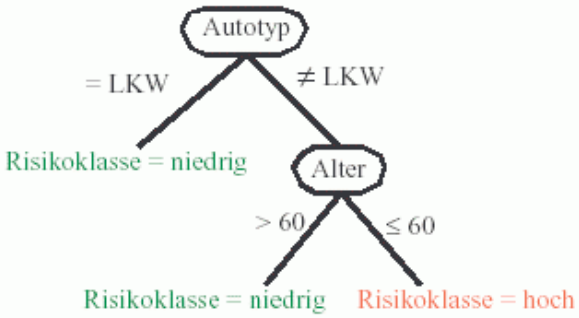
3. Entscheidungsbäume

Verfahren zum Begriffslernen (Klassifikation)

Beispiel:

ID	Alter	Autotyp	Risiko
1	23	Familie	hoch
2	17	Sport	hoch
3	43	Sport	hoch
4	68	Familie	niedrig
5	32	LKW	niedrig

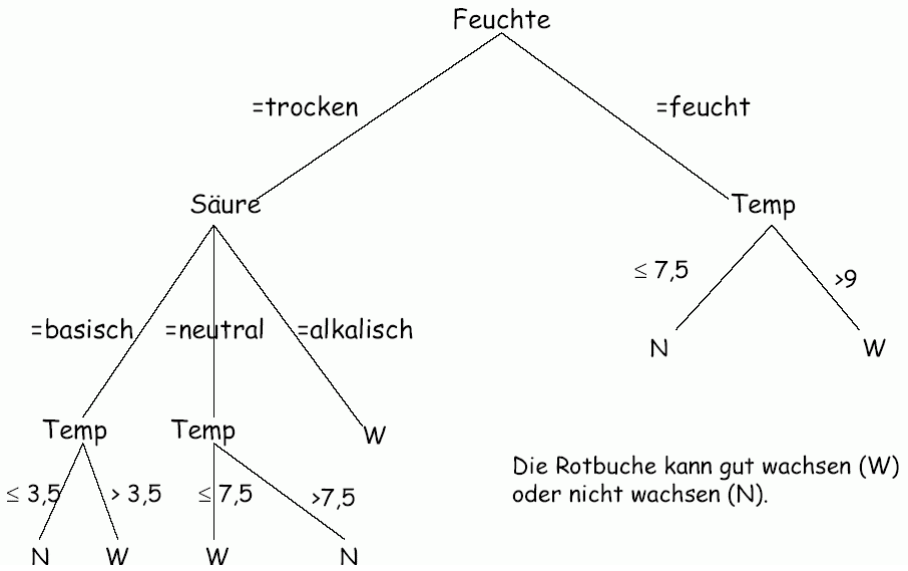


finden explizites Wissen

Entscheidungsbäume sind für die meisten Benutzer verständlich

(aus Böhm 2003)

weiteres Beispiel:



(aus Morik 2002)

- rekursive Partition der Untersuchungseinheiten
- Situation:

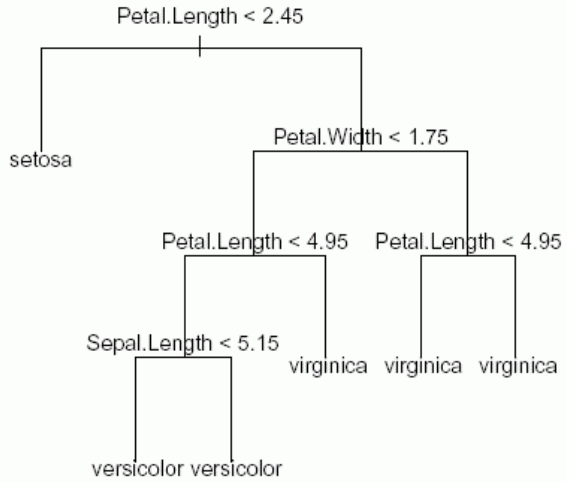
  Y Zielvariable

  $X_1, \ldots, X_p$ erklärende Variable
- Terminologie:

  falls Y kategoriell $\rightsquigarrow$ Klassifikationsbaum

  falls Y stetig $\rightsquigarrow$ Regressionsbaum

Beispiel 1. *Iris-Daten: Species: 3 Irisarten (Setosa, Versicolor, Virginica), 4 erklärende Variable: Petal Length, Petal Width, Sepal Length, Sepal Width*



IF Petal Length < 2.45 THEN SPECIES = SETOSA

IF Petal Length > 2.45 AND
Petal Width < 1.75 AND
Petal Length > 4.95
THEN SPECIES = VIRGINICA

Klassifikationsbaum bzw. Regressionsbaum ist hierarchische Sammlung vieler solcher Regeln

(aus Wilhelm 2001)

Grundbegriffe

- Ein *Entscheidungsbaum* ist ein Baum mit folgenden Eigenschaften:
  - ein innerer Knoten repräsentiert ein Attribut,
  - eine Kante repräsentiert einen Test auf dem Attribut des Vaterknotens,
  - ein Blatt repräsentiert eine der Klassen.
- Konstruktion eines Entscheidungsbaums

  anhand der Trainingsmenge

  Top-Down
- Anwendung eines Entscheidungsbaums

  Durchlauf des Entscheidungsbaum von der Wurzel zu einem der Blätter

  ➡ eindeutiger Pfad

  Zuordnung des Objekts zur Klasse des erreichten Blatts

Konstruktion eines Entscheidungsbaums

Basis-Algorithmus

- Anfangs gehören alle Trainingsdatensätze zur Wurzel.
- Das nächste Attribut wird ausgewählt (Splitstrategie).
- Die Trainingsdatensätze werden unter Nutzung des Splitattributs partitioniert.
- Das Verfahren wird rekursiv für die Partitionen fortgesetzt.

➡ lokal optimierender Algorithmus

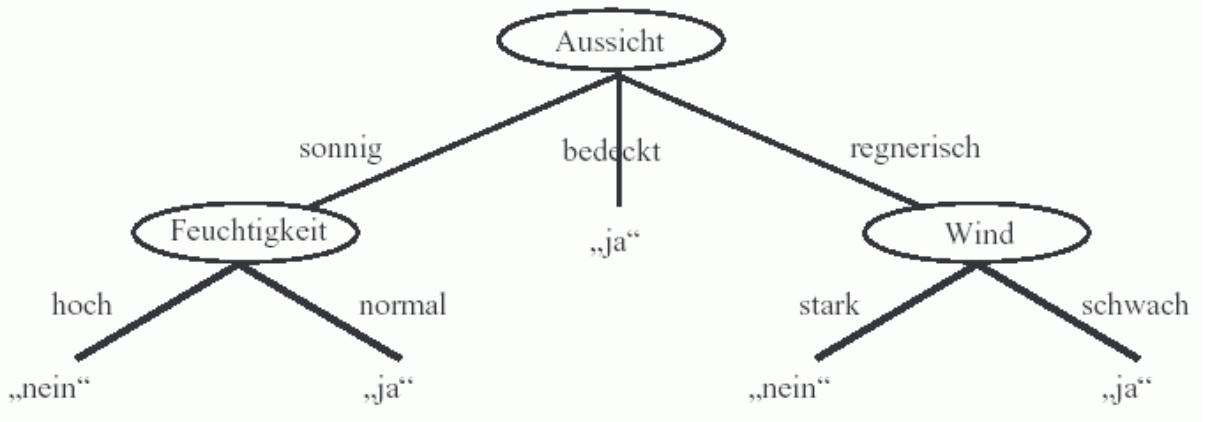
Abbruchbedingungen

- keine weiteren Splitattribute
- alle Trainingsdatensätze eines Knotens gehören zur selben Klasse

Beispiel:

Tag	Aussicht	Temperatur	Feuchtigkeit	Wind	Tennispielen
1	sonnig	heiß	hoch	schwach	nein
2	sonnig	heiß	hoch	stark	nein
3	bedeckt	heiß	hoch	schwach	ja
4	regnerisch	mild	hoch	schwach	ja
5	regnerisch	kühl	normal	schwach	ja
6	regnerisch	kühl	normal	stark	nein
7	. . .	. . .	. . .	. . .	. . .

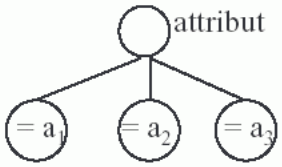
Ist heute ein Tag zum Tennisspielen?

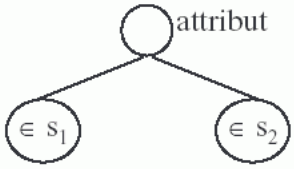


Typen von Splits

Kategorische Attribute

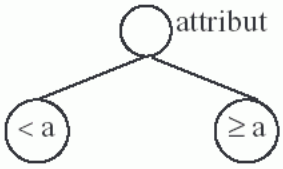
- Splitbedingungen der Form „*attribut* = *a*“ or „*attribut* ∈ *set*“
- viele mögliche Teilmengen





Numerische Attribute

- Splitbedingungen der Form „*attribut* < *a*“
- viele mögliche Splitpunkte



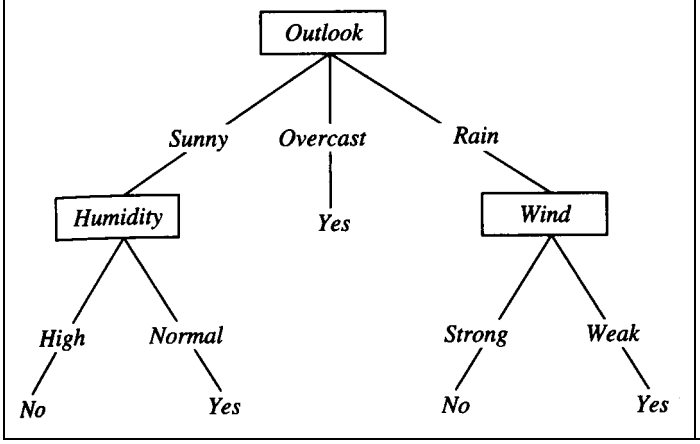
(aus Böhm 2003)

Wann sind Entscheidungsbäume sinnvoll einsetzbar?

- Instanzen werden durch eine feste Menge von Attributen und ihre Werte beschrieben
- die Zielfunktion hat diskrete Werte
- die Trainingsdaten dürfen Fehler enthalten
  Entscheidungsbaummethoden sind "robust"
- die Trainingsdaten dürfen missing values enthalten
- disjunktive Beschreibungen dürfen vorkommen

Entscheidungsbaum beschreibt Disjunktion von Konjunktionen

Beispiel:



entspr. logischer Ausdruck:

$$(Outlook = Sunny \wedge Humidity = Normal)$$
$$\vee \quad (Outlook = Overcast)$$
$$\vee \quad (Outlook = Rain \wedge Wind = Weak)$$

(leicht auf Zielfunktion mit mehr als 2 Werten verallgemeinerbar)

(aus Mitchell 1997)

Basisalgorithmus: ID3 (Quinlan 1986)

darauf aufbauende, spätere Variante: C4.5 (Quinlan 1993)

- konstruiert Baum "top-down" (siehe "Basis-Algorithmus" oben)
- zur Wahl des jeweils nächsten Split-Attributs wird für jedes verbliebene Attribut ein statistischer Test durchgeführt, um zu ermitteln, wie gut das jeweilige Attribut *allein* in der Lage ist, die Trainingsdaten zu klassifizieren.
- "Greedy-Algorithmus" – kein Backtracking

Detaillierter (nach Klawonn 2004):

Entscheidungsbaumkonstruktion

1. Man wähle ein (günstiges) Attribut als Knoten und erzeuge für jeden möglichen Wert des Attributs einen Nachfolgerknoten.
   - 1.1 Kommen im Datensatz nur Objekt einer Klasse mit der bis dahin spezifizierten Attributwertkombination vor, wird dieser Nachfolgerknoten zum Blatt und mit der entsprechenden Klasse gekennzeichnet.
   - 1.2 Kommt im Datensatz überhaupt kein Objekt mit der bis dahin spezifizierten Attributwertkombination vor, wird dieser Nachfolgerknoten zum Blatt und mit einem Fragezeichen gekennzeichnet.
   - 1.3 Kommen im Datensatz Objekte unterschiedlicher Klassen mit der bis dahin spezifizierten Attributwertkombination vor, muss in diesem Knoten ein weiteres Attribut gewählt werden, sofern noch weitere Attribute zur Verfügung stehen. Ansonsten wird auch dieser Knoten zum Blatt mit Fragezeichen.
2. Behandle alle (Nicht-Blattknoten) aus 1.3 wieder wie unter 1. Bei der Attributwahl darf kein Attribut gewählt werden, das in einem Knoten oberhalb des zu behandelnden Knotens auftritt.

- Im Schritt 1.3 wird üblicherweise anstelle eines Fragezeichens die Klasse eingetragen, die am häufigsten in dem entsprechenden Knoten vorkommt.
- Teilweise wird das sehr scharfe Kriterium 1.1 etwas abgeschwächt, indem einem Knoten schon eine Klasse zugeordnet wird, wenn zwar verschiedene Klassen unter den diesem Knoten zugeordneten Objekten vorkommen, aber eine große Mehrheit (z.B. mindestens 95%) der Objekte in dem Knoten aus einer Klasse stammen.

Wie wählt man jeweils ein *günstiges* Attribut für einen Knoten im Entscheidungsbaum?

Ideal wäre es, wenn man mit einem einzigen Attribut bereits die Klasse vorhersagen könnte, was in der Praxis aber so gut wie nie vorkommt.

Man verfolgt daher eine Greedy-Strategie und wählt in jedem Schritt das Attribut, das den größten Informationsgewinn liefert.

Informationsgewinn wird dabei als Reduktion der Entropie verstanden.

Die **Entropie** kann als ein Maß für den Informationsgewinn verstanden werden, den eine konkrete Realisierung eines Zufallsexperiments liefert.

Wird beispielsweise eine (faire) Münze geworfen, so hat man vorher eine Chance von 50% das Ergebnis korrekt vorherzusagen. In diesem Fall liefert der Münzwurf ein binäres Ereignis (Kopf (0) oder Zahl (1)).

Der Informationsgewinn ist bei der Kenntnis des Ausgangs (im Mittel) ein halbes Bit, da man in 50% der Fälle ja das richtige Bit errät.

Bei einer sehr asymmetrischen Verteilung ist der Informationsgewinn (die Entropie) deutlich geringer, da man durch Raten des wahrscheinlichsten Versuchsausgangs das Ergebnis häufiger korrekt vorhersagen kann.

Im extremen Beispiel eines sicheren Ereignisses (z.B. eine manipulierte Münze, die immer Zahl zeigt, egal wie man sie wirft), kann das Ergebnis immer korrekt vorhergesagt werden. Der Informationsgewinn bzw. die Entropie ist 0.

(H1) Entropie ist eine (Klasse von) reellwertige(n) Funktion(en) $H(\mathbf{p})$, die für alle Wahrscheinlichkeitsverteilungen $\mathbf{p} \in \mathbb{R}^n$ definiert ist ($\sum_{i=1}^{n} p_i = 1$ und $p_i \geq 0$ für alle $i = 1, \ldots, n$).

(H2) Die Entropie ist niemals negativ, d.h. $H(\mathbf{p}) \geq 0$, wobei die Gleichheit nur im Falle sicherer Ereignisse ($p_i = 1$ für ein i) auftritt.

(H3) Erhält man die Wahrscheinlichkeitsverteilung $\mathbf{q}$ aus der Wahrscheinlichkeitsverteilung $\mathbf{p}$, indem man alle Komponenten, die 0 sind (alle unmöglichen Ereignisse) entfernt, so gilt $H(\mathbf{p}) = H(\mathbf{q})$. (Unmögliche Ereignisse haben keine Auswirkung auf die Entropie.)

(H4) $H(1/n, \ldots, 1/n)$ ist monoton wachsend mit n, d.h., eine Gleichverteilung über mehr Ereignisse hat eine größere Entropie.

(H5) $H(\mathbf{p})$ ist eine stetige Funktion in $\mathbf{p}$, d.h., die Entropie ändert sich nicht sprungartig.

(H6) Bei einer Hintereinanderschaltung/ Schachtelung/Aufteilung von Experimenten kann die Entropie über eine geeignete gewichtete Summe berechnet werden.

Beispiel (für Axiom (H6)): Es werden zwei (faire) Würfel geworfen.

- Die Entropie ergibt sich aus der Verteilung $\mathbf{p} = (1/36, 2/36, 3/36, 4/36, 5/36, 6/36, 5/36, 4/36, 3/36, 2/36, 1/36)$.
- Wir betrachten zunächst das Ereignis, dass beide Würfel die gleiche Zahl zeigen. Die Entropie für dieses Ereignis ergibt sich aus der Verteilung $\mathbf{q} = (1/6, 5/6)$.

  Wir untersuchen die beiden Fälle (gleiche Zahlen/ungleiche Zahlen):

gleiche Zahlen: Die Entropie, gegeben die beiden Würfel zeigen dieselbe Zahl, ergibt sich aus der Verteilung: $\mathbf{p}^{(1)} = (1/6, 1/6, 1/6, 1/6, 1/6, 1/6)$.

ungleiche Zahlen: Die Entropie, gegeben die beiden Würfel zeigen ungleiche Zahlen, ergibt sich aus der Verteilung:

$$\begin{aligned}\mathbf{p}^{(2)} &= (2/30, 2/30, 4/30, 4/30, 6/30, \\ &\quad 4/30, 4/30, 2/30, 2/30)\end{aligned}$$

Die erwartete Entropie, wenn man mitgeteilt bekommt, ob zwei gleiche Zahlen gewürfelt wurden, ist daher

$$\frac{1}{6}H(\mathbf{p}^{(1)}) + \frac{5}{6}H(\mathbf{p}^{(2)}).$$

Das Axiom (H6) fordert, dass sich die Gesamtentropie additiv aus dieser erwarteten Entropie und der Entropie für das vorgeschaltete Ereignis (zwei gleiche Zahlen zu würfeln) ergibt:

$$H(\mathbf{p}) = H(\mathbf{q}) + \frac{1}{6}H(\mathbf{p}^{(1)}) + \frac{5}{6}H(\mathbf{p}^{(2)})$$

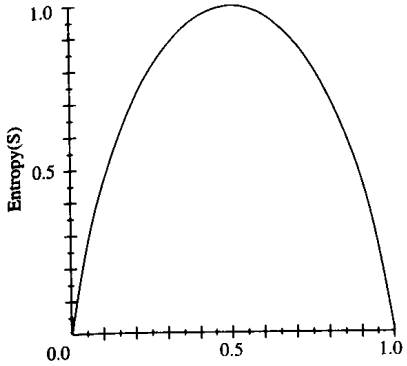
Satz: Die einzigen Funktionen, die die Axiome (H1)–(H6) erfüllen, sind

$$H(\mathbf{p}) = -C\sum_i p_i \log_2 p_i,$$

wobei C eine beliebige positive, reelle Konstante ist.

Üblicherweise wählt man $C = 1$.

Verlauf der Entropiefunktion im Fall eines zweiwertigen Attributs
(*x*-Achse: Anteil p_1 der "positiven" Fälle):



Bei der Konstruktion eines Entscheidungsbaums wählt man jeweils das Attribut, das die Entropie am stärksten verringert.

Die (Rest-)Entropie in einem Knoten des Baumes ergibt sich aus der Entropie der Häufigkeitsverteilung der Klassen in diesem Knoten k. Wir bezeichnen diesen Wert mit H_k

Spaltet man die Objekte in dem Knoten k nach einem weiteren Attribut auf, ergibt sich die neue Entropie aus

$$p_{k_1} H_{k_1} + \ldots + p_{k_t} H_{k_t}.$$

Dabei ist H_{k_i} die (Rest-)Entropie in dem Nachfolgerknoten der sich aus dem i-ten Attributwert ergibt und p_{k_1} der (relative) Anteil der Objekte im Knoten k, die für das entsprechende Attribut den i-ten Wert angenommen haben, wobei das entsprechende Attribut t verschiedene Werte annehmen kann.

Die Entropiereduktion ist daher

$$H_k - \sum_{i=1}^{t} p_{k_i} H_{k_i}.$$

Da H_k unabhängig von der Wahl des Attributs ist im Knoten k ist, wird die Entropie am stärksten reduziert, wenn wir das Attribut wählen, für das

$$\sum_{i=1}^{t} p_{k_i} H_{k_i}$$

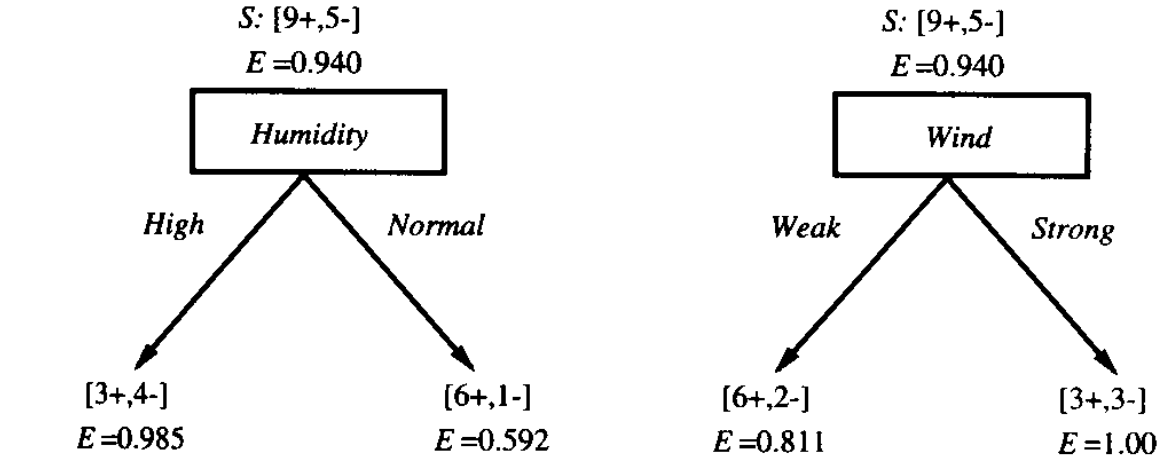
am kleinsten ist.

Bezeichnungen:
für "Entropiereduktion" auch "Informationsgewinn", *information gain*, kurz *Gain*.

Sei *S* die Menge der Trainingsdaten (Beispiele), *A* das betrachtete Attribut, *Values*(*A*) die Menge aller möglichen Werte von *A* und S_v die Teilmenge von *S*, für deren Elemente *A* den Wert *v* hat:

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Beispiel:



Gain (S, Humidity)
= .940 - (7/14).985 - (7/14).592
= .151

Gain (S, Wind)
= .940 - (8/14).811 - (6/14)1.0
= .048

"Humidity" bringt einen größeren Informationsgewinn als "Wind".

(aus Mitchell 1997)

Anstelle der Entropie wird manchmal auch der *Gini-Index* verwendet:
formal ähnlich definiert, jedoch mit p_i^2 statt $p_i \log p_i$.

- Vorteile: Unterschiede der Häufigkeiten werden stärker gewichtet; etwas effizientere Berechnung.

Gini-Index

- *Gini-Index* für eine Menge T von Trainingsobjekten

$$gini(T) = 1 - \sum_{j=1}^{k} p_j^2$$

kleiner Gini-Index $\Leftrightarrow$ geringe Unreinheit,
großer Gini-Index $\Leftrightarrow$ hohe Unreinheit

- Das Attribut A habe die Partitionierung $T_1, T_2, \ldots, T_m$ erzeugt.
- *Gini-Index* des Attributs A in Bezug auf T ist definiert als

$$gini_A(T) = \sum_{i=1}^{m} \frac{|T_i|}{|T|} \cdot gini(T_i)$$

(Böhm 2003)

Informationsgewinn und Gini-Index sind problematisch bei Attributen mit sehr vielen möglichen Werten (z.B. Datum). Wahl eines solchen Attributs würde hohen Informationsgewinn bringen, für die Praxis aber meistens nutzlos sein.

Abhilfe: Berücksichtigung eines Maßes "Split Information", das misst, wie uniform das Attribut *A* die Daten aufsplittet:

$$SplitInformation(S, A) \equiv -\sum_{i=1}^{c} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

Damit kann der Informationsgewinn (invers) gewichtet werden:

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

$\Rightarrow$ Attribute mit sehr vielen uniform verteilten Werten werden "bestraft".

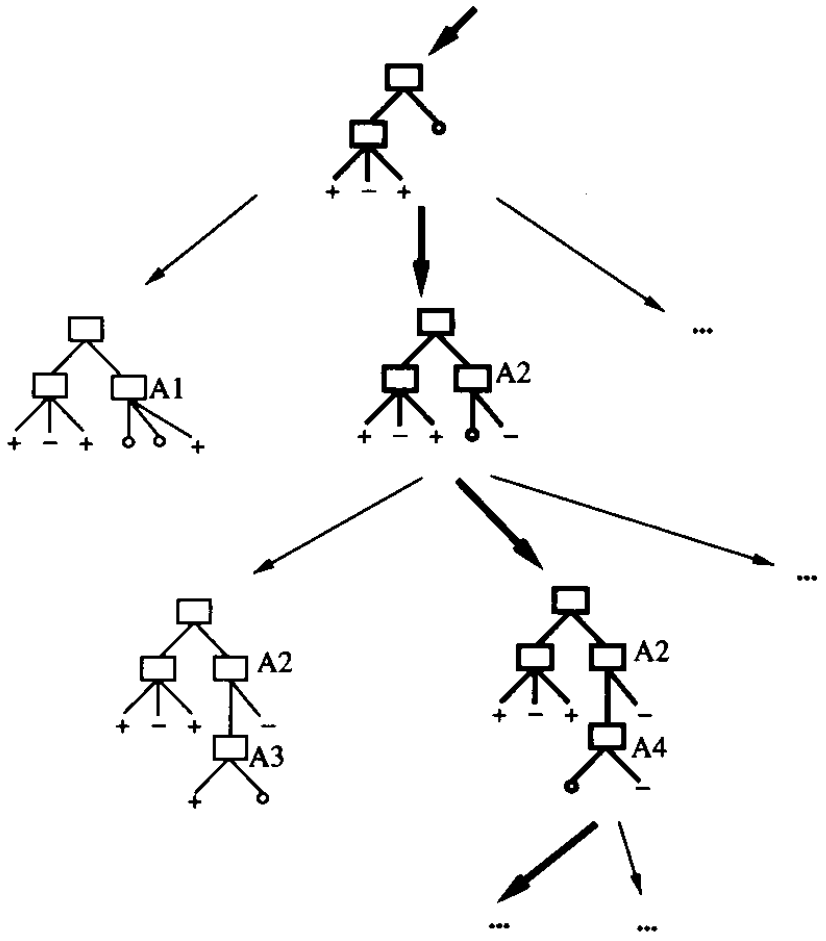
Eigenschaften des Entscheidungsbaum-Verfahrens:

- durchsuchter Hypothesenraum: Menge aller möglichen Entscheidungsbäume
- im diskreten Fall kann *jede* mögliche Zielfunktion durch einen Entscheidungsbaum repräsentiert werden (Disjunktion von Konjunktion von Attributbelegungen)
  $\Rightarrow$ Vollständigkeit!

- ID3: Hill-climbing Suchverfahren durch diesen Raum
- von einfachen zu komplexeren Entscheidungsbäumen

Funktion, die die Suche leitet: Informationsgewinn (oder Variante hiervon) – Heuristik

- keine Garantie, dass so der "beste" (einfachste) Entscheidungsbaum gefunden wird (= Nachteil der meisten "Greedy"-Verfahren, die lokale Optima suchen...)



- während der Abarbeitung des Algorithmus wird immer nur *eine* aktuelle Hypothese (zur Beschreibung der Zielfunktion) gespeichert (der jeweils aktuelle Entscheidungsbaum) – anders als bei CANDIDATE-ELIMINATION

- kein Backtracking; globales Optimum wird evtl. verpasst

- für den statistischen Test bei der Auswahl des nächsten Split-Attributs werden stets *alle* Trainingsbeispiele verwendet
  – Unterschied zu inkrementell vorgehenden Verfahren (FIND-S, CANDIDATE-ELIMINATION)

  Folgerung: ID3 ist weniger anfällig für Fehler in den Trainingsdaten!

Induktiver Bias von ID3:

bevorzugt kürzere gegenüber längeren Bäumen und bevorzugt Bäume, die hohen Informationsgewinn bei Attributen nahe der Wurzel plazieren.

Beachte: Dieser induktive Bias ist nur Folge der Anordnung der Hypothesen durch die Suchstrategie, keine Folge einer Verkleinerung des Hypothesenraumes

$\Rightarrow$ "**preference bias**", auch "search bias" genannt

Gegensatz (z.B. bei CANDIDATE-ELIMINATION):

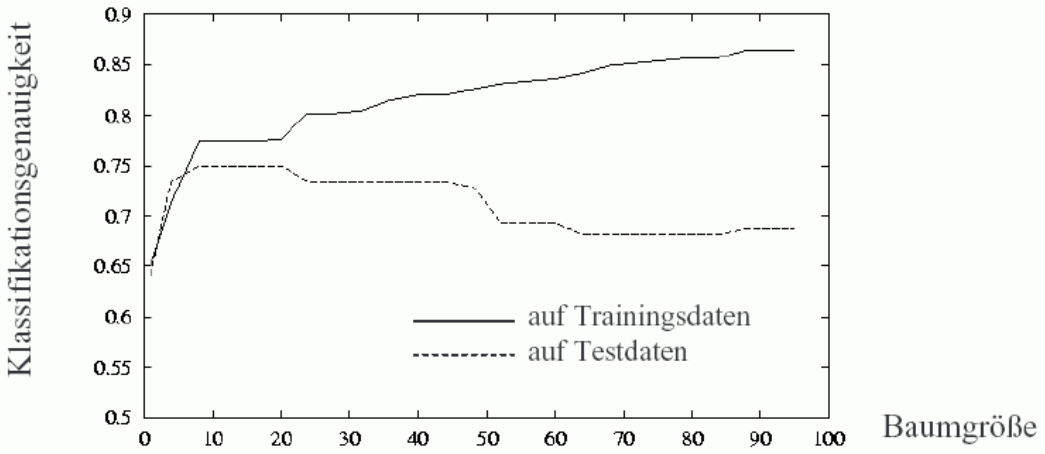
"**restriction bias**", auch "language bias"
(Algorithmus durchsucht einen unvollständigen Hypothesenraum, dieser wird aber komplett durchsucht: Die Suchstrategie erzeugt keinen zusätzlichen Bias.)

Nachteil des ID3-Algorithmus:
Gefahr des *Overfitting*

Overfitting heißt:
Eine Hypothese *H* wird gewählt, die die Trainingsdaten gut fittet, aber es gibt eine andere Hypothese, die die Trainingsdaten zwar weniger gut fittet als *H*, die aber auf der gesamten Instanzenmenge besser fittet als *H*.

Overfitting bei der Konstruktion eines Entscheidungsbaums, wenn es zwei Entscheidungsbäume *E* und *E'* gibt mit

- *E* hat auf der Trainingsmenge eine kleinere Fehlerrate als *E'*,
- *E'* hat auf der Grundgesamtheit der Daten eine kleinere Fehlerrate als *E*.



Ansätze zum Vermeiden von Overfitting

- Entfernen von fehlerhaften Trainingsdaten

  insbesondere widersprüchliche Trainingsdaten

- Wahl einer geeigneten Größe der Trainingsmenge

  nicht zu klein, nicht zu groß

- Wahl einer geeigneten Größe des minimum support

  minimum support:

  Anzahl der Datensätze, die mindestens zu einem Blattknoten des Baums gehören müssen

  minimum support >> 1

- Wahl einer geeigneten Größe der minimum confidence

  minimum confidence: Anteil, den die Mehrheitsklasse eines Blattknotens mindestens besitzen muß

  minimum confidence << 100%

  Blätter können auch fehlerhafte Datensätze oder Rauschen „absorbieren“

- nachträgliches Pruning des Entscheidungsbaums

  Abschneiden der überspezialisierten Äste

Fehlerreduktions-Pruning [Mitchell 1997]

- Aufteilung der klassifizierten Daten in Trainingsmenge und Testmenge
- Konstruktion eines Entscheidungsbaums E für die Trainingsmenge
- Pruning von E mit Hilfe der Testmenge T
  - bestimme denjenigen Teilbaum von E, dessen Abschneiden den Klassifikationsfehler auf T am stärksten reduziert
  - entferne diesen Teilbaum
  - fertig, falls kein solcher Teilbaum mehr existiert

nur anwendbar, wenn genügend viele klassifizierte Daten

(Böhm 2003)

"Pruning" (Abschneiden) eines Entscheidungsknotens heißt:

- der Teilbaum, der in diesem Knoten beginnt, wird entfernt
- der Knoten wird zum Blattknoten
- zugeordnet wird ihm die häufigste Klassifikation, die unter den Trainingsdaten auftritt, die unter diesen Blattknoten fallen

- Knoten werden nur dann entfernt, wenn sich der resultierende Baum für die Testmenge *T* nicht schlechter verhält als der ursprüngliche Baum!

Wenn die Trainingsmenge zu Overfitting geführt hat (durch zufällige Regularitäten in den Trainingsdaten), ist es sehr wahrscheinlich, dass die entsprechenden Teilbäume im Entscheidungsbaum beim Pruning wieder entfernt werden, da die zufälligen Regularitäten in der Testmenge nicht vorkommen.

Pseudocode dieses Verfahrens:

```
FehlerreduktionsPruning (Entscheidungsbaum E, Testmenge T)
   sei BesterFehler der Klassifikationsfehler von E auf T;
   B := E;
   loop
      for each Knoten K aus B do
         entferne K und seinen zugehörigen Teilbaum aus B und
           erhalte B – K;
         bestimme den Klassifikationsfehler NeuerFehler von
           B – K auf T;
         merke das Minimum der Werte für NeuerFehler in
           BesterNeuerFehler;
         if BesterNeuerFehler < BesterFehler then
            entferne den entsprechenden Knoten K und seinen
              Teilbaum aus B, d.h. B := B – K;
            merke als BesterFehler den Klassifikationsfehler
              des neuen B auf T;
         else return B;
```

(aus Ester / Sander 2000)

Somit:

Unter **Pruning** versteht man das Beschneiden/Verkleinern eines Entscheidungsbaums. (Der Begriff Pruning wird allgemein für das Beschneiden von Graphenstrukturen z.B. auch bei neuronalen Netzen verwendet.)

Eine einfache **Prepruning**-Technik (Beschneiden des Baumes während der Konstruktion des Baumes) wurde bereits erwähnt: Gehört in einem Knoten eine große Mehrheit der Objekte zu einer Klasse, wird der Baum in diesem Knoten nicht mehr tiefer fortgesetzt.

Diese Technik kann auch nach dem Erstellen des Baumes als **Postpruning** oder **Backward-Pruning** angewendet werden.

Eine weitere wichtige Postpruning-Technik ist das **Subtree Raising**.

Dabei werden einzelne Unterbäume nach oben verschoben.

Dies kann insbesondere dann sehr effizient sein, wenn es Attribute gibt, die einzeln wenig, aber in Kombination sehr stark zur Klassifikation beitragen.

Solche Attribute rutschen sehr leicht weit nach unten bei der Konstruktion des Entscheidungsbaums nach dem Entropie-Kriterium.

Attribute mit einer größeren Anzahl von Werten werden durch die Entropiebewertung ebenfalls bevorzugt, da sich durch mehr Werte eine feinere Aufteilung ergibt.

(Klawonn 2004)

Regel-Postpruning:

Es ist vorteilhaft, den Entscheidungsbaum in eine Menge von Regeln zu transformieren:

- die Regeln können dann einzeln vereinfacht werden, ohne den Baum-Kontext berücksichtigen zu müssen – d.h. für jeden Pfad im Baum separat
- Ergebnis ist dann allerdings kein Entscheidungsbaum mehr, sondern eine Regelmenge

Verfahren, das in C4.5 (Quinlan 1993) verwendet wird:

- Verallgemeinere jede Regel durch Entfernen von Voraussetzungen, so dass die geschätzte Trennschärfe erhöht wird
- sortiere die gewonnenen Regeln nach ihrer geschätzten Trennschärfe und wende sie in dieser Reihenfolge auf die zu klassifizierenden Daten an

(Bsp. s. Mitchell 1997, S. 71)

Entscheidungsbäume bei Attributen mit kontinuierlichem Wertebereich:

- Quantisierung

d.h. Aufteilung des Wertebereichs in Intervalle (Trennung durch Schwellenwerte)
jedem Intervall entspricht ein Tochterknoten

Wahl des Schwellenwertes (bei binärer Aufteilung):
wähle den Schwellenwert, der den größten Informationsgewinn (oder den größten Gini-Index) liefert

Wir betrachten ein numerisches Attribut A_j, dessen Wertebereich in t Intervalle aufgeteilt werden soll.

Dazu müssen $t-1$ Schnittpunkte $T_1, \ldots, T_{t-1}$ definiert werden.

Diese Schnittpunkte sollten so gewählt werden, dass die Entropie durch die induzierte Partition minimiert wird.

T_0 und T_t bezeichnen den linken und rechten Rand des Wertebereichs des Attributs A_j.

Nehmen wir an, es fallen n_i $(i = 1, \ldots, t)$ der n Objekte in das Intervall zwischen T_{i-1} und T_i, wenn wir nur das Attribut A_j betrachten.

k_ℓ bezeichne die Anzahl der n_i Objekte, die zur Klasse ℓ gehören. Dann ist die Entropie in diesem Intervall

$$E_i = -\sum_{\ell=1}^{c} \frac{k_\ell}{n_i} \cdot \log_2 \left(\frac{k_\ell}{n_i} \right).$$

Die durch die gesamte Partition induzierte Entropie ist dann die gewichtete Summe der Einzelentropiewerte:

$$E = \sum_{i=1}^{t} \frac{n_i}{n} \cdot E_i$$

die durch geeignete Wahl der Schnittpunkte zu minimieren ist.

Sortiert man die in den Daten vorkommenden Werte des Attributs A_j, dann müssen für eine Minimierung der Entropie sämtliche Schnittpunkte Grenzpunkte sein.

Ein Wert T im Wertebereich des Attributs A_j ist ein Grenzpunkt, wenn in der Folge der bzgl. des Attributs A_j sortierten Daten folgendes gilt:

Es existieren zwei Objekte $\mathbf{x}$ und $\mathbf{y}$, die zu unterschiedlichen Klassen gehören, mit $x_j < T < y_j$ und es gibt kein Objekt $\mathbf{z}$ mit $x_j < z_j < y_j$.

Im folgenden Beispiel sind die Grenzpunkte durch Striche markiert:

Wert:	1	2	3	3	4	5	5	6	6
Klasse:	3	3	1	1	1	2	2	1	3

7	8	8	9	10	11	11	12
3	3	3	3	2	1	1	1

Wird nur ein Schnittpunkt gesucht (binäre Diskretisierung), werden die Grenzpunkte berechnet und für alle Grenzpunkte der Entropiewert bestimmt.

Bei mehreren Schnittpunkten lässt sich eine Rekursionsformel zur Berechnung der Entropie angeben.

(Klawonn 2004)

Fazit zu Entscheidungsbaum-Verfahren:

Diskussion

+ Interpretation des gefundenen Baumes relativ einfach
+ Implizite Gewichtung der Attribute
+ Leistungsfähiger Klassifikator, häufig in der Praxis verwendet
+ Effiziente Auswertung des gefundenen Modells

- Finden eines optimalen Entscheidungsbaums ist exponentiell
- Heuristische Methoden können nur lokales Optimum finden
- Anfällig für Overfitting

(Böhm 2003)

- Entscheidungsbaum ist nicht eindeutig

- Gestalt des gefundenen Entscheidungsbaumes muss nicht viel über die Daten aussagen

+ Kosten der Erhebung eines Attributwertes (z.B. Zeitaufwand einer Messung) können leicht in die Entscheidungsbaum-konstruktion einbezogen werden:

  z.B. indem man den Informationsgewinn ersetzt durch die Größe $Gain^2(S, A)/Cost(A)$ (s. Mitchell 1997)

- identische Testfolgen (Abfragen von Attributwerten) können an verschiedenen Stellen im Baum auftreten – Redundanz