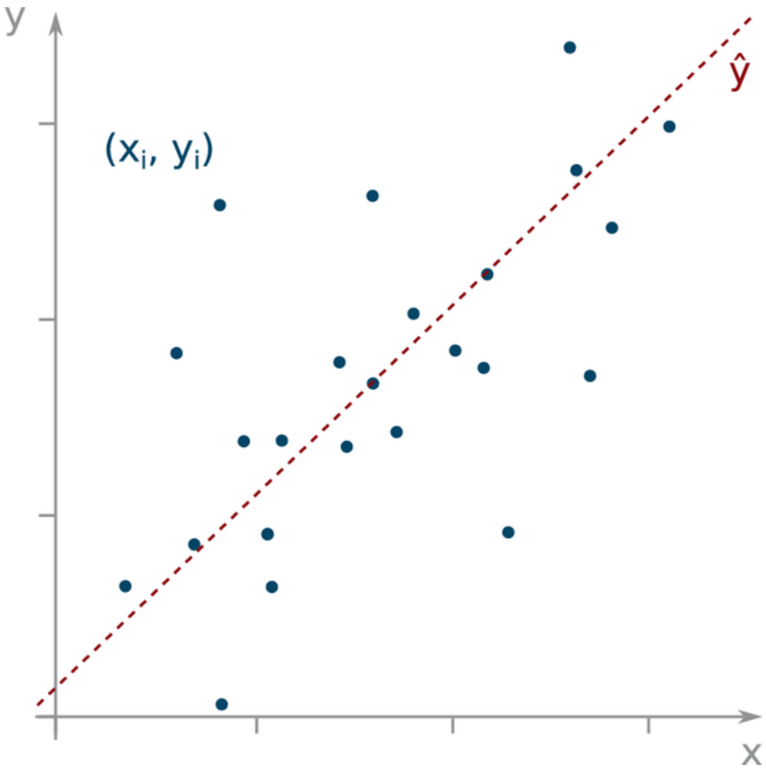
- we want to **generalize**, rather than just perform well on the training set!

Linear Regression

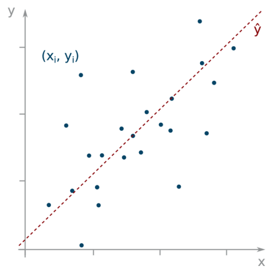
Supervised Learning

two main types:

1. **classification**
2. **regression**



residual:	Residuum
quadratic loss:	quadratischer Verlust
mean squared error (MSE):	mittlerer quadratischer Fehler



Supervised Learning: Regression

regression: $\mathcal{Y} = \mathbb{R}$

goal: predict a real-valued quantity $y \in \mathbb{R}$ instead of a class label from $\{1, \ldots, C\}$

loss function: quadratic loss or ℓ_2 loss defined by

$$\ell_2(y, \hat{y}) = (y - \hat{y})^2$$

- $\hat{y} = f(\mathbf{x}; \boldsymbol{\theta})$
- penalizes large **residuals** $|y - \hat{y}|$ more than small ones
- **most common** choice for regression

empirical risk: defined by the mean squared error or MSE defined by

$$\mathcal{L}(\boldsymbol{\theta}) \triangleq \mathrm{MSE}(\boldsymbol{\theta}) \triangleq \frac{1}{N} \sum_{n=1}^{N} \ell_2(y_n, f(\mathbf{x}_n; \boldsymbol{\theta})) = \frac{1}{N} \sum_{n=1}^{N} (y_n - f(\mathbf{x}_n; \boldsymbol{\theta}))^2$$

least squares: Methode der kleinsten Quadrate

Supervised Learning: Linear Regression

least squares solution: compute θ **minimizing** the MSE, i.e., solve

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \operatorname{MSE}(\boldsymbol{\theta})$$

- optimization problem!

example $(D = 1)$: