

# Entscheidungsbaum-Lernen: Übersicht

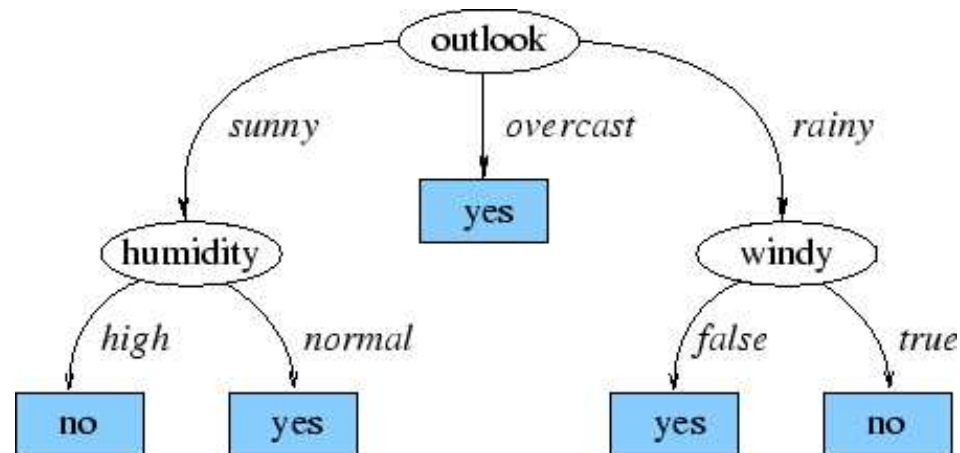
- Entscheidungs bäume als Repräsentationsformalismus
  - Semantik: Klassifikation
- Lernen von Entscheidungsbäumen
  - vollst. Suche vs. TDIDT
  - Tests, Ausdrucksfähigkeit
  - Maße: Information Gain, Information Gain Ratio, Gini Index
  - Overfitting: Pruning
  - weitere Aspekte: Kosten, fehlende Attribute, Trsf. in Regeln

2 Verfahren: ID3 und C4.5

# Repräsentationsformalismus

Ein Entscheidungsbaum ist ein Baum mit:

- Jeder **interne Knoten** enthält einen **Test**
  - für jeden **Ausgang** des Testes gibt es eine Kante.
- Jedes **Blatt** enthält einen **Klassifikationswert**



# Realer Beispielbaum für medizinische Diagnose

(C-Section: Kaiserschnitt)

Gelernt aus medizinischen Befunden von 1000 Frauen

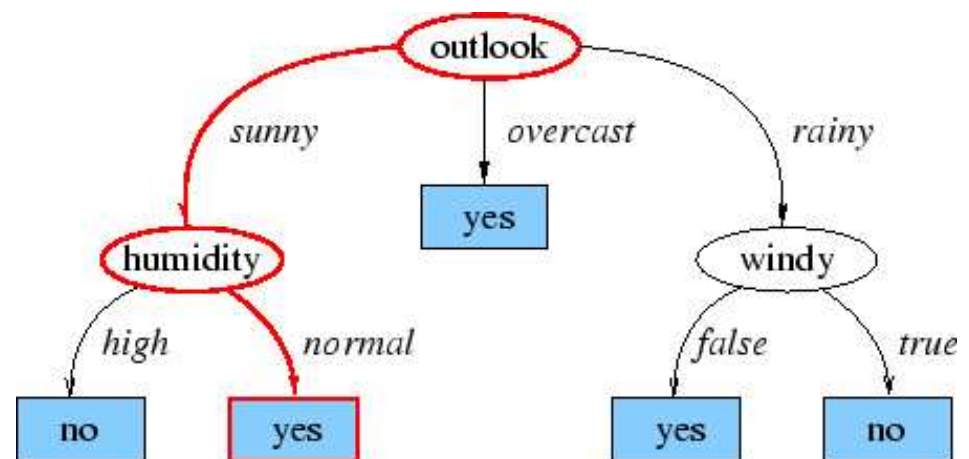
```
[833+,167-] .83+ .17-
Fetal_Presentation = 1: [822+,116-] .88+ .12-
| Previous_Csection = 0: [767+,81-] .90+ .10-
| | Primiparous = 0: [399+,13-] .97+ .03-
| | Primiparous = 1: [368+,68-] .84+ .16-
| | | Fetal_Distress = 0: [334+,47-] .88+ .12-
| | | | Birth_Weight < 3349: [201+,10.6-] .95+ .05-
| | | | Birth_Weight >= 3349: [133+,36.4-] .78+ .22-
| | | Fetal_Distress = 1: [34+,21-] .62+ .38-
| Previous_Csection = 1: [55+,35-] .61+ .39-
Fetal_Presentation = 2: [3+,29-] .11+ .89-
Fetal_Presentation = 3: [8+,22-] .27+ .73-
```

# Klassifikation

Beginnend mit der Wurzel:

- Wenn in innerem Knoten:
  - Führe Test aus
  - Verzweige entsprechend Testausgang und gehe zum Anfang
- Wenn in Blatt:
  - Gib den Klassifikationswert als Ergebnis zurück

outlook	sunny
temperature	hot
humidity	normal
windy	false
play	?



# Lernen von Entscheidungsbäumen

gegeben: Menge von Daten (Attribut-Wertpaare) zusammen mit der Zielklasse (binär)

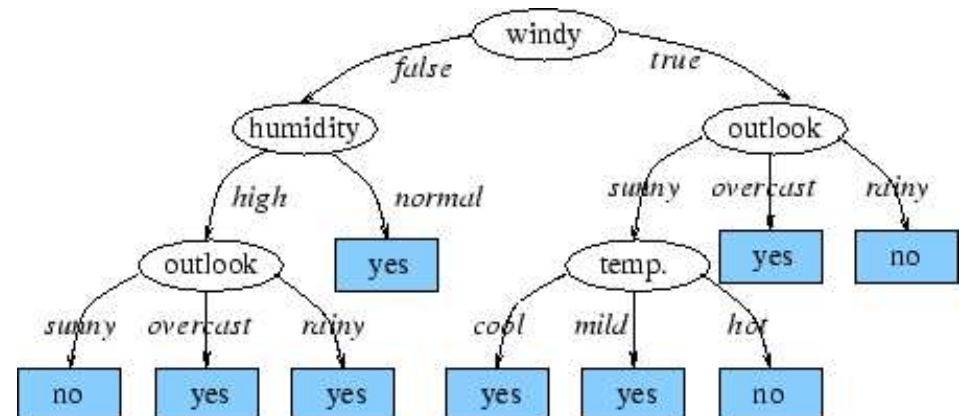
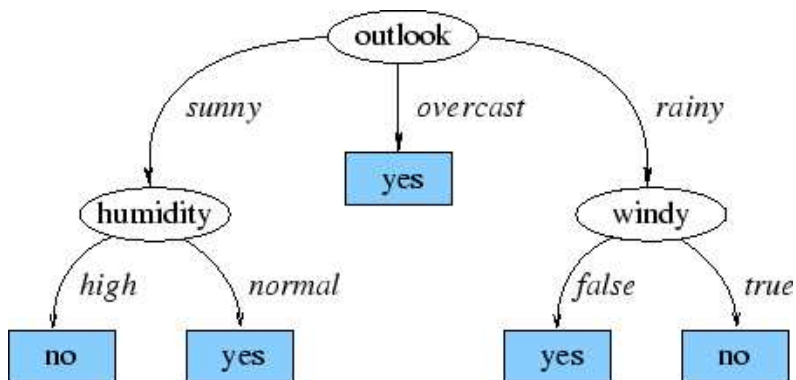
Naiver Ansatz (so wird's NICHT gemacht!)

- Erzeuge alle möglichen Entscheidungsbäume und wähle den **besten**.

? Wie viele Entscheidungsbäume sind zu durchsuchen?

? Was heißt hier 'bester'?

*Welcher Baum ist besser? Warum?*



# Top-Down Induktion of Decision Trees (TDIDT)

- Erzeuge Wurzelknoten  $n$
- Berechne TDIDT( $n$ ,Beispielmenge)

TDIDT( $n$ ,Beispielmenge):

1. Wenn der Baum genügend groß ist: weise dem Blatt  $n$  eine Klasse zu
2. Sonst:
  - (a) Bestimme “besten” Test (Attribut)  $A$  für die Beispielmenge
  - (b) Weise dem Knoten  $n$  den Test  $A$  zu
  - (c) Bestimme Menge  $TA$  aller Testausgänge (Werte) von  $A$
  - (d) Für jeden Testausgang  $t \in TA$ :
    - erzeuge einen neuen Knoten  $n_t$
    - erzeuge eine Kante von  $n$  nach  $n_t$  und beschrifte sie mit  $t$
    - initialisiere  $Bsp_t = \emptyset$
  - (e) Für jedes Beispiel  $b$  aus der Beispielmenge:
    - Wende den Test  $A$  auf  $b$  an und bestimme den Testausgang  $t$
    - Füge  $b$  zur Menge  $Bsp_t$  hinzu
  - (f) Für jeden Nachfolgeknoten  $n_t$  von  $n$ : Berechne TDIDT( $n_t$ ,  $Bsp_t$ )



# Fragestellungen

---

1. Was sind Tests?
2. Wie bestimmt man den “besten” Test?
3. Wann ist der Baum genügend groß?

---

## Ziele

- entstehender Baum sollte möglichst klein sein
- Beispiele sollten möglichst gleichmäßig über den Baum verteilt sein



# Frage 1: Tests

Wir benutzen beim Lernen von Entscheidungsbäumen ausschließlich die folgenden Tests:

für nominale Attribute:

- Test ist das Attribut selbst
- Testausgang ist jeder der möglichen Attributwerte
- *Beispiel: outlook, Ausgänge: sunny, overcast, rainy*

für numerische Attribute:

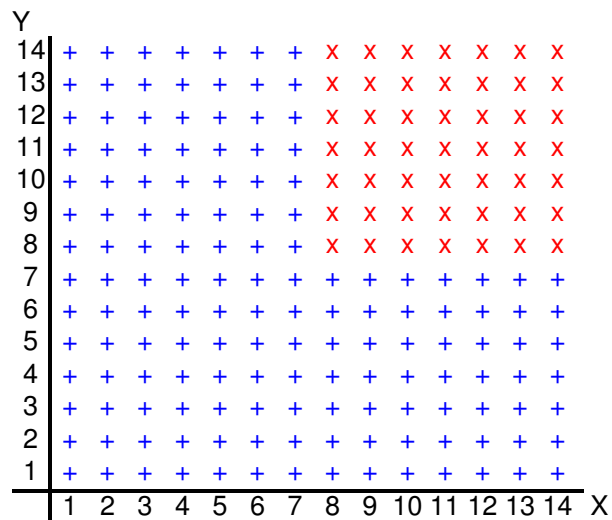
- Test ist Vergleich mit Konstante
- 2 Ausgänge
- *Beispiel: kontostand  $\leq$  1000, Ausgänge: yes, no*

# Repräsentationsfähigkeit

## Warum keine anderen Tests?

- Vergleich zweier Attribute
  - Bsp:  $\text{meinKontostand} > \text{KontostandDagobert}$
- Logische Verküpfungen
  - Bsp:  $\text{outlook} = \text{sunny} \wedge \text{humidity} = \text{high}$
- Arithmetische Verküpfungen
  - Bsp:  $\text{meinKontostand} * \text{meinAlter} > \log(\text{KontostandDagobert})$

Beispiel:



Baum mit atomaren Tests:

$X \leq 7 : +$   
 $X > 7 :$   
|  $Y \leq 7 : +$   
|  $Y > 7 : x$

Baum mit komplexen Tests:

$(X > 7 \wedge Y > 7) : x$   
 $\neg(X > 7 \wedge Y > 7) : +$

# Repräsentationsfähigkeit (cont.)

- Wie viele atomare Tests gibt es für das Beispiel?
  - Wie viele komplexe Tests gibt es für das Beispiel?
- 

Wie kann man folgende Formeln ausdrücken?

- $\wedge, \vee, \text{XOR}$
- $(A \wedge B) \vee (C \wedge \neg D \wedge E)$
- $M$  von  $N$

# Repräsentationsfähigkeit (cont.)

- Jeder atomare Test legt eine **achsenparallele Hyperebene** in den Attributraum
  - Logische Verküpfungen können durch Baum**struktur** ausgedrückt werden
    - kein Verlust an Ausdrucksfähigkeit
    - Suche wird einfacher und schneller, wenn nur atomare Tests verwendet werden.
- Ein Entscheidungsbaum zerteilt den Attributraum in **achsenparallele Hyperrechtecke**
- Je weiter oben ein Test im Baum steht, desto größeren Einfluß hat er

# Frage 2: Wie bestimmt man “besten” Test?

- Heuristische Maße gesucht
  - Gute Trennung der Beispiele
    - \* ergibt kleinen Baum
  - Gleichmäßige Aufteilung der Beispiele
- Eingabe:
  - $S$ : Menge von Trainingsbeispielen
  - $A$ : ein Test
- Ergebnis: Zahl, je größer, desto besser ist der Test

# Auswahl des besten Tests: Information-Gain

$S$ : Menge von Trainingsbeispielen,  $A$ : ein Test

$$\textit{Gain}(S, A) \equiv \textit{Entropie}(S) - \sum_{v \in \textit{Testausgang}(A)} \frac{|S_v|}{|S|} \textit{Entropie}(S_v)$$

$$\textit{Entropie}(S) \equiv -\frac{|S_{\oplus}|}{|S|} \log_2 \frac{|S_{\oplus}|}{|S|} - \frac{|S_{\ominus}|}{|S|} \log_2 \frac{|S_{\ominus}|}{|S|}$$

$S_{\oplus}$  ist die Menge aller positiven Beispiele in  $S$ ,  $S_{\ominus}$  die der negativen

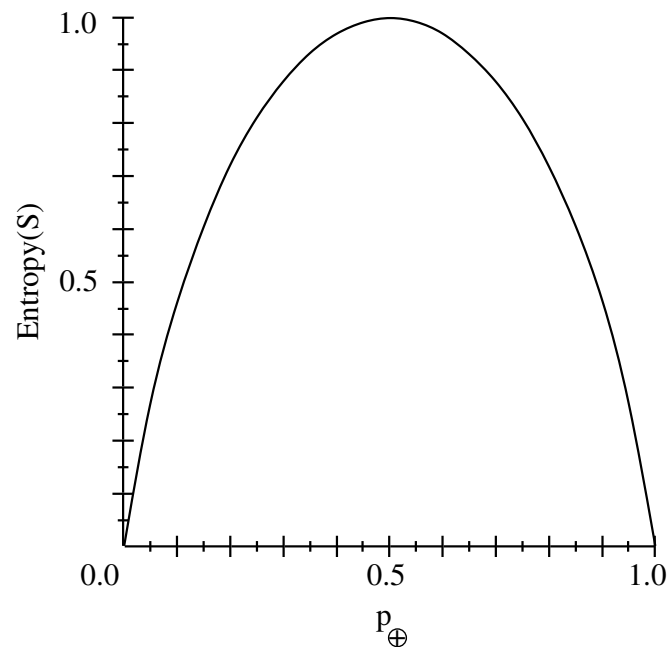
$S_v$  ist die Teilmenge von  $S$ , für die der Test  $A$  den Ausgang (Wert)  $v$  hat

$\textit{Gain}(S, A)$  = erwartete Verringerung der Entropie nach Partitionierung bzgl.  $A$

# Entropie

$$\text{Entropie}(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

$$(p_{\oplus} = \frac{|S_{\oplus}|}{|S|}, p_{\ominus} = \frac{|S_{\ominus}|}{|S|})$$



- Entropie mißt die ‘Unreinheit’ von  $S$

# Entropie

$Entropie(S)$  = erwartete Anzahl von Bits die benötigt werden, um die Klassifikation ( $\oplus$  oder  $\ominus$ ) eines zufällig gezogenen Beispiels aus  $S$  zu kodieren (unter optimaler, kürzester Kodierung)

Warum?

Informationstheorie: optimale Kodierung benötigt  $-\log_2 p$  Bits um eine Nachricht mit der Wahrscheinlichkeit  $p$  zu kodieren

→ erwartete Anzahl von Bits, um  $\oplus$  oder  $\ominus$  eines beliebig gezogenen Beispiels aus  $S$  zu kodieren:

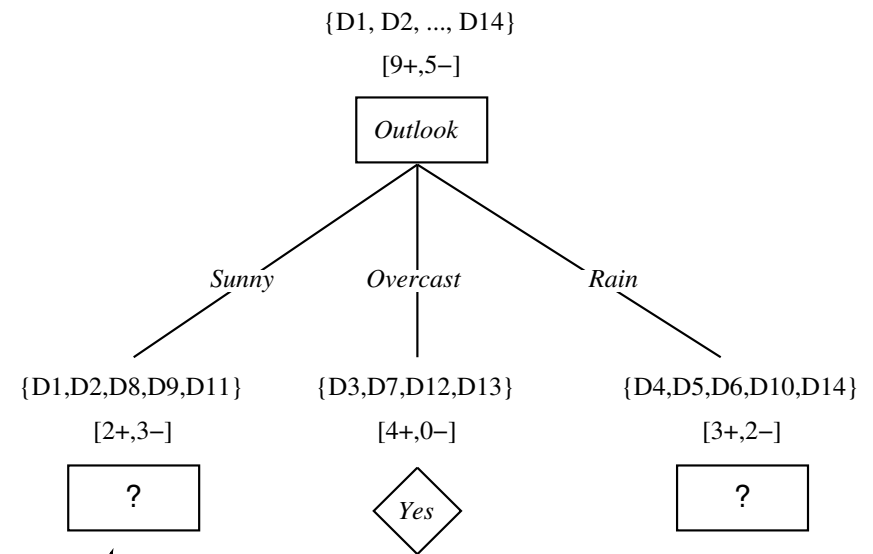
$$p_{\oplus}(-\log_2 p_{\oplus}) + p_{\ominus}(-\log_2 p_{\ominus})$$

$$Entropie(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$



# Beispiel

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No



Which attribute should be tested here?

$$S_{\text{sunny}} = \{D1, D2, D8, D9, D11\}$$

$$\text{Gain}(S_{\text{sunny}}, \text{Humidity}) = .970 - (3/5) 0.0 - (2/5) 0.0 = .970$$

$$\text{Gain}(S_{\text{sunny}}, \text{Temperature}) = .970 - (2/5) 0.0 - (2/5) 1.0 - (1/5) 0.0 = .570$$

$$\text{Gain}(S_{\text{sunny}}, \text{Wind}) = .970 - (2/5) 1.0 - (3/5) .918 = .019$$

# ID3

Das erste Lernverfahren ist fertig!

- Maß zur Auswahl des besten Tests: *Gain*
- Frage 3: Wann ist der Baum genügend groß?
  - Wenn alle Beispiele in Beispielmenge ein und dieselbe Klassifikation besitzen, dann höre auf und weise dem Blatt diese Klasse zu.
- Kein Backtracking
  - Lokale Minima...
- Statistikbasierte Suchentscheidungen
  - Robust gegenüber verrauschten Daten...
- Induktiver Bias: “bevorzuge kleine Bäume”
  - **Occam's Razor**: Wähle kleinste Hypothese, die die Daten widerspiegelt
    - \* Warum? Was ist so besonders an kleinen Hypothesen?

# Probleme mit ID3

- Wenn der Wertebereich eines Attributs sehr groß ist (d.h. sehr viele Testausgänge existieren), wird *Gain* dieses auswählen
  - Extrembeispiele: Kundennummer, Geburtsdatum, Name
- Baum wird aufgebaut, bis auch das letzte Beispiel abgedeckt ist
  - Was ist, wenn Daten fehlerhaft (“verrauscht”) sind?

# Auswahl des besten Tests: Information-Gain-Ratio

Besseres Maß: *GainRatio*

$$\textit{GainRatio}(S, A) \equiv \frac{\textit{Gain}(S, A)}{\textit{SplitInformation}(S, A)}$$

$$\textit{SplitInformation}(S, A) \equiv - \sum_{v \in \textit{Werte}(A)} \frac{|S_v|}{|S|} \log_2 \frac{|S_v|}{|S|}$$

# Auswahl des besten Tests: Gini-Index

Wähle das Attribut mit minimalem

$$gini(S, A) \equiv \sum_{v \in \text{Werte}(A)} \frac{|S_v|}{|S|} g(S_v)$$

$$g_i(S) = 1 - \left( \frac{|S_{\oplus}|}{|S|} \right)^2 - \left( \frac{|S_{\ominus}|}{|S|} \right)^2$$

$S_{\oplus}$  ist die Menge aller positiven Beispiele in  $S$ ,  $S_{\ominus}$  die der negativen

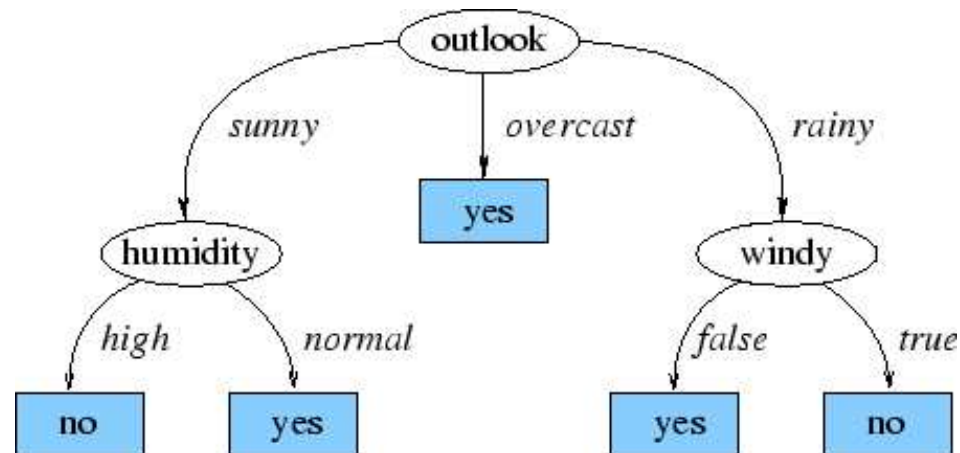
$S_v$  ist die Teilmenge von  $S$ , für die der Test  $A$  den Ausgang (Wert)  $v$  hat

# Overfitting bei Entscheidungsbäumen

Betrachte folgendes verrauschte Beispiel #15:

*Sunny, Hot, Normal, Strong, PlayTennis = No*

Was passiert mit dem vorhin erzeugten Baum?



# Overfitting

Betrachte Fehler von Hypothesen  $h$  über

- Trainingsdaten:  $error_{train}(h)$
- gesamter Verteilung  $\mathcal{D}$  der Daten:  $error_{\mathcal{D}}(h)$

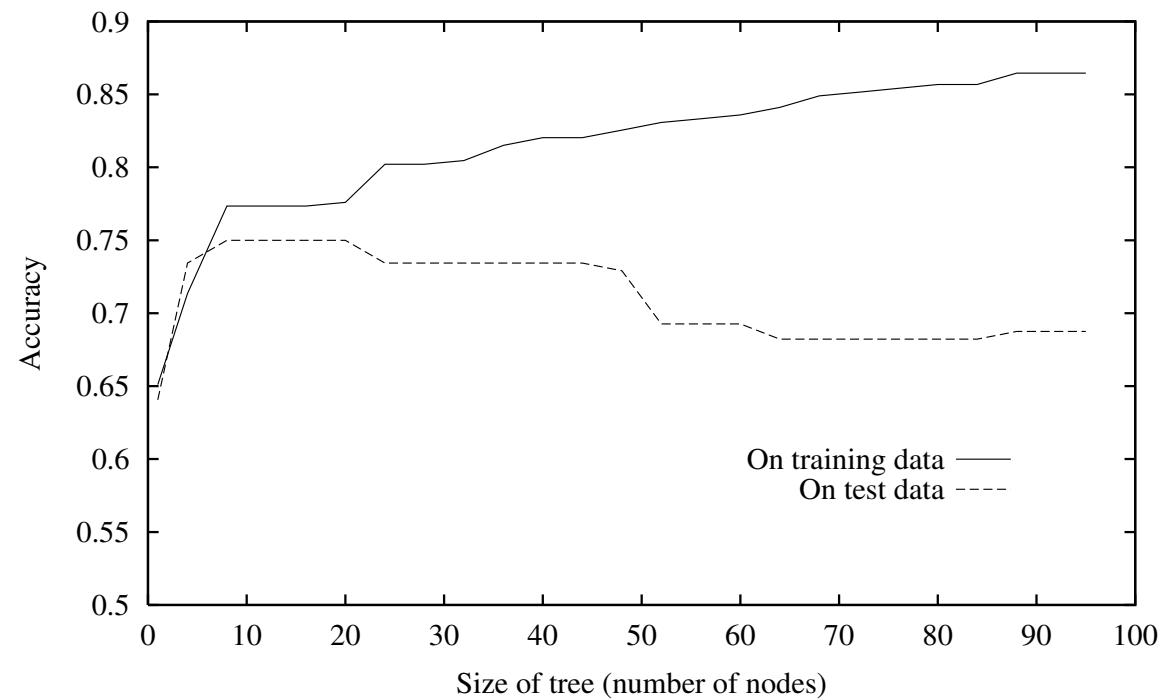
Hypothese  $h \in H$  **overfits** eine Trainingsmenge, wenn es eine alternative Hypothese  $h' \in H$  gibt mit

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

und

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

# Overfitting beim Entscheidungsbaumlernen





# Overfitting: Ein Beispiel

---

# 'Gehorsamer' Entscheidungsbaum

```
x <= 6
|   x <= 1.51
|   |   x <= 1: - (2.0)
|   |   x > 1: + (2.0)
|   x > 1.51: - (5.0)
x > 6: + (3.0)
```

# Wie kann man Overfitting verhindern?

**Pre-Pruning** Aufhören, wenn Verfeinerung keine statistisch signifikante Verbesserung mehr bringt

**Post-Pruning** Erzeuge vollständigen Baum, danach verkleinere ihn wieder

Auswahl des “besten” Baums:

- Miß Verhalten auf Trainingsdaten
- Miß Verhalten auf separaten Validationsdaten  
→ Teile ursprüngliche Daten auf!!!
- MDL: minimiere  $size(tree) + size(misclassifications(tree))$

MDL: Minimum Description Length Principle

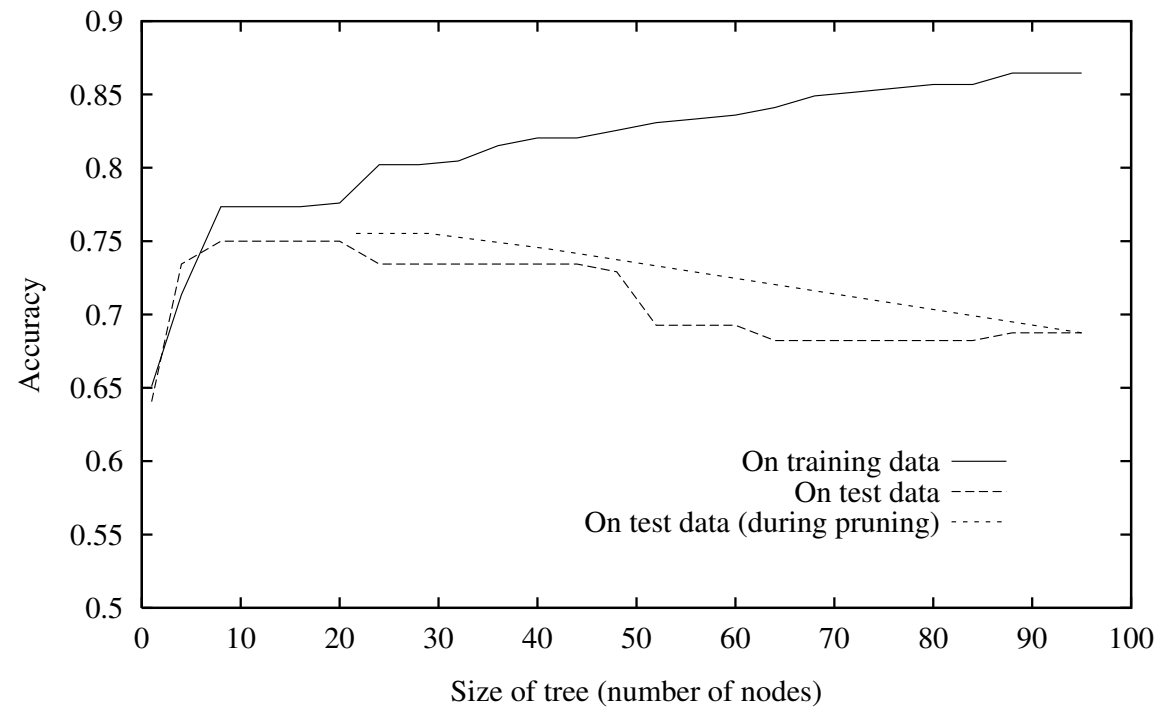
# Reduced-Error Pruning

Teile Daten auf in *Trainings-* und *Validierungsmenge*

Solange bis Pruning keine Verbesserung mehr bringt:

1. Für jeden Knoten  $n$  des Baumes
  - entferne  $n$  (und alle darunter liegenden Knoten) und bestimme Güte dieses Baumes bzgl. der Validierungsmenge
2. Entferne denjenigen Knoten, der die meiste Verbesserung auf der Validierungsmenge bewirkt
  - erzeugt **kleinste** Version des **genauesten** Baumes
  - Was, wenn die Datenmenge beschränkt ist?

# Effekt des Reduced-Error Prunings



# C4.5 / J48

C4.5 = TDIDT mit Maß **Information-Gain-Ratio**, vollständigem Aufbau des Baumes und dann **Reduced-Error-Pruning**

weitere Features (siehe weitere Folien)

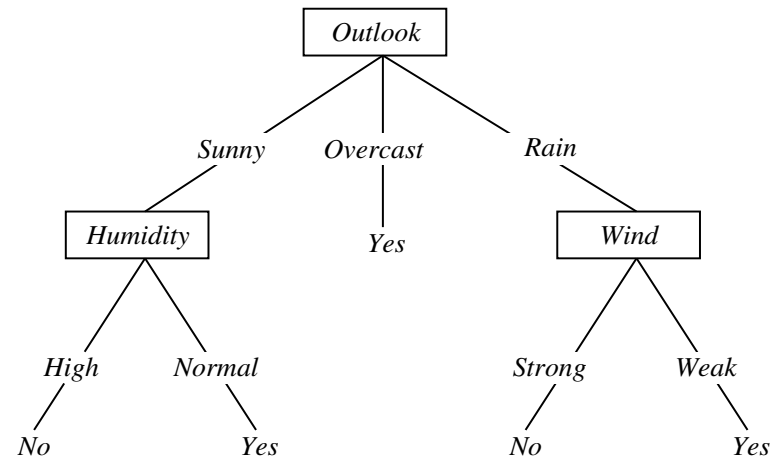
- Kontinuierliche Attribute möglich
- Unbekannte Attribute (unvollständige Beispiele) möglich
- Erzeugung von Regelmengen statt Bäumen
- Integration einiger Preprocessing- und Metalernschritte (siehe dort): Attribute Grouping, Windowing

# Regel-Post-Pruning

1. Konvertiere Baum in äquivalente Menge von Regeln
2. Prune jede Regel unabhängig von den anderen
  - Einfach Bedingungen herausstreichen
3. Sortiere Endregeln in gewünschte Reihenfolge für Benutzung
  - Jetzt können mehrere Regeln gleichzeitig feuern

Häufig genutzte Methode (z.B. als Option in C4.5)

# Konvertiere Baum in Regelmenge



IF  $(Outlook = Sunny) \wedge (Humidity = High)$   
THEN  $PlayTennis = No$

IF  $(Outlook = Sunny) \wedge (Humidity = Normal)$   
THEN  $PlayTennis = Yes$

...



# Details: Kontinuierliche Attribute

Kontinuierliche Attribute werden mit Konstante verglichen

- Mit welcher? (Es gibt überabzählbar unendlich viele)

<i>Temperature:</i>	40	48	60	72	80	90
<i>PlayTennis:</i>	No	No	Yes	Yes	Yes	No

- Es genügt, für jeden in den Daten vorkommenden Wert einen Test zu generieren
  - Warum?
  - Welchen?

# Details: Unbekannte Attributwerte

Was wenn Wert von Attribut  $A$  fehlt?

Benutze Trainingsbeispiele  $s$  trotzdem: Wenn der Knoten  $n$  das Attribut  $A$  testet, dann

- Nimm an,  $s$  hätte für  $A$  denjenigen Wert, der unter allen anderen Beispielen für Knoten  $n$  am häufigsten für  $A$  vorkommt
- Weise  $A$  den Wert zu, den die meisten Beispiele mit der gleichen Klassifikation wie  $s$  haben
- Weise Wahrscheinlichkeit  $p_i$  für jeden möglichen Wert  $v_i$  von  $A$  zu
  - Propagiere ‘Anteile’  $p_i$  der Beispiele in die Teilbäume
    - \* Beispiel: Attribut boolean, Anteil ‘+’ = 60%, ‘-’ = 40%  
Propagiere Beispiel mit Gewicht 0,6 in Zweig für ‘+’, mit Gewicht 0,4 in Zweig für ‘-’

Klassifikation erfolgt in gleicher Weise.

# Details: Attribute mit Kosten

## Beispiele

- Medizinische Diagnose, *BloodTest* kostet \$150
- Robotik, *Width\_from\_1ft* kostet 23 Sekunden.

Wie kann man einen konsistenten Baum mit geringsten Kosten lernen?

Ansatz: ersetze *Gain* bspw. durch

- Tan and Schlimmer (1990)

$$\frac{Gain^2(S, A)}{Cost(A)}.$$

- Nunez (1988)

$$\frac{2^{Gain(S,A)} - 1}{(Cost(A) + 1)^w}$$

wobei  $w \in [0, 1]$  den Einfluß der Kosten beschreibt