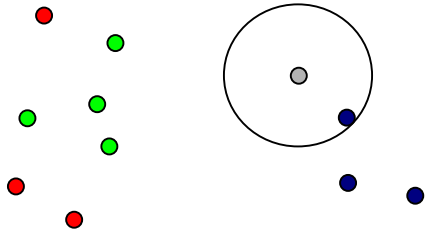
5.4 Nächste-Nachbarn-Klassifikatoren



- Schrauben
- Nägel
- Klammern

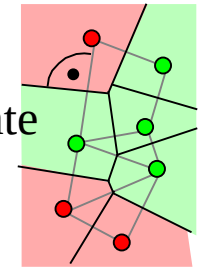
} Trainingsdaten

- Neues Objekt

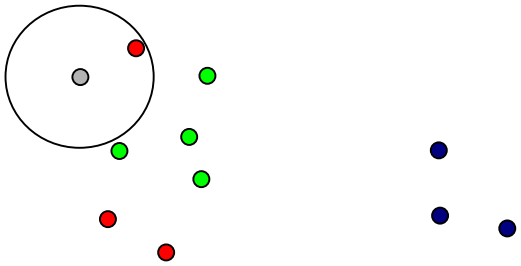
Instanzbasiertes Lernen (*instance based learning*)

- Einfacher Nächster-Nachbar-Klassifikator: Zuordnung zu der Klasse des nächsten Nachbarpunkts
- Im Beispiel: Nächster Nachbar ist eine Schraube
- Regionen der Klassenzuordnung können als *Voronoi-Diagramme* dargestellt werden:

Mittel-senkrechte

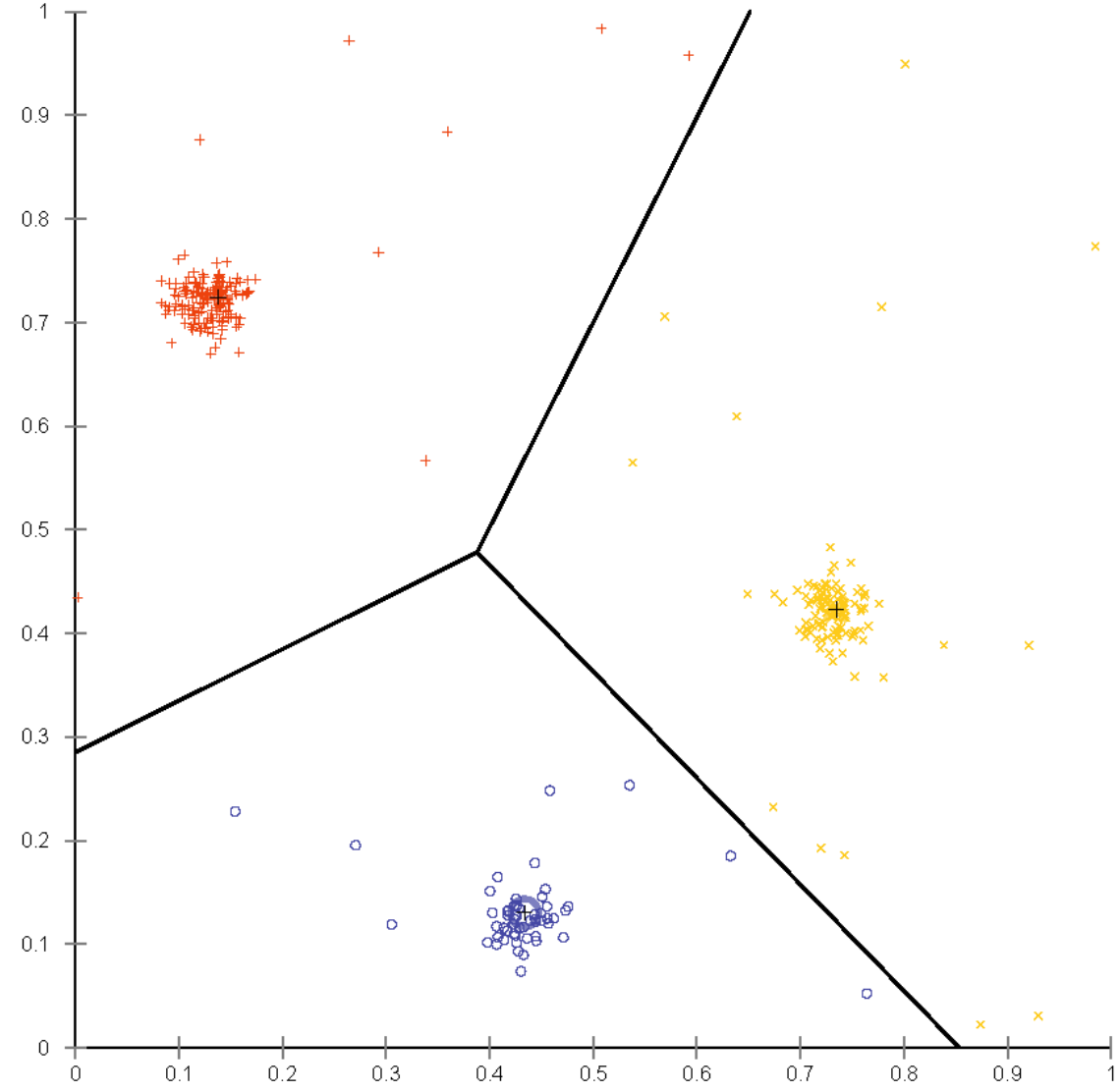


Nächste-Nachbarn-Klassifikatoren



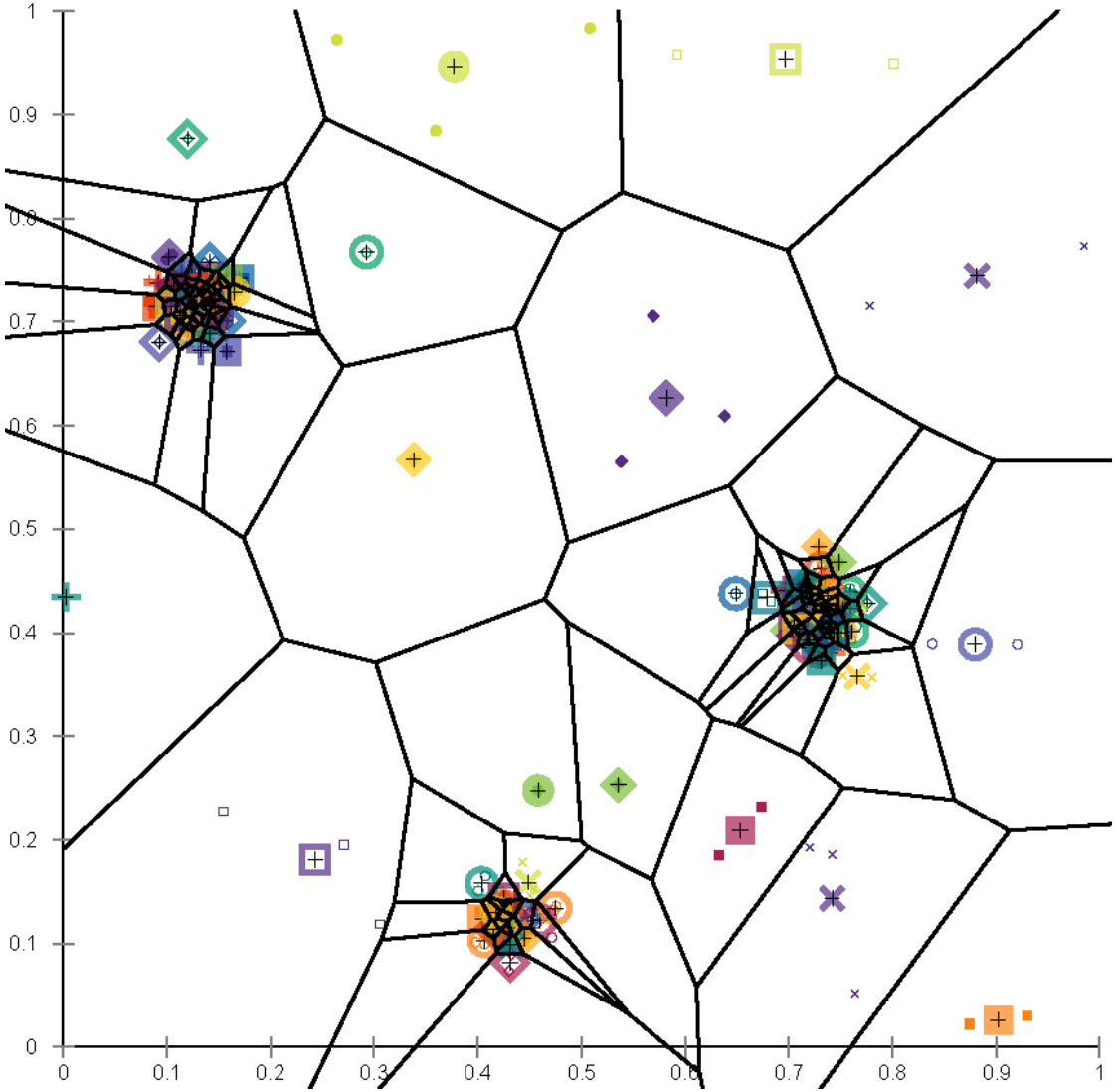
- Problem: Punkt links oben wahrscheinlich nur Ausreißer
- Besser: Betrachte mehr als nur einen Nachbarn
  $\Rightarrow$ *k*-Nächste-Nachbarn-Klassifikator
- *Entscheidungsmenge*
  die Menge der zur Klassifikation betrachteten *k*-nächsten Nachbarn
- *Entscheidungsregel*
  Wie bestimmt man aus den Klassen der Entscheidungsmenge die Klasse des zu klassifizierenden Objekts?
  - Interpretiere Häufigkeit einer Klasse in der Entscheidungsmenge als Wahrscheinlichkeit der Klassenzugehörigkeit
  - Maximum-Likelihood-Prinzip: Mehrheitsentscheidung
  - Ggf. Gewichtung

Nächste-Nachbarn-Klassifikatoren



- k=1: Voronoi-Zellen separieren „Trainingsobjekte“
- Zuordnung neuer Punkte zum nächstgelegenen „Trainings“-Objekt

Nächste-Nachbarn-Klassifikatoren

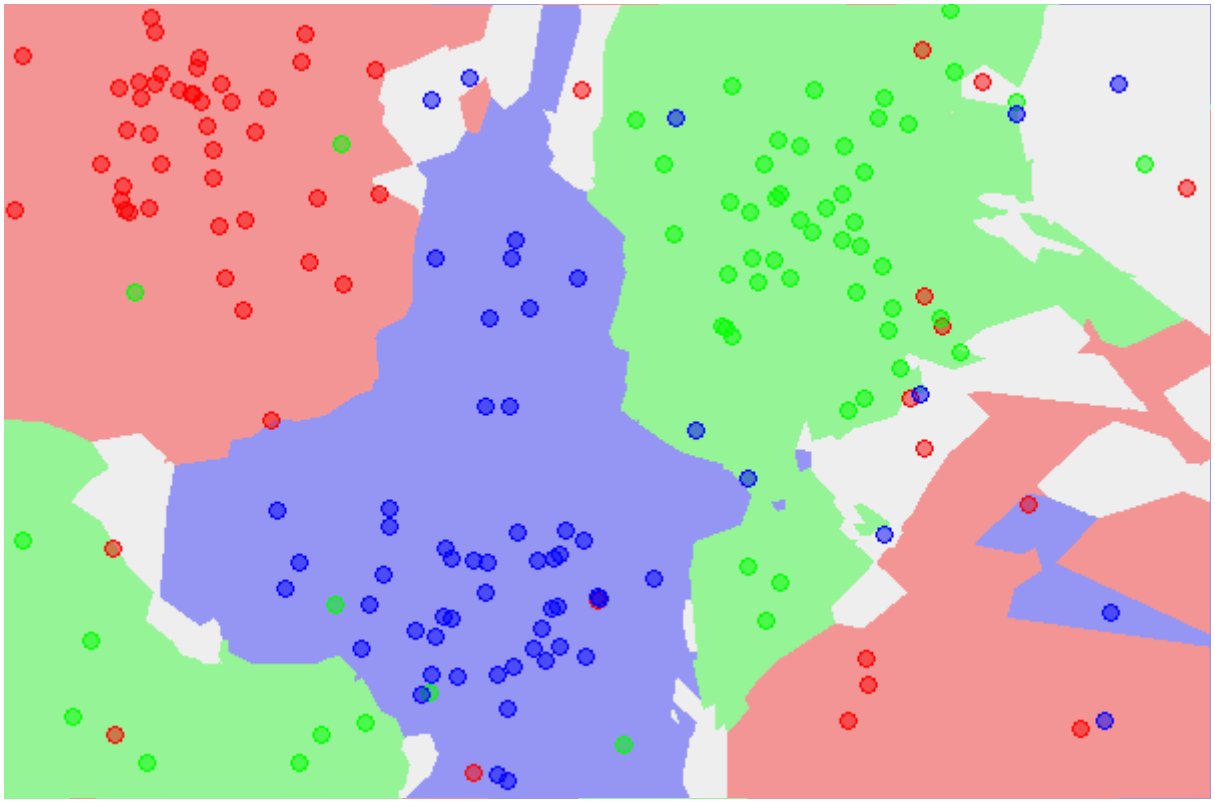


viele Trainingsobjekte

⇩

sehr kleinteilige, flexible Klassengrenzen

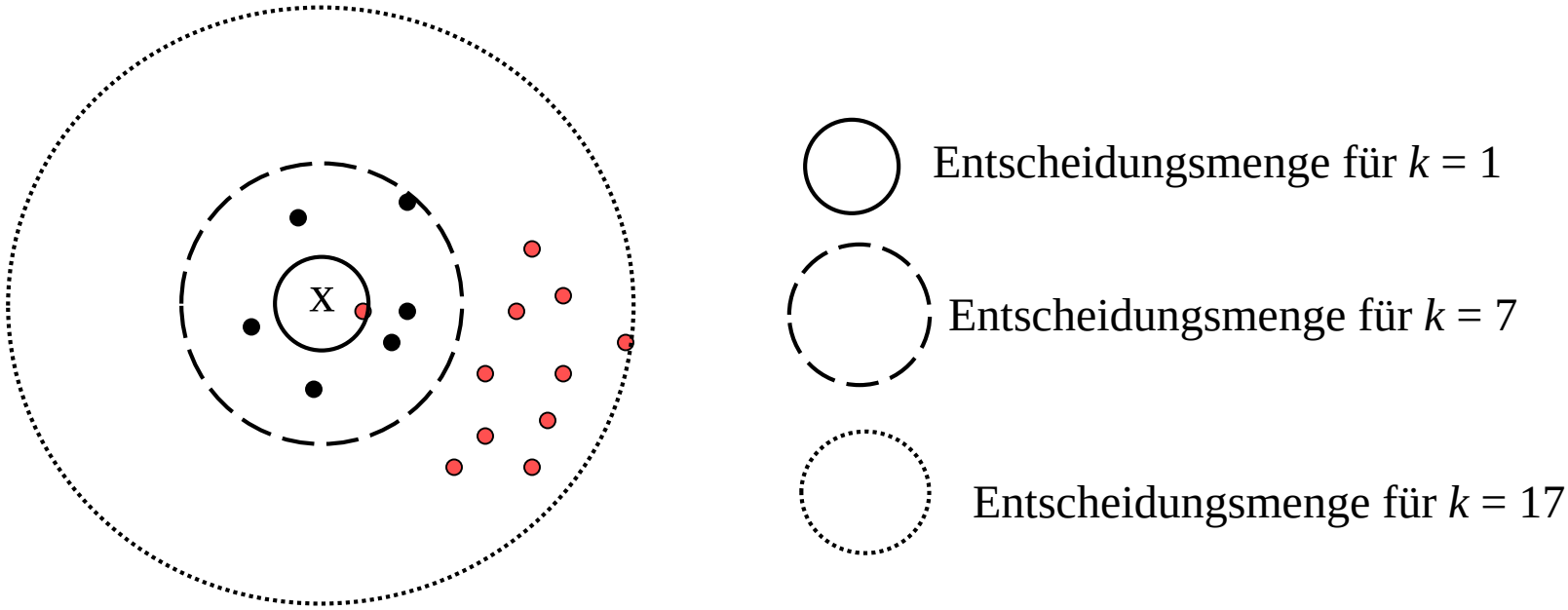
Nächste-Nachbarn-Klassifikatoren



- 1-NN
- 5-NN

Wahl des Parameters k

- „zu kleines“ k: hohe Sensitivität gegenüber Ausreißern
- „zu großes“ k: viele Objekte aus anderen Clustern (Klassen) in der Entscheidungsmenge.
- mittleres k: höchste Klassifikationsgüte, oft $1 << k < 10$



x: zu klassifizieren

Entscheidungsregel

- Standardregel

  $\Rightarrow$ wähle die Mehrheitsklasse der Entscheidungsmenge

- Gewichtete Entscheidungsregel

  gewichte die Klassen der Entscheidungsmenge

  - nach Distanz, meist invers quadriert: $weight(dist) = 1/dist^2$
  - nach Verteilung der Klassen (oft sehr ungleich!)

    Problem: Klasse mit zu wenig Instanzen ($< k/2$) in der Trainingsmenge bekommt keine Chance, ausgewählt zu werden, selbst bei optimaler Distanzfunktion

    - Klasse A: 95 %, Klasse B 5 %
    - Entscheidungsmenge = {A, A, A, A, B, B, B}
    - Standardregel $\Rightarrow$ A, gewichtete Regel $\Rightarrow$ B

Klassifikation von Sternen

Analyse astronomischer Daten

Entfernen von Rauschen

Bildsegmentierung

Feature-Extraktion

Automatische *Klassifikation* des Sterntyps

Manuelle Analyse der interessanten Sterntypen

Klassifikation des Sterntyps mit Nächste-Nachbarn Klassifikator basierend auf dem Hipparcos-Katalog

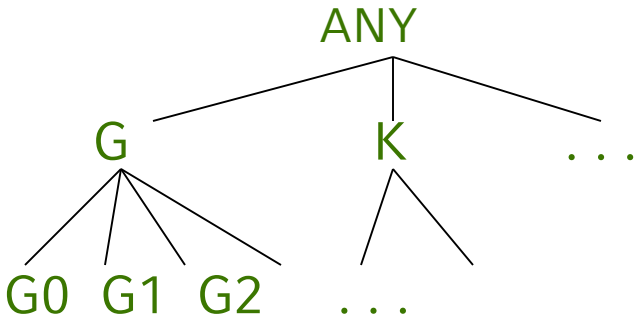
Klassifikation von Sternen

Hipparcos-Katalog [ESA 1998]

- enthält ca. 118 000 Sterne
- mit 78 Attributen (Helligkeit, Entfernung, Farbe,. . .)
- Klassenattribut: Spektraltyp (Attribut H76)

z.B.

H76: G0
H76: G7.2
H76: KIII/IV



- Werte des Spektraltyps sind vage

➡ Hierarchie von Klassen: benutze die erste Ebene der Klassenhierarchie

Klassifikation von Sternen

Verteilung der Klassen

Klasse	#Instanzen	Anteil Instanzen	
K	32 036	27.0	häufige Klassen
F	25 607	21.7	
G	22 701	19.3	
A	18 704	15.8	
B	10 421	8.8	
M	4 862	4.1	
O	265	0.22	seltene Klassen
C	165	0.14	
R	89	0.07	
W	75	0.06	
N	63	0.05	
S	25	0.02	
D	27	0.02	

Klassifikation von Sternen

Experimentelle Untersuchung [Poschenrieder 1998]

- Distanzfunktion
  - mit 6 Attributen (Farbe, Helligkeit und Entfernung)
  - mit 5 Attributen (ohne Entfernung)

  $\Rightarrow$ beste Klassifikationsgenauigkeit mit 6 Attributen
- Anzahl k der Nachbarn

  $\Rightarrow$ beste Klassifikationsgenauigkeit für $k = 15$
- Entscheidungsregel
  - Gewichtung nach Distanz
  - Gewichtung nach Klassenverteilung

  $\Rightarrow$ beste Klassifikationsgenauigkeit bei Gewichtung nach Distanz aber nicht nach Klassenverteilung

Klassifikation von Sternen

Klasse	Falsch klassifiziert	Korrekt klassifiziert	Klassifikations-genauigkeit
K	408	2338	85.1%
F	350	2110	85.8%
G	784	1405	64.2%
A	312	975	75.8%
B	308	241	43.9%
M	88	349	79.9%
C	4	5	55.6%
R	5	0	0%
W	4	0	0%
O	9	0	0%
N	4	1	20%
D	3	0	0%
S	1	0	0%
Total	2461	7529	75.3%

hohe Klassifikationsgenauigkeit für die häufigen Klassen, schlechte Genauigkeit für die seltenen Klassen

die meisten seltenen Klassen besitzen weniger als $k / 2 = 8$ Instanzen!

Nächste-Nachbarn-Klassifikatoren

Diskussion

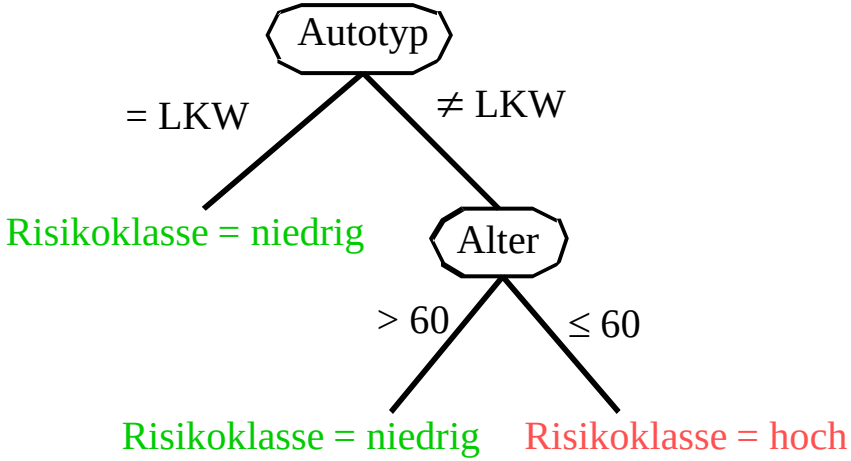
+ Anwendbarkeit: erfordert nur die Trainingsdaten
+ hohe Klassifikationsgenauigkeit in vielen Anwendungen
+ inkrementell: Klassifikator kann sehr einfach an neue Trainingsobjekte adaptiert werden
+ keine Trainingsphase erforderlich („lazy learner“)

- Ineffizienz: Auswertung des “Modells” erfordert k-nächste-Nachbarn Anfrage an die Datenbank
- liefert kein explizites Wissen über die Klassen
- Wahl von k problematisch bei sehr unterschiedlichen Klassengrößen

5.5 Entscheidungsbaum-Klassifikatoren

Motivation

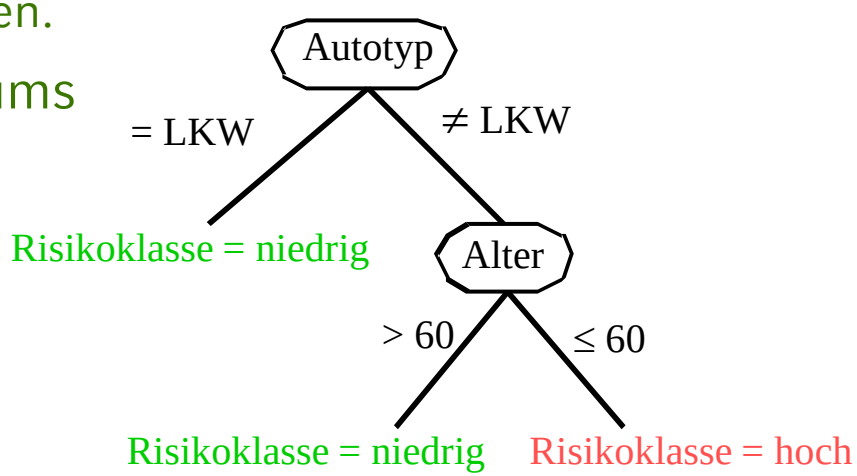
ID	Alter	Autotyp	Risiko
1	23	Familie	hoch
2	17	Sport	hoch
3	43	Sport	hoch
4	68	Familie	niedrig
5	32	LKW	niedrig



- finden explizites Wissen
- Entscheidungsbäume liefern ein für den Benutzer verständliches Modell (Hierarchie von Regeln)

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

  $\Rightarrow$ 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.

  $\Rightarrow$ lokal optimierender Algorithmus

Abbruchbedingungen

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

Entscheidungsbaum-Klassifikatoren

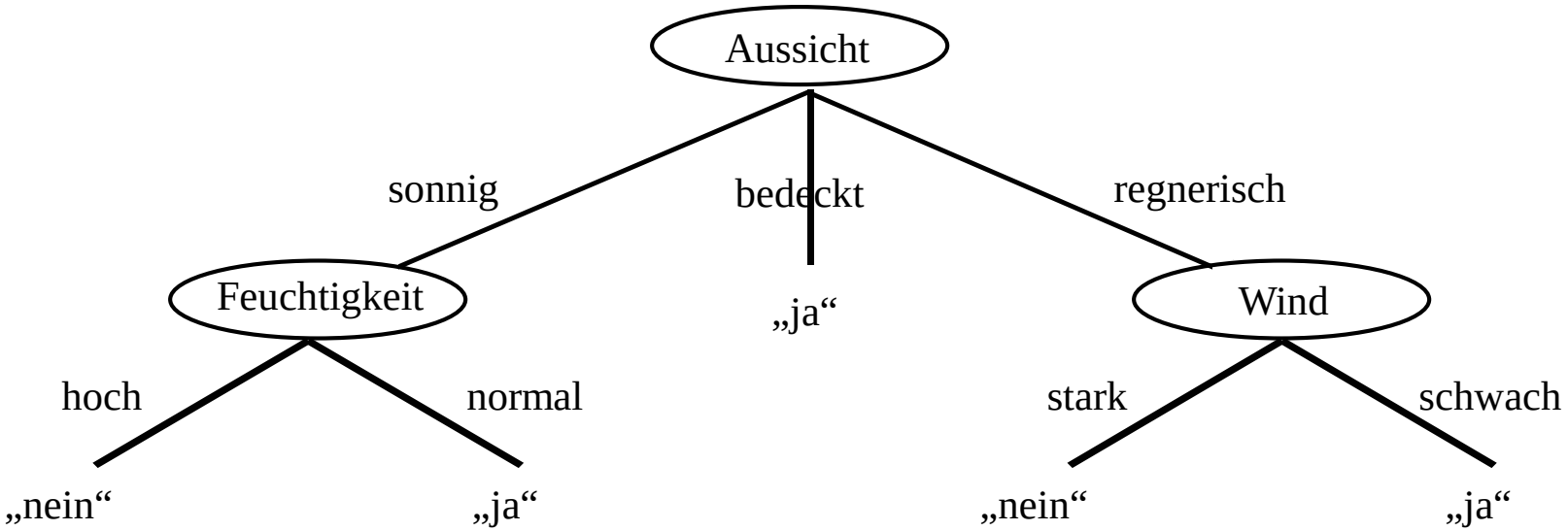
Beispiel

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	bedeckt	kühl	normal	stark	ja
8	sonnig	mild	hoch	schwach	nein
9	sonnig	kühl	normal	schwach	ja
10	regnerisch	mild	normal	schwach	ja
11	sonnig	mild	normal	stark	ja
12	bedeckt	mild	hoch	stark	ja
13	bedeckt	heiß	normal	schwach	ja
14	regnerisch	mild	hoch	stark	nein

Ist heute ein Tag zum Tennisspielen?

Entscheidungsbaum-Klassifikatoren

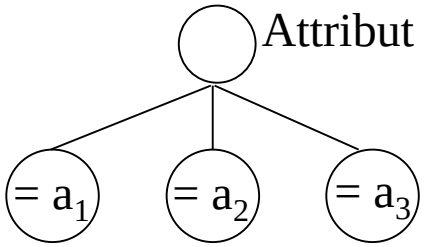
Beispiel

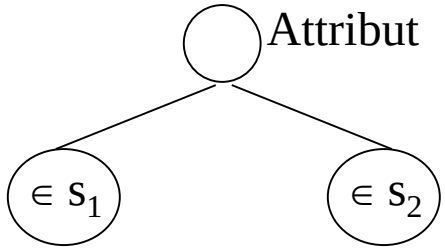


Splitstrategien

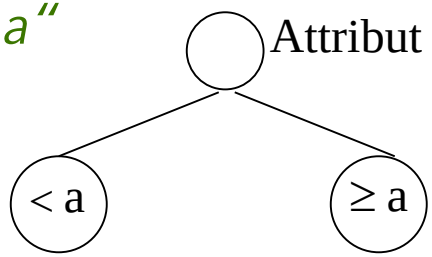
Typen von Splits

- Kategorische Attribute
  Splitbedingungen der Form „*Attribut = a*“ or „*Attribut ∈ set*“
  viele mögliche Teilmengen





- Numerische Attribute
  Splitbedingungen der Form „*Attribut < a*“
  viele mögliche Splitpunkte



Numerische Splitgrenzen

Wo sollen diskrete Attribute gesplittet werden?

=> An den Stellen, die die Qualität maximieren.

Idee: Ordnen der numerischen Attributwerte

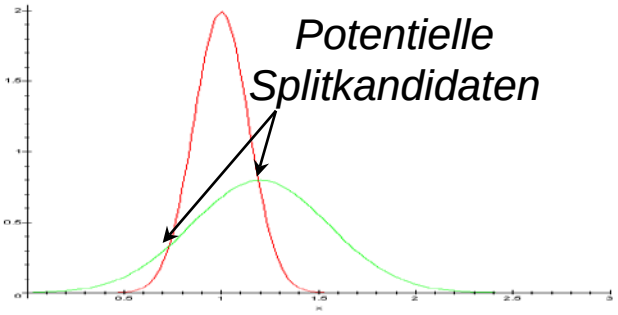
Wert	0.9	0.8	0.65	0.5	0.45	0.3	0.15	0.01
Klasse	A	A	B	B	B	A	A	A

Potentielle Splitkandidaten

Teste die Kombination, die den höchsten Information Gain erzielen.

Schnellere Methode:

- Bilde Gauß-Kurve über alle Klassen
- Wähle Schnittpunkte der Gauß-Kurven als Kandidaten.



Splitstrategien

Qualitätsmaße für Splits

Gegeben

- eine Menge T von Trainingsobjekten
- eine disjunkte, vollständige Partitionierung $T_1, T_2, \ldots, T_m$ von T
- p_i die relative Häufigkeit der Klasse c_i in T

Gesucht

- ein Maß der *Unreinheit* einer Menge S von Trainingsobjekten in Bezug auf die Klassenzugehörigkeit
- ein Split von T in $T_1, T_2, \ldots, T_m$, der dieses Maß der Unreinheit *minimiert*

  $\Rightarrow$ Informationsgewinn, Gini-Index

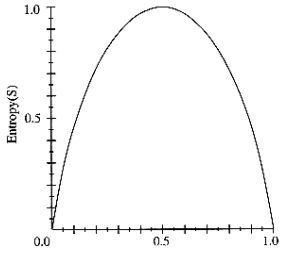
Splitstrategien

Informationsgewinn

Entropie: minimale Anzahl von Bits zum Codieren der Nachricht, mit der man die Klasse eines zufälligen Trainingsobjekts mitteilen möchte.

Die *Entropie* für eine Menge T von Trainingsobjekten ist definiert als

$$entropie(T) = -\sum_{i=1}^{k} p_i \cdot \log p_i$$



$entropie(T) = 0$, falls $p_i = 1$ für ein i

$entropie(T) = 1$ für $k = 2$ Klassen mit $p_i = 1/2$

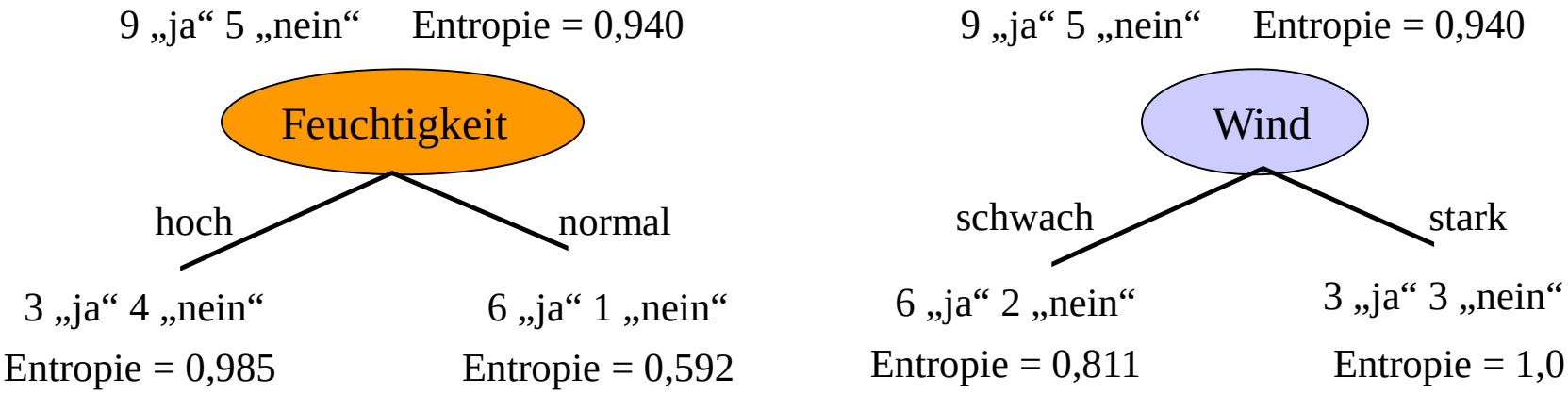
Das Attribut A habe die Partitionierung $T_1, T_2, \ldots, T_m$ erzeugt.

Der *Informationsgewinn* des Attributs A in Bezug auf T ist definiert als

$$Informationsgewinn(T, A) = entropie(T) - \sum_{i=1}^{m} \frac{|T_i|}{|T|} \cdot entropie(T_i)$$

Splitstrategien

Beispiel



$$Informationsgewinn(T, Feuchtigkeit) = 0{,}94 - \frac{7}{14} \cdot 0{,}985 - \frac{7}{14} \cdot 0{,}592 = 0{,}151$$

$$Informationsgewinn(T, Wind) = 0{,}94 - \frac{8}{14} \cdot 0{,}811 - \frac{6}{14} \cdot 1{,}0 = 0{,}048$$

$\Rightarrow$ Feuchtigkeit liefert den höheren Informationsgewinn

Gini-Index

Gini-Index für eine Menge *T* von Trainingsobjekten

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

- kleiner Gini-Index ⇔ geringe Unreinheit,
- großer Gini-Index ⇔ hohe Unreinheit

Das Attribut *A* habe die Partitionierung $T_1, T_2, \ldots, T_m$ erzeugt.

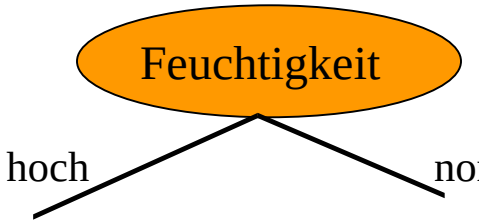
Gini-Index des Attributs *A* in Bezug auf *T* ist definiert als

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

Splitstrategien

Beispiel

9 „ja“ 5 „nein“, |T|=14



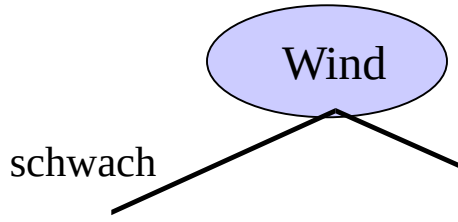
3 „ja“ 4 „nein“

$$gini(T) = 1 - \left(\frac{3^2}{7^2} + \frac{4^2}{7^2}\right) = \frac{24}{49}$$

6 „ja“ 1 „nein“

$$gini(T) = 1 - \left(\frac{6^2}{7^2} + \frac{1^2}{7^2}\right) = \frac{12}{49}$$

9 „ja“ 5 „nein“, |T|=14



6 „ja“ 2 „nein“

$$gini(T) = 1 - \left(\frac{6^2}{8^2} + \frac{2^2}{8^2}\right) = \frac{24}{64} = \frac{3}{8}$$

3 „ja“ 3 „nein“

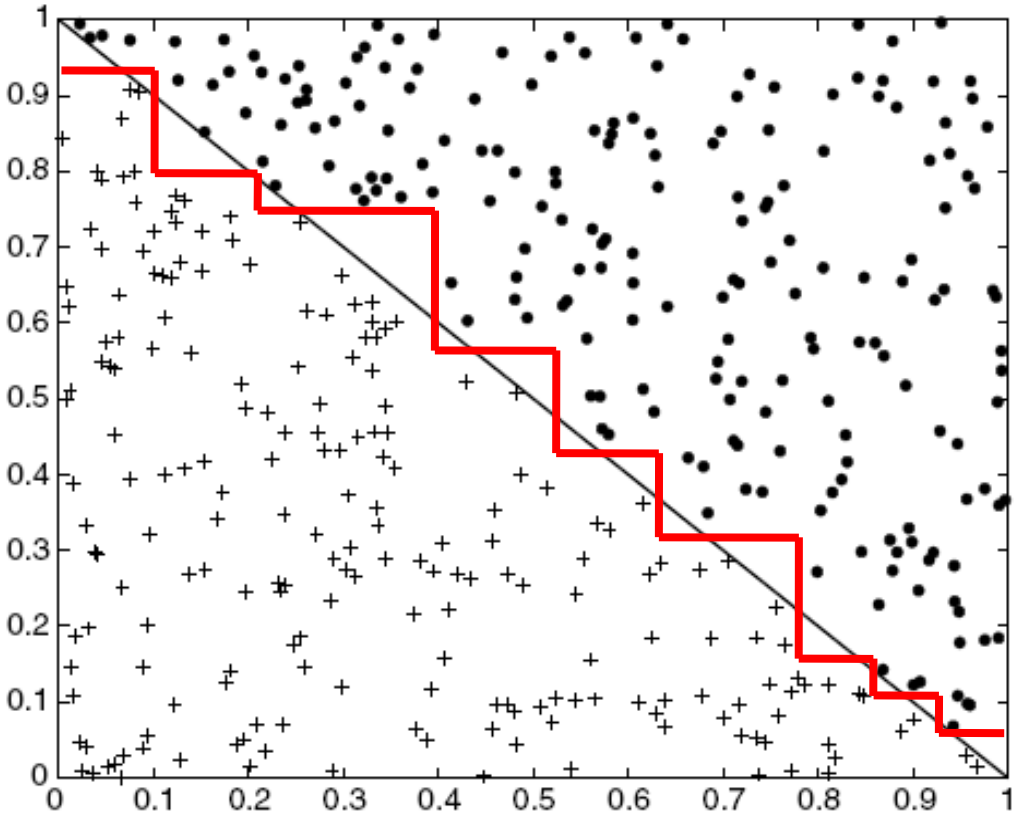
$$gini(T) = 1 - \left(\frac{3^2}{6^2} + \frac{3^2}{6^2}\right) = \frac{18}{36} = \frac{1}{2}$$

$$gini_{Feuchtigkeit}(T) = \frac{7}{14} \cdot \frac{24}{49} + \frac{7}{14} \cdot \frac{12}{49} = \frac{18}{49}$$

$$gini_{Wind}(T) = \frac{8}{14} \cdot \frac{3}{8} + \frac{6}{14} \cdot \frac{1}{2} = \frac{3}{7} = \frac{21}{49}$$

⇒ Feuchtigkeit liefert kleineren Gini-Index (geringere Unreinheit)

Bias

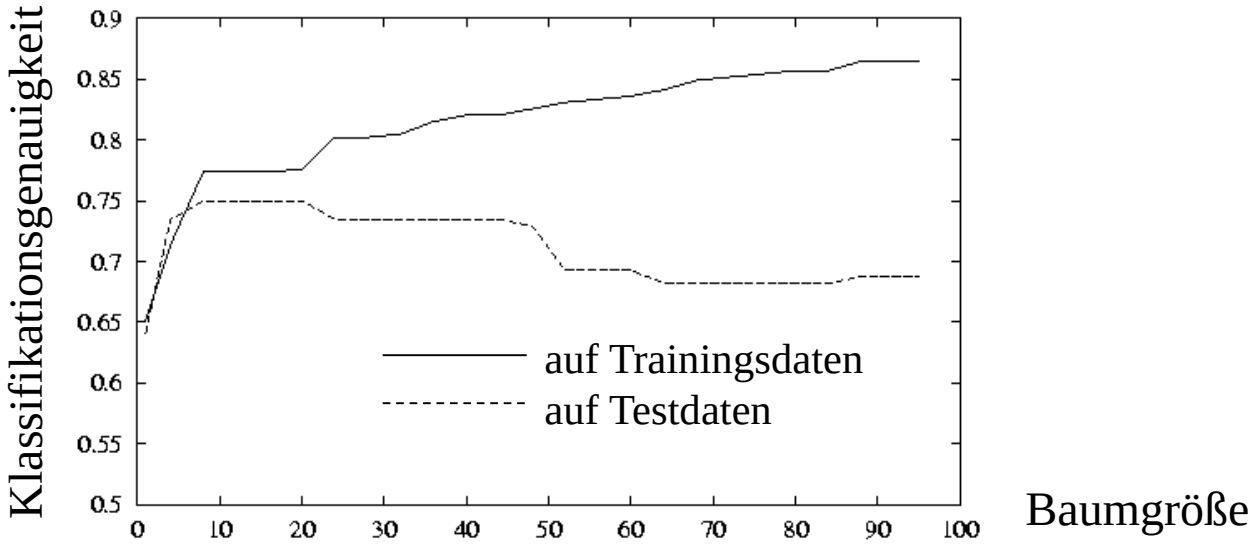


Overfitting

Einführung

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.



Overfitting

Trainingsdaten

Name	Body Temp	Gives Birth	Four-legged	Hibernates	Mammal
porcupine	warm-blooded	yes	yes	yes	yes
cat	warm-blooded	yes	yes	no	yes
bat	warm-blooded	yes	no	yes	no*
whale	warm-blooded	yes	no	no	no*
salamander	cold-blooded	no	yes	yes	no
komodo dragon	cold-blooded	no	yes	no	no
python	cold-blooded	no	no	yes	no
salmon	cold-blooded	no	no	no	no
eagle	warm-blooded	no	no	no	no
guppy	cold-blooded	yes	no	no	no

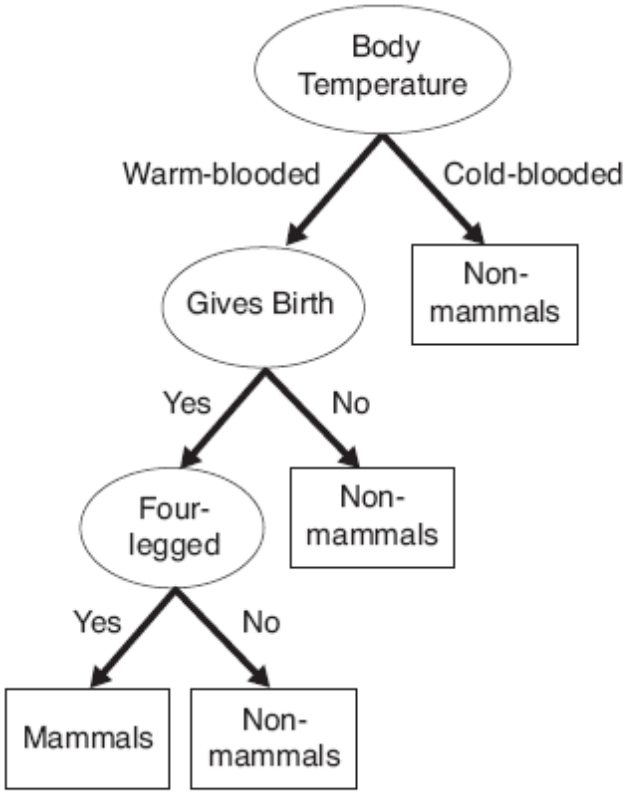
*Fehlerhafte Daten

Overfitting

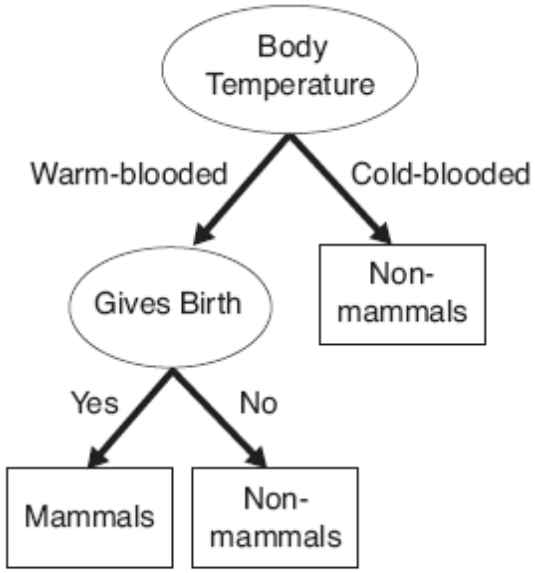
Testdaten

Name	Body Temp	Gives Birth	Four-legged	Hibernates	Mammal
human	warm-blooded	yes	no	no	yes
pigeon	warm-blooded	no	no	no	no
elephant	warm-blooded	yes	yes	no	yes
leopard shark	cold-blooded	yes	no	no	no
turtle	cold-blooded	no	yes	no	no
penguin	warm-blooded	no	no	no	no
eel	cold-blooded	no	no	no	no
dolphin	warm-blooded	yes	no	no	yes
spiny anteater	warm-blooded	no	yes	yes	yes
gila monster	cold-blooded	no	yes	yes	no

Overfitting



(a) Model M1



(b) Model M2

Trainingserror: 0%
Testerror: 30%

Trainingserror: 20%
Testerror: 10%

M1:

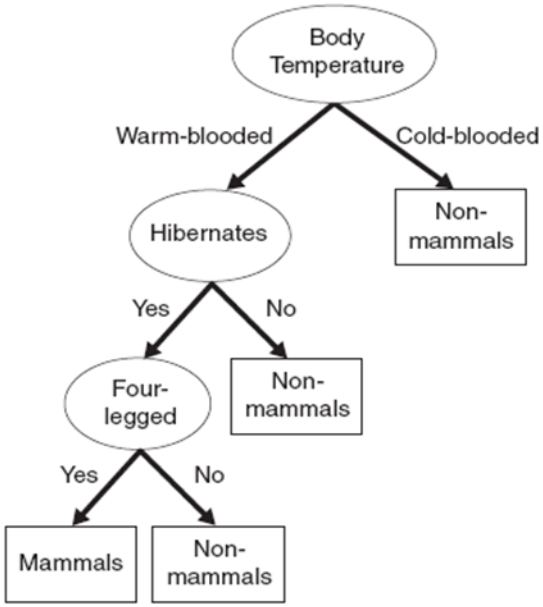
- Fehlklassifikation von human und dolphin aufgrund von fehlerhaften Trainingsdaten
- spiny anteater: ungewöhnlicher Fall

M2:

ungewöhnlicher Fall kann nicht vermieden werden

Overfitting

Name	Body Temp	Gives Birth	Four-legged	Hibernates	Mammal
salamander	cold-blooded	no	yes	yes	no
poorwill	warm-blooded	no	no	yes	no
platypus	warm-blooded	no	yes	yes	yes
eagle	warm-blooded	no	no	no	no
guppy	cold-blooded	yes	no	no	no



Problem:

Mangel an repräsentativen Daten

$\Rightarrow$ Fehlerrate auf Testset: 30%

Overfitting

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

Overfitting

Ansätze zum Vermeiden von Overfitting

- 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

  kleinere Bäume sind weniger anfällig für Overfitting (Occam‘s Razor)

Pruning von Entscheidungsbäumen

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

Entscheidungsbaum-Klassifikatoren

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 (besondere Methoden zur Vermeidung von Overfitting für Entscheidungsbäume)