Data Mining

I. H. Witten and E. Frank

übersetzt von N. Fuhr

4

Algorithmen

Die grundlegenden Methoden

- Das Einfachste zuerst: 1R
- Berücksichtigung aller Attribute: Der Naive Bayes'sche Ansatz
- Entscheidungsbäume: ID3
- Abdeckungsalgorithmen: Entscheidungsregeln: PRISM
- Assoziationsregeln
- Lineare Modelle
- Instanzbasiertes Lernen

Das Einfachste zuerst

- Einfache Algorithmen funktionieren oft erstaunlich gut!
- Es gibt viele Arten von einfachen Strukturen, z. B.:
  - Nur ein Attribut betrachten
  - Alle Attribute tragen in gleichem Maße und unabhängig voneinander zum Ergebnis bei
  - Eine gewichtete Linearkombination liefert das Gewünschte
  - Instanzbasiert: Benutze eine wenige Prototypen
  - Benutze einfache logische Regeln
- Erfolg der Methoden hängt vom Anwendungsgebiet ab

Inferieren rudimentärer Regeln

- 1R: lernt einen einstufigen Entscheidungsbaum
  - D. h., alle Regeln testen ein bestimmtes Attribut
- Grundversion
  - Ein Zweig für jeden möglichen Wert
  - In jedem Zweig wird häufigste Klasse zugewiesen
  - Fehlerquote: Anteil der Instanzen, die nicht zur Mehrheitsklasse im jeweiligen Zweig gehören
  - Wähle Attribut mit der geringsten Fehlerquote

(*für nominale Attribute*)

Pseudocode für 1R

```
For each attribute,
   For each value of the attribute,
    make a rule as follows:
      count how often each class appears
      find the most frequent class
      make the rule assign that class to
       this attribute-value
   Calculate the error rate of the rules
Choose the rules with the smallest error rate
```

- Anmerkung: “missing” wird als eigener Attributwert behandelt

Evaluierung der Wetter-Attribute

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Attribute	Rules	Errors	Total errors
Outlook	Sunny → No	2/5	4/14
	Overcast → Yes	0/4	
	Rainy → Yes	2/5	
Temp	Hot → No*	2/4	5/14
	Mild → Yes	2/6	
	Cool → Yes	1/4	
Humidity	High → No	3/7	4/14
	Normal → Yes	1/7	
Windy	False → Yes	2/8	5/14
	True → No*	3/6	

* Keine eindeutige Entscheidung

Behandlung nummerischer Attribute

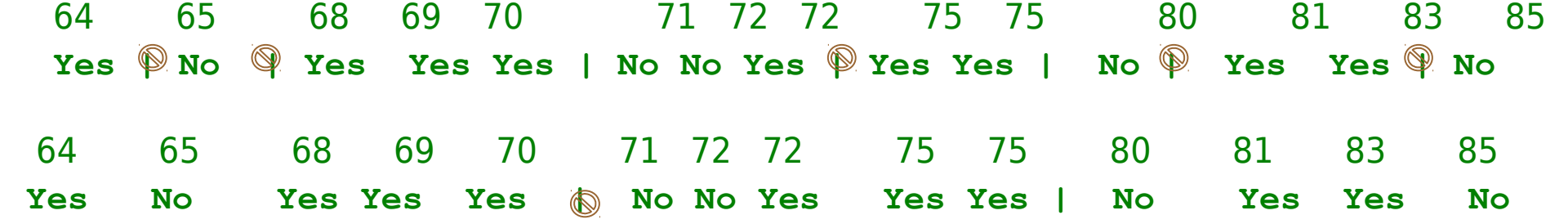
- Diskretisiere nummerische Attribute
- Aufteilung jedes Wertebereichs in Intervalle
  - Sortiere Instanzen nach den Attributwerten
  - Platzie
    ändert
    (die Me
  - Führt z
- Beispiel: *temperature* aus den Wetterdaten

Outlook	Temperature	Humidity	Windy	Play
Sunny	85	85	False	No
Sunny	80	90	True	No
Overcast	83	86	False	Yes
Rainy	75	80	False	Yes
...	...	...	...	...

```
 64    65    68   69   70    71   72   72    75   75    80    81   83    85
Yes | No |  Yes  Yes  Yes | No   No  | Yes   Yes  Yes |  No | Yes  Yes  | No
```

Das Problem der Überadaption

- Vorgeschlagenes Verfahren ist sehr empfindlich gegenüber Rauschen
  - Einzelne Instanz mit fehlerhafter Klassenzuordnung kann ein neues Intervall generieren
- Außerdem: Attribut *time stamp* würde Fehlerquote 0 erzeugen
- Einfache Lösung:
  Fordere minimale Anzahl von Instanzen der Mehrheitsklasse pro Intervall
- Beispiel (mit min = 3):



Mit der Vermeidung von Überadaption

- Resultierende Regelmenge:

Attribute	Rules	Errors	Total errors
Outlook	Sunny → No	2/5	4/14
	Overcast → Yes	0/4	
	Rainy → Yes	2/5	
Temperature	≤ 77.5 → Yes	3/10	5/14
	> 77.5 → No*	2/4	
Humidity	≤ 82.5 → Yes	1/7	3/14
	> 82.5 and ≤ 95.5 → No	2/6	
	> 95.5 → Yes	0/1	
Windy	False → Yes	2/8	5/14
	True → No*	3/6	

Diskussion von 1R

- 1R wurde in einem Artikel von Holte (1993) beschrieben
  - Enthält experimentelle Evaluierung von 16 Datensätzen (Benutzung von Cross-Validierung, so dass die Ergebnisse repräsentativ für zukünftige Daten sind)
  - Minimale Anzahl von Instanzen wurde auf 6 festgelegt (nach Vorexperimenten)
  - 1R-Regeln funktionierten kaum schlechter als wesentlich komplexere Entscheidungsbäume
- Einfachheit zahlt sich aus!

Very Simple Classification Rules Perform Well on Most Commonly Used Datasets

Robert C. Holte, Computer Science Department, University of Ottawa

Statistische Modellierung

- "Gegenteil" von 1R: Benutze alle Attribute
- Zwei Annahmen: Attribute sind
  - *Gleich wichtig*
  - *Statistisch unabhängig* (in Bezug auf den Klassenwert)
    - D. h., Kenntnis des Wertes eines Attributs sagt nichts über den Wert eines anderen Attributs (wenn die Klasse bekannt ist)
- Unabhängigkeitsannahme stimmt nie!
- Aber ... dieses Verfahren funktioniert gut in der Praxis

Wahrscheinlichkeiten für die Wetterdaten

Outlook			Temperature			Humidity			Windy			Play	
	Yes	*No*		*Yes*	*No*		*Yes*	*No*		*Yes*	*No*	*Yes*	*No*
Sunny	2	3	Hot	2	2	High	3	4	False	6	2	9	5
Overcast	4	0	Mild	4	2	Normal	6	1	True	3	3		
Rainy	3	2	Cool	3	1								
Sunny	2/9	3/5	Hot	2/9	2/5	High	3/9	4/5	False	6/9	2/5	9/14	5/14
Overcast	4/9	0/5	Mild	4/9	2/5	Normal	6/9	1/5	True	3/9	3/5		
Rainy	3/9	2/5	Cool	3/9	1/5								

Rainy	**Mild**	**High**	**False**	**Yes**
Rainy	**Cool**	**Normal**	**False**	**Yes**
Rainy	**Cool**	**Normal**	**True**	**No**
Overcast	**Cool**	**Normal**	**True**	**Yes**
Sunny	**Mild**	**High**	**False**	**No**
Sunny	**Cool**	**Normal**	**False**	**Yes**
Rainy	**Mild**	**Normal**	**False**	**Yes**
Sunny	**Mild**	**Normal**	**True**	**Yes**
Overcast	**Mild**	**High**	**True**	**Yes**
Overcast	**Hot**	**Normal**	**False**	**Yes**
Rainy	**Mild**	**High**	**True**	**No**

Wahrscheinlichkeiten für die Wetterdaten

Outlook			Temperature			Humidity			Windy			Play	
	Yes	*No*		*Yes*	*No*		*Yes*	*No*		*Yes*	*No*	*Yes*	*No*
Sunny	2	3	Hot	2	2	High	3	4	False	6	2	9	5
Overcast	4	0	Mild	4	2	Normal	6	1	True	3	3		
Rainy	3	2	Cool	3	1								
Sunny	2/9	3/5	Hot	2/9	2/5	High	3/9	4/5	False	6/9	2/5	9/14	5/14
Overcast	4/9	0/5	Mild	4/9	2/5	Normal	6/9	1/5	True	3/9	3/5		
Rainy	3/9	2/5	Cool	3/9	1/5								

- Ein neuer Tag:

Outlook	Temp.	Humidity	Windy	Play
Sunny	Cool	High	True	?

Likelihood of the two classes

For “yes” = $2/9 \times 3/9 \times 3/9 \times 3/9 \times 9/14 = 0.0053$

For “no” = $3/5 \times 1/5 \times 4/5 \times 3/5 \times 5/14 = 0.0206$

Conversion into a probability by normalization:

P(“yes”) = 0.0053 / (0.0053 + 0.0206) = 0.205

P(“no”) = 0.0206 / (0.0053 + 0.0206) = 0.795

Bayes'sche Regel

- Wahrscheinlichkeit des Ereignisses *H*, *wenn die Evidenz E gegeben ist* :

$$\Pr[H \mid E] = \frac{\Pr[E \mid H]\Pr[H]}{\Pr[E]}$$

$$Pr[Grippe \mid Schnupfen, Fieber] = \frac{Pr[Schnupfen, Fieber \mid Grippe]\, Pr[Grippe]}{Pr[Schnupfen, Fieber]}$$

- *Apriori*-Wahrscheinlichkeit von *H* : $\Pr[H]$
  - Wahrscheinlichkeit eines Ereignisses, *bevor* die Evidenz bekannt ist
- *Aposteriori*-Wahrscheinlichkeit *von H* : $\Pr[H \mid E]$
  - Wahrscheinlichkeit eines Ereignisses, *nachdem* die Evidenz bekannt ist

Naiver Bayes für die Klassifikation

- Lernen von Klassifikationen: Wahrscheinlichkeit einer Klasse, wenn die Instanz gegeben ist?
  - Evidenz E = Instanz
  - Ereignis H = Klassenwert der Instanz
- Naive Annahme: Evidenz kann aufgeteilt werden auf die Attribute, die *unabhängig* voneinander sind

$$\Pr[H|E] = \frac{\Pr[E_1|H]\Pr[E_2|H]\ldots\Pr[E_n|H]\Pr[H]}{\Pr[E]}$$

Wetterdaten-Beispiel

Outlook	Temp.	Humidity	Windy	Play
Sunny	Cool	High	True	?

← ***Evidenz E***

$$\Pr[yes|E] = \Pr[Outlook=Sunny|yes] \times$$

$$\Pr[Temperature=Cool|yes] \times$$

$$\Pr[Humidity=High|yes] \times$$

$$\Pr[Windy=True|yes] \times$$

$$\frac{\Pr[yes]}{\Pr[E]}$$

$$= \frac{\frac{2}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{9}{14}}{\Pr[E]}$$

Wahrscheinlichkeit der Klasse "yes"

Das Problem der Häufigkeit 0

- Was tun, wenn ein Attributwert nicht mit jedem Klassenwert vorkommt?
  (z. B. “Outlook=overcast” für die Klasse “no”)
  - Wahrscheinlichkeit ergibt sich zu 0!
  - *Aposteriori*–Wahrscheinlichkeit ergibt sich ebenfalls zu 0!
    (Egal, wie groß die anderen Wahrscheinlichkeiten sind!)

$$\Pr[Outlook=overcast|no]=0 \qquad \Pr[no|E]=0$$

- Abhilfe: Addiere 1 zum Zähler für jede Attributwert-Klassenwert-Kombination (*Laplace-Schätzer)*
- Ergebnis: Wahrscheinlichkeiten können nie 0 werden!
  (außerdem: die Wahrscheinlichkeitsschätzungen werden stabilisiert)

Modifizierte Wahrscheinlichkeitsschätzer

- In einigen Fällen liefert die Addition einer von 1 verschiedenen Konstante bessere Ergebnisse
- Beispiel: Attribut *outlook* für die Klasse *yes*

$$\frac{2+\mu/3}{9+\mu} \qquad \frac{4+\mu/3}{9+\mu} \qquad \frac{3+\mu/3}{9+\mu}$$

Sunny ***Overcast*** ***Rainy***

- Die Gewichte müssen nicht gleich sein (aber deren Summe muss 1 sein)

$$\frac{2+\mu p_1}{9+\mu} \qquad \frac{4+\mu p_2}{9+\mu} \qquad \frac{3+\mu p_3}{9+\mu}$$

Fehlende Werte

- Training: Instanz wird bei der Häufigkeitszählung für die Attributwert-Klassenkombination nicht berücksichtigt
- Klassifikation: Attribut wird bei der Berechnung ausgelassen
- Beispiel:

Outlook	Temp.	Humidity	Windy	Play
?	Cool	High	True	?

Likelihood of “yes” = $3/9 \times 3/9 \times 3/9 \times 9/14 = 0.0238$

Likelihood of “no” = $1/5 \times 4/5 \times 3/5 \times 5/14 = 0.0343$

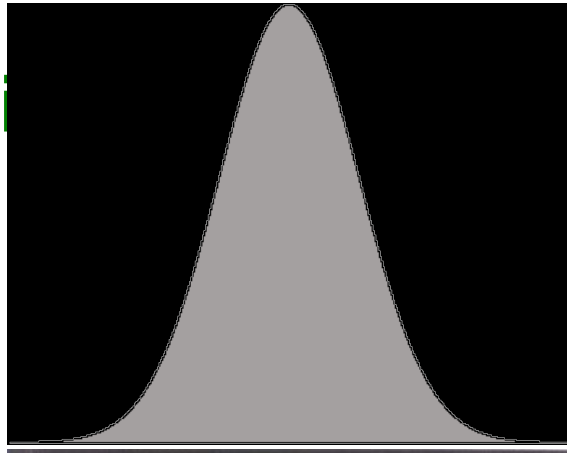
P(“yes”) = 0.0238 / (0.0238 + 0.0343) = 41%

P(“no”) = 0.0343 / (0.0238 + 0.0343) = 59%

Nummerische Attribute

- Übliche Annahme: Attributwerte sind normalverteilt (innerhalb jeder Klasse)
- Die *Dichtefunktion* für die Normalverteilung ist durch zwei Parameter bestimmt:
  - *Mittelwert* μ

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$



  - *Standardabweichung* σ

$$\sigma = \frac{1}{n-1}\sum_{i=1}^{n} (x_i - \mu)^2$$

  - Dichtefunktion *f(x):*

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Anwendung auf die Wetterdaten

Outlook			Temperature		Humidity		Windy			Play	
	Yes	*No*	*Yes*	*No*	*Yes*	*No*		*Yes*	*No*	*Yes*	*No*
Sunny	2	3	64, 68,	65, 71,	65, 70,	70, 85,	False	6	2	9	5
Overcast	4	0	69, 70,	72, 80,	70, 75,	90, 91,	True	3	3		
Rainy	3	2	72, …	85, …	80, …	95, …					
Sunny	2/9	3/5	μ =73	μ =75	μ =79	μ =86	False	6/9	2/5	9/14	5/14
Overcast	4/9	0/5	σ =6.2	σ =7.9	σ =10.2	σ =9.7	True	3/9	3/5		
Rainy	3/9	2/5									

- Beispiel für Dichtewert:

$$f(temperature=66|yes)=\frac{1}{\sqrt{2\pi}6.2}e^{-\frac{(66-73)^2}{2*6.2^2}}=0.0340$$

Klassifikation eines neuen Tages

- Ein neuer Tag:

Outlook	Temp.	Humidity	Windy	Play
Sunny	66	90	true	?

Likelihood of “yes” = $2/9 \times 0.0340 \times 0.0221 \times 3/9 \times 9/14 = 0.000036$

Likelihood of “no” = $3/5 \times 0.0291 \times 0.0380 \times 3/5 \times 5/14 = 0.000136$

P(“yes”) = 0.000036 / (0.000036 + 0. 000136) = 20.9%

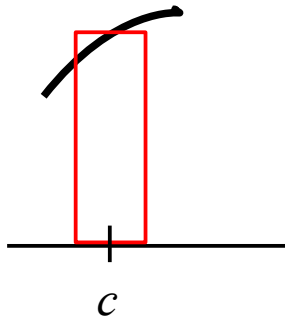
P(“no”) = 0.000136 / (0.000036 + 0. 000136) = 79.1%

- Fehlende Werte in den Trainingsdaten werden bei der Berechnung von Mittelwert und Standardabweichung nicht berücksichtigt

Wahrscheinlichkeitsdichten

- Beziehung zwischen Wahrscheinlichkeit und Dichte:

$$\Pr[c-\frac{\varepsilon}{2}<x<c+\frac{\varepsilon}{2}]\approx\varepsilon*f(c)$$



- Aber: dies hat keinen Einfluss auf die *aposteriori*-Wahrscheinlichkeiten, da ε herausfällt
- Exakte Beziehung:

$$\Pr[a\le x\le b]=\int_a^b f(t)dt$$

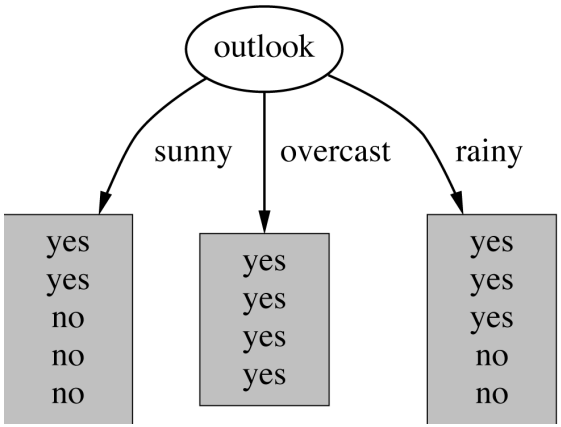
Naiver Bayes: Diskussion

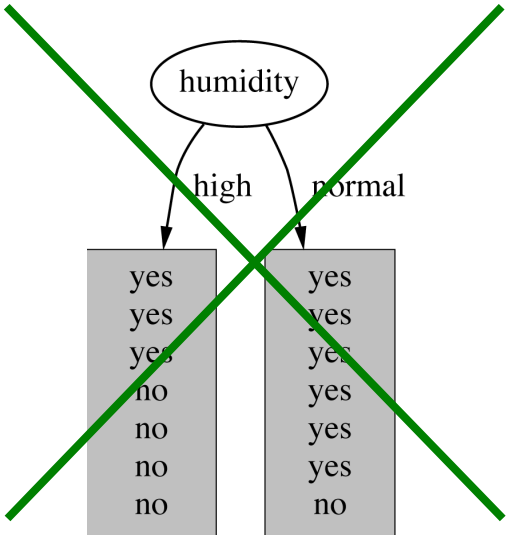
- Naiver Bayes funktioniert überraschend gut (selbst wenn die Unabhängigkeitsannahme klar verletzt ist)
- Warum? Weil die Klassifikation keine exakten Wahrscheinlichkeitsschätzungen benötigt, *solange die maximale Wahrscheinlichkeit der korrekten Klasse zugewiesen wird*
- Allerdings: Hinzufügen zu vieler redundanter Attribute führt zu Problemen (z. B. identische Attribute)
- Außerdem: viele nummerische Attribute sind nicht normalverteilt ($\rightarrow$ *kernel density-Schätzer*)

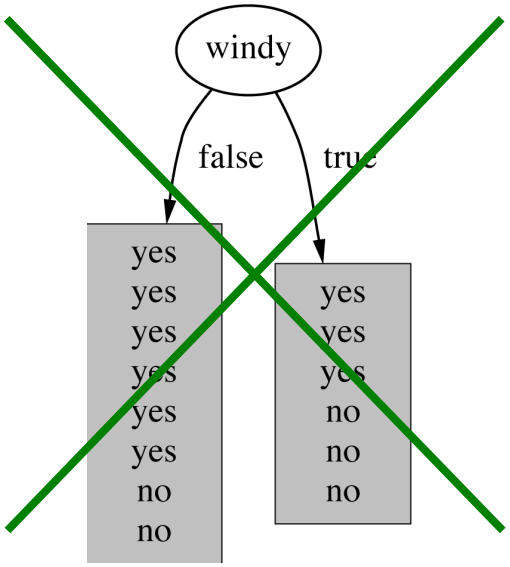
Konstruktion von Entscheidungsbäumen

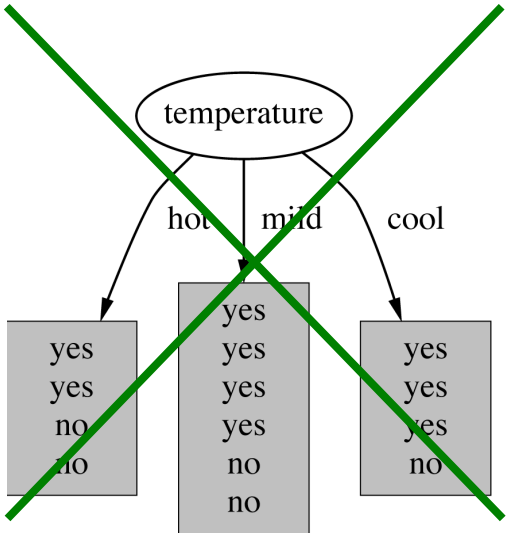
- Strategie: top-down
  Rekursiver *teile-und-herrsche*-Ansatz
  1. Beginn: wähle Attribut für den Wurzelknoten. Erzeuge einen Zweig für jeden möglichen Attributwert
  2. Dann: Teile die Instanzen in Teilmengen auf – eine für jeden Zweig vom Wurzelknoten
  3. Weiter: wiederhole Schritte 1 und 2 rekursiv für jeden Zweig, wobei nur die Instanzen berücksichtigt werden, die zum jeweiligen Zweig gehören
- Stopp, wenn alle Instanzen zur selben Klasse gehören

Attributauswahl









Kriterium für die Attributauswahl

- Welches ist das beste Attribut?
  - Ziel: möglichst kleiner Baum
  - Heuristik: wähle dasjenige Attribut, das die “reinsten” Knoten erzeugt
- Populäres *Unreinheits-Kriterium: Informationsgewinn (information gain)*
  - Informationsgewinn steigt mit der durchschnittlichen Reinheit der Teilmengen
- Strategie: wähle das Attribut mit dem größten Informationsgewinn

Berechnung der Information

- Messen der Informationsmenge in *Bits*
  - Für eine gegebene Wahrscheinlichkeitsverteilung ist die Information, die benötigt wird, um ein Ereignis vorherzusagen, die *Entropie* der Verteilung
  - Entropie misst die benötigte Information in Bits
    (dies können auch Bruchteile von Bits sein)
- Formel zur Berechnung der Entropie:

$$entropy(p_1, p_2, \ldots, p_n) = -p_1 \log(p_1) - p_2 \log(p_2) - \ldots - p_n \log(p_n)$$

Claude Shannon

"Father of information theory"

Born: 30 April 1916

Died: 23 February 2001

Claude Shannon, who has died aged 84, perhaps more than anyone laid the groundwork for today's digital revolution. His exposition of information theory, stating that all information could be represented mathematically as a succession of noughts and ones, facilitated the digital manipulation of data without which today's information society would be unthinkable.

Shannon's master's thesis, obtained in 1940 at MIT, demonstrated that problem solving could be achieved by manipulating the symbols 0 and 1 in a process that could be carried out automatically with electrical circuitry. That dissertation has been hailed as one of the most significant master's theses of the 20th century. Eight years later, Shannon published another landmark paper, *A Mathematical Theory of Communication*, generally taken as his most important scientific contribution.

Shannon applied the same radical approach to cryptography research, in which he later became a consultant to the US government.

Many of Shannon's pioneering insights were developed before they could be applied in practical form. He was truly a remarkable man, yet unknown to most of the world.

Beispiel: Attribut *Outlook*

- *Outlook = Sunny* :

$$\text{info}([2,3]) = \text{entropy}(2/5, 3/5) = -2/5\log(2/5) - 3/5\log(3/5) = 0.971 \text{ bits}$$

- *Outlook = Overcast* :

$$\text{info}([4,0]) = \text{entropy}(1,0) = -1\log(1) - 0\log(0) = 0 \text{ bits}$$

Anmerkung: dies ist normalerweise undefiniert.

- *Outlook = Rainy* :

$$\text{info}([3,2]) = \text{entropy}(3/5, 2/5) = -3/5\log(3/5) - 2/5\log(2/5) = 0.971 \text{ bits}$$

- Erwartete Information für das Attribut:

$$\text{info}([3,2],[4,0],[3,2]) = (5/14)\times 0.971 + (4/14)\times 0 + (5/14)\times 0.971$$

$$= 0.693 \text{ bits}$$

Berechnung des Informationsgewinns

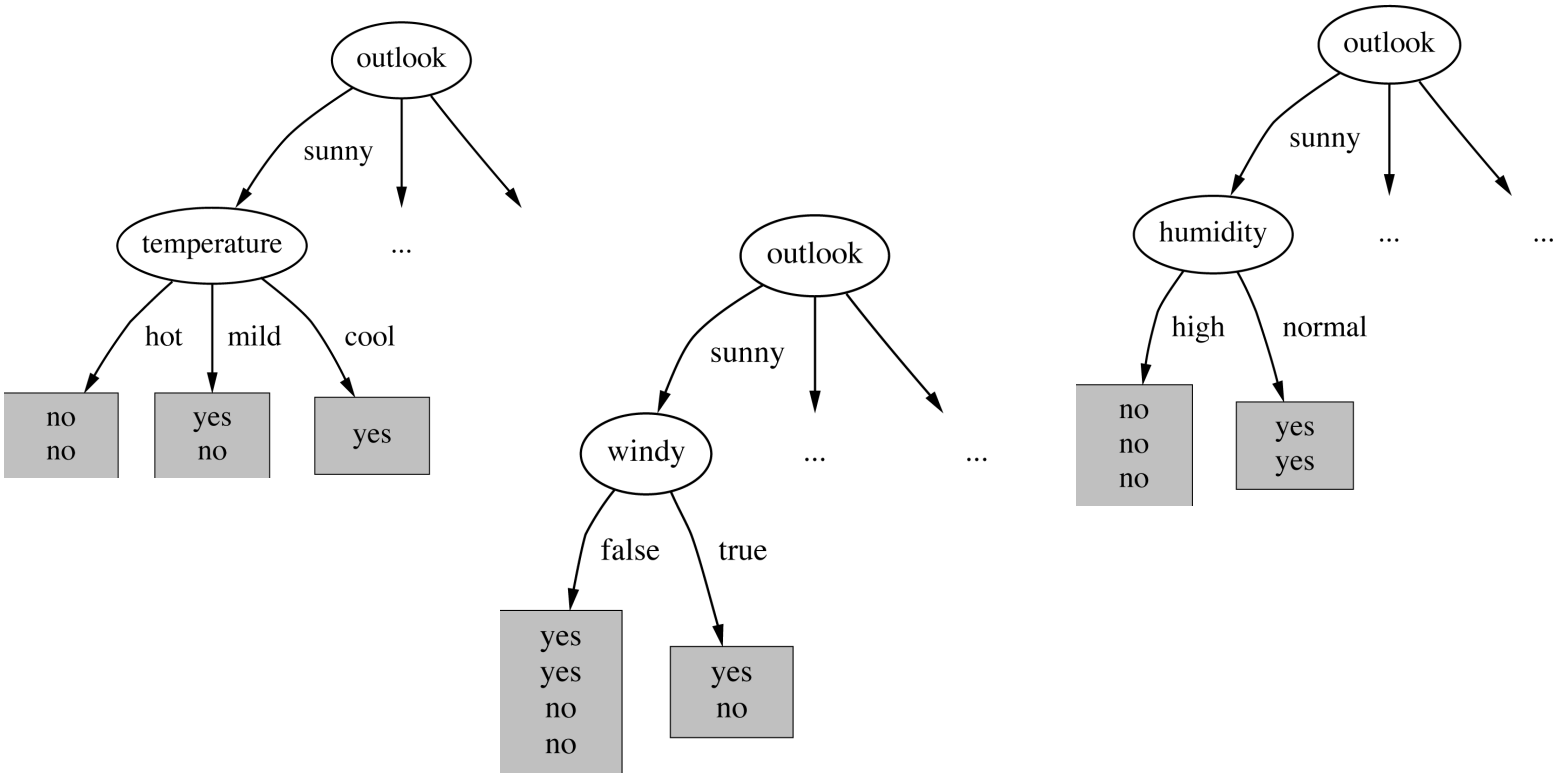
- Informationsgewinn: Information vor der Aufteilung – Information nach der Aufteilung

gain(*Outlook*)= info([9,5]) – info([2,3],[4,0],[3,2])
= 0.940 – 0.693
= 0.247 bits

- Informationsgewinn für die Attribute der Wetterdaten:

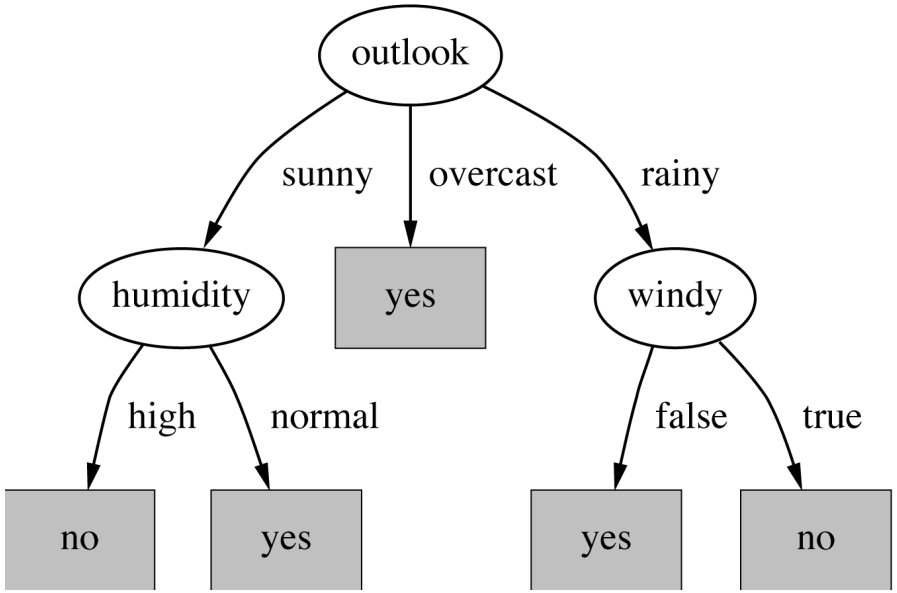
gain(*Outlook*) = 0.247 bits
gain(*Temperature*) = 0.029 bits
gain(*Humidity*) = 0.152 bits
gain(*Windy*) = 0.048 bits

Weitere Aufteilung



gain(*Temperature*) = 0.571 bits
gain(*Humidity*) = 0.971 bits
gain(*Windy*) = 0.020 bits

Vollständiger Entscheidungsbaum



- Anm.: nicht alle Blätter müssen rein sein; manchmal gehören gleiche Instanzen zu verschiedenen Klassen
  - ⇒ Aufteilung stoppt, wenn die Daten nicht weiter aufgeteilt werden können

Anforderungen an ein Reinheitsmaß

- Eigenschaften eines Reinheitsmaßes:
  - Für reine Knoten soll das Maß den Wert 0 liefern
  - Wenn die Unreinheit maximal ist (d.h. alle Klassen gleich wahrscheinlich), sollte das Maß den maximalen Wert liefern
  - Das Maß sollte die *Mehrstufigkeits-Eigenschaft besitzen* (d. h. Entscheidungen können in mehreren Schritten gefällt werden):

$$\text{measure}([2,3,4]) = \text{measure}([2,7]) + (7/9) \times \text{measure}([3,4])$$

- Entropie ist die einzige Funktion, die alle drei Eigenschaften besitzt!

Eigenschaften der Entropie

- Die Mehrstufigkeitseigenschaft:

$$\text{entropy}(p,q,r)=\text{entropy}(p,q+r)+(q+r)\times\text{entropy}\left(\frac{q}{q+r},\frac{r}{q+r}\right)$$

- Vereinfachung der Berechnung

$$\text{info}([2,3,4])=-2/9\times\log(2/9)-3/9\times\log(3/9)-4/9\times\log(4/9)$$
$$=[-2\log 2-3\log 3-4\log 4+9\log 9]/9$$

- Anm.: anstelle der Maximierung des Informationsgewinns kann man auch einfach die Information minimieren

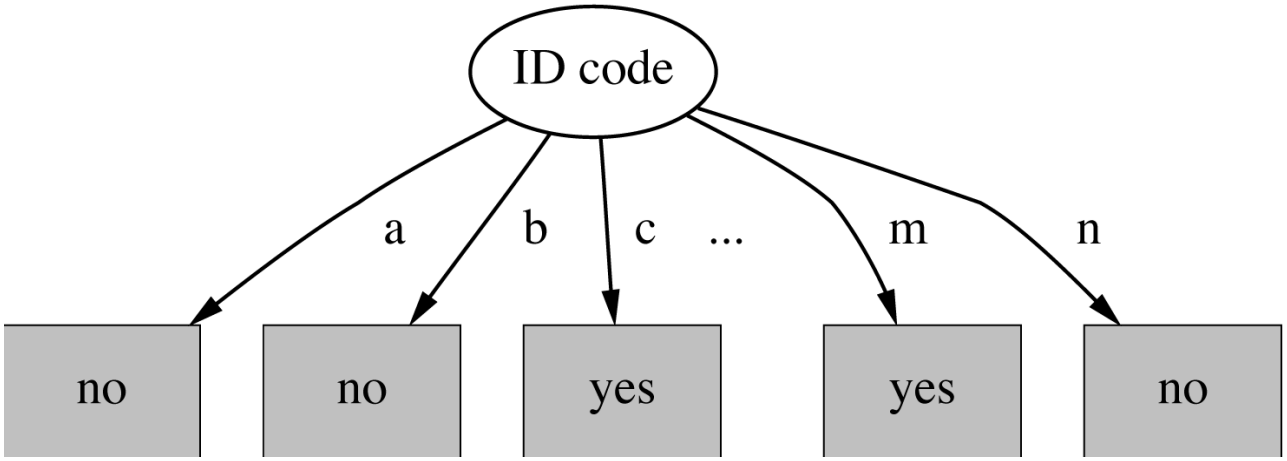
Stark verzweigende Attribute

- Problematisch: Attribute mit einer hohen Anzahl von Werten (Extremfall: ID-Code)
- Teilmengen sind eher rein, wenn es eine große Menge von Werten gibt
  - ⇒ Informationsgewinn verzerrt, indem Attribute mit hoher Anzahl von Werten bevorzugt werden
  - ⇒ Dies kann zur *Überadaption* führen (Auswahl eines Attributs, das nicht optimal für die Vorhersage ist)
    - fehlerhafte Wahrscheinlichkeitsschätzung aufgrund zu kleiner Beobachtungszahlen
    - Beobachtete Werte nicht repräsentativ für die Anwendung (ID-Code, Timestamp)
- Anderes Problem: *Fragmentierung*

Wetterdaten mit *ID-Codes*

ID code	Outlook	Temp.	Humidity	Windy	Play
A	Sunny	Hot	High	False	No
B	Sunny	Hot	High	True	No
C	Overcast	Hot	High	False	Yes
D	Rainy	Mild	High	False	Yes
E	Rainy	Cool	Normal	False	Yes
F	Rainy	Cool	Normal	True	No
G	Overcast	Cool	Normal	True	Yes
H	Sunny	Mild	High	False	No
I	Sunny	Cool	Normal	False	Yes
J	Rainy	Mild	Normal	False	Yes
K	Sunny	Mild	Normal	True	Yes
L	Overcast	Mild	High	True	Yes
M	Overcast	Hot	Normal	False	Yes
N	Rainy	Mild	High	True	No

Baumstumpf für das ID-Code-Attribut



- Entropie dieser Aufteilung::

$$\text{info}(IDcode) = \text{info}([0,1]) + \text{info}([0,1]) + \ldots + \text{info}([0,1]) = 0 \text{ bits}$$

$\Rightarrow$ Informationsgewinn ist maximal für ID-Code (nämlich 0.940 Bits)

Gewinnverhältnis

- *Gewinnverhältnis*: eine Modifikation des Informationsgewinns, die dessen Bias reduziert
- Gewinnverhältnis berücksichtigt die Anzahl und die Größe der Zweige bei der Auswahl des besten Attributs
  - Es korrigiert den Informationsgewinn durch Berücksichtigung der *intrinsischen Information* einer Aufteilung
- Intrinsische Information: Entropie der Verteilung der Instanzen auf die Zweige (d. h. wieviel Information benötigen wir, um zu bestimmen, zu welchem Zweig eine Instanz gehört)

Berechnung des Gewinnverhältnisses

- Beispiele: intrinsische Information von ID-Code

$$\text{info}([1,1,\ldots,1]) = 14(-1/14 \log(1/14)) = 3.807 \text{ bits}$$

- Wert eines Attributes reduziert sich, wenn die intrinsische Information wächst
- Definition des Gewinnverhältnisses:

$$\text{gain\_ratio}(Attribute) = \frac{\text{gain}(Attribute)}{\text{intrinsic\_info}(Attribute)}$$

- Beispiel:

$$\text{gain\_ratio}(ID_{code}) = \frac{0.940 \text{ bits}}{3.807 \text{ bits}} = 0.246$$

Gewinnverhältnisse für die Wetterdaten

Outlook		Temperature	
Info:	0.693	Info:	0.911
Gain: 0.940-0.693	0.247	Gain: 0.940-0.911	0.029
Split info: info([5,4,5])	1.577	Split info: info([4,6,4])	1.362
Gain ratio: 0.247/1.577	0.156	Gain ratio: 0.029/1.362	0.021
Humidity		**Windy**	
Info:	0.788	Info:	0.892
Gain: 0.940-0.788	0.152	Gain: 0.940-0.892	0.048
Split info: info([7,7])	1.000	Split info: info([8,6])	0.985
Gain ratio: 0.152/1	0.152	Gain ratio: 0.048/0.985	0.049

Mehr zum Gewinnverhältnis

- “Outlook” ist immer noch das beste Attribut
- Aber: “ID code” hat ein besseres Gewinnverhältnis
  - Standard-Korrektur: *adhoc-Test,* um Aufteilung nach diesem Attribut zu verhindern
- Problem mit dem Gewinnverhältnis: es kann überkompensieren
  - Es kann ein Attribut wählen, nur weil dessen intrinsische Information sehr niedrig ist
  - Standard-Korrektur: Nur solche Attribute berücksichtigen, deren Informationsgewinn größer als der Durchschnitt ist

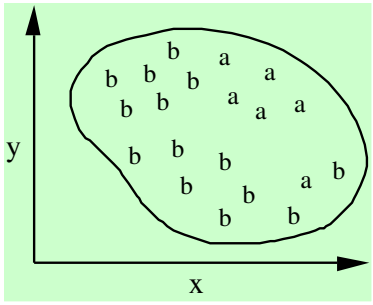
Diskussion

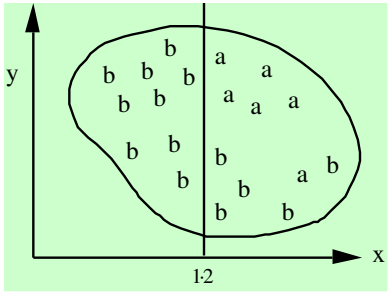
- Top-down Generierung von Entscheidungsbäumen: ID3-Algorithmus, entwickelt von Ross Quinlan
  - Gewinnverhältnis ist nur eine Modifikation des grundlegenden Algorithmus
  - $\Rightarrow$ C4.5: behandelt nummerische Attribute, fehlende Werte, verrauschte Daten
- Ähnlicher Ansatz: CART
- Es gibt viele andere Kriterien zur Attributauswahl!
  (Aber nur geringe Unterschiede in der Qualität der Ergebnisse)

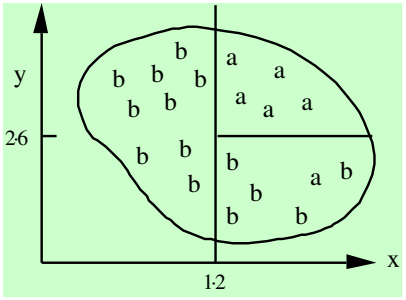
Abdeckungsalgorithmen

- Konvertiere Entscheidungsbaum in Regelmenge
  - Einfach, aber die Regelmenge ist unnötig komplex
  - Effektivere Konvertierungen sind nicht trivial
- Stattdessen kann man die Regelmenge direkt generieren
  - Bestimme für jede Klasse diejenige Regelmenge, die alle zugehörigen Instanzen abdeckt (und Instanzen anderer Klassen zurückweist)
- Wird als *Abdeckungs*-Ansatz bezeichnet:
  - In jedem Schritt wird eine Regel bestimmt, die einige Instanzen abdeckt

Beispiel: Generieren einer Regel








```
If true
   then class = a
```

```
If x > 1.2
   then class = a
```

```
If x > 1.2 and y > 2.6
   then class = a
```

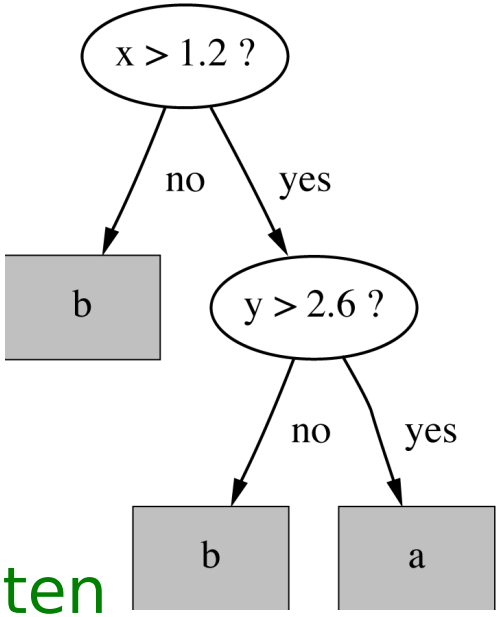
- Mögliche Regelmenge für die Klasse “b”:

```
If x ≤ 1.2 then class = b
If x > 1.2 and y ≤ 2.6 then class = b
```

- Weitere Regeln können hinzugefügt werden, um “perfekte” Regelmenge zu gewinnen

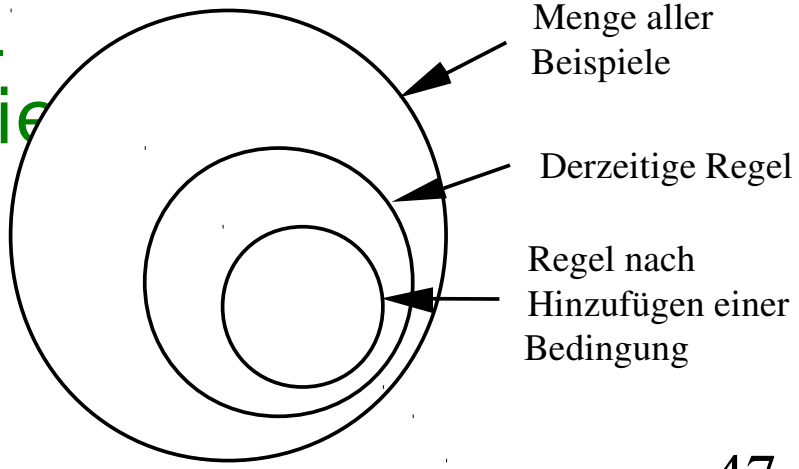
Regeln vs. Bäume

- Korrespondierender Entscheidungsbaum: (produziert exakt die gleichen Vorhersagen)
- Aber: Regelmengen *können* kompakter sein, während Entscheidungsbäume oft replizierte Teilbäume enthalten
- Außerdem: Bei mehr als 2 Klassen konzentriert sich der Abdeckungsalgorithmus jeweils auf eine Klasse, während Entscheidungsbäume stets alle Klassen berücksichtigen



Einfacher Abdeckungsalgorithmus

- Generiert eine Regel durch Hinzufügen von Bedingungen, die die Präzision der Regel erhöhen
- Ähnlich zur Situation bei Entscheidungsbäumen: Auswahl eines Attributs als Aufteilungskriterium
  - Aber: Entscheidungsbaum-Lerner maximiert die globale Reinheit
- Jede zusätzliche Bedingung reduziert die Abdeckung einer Regel:



Auswahl einer Bedingung

- Ziel: Maximierung der Präzision
  - t Anzahl der durch die Regel abgedeckten Instanzen
  - p Anzahl positiver, abgedeckter Beispiele
  - $t - p$ Anzahl der Fehler der Regel
  - $\Rightarrow$ Wähle Bedingung, die *p/t maximiert*
- Fertig, wenn $p/t = 1$ oder die Menge der Instanzen nicht weiter aufgeteilt werden kann

Beispiel: Kontaktlinsen-Daten

- Gesuchte Regel:

```
If ?
    then recommendation = hard
```

- Mögliche Bedingungen:

Age = Young	2/8
Age = Pre-presbyopic	1/8
Age = Presbyopic	1/8
Spectacle prescription = Myope	3/12
Spectacle prescription = Hypermetrope	1/12
Astigmatism = no	0/12
Astigmatism = yes	4/12
Tear production rate = Reduced	0/12
Tear production rate = Normal	4/12

Modifizierte Regel und resultierende Daten

- Regel mit bester Bedingung:

```
If astigmatism = yes
    then recommendation = hard
```

- Dadurch abgedeckte Instanzen:

Age	Spectacle prescription	Astigmatism	Tear production rate	Recommended lenses
Young	Myope	Yes	Reduced	None
Young	Myope	Yes	Normal	Hard
Young	Hypermetrope	Yes	Reduced	None
Young	Hypermetrope	Yes	Normal	hard
Pre-presbyopic	Myope	Yes	Reduced	None
Pre-presbyopic	Myope	Yes	Normal	Hard
Pre-presbyopic	Hypermetrope	Yes	Reduced	None
Pre-presbyopic	Hypermetrope	Yes	Normal	None
Presbyopic	Myope	Yes	Reduced	None
Presbyopic	Myope	Yes	Normal	Hard
Presbyopic	Hypermetrope	Yes	Reduced	None
Presbyopic	Hypermetrope	Yes	Normal	None

Weitere Verfeinerung

- Momentaner Zustand:

```
If astigmatism = yes
    and ?
  then recommendation = hard
```

- Mögliche Bedingungen

Age = Young	2/4
Age = Pre-presbyopic	1/4
Age = Presbyopic	1/4
Spectacle prescription = Myope	3/6
Spectacle prescription = Hypermetrope	1/6
Tear production rate = Reduced	0/6
Tear production rate = Normal	4/6

Modifizierte Regel und resultierende Daten

- Regel mit bester Bedingung:

```
If astigmatism = yes
    and tear production rate = normal
then recommendation = hard
```

- Dadurch abgedeckte Instanzen:

Age	Spectacle prescription	Astigmatism	Tear production rate	Recommended lenses
Young	Myope	Yes	Normal	Hard
Young	Hypermetrope	Yes	Normal	hard
Pre-presbyopic	Myope	Yes	Normal	Hard
Pre-presbyopic	Hypermetrope	Yes	Normal	None
Presbyopic	Myope	Yes	Normal	Hard
Presbyopic	Hypermetrope	Yes	Normal	None

Weitere Verfeinerung

- Momentaner Zustand:

```
If astigmatism = yes
     and tear production rate = normal
     and ?
   then recommendation = hard
```

- Mögliche Bedingungen:

Age = Young	2/2
Age = Pre-presbyopic	1/2
Age = Presbyopic	1/2
Spectacle prescription = Myope	3/3
Spectacle prescription = Hypermetrope	1/3

- Unentschieden zwischen erster und vierter Bedingung
  - Wähle die mit der größeren Abdeckung

Das Endergebnis

- Vollständige Regel:

```
If astigmatism = yes
   and tear production rate = normal
   and spectacle prescription = myope
   then recommendation = hard
```

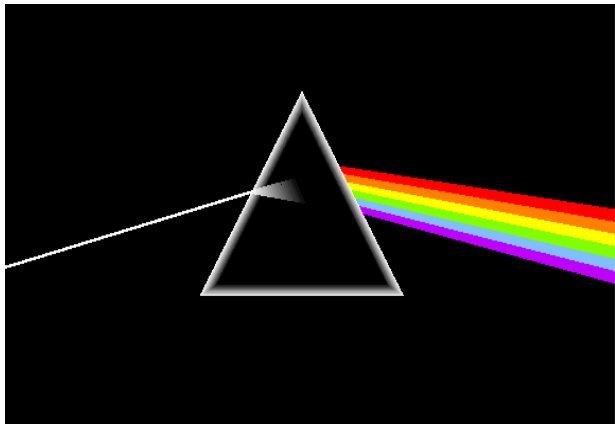
- Zweite Regel zur Empfehlung harter Linsen:
  (für die Instanzen, die durch die erste Regel nicht abgedeckt werden)

```
If age = young and astigmatism = yes
   and tear production rate = normal
   then recommendation = hard
```

- Diese zwei Regeln decken alle harten Linsen ab
  - Prozess wird für die beiden anderen Klassen wiederholt

Pseudo-Code für PRISM

```
For each class C
  Initialize E to the instance set
  While E contains instances in class C
    Create a rule R with an empty left-hand side that predicts class C
    Until R is perfect (or there are no more attributes to use) do
      For each attribute A not mentioned in R, and each value v,
        Consider adding the condition A = v to the left-hand side of R
        Select A and v to maximize the accuracy p/t
          (break ties by choosing the condition with the largest p)
      Add A = v to R
    Remove the instances covered by R from E
```



Regeln vs. Entscheidungslisten

- PRISM ohne die äußerste Schleife generiert eine Entscheidungsliste für eine Klasse
  - Nachfolgende Regeln fokussieren auf Beispiele, die durch vorangegangene Regeln nicht abgedeckt wurden
  - Aber: Reihenfolge ist egal, da alle Regeln die gleiche Klasse vorhersagen
- Äußerste Schleife berücksichtigt jede Klasse einzeln
  - Keine Reihenfolge impliziert
- Probleme: überlappende Regeln, Notwendigkeit für Default-Regel

Trenne und herrsche

- Methoden wie PRISM (zur Behandlung einer Klasse) sind *trenne-und-herrsche-*Algorithmen:
  - Beginn: Identifiziere sinnvolle Regel
  - Dann, separiere alle dadurch abgedeckten Instanzen
  - Schließlich, “herrsche” über die restlichen Instanzen
- Unterschied zu teile-und-herrsche-Algorithmen:
  - Die durch eine Regel abgedeckte Teilmenge muss nicht weiter betrachtet werden

Assoziationsregeln

- Assoziationsregeln…
  - … können beliebige Attribute und Attributkombinationen vorhersagen
  - … sollten nicht zusammen als Regelmenge benutzt werden
- Problem: große Anzahl von möglichen Assoziationen
  - Ausgabe muss auf die aussagekräftigsten Assoziationen beschränkt werden ⇒ Regeln mit großer *Unterstützung* und hoher *Konfidenz*

Unterstützung und Konfidenz einer Regel

- Unterstützung: Anzahl Instanzen, die korrekt vorhergesagt werden
- Konfidenz: Anzahl korrekter Vorhersagen, im Verhältnis zu allen Instanzen, auf die die Regel anwendbar ist
- Beispiel: 4 kühle Tage mit normaler Luftfeuchtigkeit

```
If temperature = cool then humidity = normal
```

⇒ Unterstützung = 4, Konfidenz = 100%

- Normalerweise: minimale Unterstützung und Konfidenz vorgegeben (z. B. 58 Regeln mit Unterstützung ≥ 2 und Konfidenz ≥ 95% für die Wetterdaten)

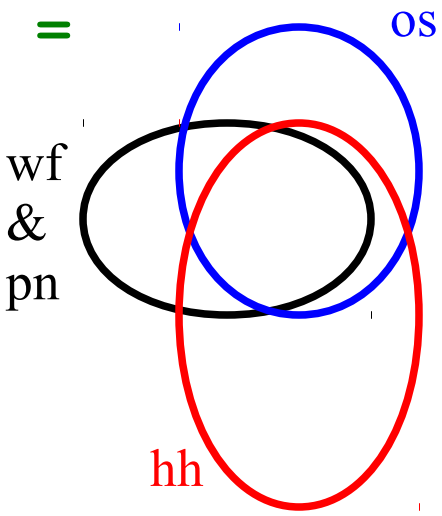
Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

If temperature = cool then humidity = normal

Interpretation von Assoziationsregeln

- Interpretation ist nicht offensichtlich:
  - **If windy = false and play = no then outlook = sunny and humidity = high**

  ist nicht das Gleiche wie
  - **If windy = false and play = no then outlook = sunny**
  - **If windy = false and play = no then humidity = high**



- Allerdings wird die folgende Regel impliziert:
  - **If humidity = high and windy = false and play = no then outlook = sunny**

Gewinnung von Assoziationsregeln

- Naive Methode zur Gewinnung von Assoziationsregeln :
  - Benutze trenne-und-herrsche-Methode
  - Behandle jede mögliche Kombination von Attributwerten als eigene Klasse
- Zwei Probleme:
  - Berechnungskomplexität
  - Resultierende Anzahl von Regeln (die gekürzt werden muss auf der Basis der Schwellenwerte für Unterstützung und Konfidenz)
- Aber: wir können direkt nach Regeln mit hoher Unterstützung suchen!

Item sets

- Unterstützung: Anzahl der Instanzen, die korrekt durch eine Assoziationsregel abgedeckt werden
  - = der Anzahl Instanzen, die *alle* Bedingungen der Regel erfüllen (linke und rechte Seite!)
- *Item*: ein Bedingung-Attributwert-Paar
- *Itemset* : alle Items, die in einer Regel vorkommen
- Ziel: nur Regeln, die den Schwellenwert für die Unterstützung mindestens erreichen

  ⇒ Gewinnung durch Auffinden aller Itemsets mit der vorgegebenen minimalen Unterstützung, anschließende Regelgenerierung hieraus!

Item sets für die Wetterdaten

One-item sets	Two-item sets	Three-item sets	Four-item sets
Outlook = Sunny (5)	Outlook = Sunny Temperature = Hot (2)	Outlook = Sunny Temperature = Hot Humidity = High (2)	Outlook = Sunny Temperature = Hot Humidity = High Play = No (2)
Temperature = Cool (4)	Outlook = Sunny Humidity = High (3)	Outlook = Sunny Humidity = High Windy = False (2)	Outlook = Rainy Temperature = Mild Windy = False Play = Yes (2)
...	...	...	...

- Insgesamt: 12 1-Itemsets, 47 2-Itemsets, 39 3-Itemsets, 6 4-Itemsets und 0 5-Itemsets

  (mit minimaler Unterstützung von 2)

Generierung von Regeln aus einem Itemset

- Nachdem alle Itemsets mit mindestens minimaler Unterstützung generiert worden sind, können wir daraus Regeln generieren
- Beispiel:

```
Humidity = Normal, Windy = False, Play = Yes (4)
```

- Sieben (2^N-1) potenzielle Regeln:

```
If Humidity = Normal and Windy = False then Play = Yes    4/4
If Humidity = Normal and Play = Yes then Windy = False    4/6
If Windy = False and Play = Yes then Humidity = Normal    4/6
If Humidity = Normal then Windy = False and Play = Yes    4/7
If Windy = False then Humidity = Normal and Play = Yes    4/8
If Play = Yes then Humidity = Normal and Windy = False    4/9
If True then Humidity = Normal and Windy = False
   and Play = Yes                                         4/12
```

Regeln für die Wetterdaten

- Regeln mit Unterstützung > 1 und Konfidenz = 100%:

	Association rule		Sup.	Conf.
1	Humidity=Normal Windy=False	⇒ Play=Yes	4	100%
2	Temperature=Cool	⇒ Humidity=Normal	4	100%
3	Outlook=Overcast	⇒ Play=Yes	4	100%
4	Temperature=Cold Play=Yes	⇒ Humidity=Normal	3	100%
	...	...	...	...
58	Outlook=Sunny Temperature=Hot	⇒ Humidity=High	2	100%

- Insgesamt:
  - 3 Regeln mit Unterstützung 4
  - 5 mit Unterstützung 3
  - 50 mit Unterstützung 2

Beispielregeln für dieselbe Menge

- Itemset:

```
Temperature = Cool, Humidity = Normal, Windy = False, Play = Yes (2)
```

- Resultierende Regeln (alle mit 100% Konfidenz):

```
Temperature = Cool, Windy = False ⇒ Humidity = Normal, Play = Yes
Temperature = Cool, Windy = False, Humidity = Normal ⇒ Play = Yes
Temperature = Cool, Windy = False, Play = Yes ⇒ Humidity = Normal
```

Basierend auf den folgenden “häufigen” Itemsets:

```
Temperature = Cool, Windy = False                      (2)
Temperature = Cool, Humidity = Normal, Windy = False   (2)
Temperature = Cool, Windy = False, Play = Yes          (2)
```

Effiziente Generierung von Itemsets

- Wie können effizient alle häufigen Itemsets gefunden werden?
- Auffinden der 1-Itemsets ist einfach
- Idee: Benutze 1-Itemsets, um 2-Itemsets zu generieren, benutze 2-Itemsets, um 3-Itemsets zu generieren, ...
  - Wenn (A B) ein häufiger Itemset ist, dann müssen auch (A) und (B) häufige Itemsets sein!
  - Allgemein: Falls X ein häufiger *k*-Itemset ist, dann sind alle (*k*-1)-Item-subsets von X auch häufig

  $\Rightarrow$ Berechne *k*-Itemset durch Mischen der (*k*-1)-Itemsets

Beispiel

- Gegeben: fünf 3-Itemsets

  (A B C), (A B D), (A C D), (A C E), (B C D)

- Lexikographisch geordnet!
- Kandidaten für 4-Itemsets:

  (A B C D) **OK wegen (B C D) (A C D)**

  (A C D E) **Nicht OK wegen (C D E)**

- Abschließender Test durch Auszählen der Instanzen auf der Datenmenge!
- (*k* –1)-Itemsets werden in Hashtabelle gespeichert

Algorithmus zur Gewinnung von Itemsets

Durchlaufe alle Datensätze und zähle alle 1-Itemsets aus

k=2

repeat

  für jeden noch möglichen k-Itemset (aus den k-1-Itemsets)

    teste, ob alle darin enthaltenen k-1-Itemsets vorhanden sind

    wenn ja, trage k-Itemset als Kandidaten ein

  Durchlaufe alle Datensätze und zähle die k-Itemsets aus

  Lösche alle Itemsets mit unzureichender Unterstützung

  k=k+1

until k > Anzahl Attribute

Effiziente Generierung von Regeln

- Suche nach Regeln mit hoher Konfidenz
  - Unterstützung eines Antezendenten kann aus der Hash-Tabelle entnommen werden
  - Aber: Aufwand für Brute-force-Methode ist (2^N-1)
- Besserer Weg: Ableitung der Regeln mit (c+1) Konsequenzen aus denen mit c *Konsequenzen*
  - Beobachtung: Regel mit (c + 1) Konsequenzen kann nur gelten, wenn alle darin enthaltenen Regeln mit *c Konsequenzen ebenfalls gelten*
- Resultierender Algorithmus ist ähnlich zum Algorithmus für große Itemsets

Beispiel

- Regeln mit 1 Konsequenz

```
If Outlook = Sunny and Windy = False and Play = No
   then Humidity = High (2/2)
```

```
If Humidity = High and Windy = False and Play = No
   then Outlook = Sunny (2/2)
```

- Zugehörige Regel mit 2 Konsequenzen:

```
If Windy = False and Play = No
   then Outlook = Sunny and Humidity = High (2/2)
```

- Abschließender Check des Antezendenten gegen die Hash-Tabelle!

Algorithmus zur Regelgewinnung

c=1

Für jeden k-Itemset

  für jede daraus mögliche Regel mit c Konsequenzen

  bestimme Hfk. des Antezendenten (aus Hashtabelle)

  wenn minimale Konfidenz erreicht wird, füge Regel ein

repeat c=c+1

  Aus den Regeln mit c-1 Konsequenzen bilde alle möglichen Regeln mit c Konsequenzen

  Für jede Regel mit c Konsequenzen

    Teste, ob alle darin enthaltenen Regeln mit c-1 Konsequenzen vorhanden;

    wenn nein, lösche die Regel

    Bestimme Hfk. des Antezendenten (aus Hashtabelle), berechne Konfidenz

    lösche die Regel, falls minimale Konfidenz nicht erreicht wird.

until c=k

Assoziationsregeln: Diskussion

- Vorstehende Methode durchläuft den Datensatz für jede Größe der Itemsets genau einmal
  - Alternative: Generiere (k+2)-Itemsets direkt nach der Generierung der (k+1)-Itemsets
  - Ergebnis: Es werden deutlich mehr (k+2)-Itemsets als notwendig generiert, aber man benötigt weniger Durchläufe durch die Daten
  - Sinnvolle Methode, wenn die Daten alle in den Hauptspeicher passen
- Praktischer Aspekt: Generierung einer bestimmten Anzahl von Regeln (z.B. durch inkrementelles Reduzieren des Minimums für die Unterstützung)

Weitere Aspekte

- Standard-ARFF-Format sehr ineffizient für typische *Warenkorb-Daten*
  - Attribute repräsentieren Objekte in einem Warenkorb, aber die meisten Objekte fehlen üblicherweise
  - Methode zur Repräsentation spärlicher Daten benötigt
- Instanzen werden auch *Transaktionen* genannt
- Konfidenz ist nicht notwendigerweise das beste Maß
  - Beispiel: Milch kommt in fast jeder Supermarkt-Transaktion vor
  - Daher wurden andere Maße vorgeschlagen (z.B. lift)

Lineare Modelle

- Arbeiten auf die naheliegendste Weise mit nummerischen Attributen
- Standardtechnik zur nummerischen Vorhersage: lineare Regression
  - Ergebnis ist eine Linearkombination von Attributen

$$x = w_0 a_0 + w_1 a_1 + w_2 a_2 + \ldots + w_k a_k$$

- Gewichte werden aus den Trainingsdaten berechnet
- Vorhersagewert für die erste Trainingsinstanz $\mathbf{a}^{(1)}$

$$w_0 a_0^{(1)} + w_1 a_1^{(1)} + w_2 a_2^{(1)} + \ldots + w_k a_k^{(1)} = \sum_{j=0}^{k} w_j a_j^{(1)}$$

Minimierung des quadratischen Fehlers

- Wähle $k+1$ Koeffizienten, um den quadratischen Fehler auf den Trainingsdaten zu minimieren
- Quadratischer Fehler: $\sum_{i=1}^{n}\left(x^{(i)}-\sum_{j=0}^{k} w_j a_j^{(i)}\right)^2$
- Berechnung der Koeffizienten durch Standard-Matrixoperationen
- Anwendbar, wenn es mehr Instanzen als Attribute gibt (grob gesprochen)
- Minimierung des *absoluten Fehlers* ist schwieriger

Klassifikation

- *Jedes* Regressionsverfahren kann zur Klassifikation verwendet werden
  - Training: führe Regression für jede Klasse durch: setze Ausgabewert=1 für Trainingsinstanzen der Klasse, und =0 für die anderen
  - Vorhersage: wähle die Klasse des Modells mit dem größten Ausgabewert (*Grad der Zugehörigkeit*)
- Bei linearer Regression wird dieses Verfahren als *multi-response linear regression* bezeichnet

Theoretische Rechtfertigung

Beobachteter Zielwert (0 oder 1)

Modell

Instanz

Verfahren minimiert diesen Wert

$$E_y\{(f(X)-Y)^2|X=x\}$$

Tatsächliche Klassenwahrscheinlichkeit

Erweitern:

$$=E_y\{(f(X)-P(Y=1|X=x)+P(Y=1|X=x)-Y)^2|X=x\}$$

Ausmultiplizieren:

$$(f(x)-P(Y=1|X=x))^2+2\times(f(x)-P(Y=1|X=x))\times$$
$$E_y\{P(Y=1|X=x)-Y|X=x\}+E_y\{(P(Y=1|X=x)-Y)^2|X=x\}$$

$$(f(x)-P(Y=1|X=x))^2+2\times(f(x)-P(Y=1|X=x))\times$$
$$(P(Y=1|X=x)-E_y\{Y|X=x\})+E_y\{(P(Y=1|X=x)-Y)^2|X=x\}$$

$$=(f(x)-P(Y=1|X=x))^2+E_y\{(P(Y=1|X=x)-Y)^2|X=x\}$$

Zu minimierender Wert

Konstante

Paarweise Regression

- Alternative Methode, um Regression zur Klassifikation anzuwenden:
  - Eine Regressionsfunktion für jedes *Paar* von Klassen, wobei nur Instanzen dieser beiden Klassen betrachtet werden
  - Setze Ausgabe= +1 für ein Element der ersten Klasse, –1 für die andere
- Vorhersage geschieht durch “abstimmen” (voting)
  - Klasse mit den meisten Stimmen wird vorhergesagt
  - Alternative: “don’t know”, wenn es keine klare Mehrheit gibt
- Meist höhere Präzision, aber aufwändiger

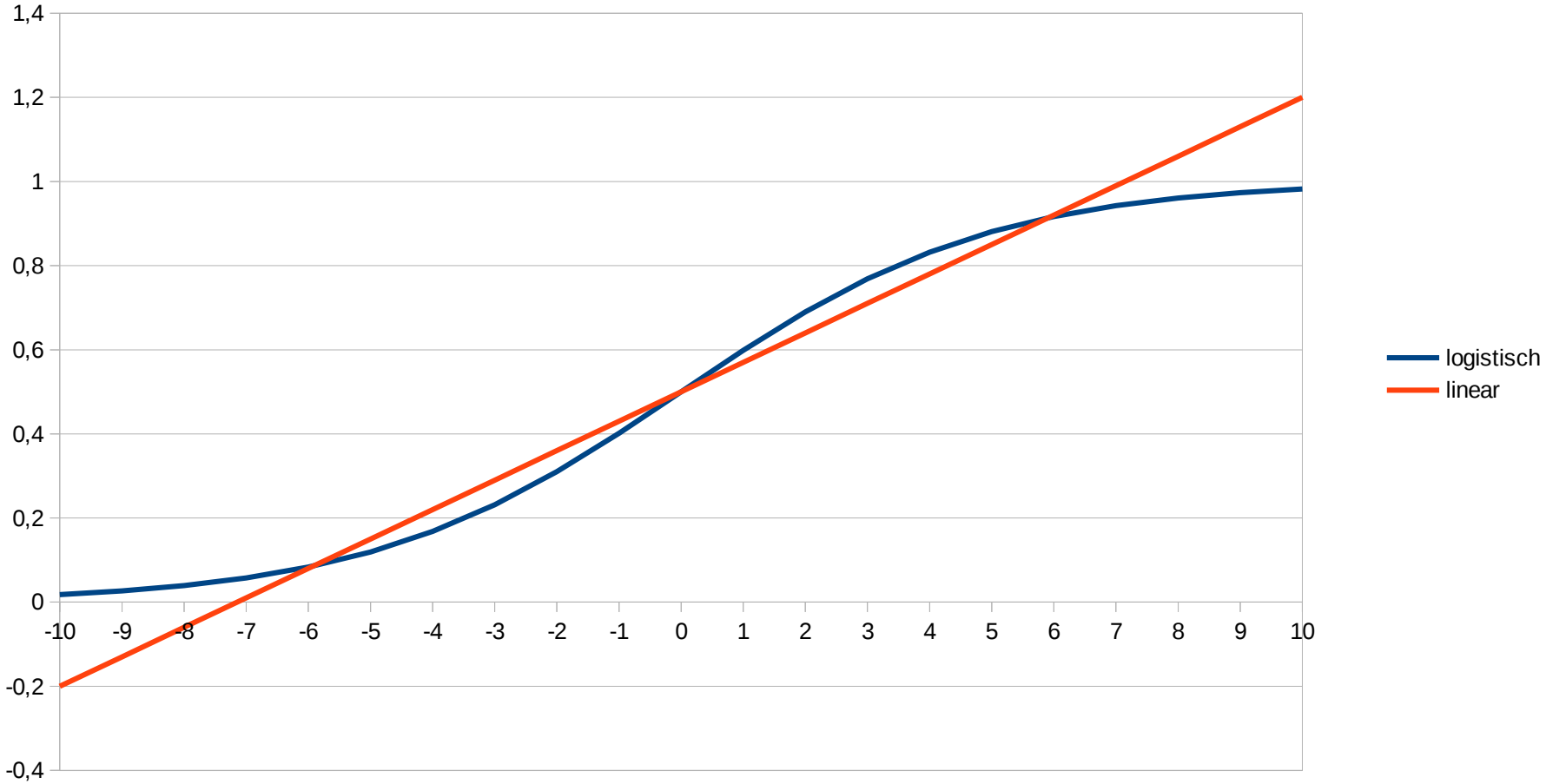
Logistische Regression

- Problem: Einige Annahmen werden verletzt, wenn lineare Regression zur Klassifikation verwendet wird
- *Logistische* Regression: Alternative zu linearer Regression
  - Entwickelt für Klassifikationsprobleme
  - Versucht, die Klassenwahrscheinlichkeit direkt zu schätzen
    - Unter Verwendung der *maximum likelihood* Methode
  - Zugrundeliegendes lineares Modell:

$$\log\left(\frac{P}{1-P}\right) = w_0 a_0 + w_1 a_1 + w_2 a_2 + \ldots + w_k a_k$$

Klassenwahrscheinlichkeit

Lineare vs. logistische Regression



Diskussion linearer Modelle

- Nicht anwendbar, wenn nichtlineare Zusammenhänge in den Daten existieren
- Aber: kann als Teil komplexerer Verfahren eingesetzt werden (z.B. Modellbäume)
- Beispiel: multi-response linear regression definiert eine *Hyperebene* für zwei beliebige Klassen:

$$(w_0^{(1)} - w_0^{(2)})a_0 + (w_1^{(1)} - w_1^{(2)})a_1 + (w_2^{(1)} - w_2^{(2)})a_2 + \ldots + (w_k^{(1)} - w_k^{(2)})a_k > 0$$

Instanzbasierte Repräsentation

- Einfachste Form des Lernens: *Auswendiglernen*
  - Zu einer neuen Instanz werden die dazu ähnlichsten in den Trainingsinstanzen gesucht
  - Die Instanzen selbst repräsentieren das Wissen
  - Wird auch *instanzbasiertes* Lernen genannt
- Ähnlichkeitsfunktion repräsentiert das "Gelernte""
- Instanzbasiertes Lernen ist *faules* Lernen
- Methoden:
  - *Nächster Nachbar*
  - *k-nächster Nachbar*
  - *...*

Die Distanzfunktion

- Einfachster Fall: ein nummerisches Attribut
  - Distanz ist die Differenz zwischen den zwei beteiligten Attributwerten (oder eine Funktion hiervon)
- Mehrere nummerische Attribute: Meist Verwendung der Euklidischen Distanz, wobei Attribute normalisiert werden
- Nominale Attribute: Distanz=1, falls Werte verschieden, =0 bei Gleichheit
- Sind alle Attribute gleich wichtig?
  - Gewichtung der Attribute evtl. notwendig

Instanzbasiertes Lernen

- Distanzfunktion repräsentiert das Gelernte
- Die meisten Verfahren benutzen die *euklidische Distanz*:

$$\sqrt{(a_1^{(1)}-a_1^{(2)})^2+(a_2^{(1)}-a_2^{(2)})^2+\ldots+(a_k^{(1)}-a_k^{(2)})^2}$$

  $\mathbf{a}^{(1)}$ und $\mathbf{a}^{(2)}$: zwei Instanzen mit k Attributen
- Quadratwurzel nicht erforderlich, wenn nur Distanzen verglichen werden
- Andere populäre Metrik: *Manhattan-Distanz*
  - Addiert die Differenzen, ohne sie zu quadrieren

Normalisierung und andere Aspekte

- Verschiedene Attribute werden mit verschiedenen Skalen gemessen ⇒ Notwendigkeit der *Normalisierung*:

$$a_i = \frac{v_i - \min v_i}{\max v_i - \min v_i}$$

  v_i : Wert für das Attribut i

- Nominale Attribute: Distanz ist entweder 0 oder 1
- Vorgehensweise bei fehlenden Werten: werden als maximal entfernt angenommen (bei normalisierten Attributen)

Diskussion von 1-NN

- Oft sehr genau
- … aber langsam:
  - Einfache Versionen scannen die gesamten Trainingsdaten für eine Vorhersage
- Annahme, dass alle Attribute gleich wichtig sind
  - Abhilfe: Attributselektion oder Gewichtung
- Mögliche Abhilfe bei verrauschten Instanzen:
  - Mehrheitsvotum über die *k* nächsten Nachbarn
  - Eliminierung verrauschter Instanzen (schwierig!)
- Statistiker benutzen *k*-NN seit 50 Jahren
  - Für $n \to \infty$ und $k/n \to 0$ wird der Fehler minimal

Anmerkungen zu den grundlegenden Methoden

- Bayes' Regel stammt aus seinem Aufsatz "Essay towards solving a problem in the doctrine of chances" (1763)
  - Hauptproblem: Schätzung der apriori-Wahrscheinlichkeiten
- Erweiterung von Naivem Bayes: Bayes'sche Netzwerke
- Algorithmus für die Assoziationsregeln: APRIORI
- Minsky und Papert (1969) beschreiben die Beschränkungen linearer Klassifikatoren (können z. B. kein EXOR lernen)
  - Aber: Kombinationen davon können (→ Neuronale Netze)