Aus Beobachtungen Lernen

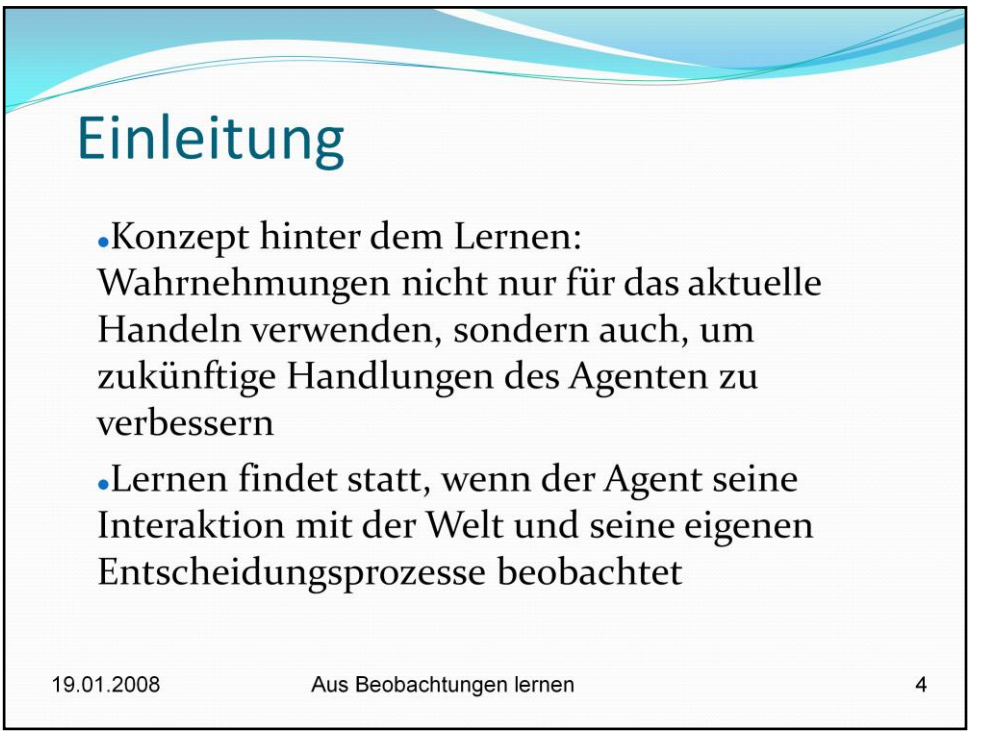
Tanja Neubacher 11042463

Inhaltsübersicht

- Lernformen
- Induktives Lernen
- Lernentscheidungsbäume
- Entscheidungsbäume als Leistungselemente
- Ausdruckskraft von Entscheidungsbäumen
- Entscheidungsbäume per Induktion aus Beispielen ableiten

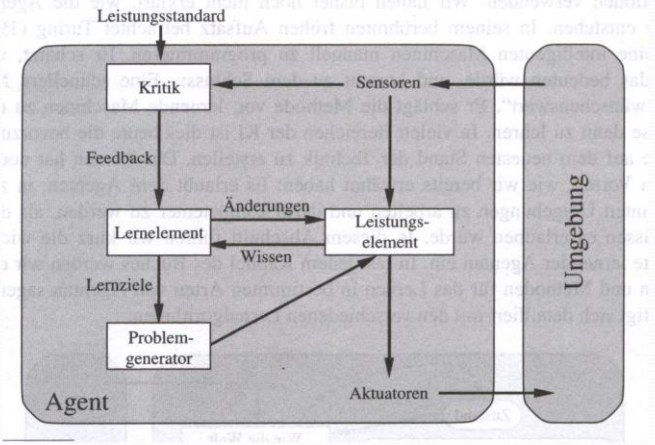
Inhaltsübersicht II

- Auswahl von Attributtests
- Leistungsabschätzung des Lernalgorithmus
- Überanpassung
- Gruppenlernen
- Warum das Lernen funktioniert: Computer-Lerntheorie
- Fazit



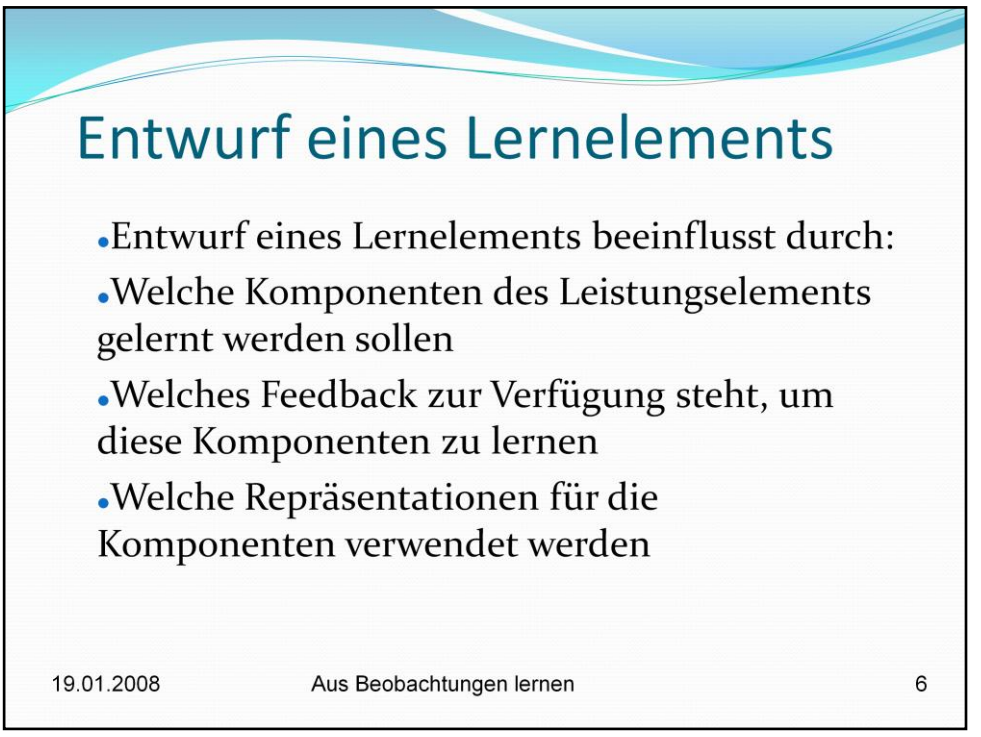
- Lernen kann von einem einfachen Erinnern der Erfahrungen, wie wir es beim Agenten der Wumpus-Welt gesehen haben (Kap. 10),
- bis hin zur Erstellung ganzer wissenschaftlicher Theorien reichen (z.B. Albert Einstein)
- Konzept des Kapitels:
  - Induktives Lernen (aus Beobachtungen)
  - Beschreibung wie einfache Theorien aus der Aussagenlogik gelernt werden

Lernender Agent

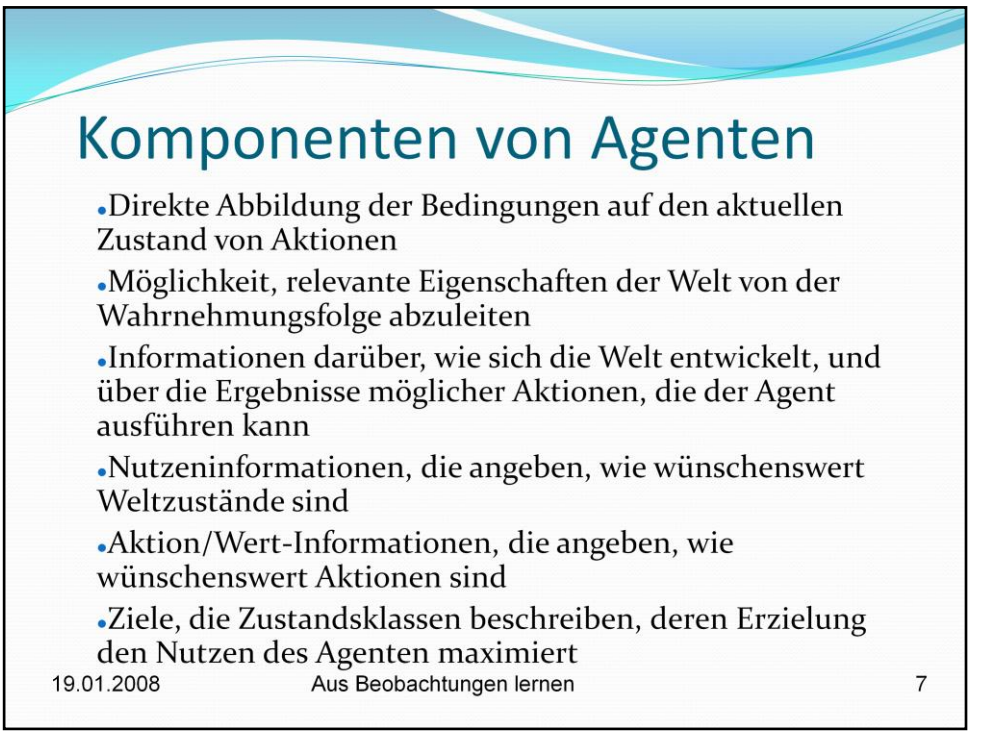


19.01.2008 Aus Beobachtungen lernen 5

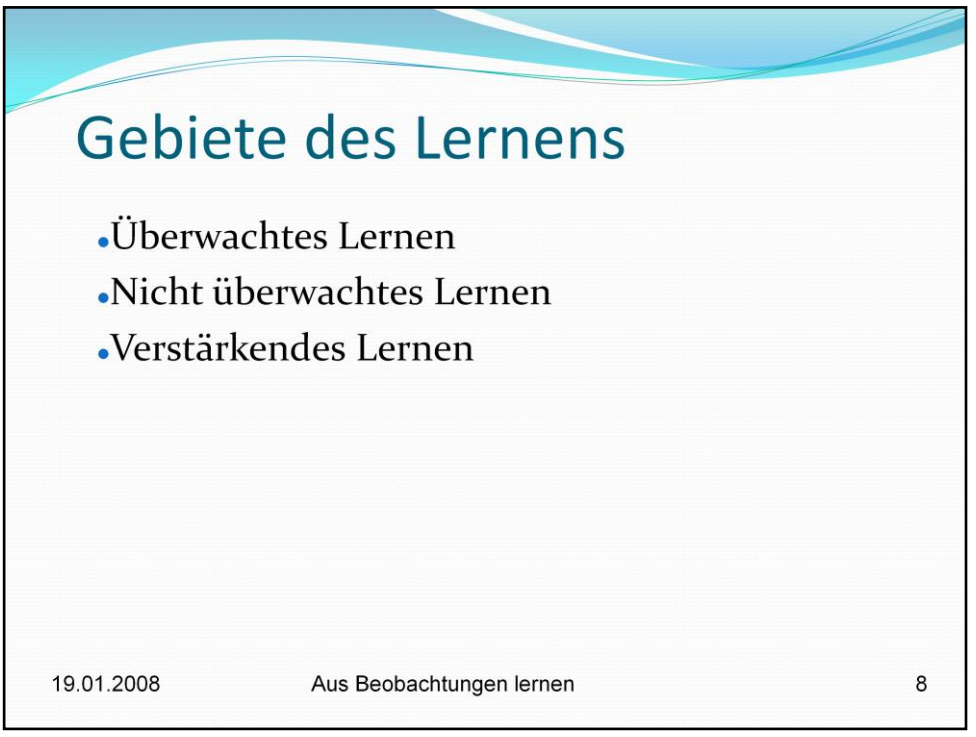
- Hier Bild aus Kapitel 2 einfügen (Lernender Agent)
  - Leistungselement
    - entscheidet welche Aktionen ausgeführt werden sollen
  - Lernelement
    - Passt Leistungselement an, so dass es bessere Entscheidungen trifft



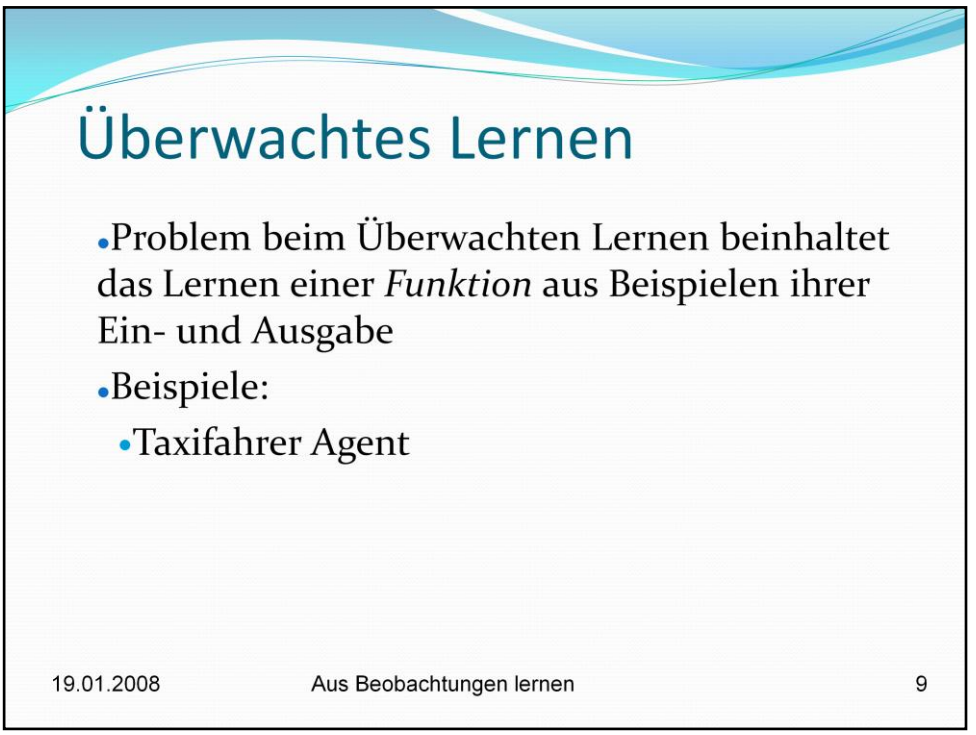
- Forscher im Bereich des maschinellen Lernens haben eine Vielzahl von Lernelementen entwickelt
- Um sie zu verstehen, betrachten, wie ihr Entwurf durch den Kontext beeinflusst wird, in dem Sie arbeiten
- Entwurf eines Lernelements wird hauptsächlich durch drei Aspekte beeinflusst: (s.o.)
- Diese Aspekte werden gleich näher analysiert



- Viele Möglichkeiten das Leistungselement eines Agenten aufzubauen
- Jede Komponent kann aus geeignetem Feedback gelernt werden
- Beispielsweise Agent der Taxifahrer lernen möchte
- Immer wenn der Lehrer bremsen ruft kann der Agent eine Bedingung/Aktion-Regel lernen, wann er bremsen soll (Komponente 1)
- Durch Betrachtung verschiedener Kamerabilder, auf denen Busse zu sehen sind, lernt er Busse zu erkennen (Komponente 2)
- usw.

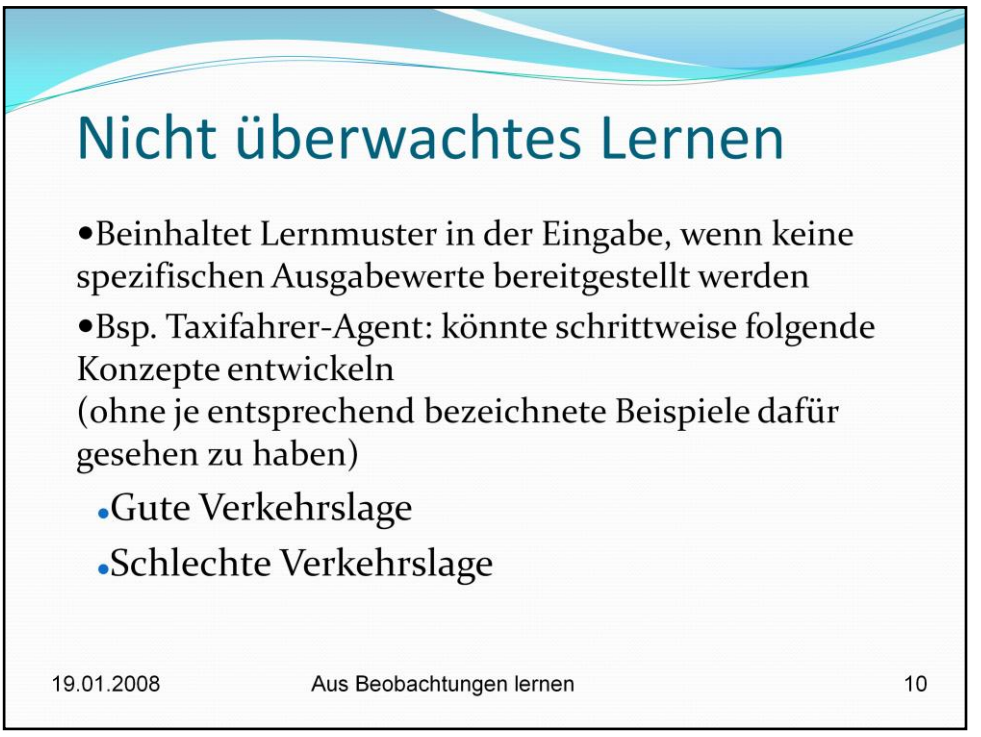


Verfügbare Feedbacktyp ist im Allgemeinen der wichtigste Faktor bei der Entscheidung, welchem Lernproblem der Agent gegenübersteht
Gebiet des Maschinenlernens unterscheidet 3 Fälle (s.o.)

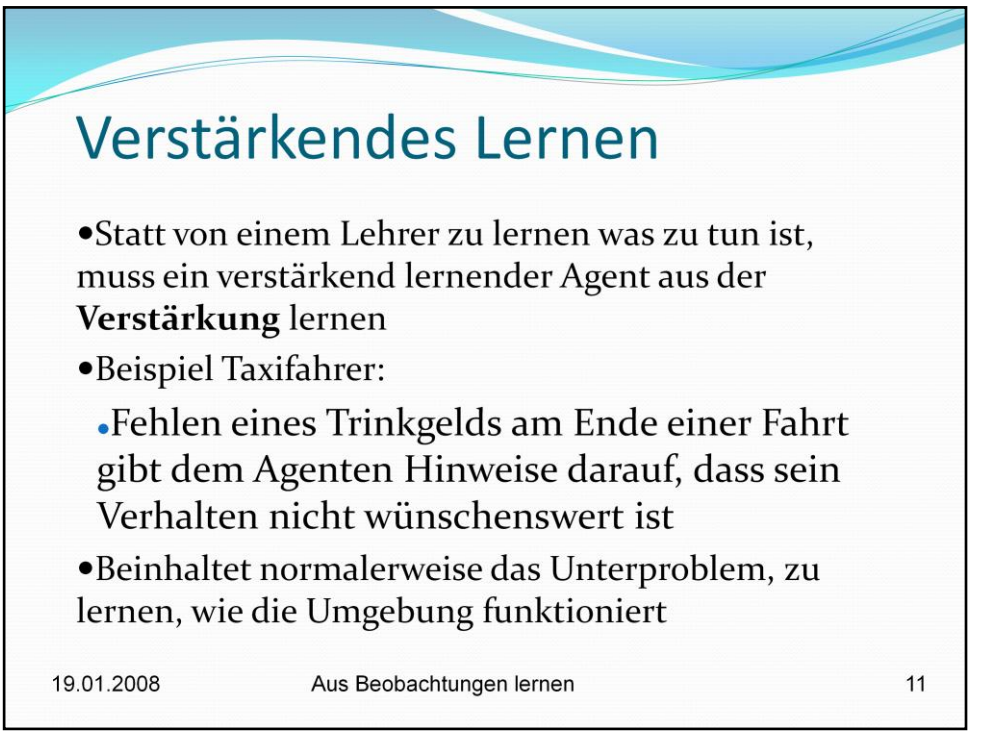


Beispiele für Überwachtes Lernen:

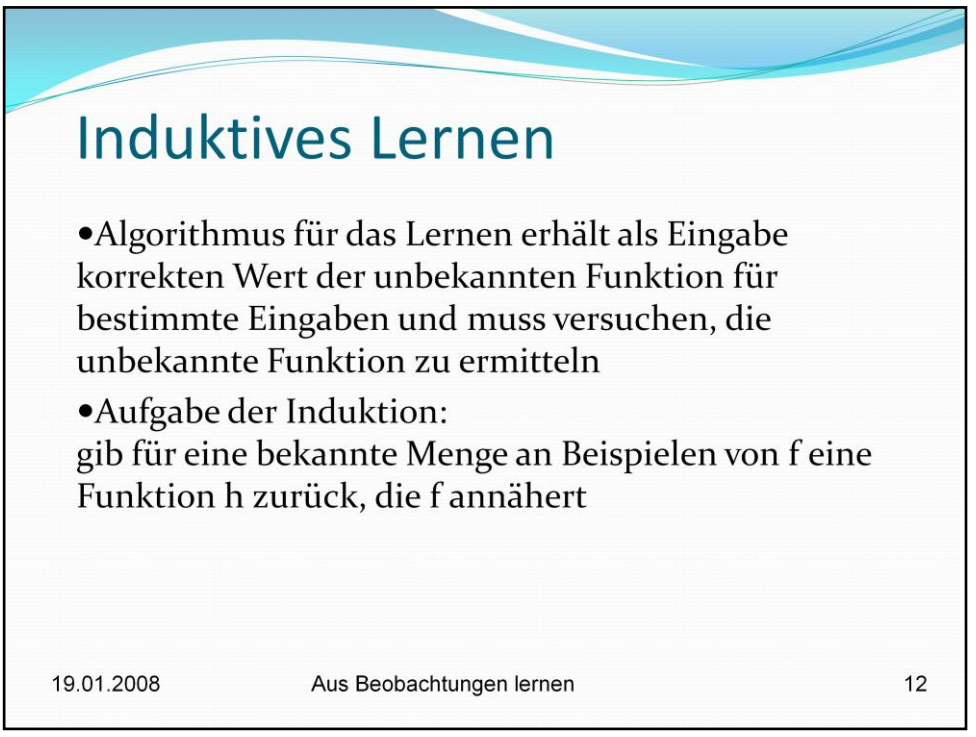
- Taxifahrer-Agent:
- In (1) lernt der Agent die Bedingung/Aktion-Regeln für das Bremsen – das ist die Funktion von Zuständen zu einer Boolschen Ausgabe (bremsen/nicht-bremsen)
- In (2) lernt der Agent eine Funktion von Bildern zu einer Boolschen Ausgabe (ob Bild einen Bus enthält)
- In (3) ist die Bremstheorie eine Funktion von Zuständen und Bremsaktionen zu beispielsweise dem Halteweg in Metern
- Beachte: In den Fällen (1) und (2) stellt ein Lehrer die korrekten Ausgabewerte bereit
- in (3) hingegen stand der Ausgabewert direkt aus den Wahrnehmungen des Agenten zur Verfügung
- Für vollständig beobachtbare Umgebungen ist es immer der Fall, das ein Agent die Wirkung seiner Aktionen beobachten und damit überwachte Lernmethoden nutzen kann, um sie vorherzusagen
- Für partiell beobachtbare Umgebungen hingegen ist das Problem schwieriger, weil die unmittelbare Wirkung evtl. nicht sichtbar ist



- reiner nicht überwachter lernender Agent kann nicht lernen, was zu tun ist, weil er keine Informationen darüber hat, was eine korrekte Aktion oder ein wünschenswerter Zustand ist
- nicht überwachtes Lernen hauptsächlich im Kontext probalistischer Inferenzsysteme betrachten (s. Kap. 20)



Begriff **Gewinn**, wie er in Kp 17 verwendet wird, ist ein Synonym für **Verstärkung**



•Algorithmus für deterministisches überwachtes Lernen erhält als Eingabe den korrekten Wert der unbekannten Funktion für bestimmte Eingaben und muss versuchen, die unbekannte Funktion oder etwas hinreichend Ähnliches zu ermitteln

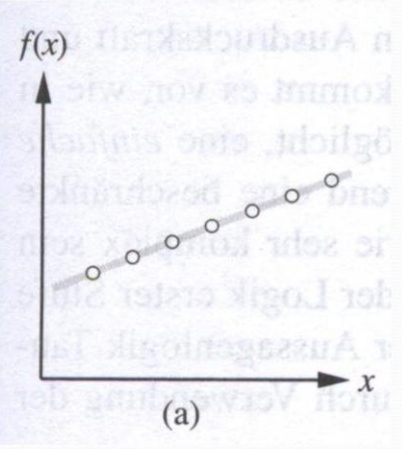
•Aufgabe der **reinen induktiven Inferenz** ist: gib für eine bekannte Menge an Beispielen von f eine Funktion h zurück, die f annähert

Induktionsproblem

- Funktion h wird als **Hypothese** bezeichnet
- Grund warum Lernen aus konzeptueller Perspektive so schwierig ist man kann nicht so einfach sagen, ob ein bestimmtes h eine gute Annäherung von f ist
- -> grundlegendes **Induktionsproblem**
- Gute Hypothese **verallgemeinert** gut, d.h. sie sagt zuvor nicht bekannte Beispiele korrekt voraus

Ermittlung einer Funktion

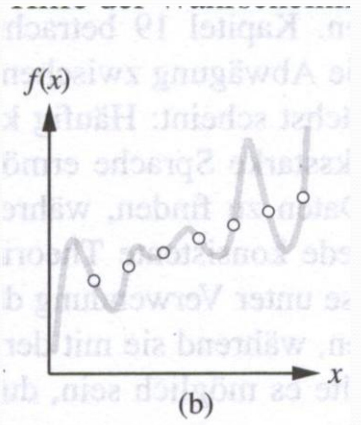
- zeigt einige Daten mit genauer Übereinstimmung durch eine gerade Linie (Polynom 1. Grads)
- Linie: **konsistente** Hypothese



- Beispiele sind (x, f(x))-Paare, wobei sowohl x als auch f(x) reelle Zahlen sind
- Zeigt ein bekanntes Beispiel: Ermittlung einer Funktion einer einzelnen Variablen für ein paar Datenpunkte
- **Hypothesenraum h** wählen (Menge der betrachteten Hypothesen) als Menge der Polynome mit maximalem Grad k (z.B. 3*x^3 + 2, x^17 – 4*x^3, usw.)
- Linie wird als **konsistente** Hypothese bezeichnet, weil sie mit allen Daten übereinstimmt

Polynom höheren Grades

- zeigt Polynom höheren Grades, das ebenfalls konsistent zu denselben Daten ist

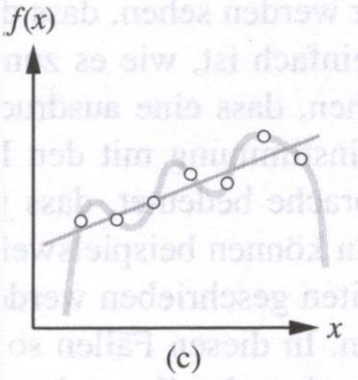


19.01.2008 Aus Beobachtungen lernen 15

- -> Erkennung des erstes Problems beim induktiven Lernen:
- Wie wählen wir aus mehreren konsistenten Hypothesen aus?
- Antwort ist das sog. **Ockham-Rasiermesser** Bevorzugung der *einfachsten* Hypothese, die konsistent mit den Daten ist
- (Intuitiv scheint dies sinnig zu sein, weil Hypothesen, die nicht einfacher als die eigentlichen Daten sind, keine *Muster* aus den Daten ableiten können)
- Definition der Einfachheit ist nicht einfach, aber es scheint sinnvoll zu sein, zu sagen, ein Polynom 1. Grades ist einfacher als ein Polynom 12. Grades

Keine konsistente gerade Linie

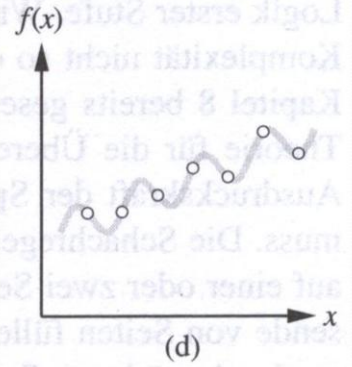
- Zeigt 2te Datenmenge gibt keine konsistente gerade Linie für diese Datenmenge
- Vielmehr braucht man ein Polynom 6. Grades (mit 7 Parametern) um eine genaue Übereinstimmung zu finden



- Es scheint also kein Muster in den Daten zu erkennen, und wir erwarten nicht, dass es gut verallgemeinert
- Es wäre besser eine einfache gerade Linie zu finden, die nicht genau konsistent ist, die aber vielleicht sinnvolle Vorhersagen erlaubt
- Damit akzeptieren wir letztlich, dass die tatsächliche Funktion nicht deterministisch ist (oder einfacher ausgedrückt, dass die tatsächliche Eingaben nicht vollständig beobachtbar sind)
- *Für nicht deterministische Funktionen gibt es eine unvermeidbare Abwägung zwischen der Komplexität der Hypothese und ihrem Grad der Übereinstimmung mit den Daten*
- Kap 20 erklärt, wie diese Abwägung mit Hilf der Wahrscheinlichkeitstheorie getroffen werden kann

Auswahl des Hypothesenraums

- Möglichkeit oder Unmöglichkeit eine einfache konsistente Hypothese zu finden ist sehr von dem gewählten Hypothesenraum abhängig



- Die Daten können genau durch eine einfache Funktion der Form ax + b + c * sin x abgedeckt werden
- Beispiel zeigt, welche Bedeutung die Auswahl des Hypothesenraums hat
- z.B. Hypothesenraum, der aus Polynomen endlichen Grads besteht, kann Sinusfunktionen nicht exakt repräsentieren
- deshalb ist ein Lerner, der diesen Hypothesenraum verwendet, nicht in der Lage, aus sinusförmigen Daten zu lernen

Lernprogramm

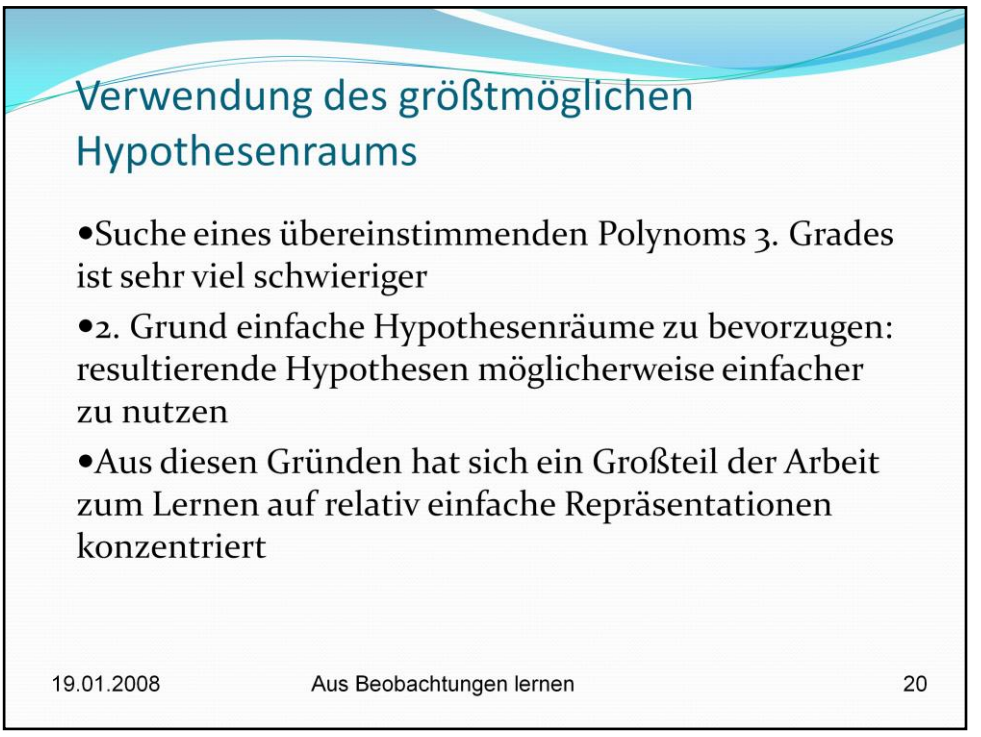
- Lernprogramm **erkennbar**, wenn Hypothesenraum richtige Funktion enthält, andernfalls ist es **nicht erkennbar**
- Leider nicht immer feststellbar, ob bestimmtes Lernproblem erkennbar, weil richtige Funktion nicht bekannt ist
- Umgehung der Grenze, indem man *Vorwissen* verwendet, um Hypothesenraum abzuleiten, von dem bekannt ist, dass richtige Funktion darin liegen muss

•(-> Kap. 19)

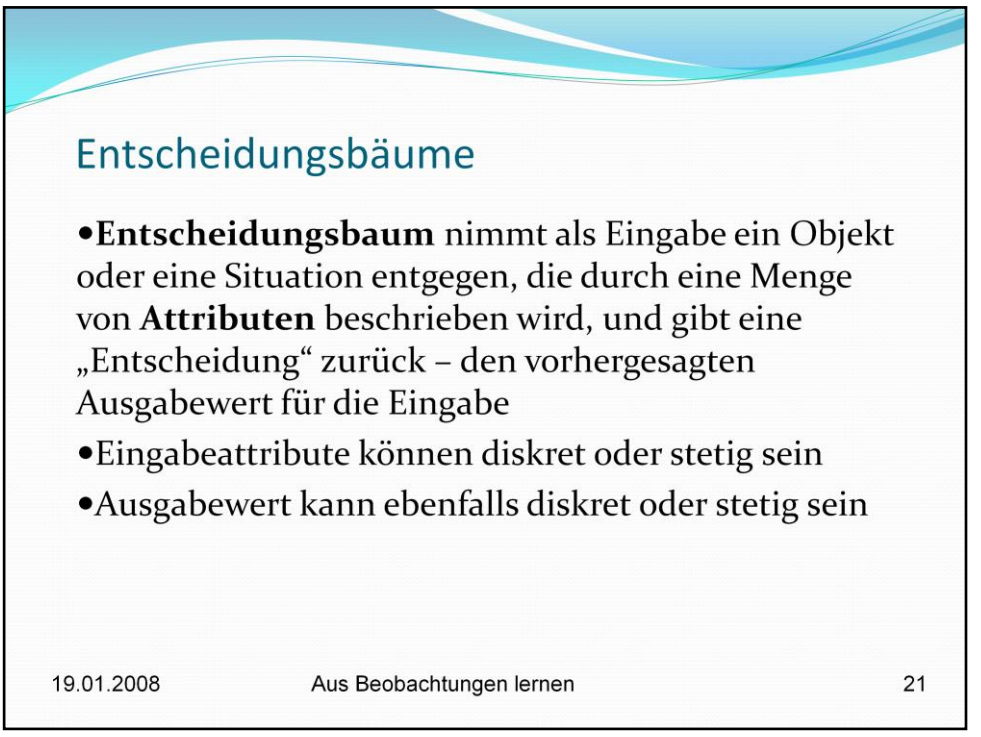
Verwendung des größtmöglichen Hypothesenraums

- Problem:
berücksichtigt nicht die *Rechenkomplexität des Lernens*
- *Abwägung zwischen der Ausdruckskraft eines Hypothesenraums und der Komplexität, einfache, konsistente Hypothesen in diesem Raum zu finden*
- Beispielsweise sehr einfach, gerade Linien zu finden, wenn sie mit den Daten übereinstimmen

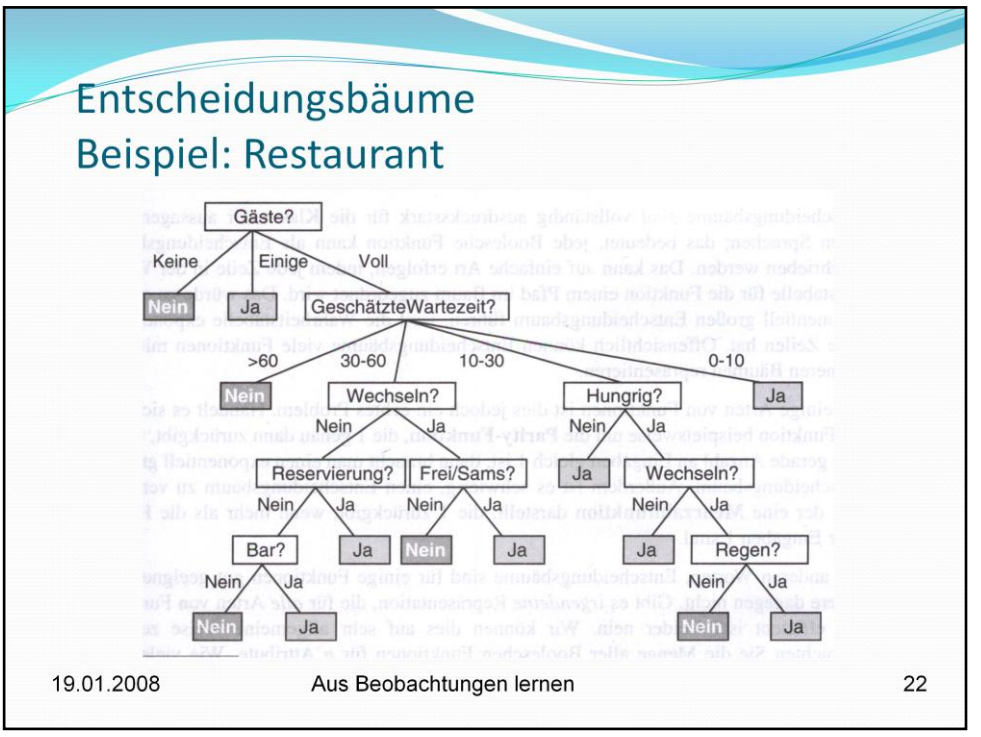
- z.B.: Klasse aller Turing-Maschinen
- schließlich kann jede berechenbare Funktion durch eine Turing-Maschine dargestellt werden, und das ist das Beste, was man tun könnte



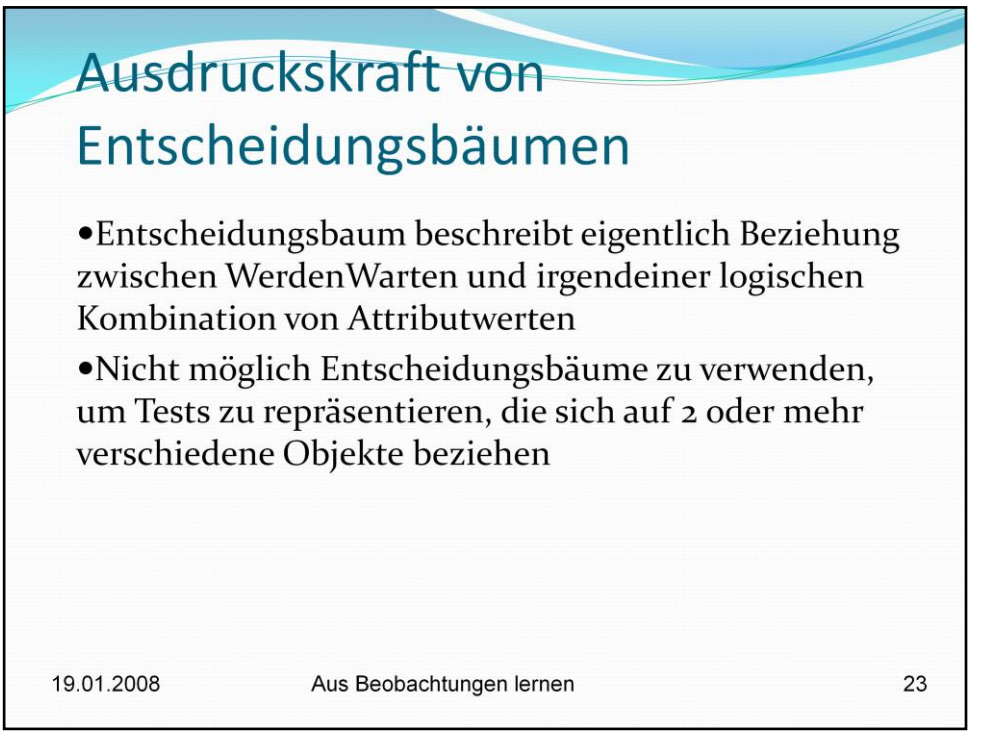
- Zu 2. – d.h., es ist schneller, h(x) zu berechnen, wenn h eine lineare Funktion ist, als wenn es sich dabei um eine beliebiges Turing-Maschinenprogramm handelt
- Suche nach übereinstimmenden Turing-Maschinen ist sehr schwer, weil die Feststellung, ob eine bestimmte Turing-Maschine konsistent mit den Daten ist, im Allgemeinen nicht einmal entscheidbar ist
- In diesem Kapitel hauptsächlich Beschäftigung mit Aussagenlogik und verwandten Sprachen
- In Kap 19 Betrachtung von Lerntheorien der Logik erster Stufe
- Abwägung zwischen Ausdruckskraft und Komplexität nicht so einfach ist, wie es zunächst scheint
- Häufig kommt es vor, wie in Kap. 8 bereits gezeigt, dass eine ausdrucksstarke Sprache ermöglicht eine einfache Theorie für die Übereinstimmung mit den Daten zu finden, während eine beschränkte Ausdruckskraft für die Sprache bedeutet, dass jede konsistente Theorie sehr komplex sein muss (Beispiel Schachregeln)



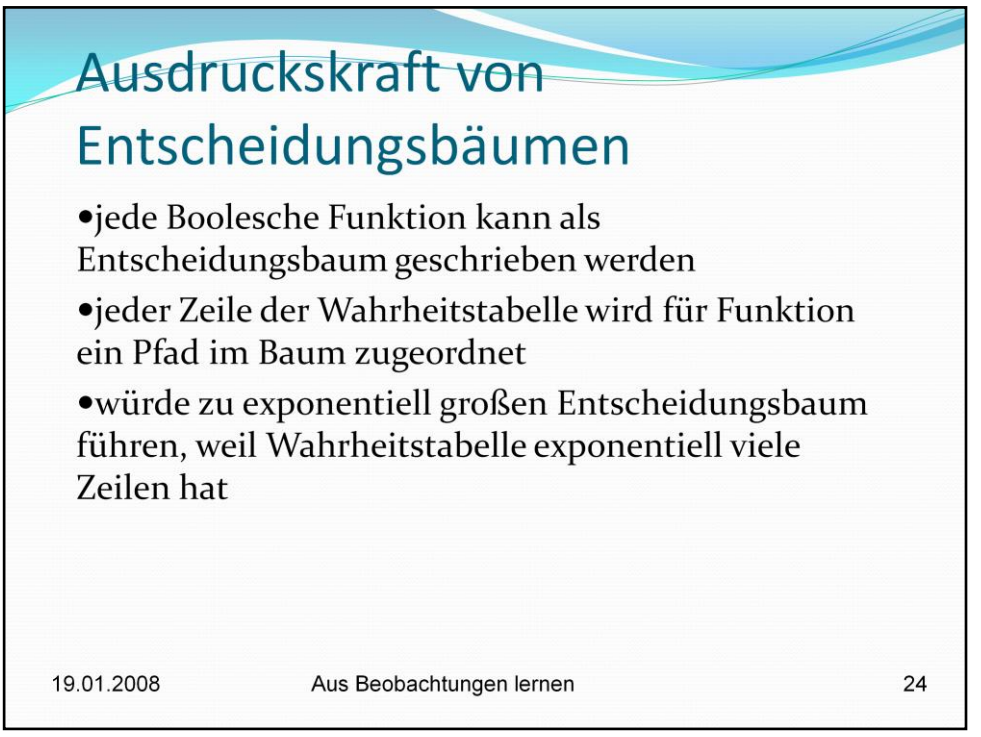
- Lernen einer diskretwertigen Funktion wird auch als **klassifizierendes** Lernen bezeichnet
- Lernen einer stetigen Funktion ist eine sog. **Regression**
- Konzentration auf boolesche Klassifizierungen, wobei jedes Beispiel als true (**positiv**) oder false (**negativ**) klassifiziert wird



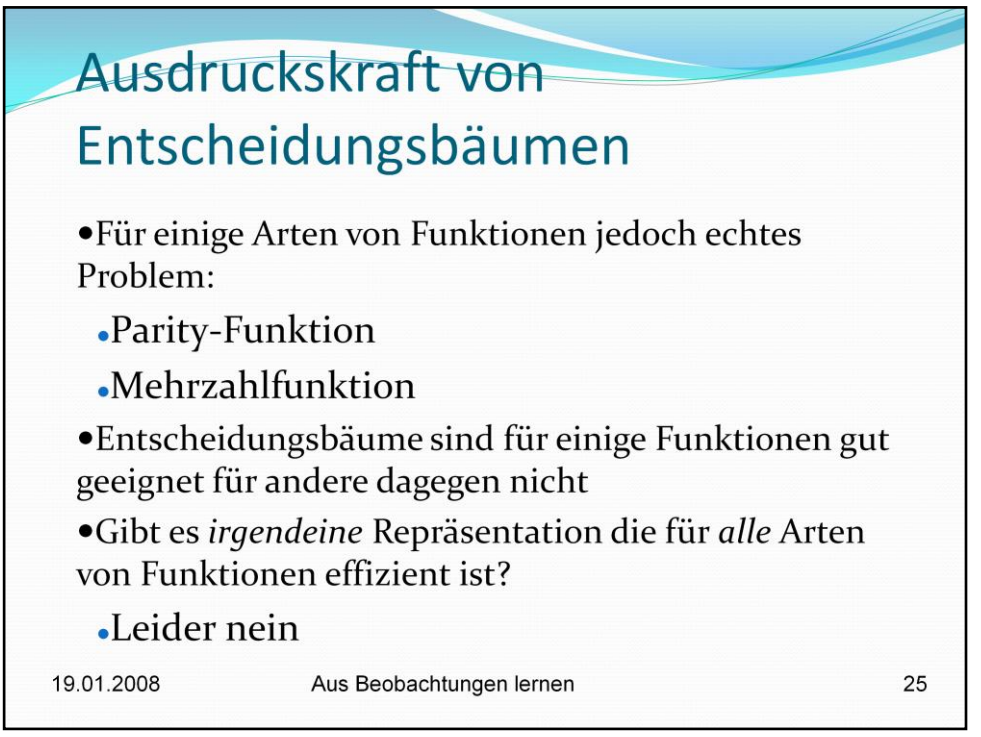
- Trifft eine Entscheidung, in dem er eine Reihe von Tests ausführt
- Jeder interne Knoten im Baum entspricht einem Test des Werts einer der Eigenschaften
- die Verzweigungen von dem Knoten aus werden mit den möglichen Werten des Tests beschriftet
- Attribute Preis und Typ nicht verwendet, da als irrelevant erachtet
- Beispiele von der Wurzel aus verarbeitet, wobei die zutreffenden Verzweigungen verfolgt werden, bis ein Blattknoten erreicht ist
- Problem ob man auf einen Tisch in einem Restaurant warten soll
- Hier ist das Ziel, eine Definition für das **Zielprädikat** *WerdenWarten* zu lernen
- Für Einrichtung als Lernproblem zuerst Festlegung welche Attribute zur Verfügung stehen, um Beispiele in der Domäne zu beschreiben
- -> in Kap. 19 wie diese Aufgabe automatisiert werden kann



- Jede Bedingung $P_i(s)$ eine Konjunktion der Tests entsprechend einem Pfad von der Wurzel des Baumes zu einem Blattknoten mit positiven Ergebnis
- Logisch ausgedrückt, kann eine beliebige Entscheidungsbaum-Hypothese für das Zielprädikat *WerdenWarten* als Zusicherung der unten dargestellten Form betrachtet werden
- Vs WerdenWarten(s) <=> ($P_1(s)$ v $P_2(s)$ v ... v $P_n(s)$)
- Obwohl es wie ein Satz aus der Logik erster Stufe ausschaut ist es in gewisser Weise aussagenlogisch, weil es nur eine einzige Variable enthält und alle Prädikate unär sind
- Entscheidungsbaum beschreibt eigentlich die Beziehung zwischen *WerdenWarten* und irgendeiner logischen Kombination von Attributwerten
- Er_2 Nähe(r_2,r) ^ Preis(r,p) ^ Preis(r_2,p_2) ^ Billiger(p_2,p)
- z.B. Gibt es ein billigeres Restaurant in der Nähe?
- Könnten ein weiteres boolesches Attribut mit Namen *BilligeresRestaurantInDerNähe* einführen, abes es ist unmöglich, *alle* diese Attribute zu handhaben
- -> Kap 19 beschäftigt sich eingehender mit dem Problem, in der Logik erster Stufe korrekt zu lernen



- Zu 1. Entscheidungsbäume sind vollständig ausdrucksstark für Klasse der aussagenlogischen Sprachen, ...
- Offensichtlich können Entscheidungsbäume viele Funktionen mit viel kleineren Bäumen repräsentieren



- Handelt es sich bei der Funktion beispielsweise um die **Parity-Funktion** die 1 genau dann zurückgibt wenn eine gerade Anzahl an Eingaben gleich 1 ist, dann braucht man einen exponentiell großen Entscheidungsbaum
- Außerdem ist es schwierig einen Entscheidungsbaum zu verwenden der eine **Mehrzahlfunktion** darstellt die 1 zurückgibt wenn mehr als die Hälfte ihrer Eingaben 1 sind

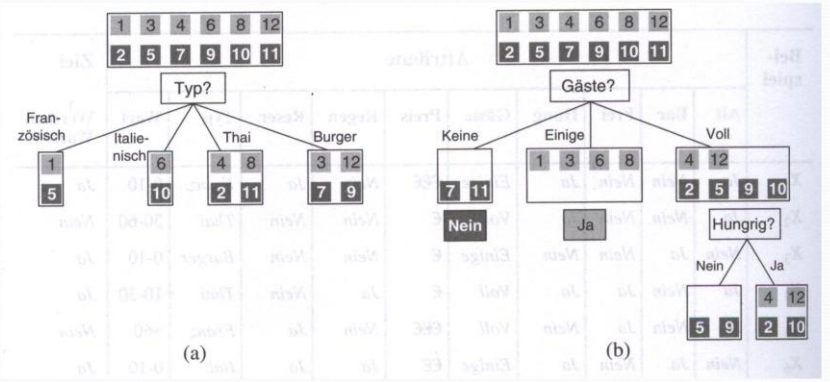
•Menge aller Booleschen Funktionen für *n* Attribute
•Wie viele versch. Funktionen befinden sich in dieser Menge? Genau die Anzahl der versch. Wahrheitstabellen, die man aufschreiben kann. Weil die Funktion durch ihre Wahrheitstabelle definiert ist
•Wahrheitstabelle hat 2^n Zeilen, weil jeder Eingabefall durch *n* Attribute beschrieben
•Antwort-Spalte der Vorbedingung als 2^n-Zahl betrachten, die die Funktion definiert
•Egal welche Repräsentation verwendet wird, brauchen einige Funktionen (fast alle) mindestens so viele Bits für die Repräsentation
•Wenn 2n Bits benötigt um Funktion zu definieren, gibt es für *n* Attribute 2^{2^n} versch. Funktionen
•Bei nur 6 versch Attributen gibt es also 2^{2^6}

Beispiel für booleschen Entscheidungsbaum

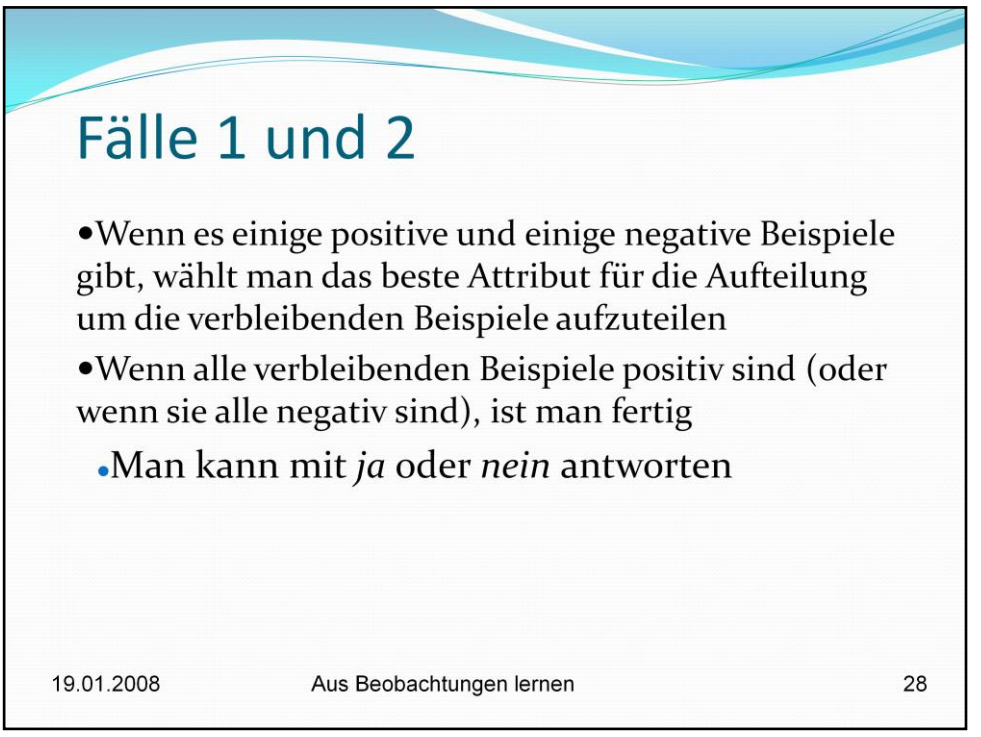
Beispiel	Attribute										Ziel
	Alt	Bar	Frei	Hung	Gäste	Preis	Regen	Reser.	Typ	Wart	Werden-Warten
X_1	*Ja*	*Nein*	*Nein*	*Ja*	*Einige*	€€€	*Nein*	*Ja*	*Franz.*	0-10	*Ja*
X_2	*Ja*	*Nein*	*Nein*	*Ja*	*Voll*	€	*Nein*	*Nein*	*Thai*	30-60	*Nein*
X_3	*Nein*	*Ja*	*Nein*	*Nein*	*Einige*	€	*Nein*	*Nein*	*Burger*	0-10	*Ja*
X_4	*Ja*	*Nein*	*Ja*	*Ja*	*Voll*	€	*Ja*	*Nein*	*Thai*	10-30	*Ja*
X_5	*Ja*	*Nein*	*Ja*	*Nein*	*Voll*	€€€	*Nein*	*Ja*	*Franz.*	>60	*Nein*
X_6	*Nein*	*Ja*	*Nein*	*Ja*	*Einige*	€€	*Ja*	*Ja*	*Ital.*	0-10	*Ja*
X_7	*Nein*	*Ja*	*Nein*	*Nein*	*Keine*	€	*Ja*	*Nein*	*Burger*	0-10	*Nein*
X_8	*Nein*	*Nein*	*Nein*	*Ja*	*Einige*	€€	*Ja*	*Ja*	*Thai*	0-10	*Ja*
X_9	*Nein*	*Ja*	*Ja*	*Nein*	*Voll*	€	*Ja*	*Nein*	*Burger*	>60	*Nein*
X_{10}	*Ja*	*Ja*	*Ja*	*Ja*	*Voll*	€€€	*Nein*	*Ja*	*Ital.*	10-30	*Nein*
X_{11}	*Nein*	*Nein*	*Nein*	*Nein*	*Keine*	€	*Nein*	*Nein*	*Thai*	0-10	*Nein*
X_{12}	*Ja*	*Ja*	*Ja*	*Ja*	*Voll*	€	*Nein*	*Nein*	*Burger*	30-60	*Ja*

- Beispiel für einen Entscheidungsbaum besteht aus einem Vektor mit Eingabeattributen, X, und einem einzigen Booleschen Ausgabewert , y
- Positiven Beispiele sind diejenigen, wo das Ziel *WerdenWarten* true ist
- Negative Beispiele sind diejenigen, wo das Ziel *WerdenWarten* false ist
- Vollständige Beispielmenge wird auch als **Trainingsmenge** bezeichnet

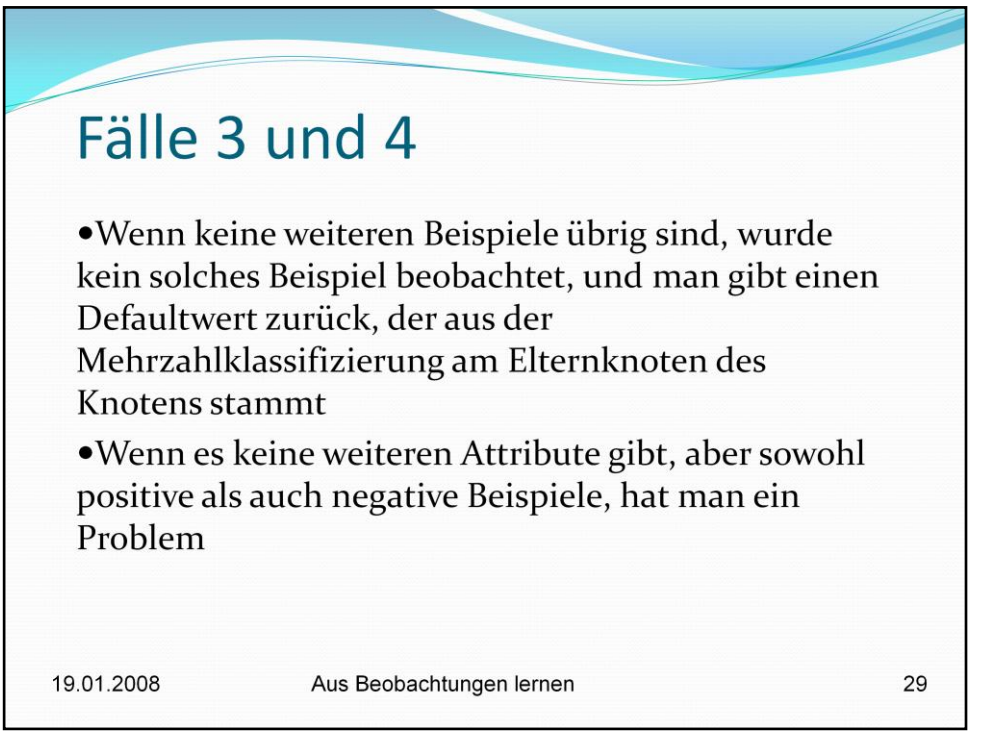
Aufteilung der Beispiele durch Testen von Attributen



- Allg geht es darum, einen kleinen (nicht zwangsläufig den kleinsten) Entscheidungsbaum zu finden
- Deshalb versuch Attribute nach Wichtigkeit zu wählen (das in der Klassifizierung eines Beispiels den größten Unterschied verursacht)
- Aufteilung nach dem Typ bringt einen näher zur Unterscheidung zwischen positiven und negativen Beispielen
- Aufteilung nach Gästen z.B. ist eine sinnvolle Maßnahme zur Unterscheidung zwischen positiven und negativen Beispielen
- Nach der Aufteilung nach Gästen ist Hungrig ein recht sinnvoller zweiter Test
- Zeigt wie Algorithmus gestartet wird
- 12 Trainingsbeispiele, die in positive und negative Mengen unterteilt werden
- Anschließend entscheiden welche Attribute zuerst im Baum ausgewertet werden
- 18.4 a schlecht, da Typ 4 mögliche Ergebnisse bringt, die wiederum dieselbe
- Anzahl positiver und negativer Beispiele haben
- 18.4 b hingegen gut, da Gäste ein wichtiges Attribut ist, denn wenn sein Wert gleich *Keine* oder *Einige* ist dann erhalten wir Beispielmengen für die man definitiv antworten kann Nein bzw. Ja
- Ist der Wert gleich *Voll* erhält man eine gemischte Menge an Beispielen



- Zu 1. (s. *Hungrig* Abb. 18.4 b)
- Zu 3. (s. Abb. 18.4 b, Fall *Keine* und *Einige*)
- Im Allgemeinen ist nach der ersten Aufteilung der Beispiele mit dem ersten Attributtest jedes Ergebnis ein neues Entscheidungsbaum-Lernproblem – mit weniger Beispielen und einem Attribut weniger
- Für diese rekursiven Probleme müssen 4 Fälle berücksichtigt werden:

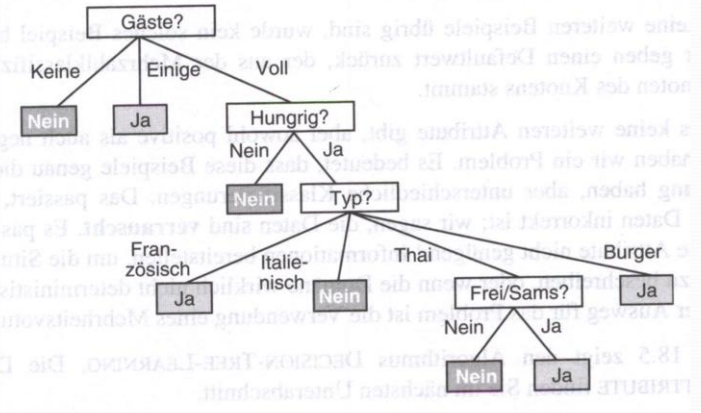


- Es bedeutet, dass diese Beispiele genau dieselbe Beschreibung haben, aber unterschiedliche Klassifizierungen
- Das passiert, wenn ein Teil der Daten inkorrekt ist; man sagt die Daten sind **verrauscht**
- Es passiert auch wenn die Attribute nicht genügend Informationen bereitstellen, um die Situation vollständig zu beschreiben, oder wenn die Domäne wirklich nicht deterministisch ist
- Ein einfacher Ausweg für das Problem ist die Verwendung eines Mehrheitsvolums

DECISION-TREE-LEARNING

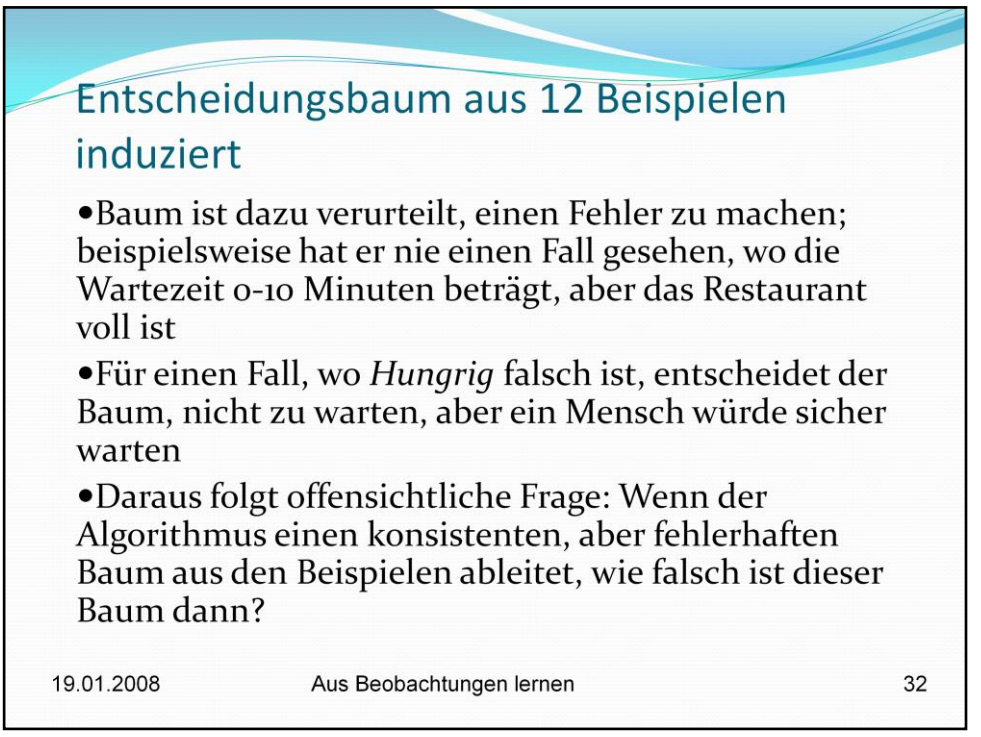
```
function DECISION-TREE-LEARNING(examples, attribs, default) returns einen
    Entscheidungsbaum
  inputs examples, Beispielmenge
         attribs, Attributmenge
         default, Defaultwert für das Zielprädikat

  if examples ist leer then return default
  else if alle examples haben dieselbe Klassifikation then return die
       Klassifikation
  else if attribs ist leer then return MAJORITY-VALUE(examples)
  else
    best ← CHOOSE-ATTRIBUTE(attribs, examples)
    tree ← ein neuer Entscheidungsbaum mit dem Wurzeltest best
    m ← MAJORITY-VALUE(examples_i)
    for each Wert v_i von best do
      examples_i ← {Elemente von examples mit best = v_i}
      subtree ← DECISION-TREE-LEARNING(examples_i, attribs − best, m)
      füge Zweig zu tree mit Beschriftung v_i und Unterbaum subtree hinzu
    return tree
```

Entscheidungsbaum aus 12 Beispielen induziert

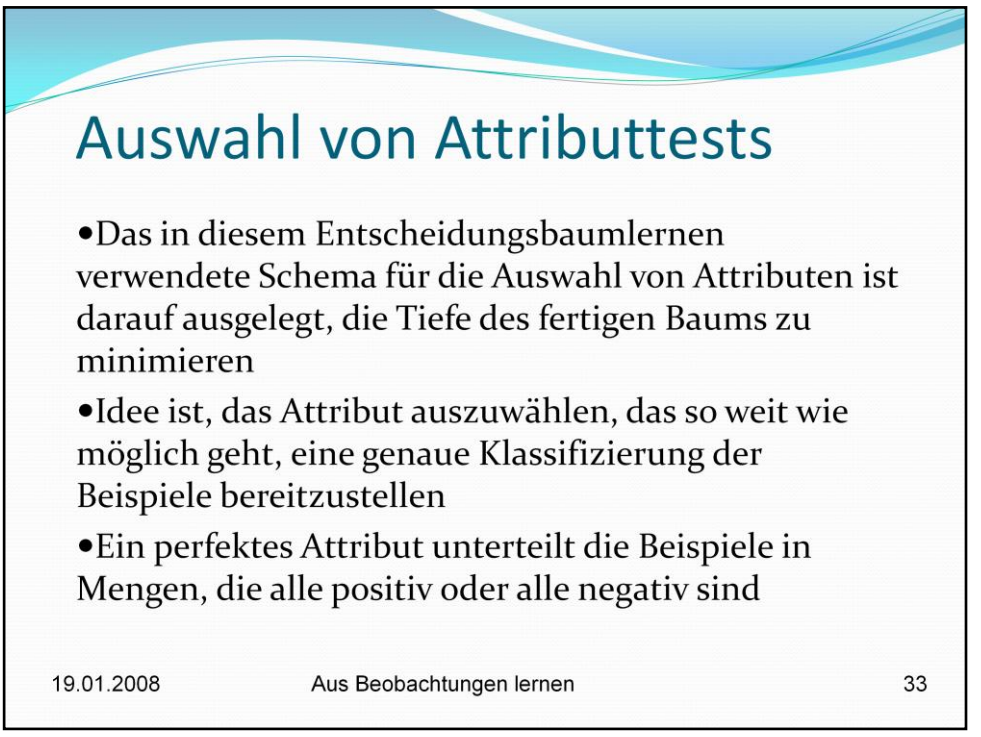


19.01.2008 Aus Beobachtungen lernen 31

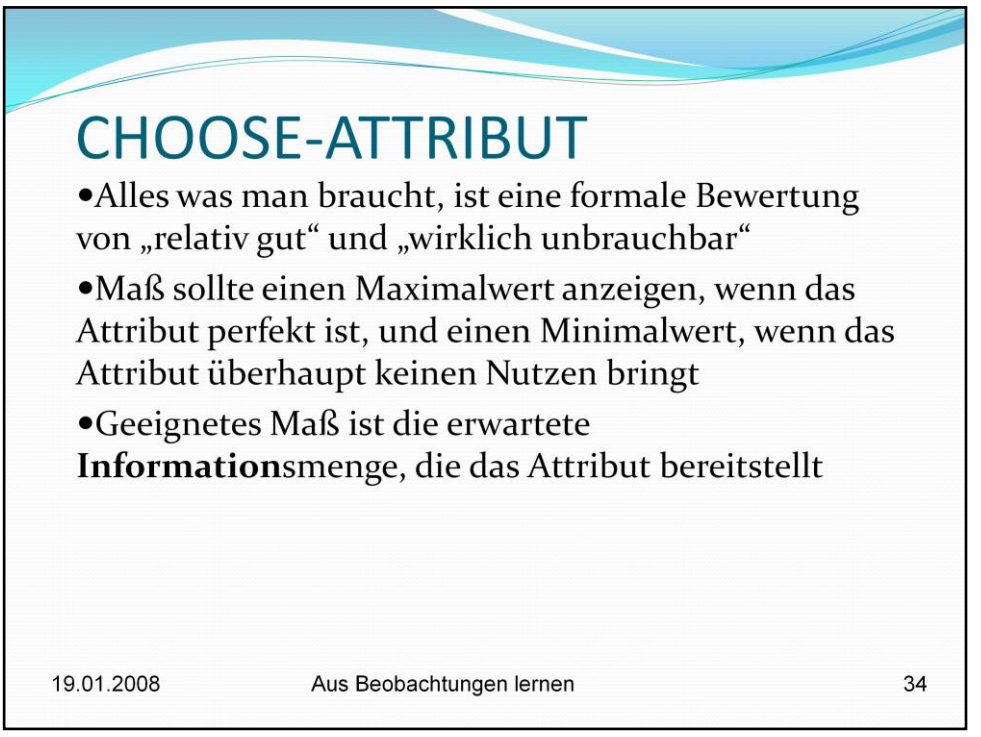
- Baum unterscheidet sich offensichtlich vom Originalbaum(Bild 18.2) – trotz der Tatsache dass die Daten von einem Agenten erzeugt wurden, der den Originalbaum heranzog
- Man könnte daraus schließen, dass der Lernalgorithmus keine gute Leistung beim Lernen der korrekten Funktion gezeigt hat
- Das wäre jedoch ein falscher Schluss
- Der Lernalgorithmus betrachtet die Beispiele, nicht die korrekte Funktion, und tatsächlich stimmt seine Hypothese nicht mit allen Beispielen überein, sondern ist wesentlicher einfacher als der Originalbaum
- Der Lernalrogithmus hat keinen Grund, Tests für Regen und Reservierung aufzunehmen, weil er alle 12 Beispiele auch ohne sie klassifizieren kann
- Außerdem hat er ein interessantes und zuvor nicht vermutetes Muster erkannt: Autor wartet am Wochenende auf Thai-Essen



- Zu 1. (Abb 18.6)
- Wenn man weitere Beispiele sammeln würde, könnte man einen Baum ähnlich dem Original induzieren
- Man kann zeigen, wie diese Frage experimentell analysieren kann, nachdem man die Details des Attributauswahlschritts aufgezeigt hat



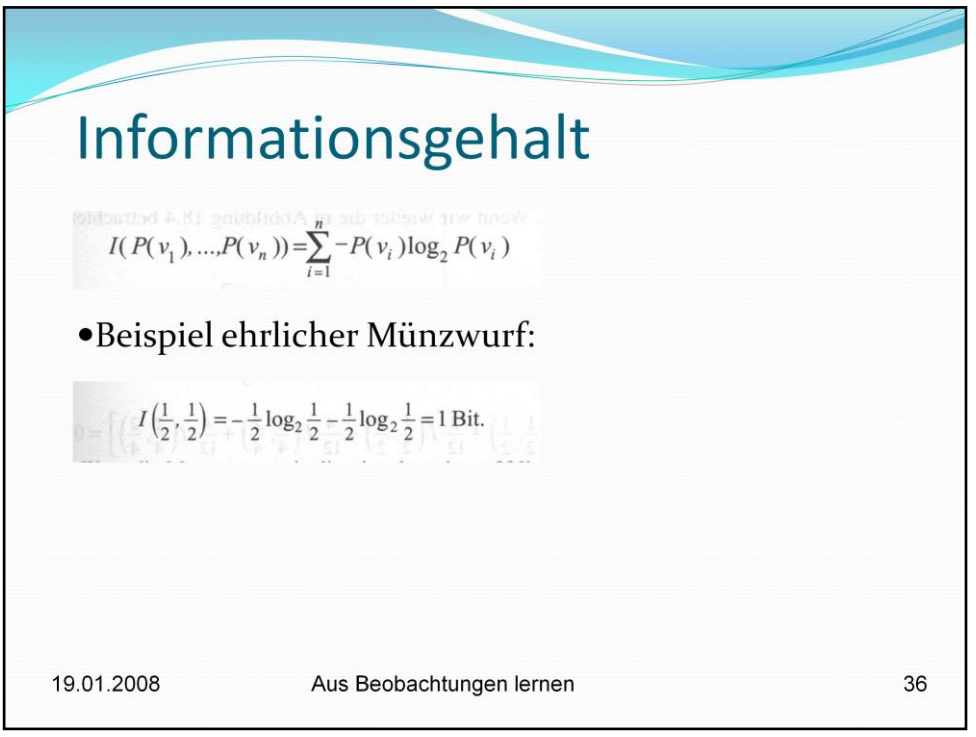
- Attribut *Gäste* ist nicht perfekt, aber es ist relativ gut
- Ein wirklich unbrauchbares Attribut, wie etwa *Typ*, hinterlässt die Beispielmengen mit etwa demselben Verhältnis an positiven und negativen Beispielen wie in der Originalmenge



- und man könnte die Funktion CHOOSE-ATTRIBUT aus Abb. 18.5 implementieren
- wobei man den Begriff in mathematischen Sinne verwendet, wie zuerst von Shannon und Weaver definiert

Konzept der Information

- Bietet Antwort auf eine Frage (z.B. bleibt die Münze auf Kopf oder Zahl liegen)
- In Antwort enthaltene Informationsbeitrag ist von dem eigenen Vorwissen abhängig
- Je weniger man weiß, desto mehr Information wird bereitgestellt
- Informationstheorie misst den Informationsgehalt in **Bits**
- 1 Bit Information ist ausreichend um eine Ja/Nein-Frage zu beantworten



- Möglichen Antworten v_i die Wahrscheinlichkeiten $P(v_i)$ haben, ist der Informationsgehalt I der tatsächlichen Antwort im Allgemeinen gegeben
- Wenn Münze so manipuliert ist, dass sie zu 99% Kopf erzielt, erhalten wir I(1/100, 99/100) = 0,08 Bit
- Wenn Wahrscheinlichkeit von Kopf gegen 1 geht, geht die Information der tatsächlichen Antwort gegen 0

Korrekte Klassifizierung

- Entscheidungsbaumlernen ist die Frage, was ist die korrekte Klassifizierung für ein gegebenes Beispiel
- Ein korrekter Entscheidungsbaum beantwortet diese Frage
- Ein Schätzung der Wahrscheinlichkeiten der möglichen Antworten, bevor eines der Attribute überprüft wurde, erhält man durch das Verhältnis der positiven und negativen Beispiele in der Trainingsmenge

Korrekte Klassifizierung

- Angenommen Traningsmenge enthält p positive und n negative Beispiele
- Schätzung der in einer korrekten Antwort enthaltenen Information lautet dann:

$$I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

- Restaurant Trainingsmenge aus Abb 18.3 hat $p = n = 6$
- -> man braucht also 1 Bit Information

Informationsgewinn

- Test auf ein einzelnes Attribut A vermittelt diese Information im Allgemeinen nicht, sondern nur einen Teil davon
- Man könnte genau messen, wie viel Information sie erbringt, indem man feststellt, wie viel Information man *nach* dem Attributtest noch braucht

Informationsgewinn

- Attribut A unterteilt die Traningsmenge E in Untermengen $E_1, ..., E_v$ gemäß ihren Werten für A, wobei A v verschiedene Werte haben kann
- Jede Untermenge E_i hat p_i positive und n_i negative Beispiele
- Wenn man also diesen Zweig verfolgt, braucht man zusätzliche $I(p_i/(p_i+n), n_i/(p_i+n_i))$ Bits an Informationen, um die Frage zu beantworten

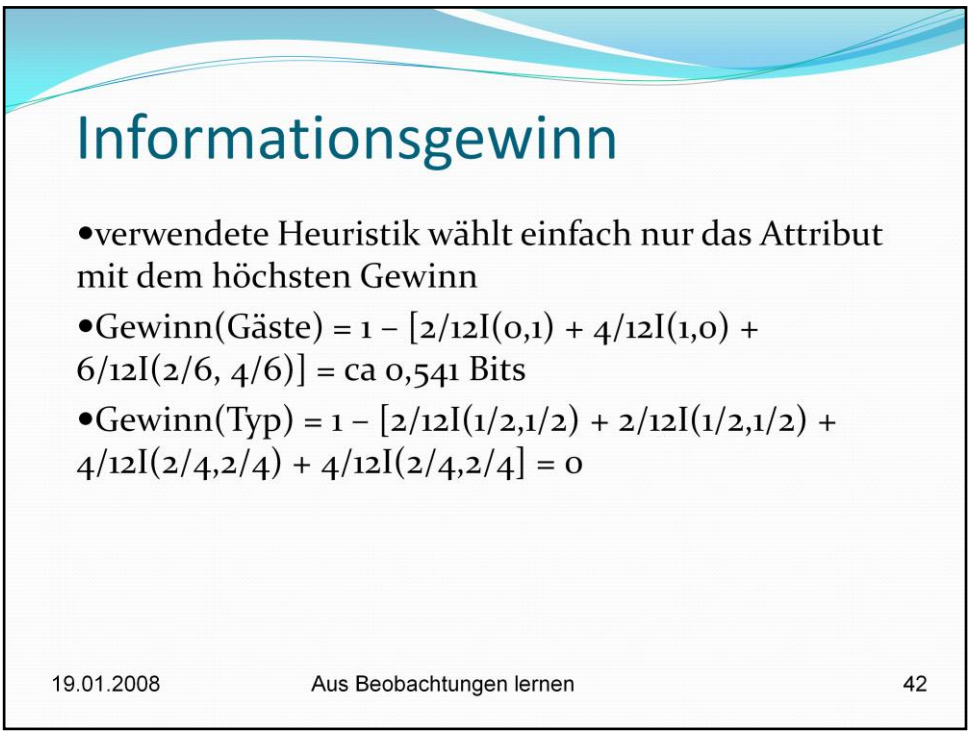
- Ein zufällig aus der Trainingsmenge ausgewähltes Beispiel hat den *i*-ten Wert für das Attribut mit der Wahrscheinlichkeit $(p_i+n_i)/(p+n)$

Informationsgewinn

$$Rest(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p+n} I\left(\frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i}\right)$$

- Informationsgewinn aus dem Attributtest ist die Differenz zwischen der ursprünglichen Informationsanforderung und der neuen Anforderung:
- Gewinn(A) = I (p/p+n, n/p+n) – Rest(A)

- Man braucht also nach dem Testen des Attributs durchschnittlich s.o. Bits Informationen um das Beispiel zu klassifizieren



- Zu 1. Die in Funktion CHOOSE-ATTRIBUTE ...
- Bestätigt die Vermutung, dass *Gäste* ein besseres Attribut für die Aufteilung ist
- Tatsächlich hat *Gäste* den höchsten Gewinn aller Attribute und würde vom Entscheidungsbaum-Lernalgorithmus als Wurzel gewählt

Leistungsabschätzung des Lernalgorithmus

- Lernalgorithmus ist gut, wenn er Hypothesen erzeugt, die die Klassifizierung noch nicht bekannter Beispiele gut vorhersagen
- Offensichtlich ist Vorhersagen gut, wenn sie sich als wahr erweist
- Man könnte also die Qualität einer Hypothese einschätzen, indem man ihre Vorhersage mit der korrekten Klassifizierung vergleicht, nachdem man sie kennt

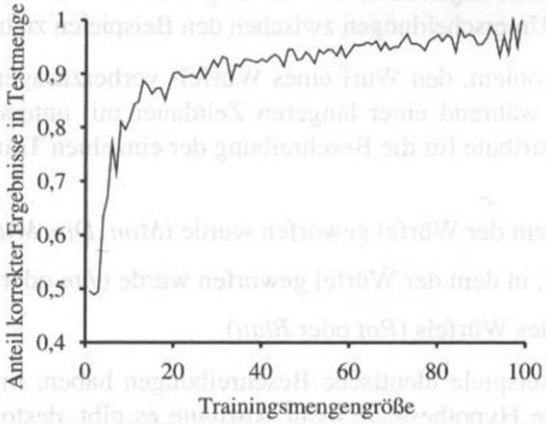
•Betrachtung einer Methode zur Einschätzung der Vorhersagequalität nach der Tatsache

•Das erledigt man für eine Beispielmenge (**Testmenge**)

Leistungsabschätzung des Lernalgorithmus

- Große Beispielmenge sammeln
- Beispielmenge in 2 separate Mengen unterteilen:
  - **Trainingsmenge** und **Testmenge**
- Lernalgorithmus auf Trainingsmenge anwenden und Hypothese h zu erzeugen
- Prozentsatz der Beispiele in der Testmenge messen, die durch h korrekt klassifiziert werden
- Vorhergegangene Schritte für verschiedene Größen von Traningsmengen und verschiedene zufällig ausgewählte Trainingsmengen beliebiger Größe wiederholen

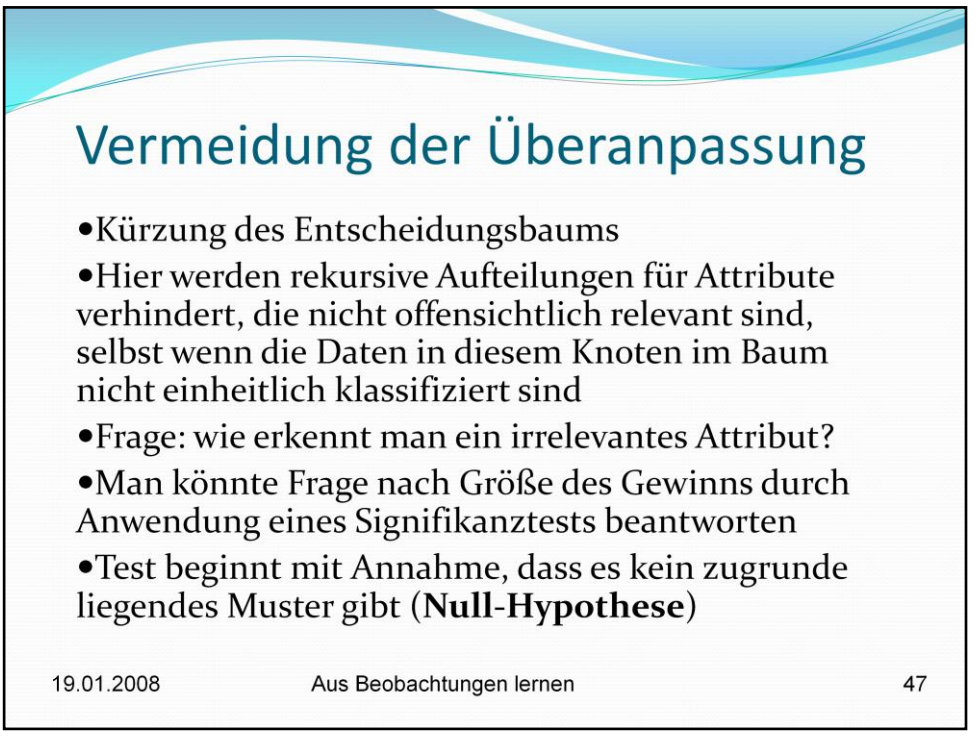
Lernkurve DECISION-TREE-LEARNING



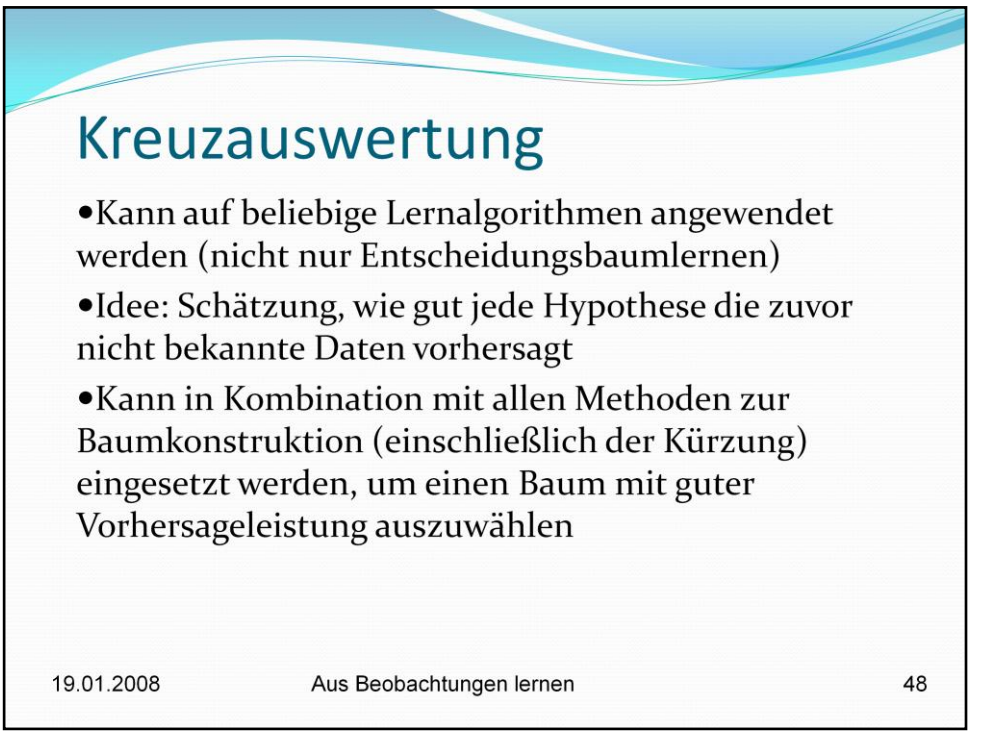
- Ergebnis der Prozedur ist eine Datenmenge die verarbeitet werden kann um die durchschnittliche Vorhersagequalität als Funktion der Größe der Traningsmenge zu ermitteln
- Kann als Graph dargestellt werden (Lernkurve) für den Algorithmus in der betreffenden Domäne
- Bild zeigt Lernkurve angewendet auf die Restaurantbeispiele (100 zufällig erzeugte Beispiele in der Restaurantdomäne, Graph zeigt 20 Versuche)
- Vorhersagequalität steigt mit wachsender Trainingsmenge (solche Kurven auch als **Happy-Graphen** bezeichnet)
- gutes Zeichen dafür, dass es wirklich ein Muster innerhalb der Daten gibt und dass der Lernalgorithmus es erkennt

Überanpassung

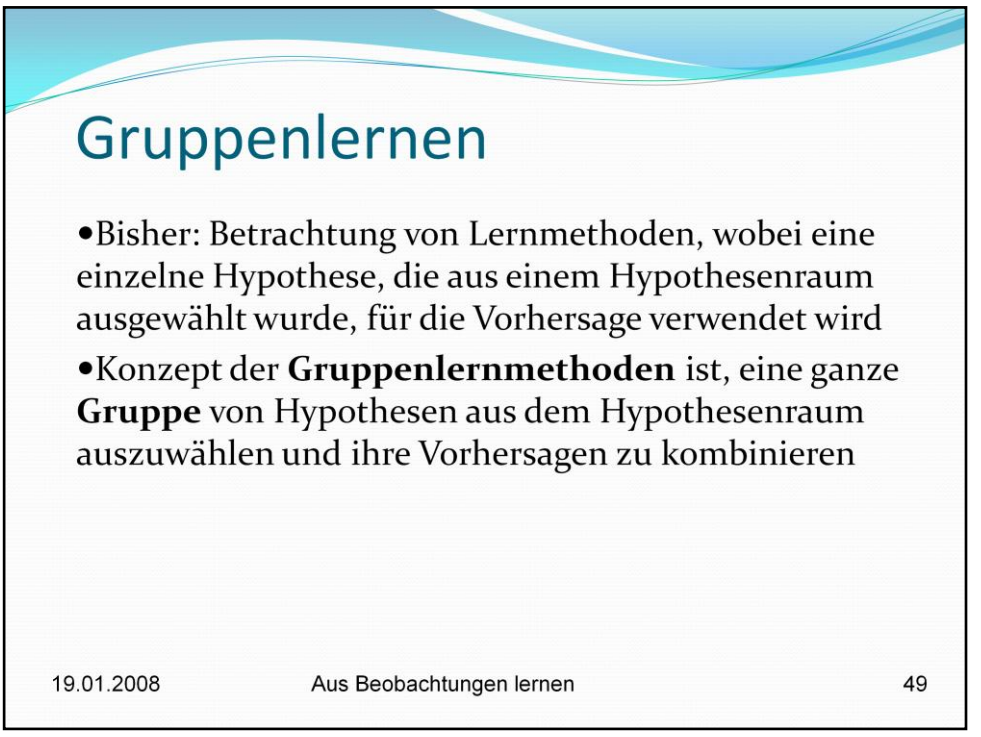
- Wenn große Menge möglicher Hypothesen muss man sehr sorgfältig vorgehen, damit man resultierende Freiheit, bedeutungslose „Regelmäßigkeiten“ in den Daten zu erkennen, nicht anwendet (Überanpassung)
- Tritt als allgemeines Phänomen häufig auf, wenn Zielfunktion nicht zufällig ist
- Gefährdet jede Art von Lernalgorithmus (nicht nur Entscheidungsbäume)



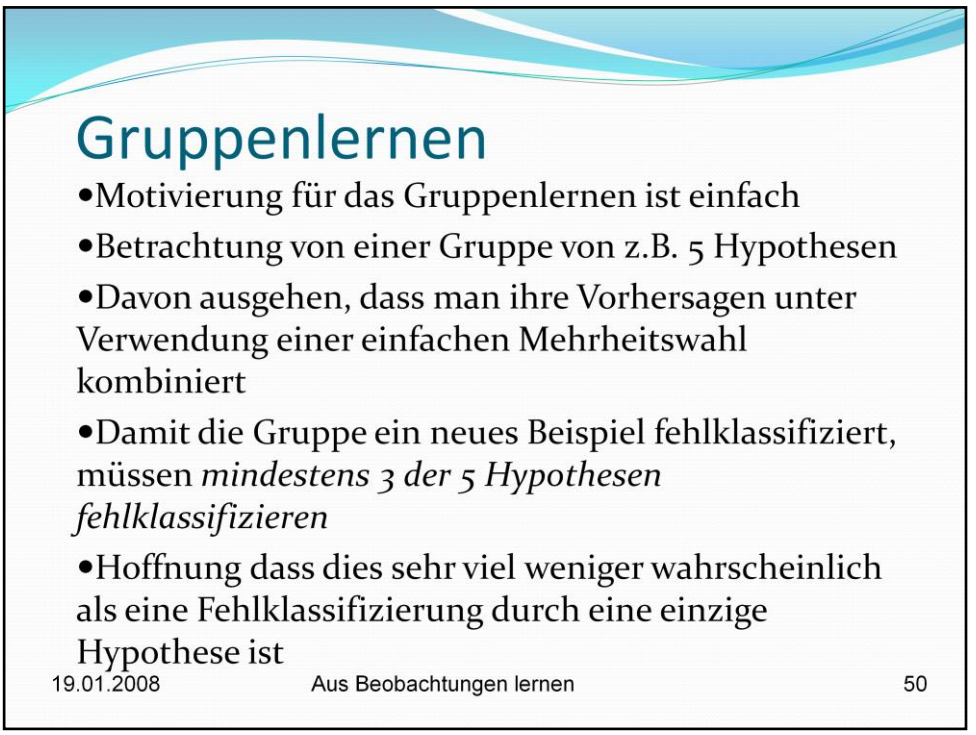
- Zerlegung der Beispielmenge unter Verwendung eines irrelevanten Attributs
- Man erwartet, dass die resultierenden Untermengen in etwa dieselben Verhältnisse aller Klassen wie Originalmenge haben (in diesem Fall liegt der Informationsgewinn nahe null)
- Informationsgewinn also ein guter Hinweis auf Irrelevanz
- Nun noch Frage, wie groß man einen Gewinn fordern sollte, um nach einem bestimmten Attribut zu unterteilen
- Anschließend werden die tatsächlichen Daten analysiert, um das Ausmaß zu berechnen, wie weit sie von einem perfekten Fehlen eines Musters abweichen
- Ist Abweichungsgrad statistisch unwarhscheinlich (man geht normalerweise von einem Mittelwert einer 5-prozentigen Wahrcheinlichkeit oder weniger aus), wird dies als guter Beweis für das Vorliegen eines signifikanten Musters in den Daten betrachtet
- Wahrscheinlichkeiten werden aus Standardverteilungen des Abweichungsbetrags berechnet, den man bei zufälligen Stichproben erwartet



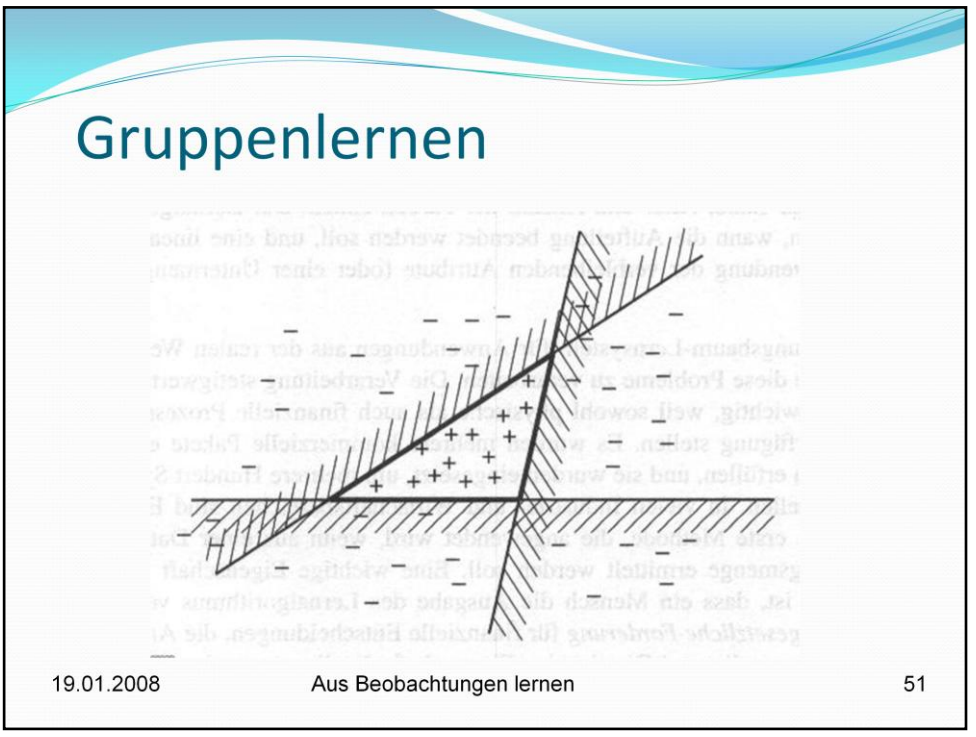
- x^2-Kürzung (nicht weiter beschrieben)
- Man legt einen Teil der bekannten Daten beiseite und verwendet sie, um die Vorhersageleistung einer Hypothese zu testen, die aus den restlichen Daten induziert wurde
- *k*-fache Kreuzauswertung bedeutet, dass man *k* Experimente durchführt und jede Mal ein anderes 1/*k* der Daten beiseite legt, um damit zu testen und dann ein Mittel über die Ergebnisse zu erzeugen
  - (beliebte Werte für k sind 5 und 10)
  - Extremfall k = n (Eins-Auslassen-Kreuzauswertung
- Um nicht erlaubte Einblicke zu vermeiden müsste man diese Leistung dann mit Hilfe einer neuen Testmenge überprüfen



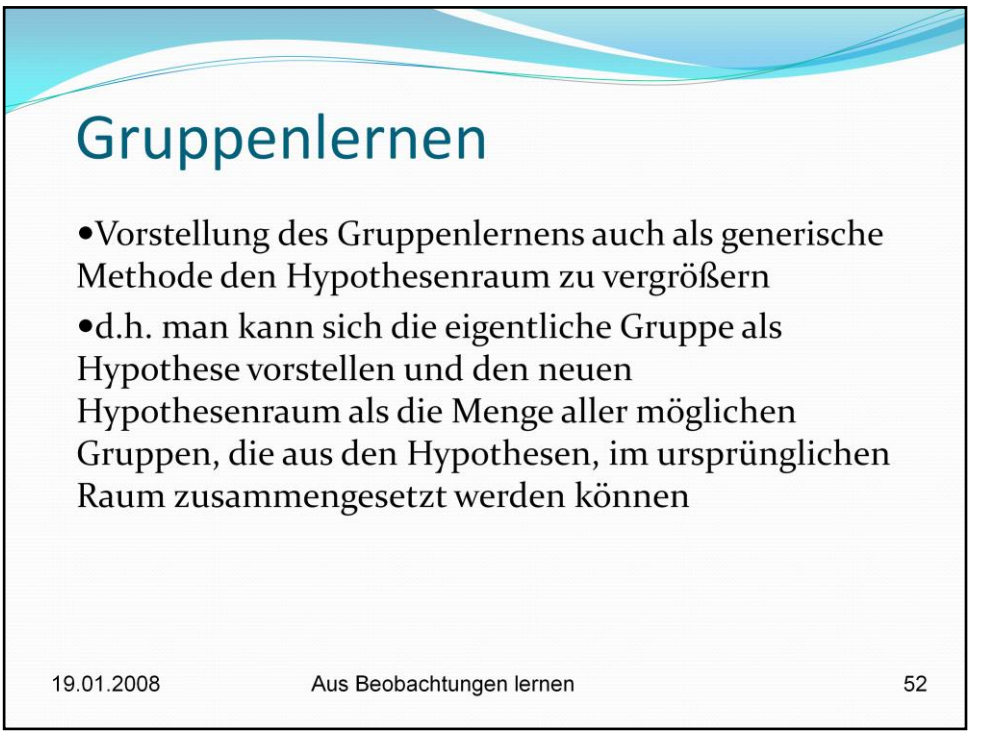
- Beispielsweise könnte man 100 verschiedene Entscheidungsbäume aus derselben Traningsmenge erstellen und sie zur besten Klassifizierung eines neuen Beispiels abstimmen lassen



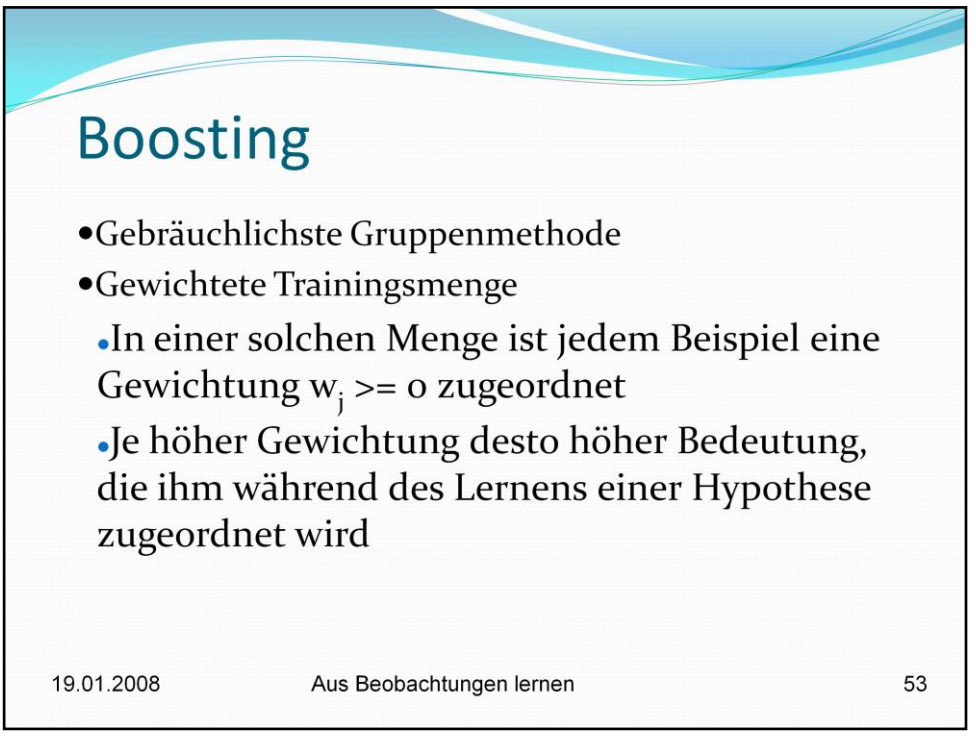
- Angenommen jede Hypothese h_i in der Gruppe den Fehler p hat d.h. Die Wahrscheinlichkeit dass ein zufällig gewähltes Beispiel durch h_i fehlklassifiziert wird, ist gleich p
- Die von jeder Hypothese erzeugten Fehler sind voneinander *unabhängig*
- In diesem Fall ist, wenn p klein ist, die Wahrscheinlichkeit einer großen Anzahl an Fehlklassifizierungen sehr klein
- Anname der Unabhängigkeit unvernünftig, weil Hypothesen wahrscheinlich durch irreführende Aspekte der Traningsdaten auf dieselbe Weise irregeführt werden
- Wenn sich Hypothesen jedoch zumindest ein bisschen unterscheiden und dabei die Korrelation zwischen ihren Fehlern reduzieren, kann das Gruppenlernen sehr sinnvoll sein



- Darstellung der gesteigerten Ausdruckskraft durch Gruppenlernen
- 3 Hypothesen mit linearen Schwellen, die jeweils die nicht schattierte Seite positiv klassifizieren und die zusammen ein Beispiel als positiv klassifizieren, das von allen dreien als positiv klassifiziert wurde
- Resultierende dreieckige Bereich ist eine Hypothese, die im ursprünglichen Hypothesenraum nicht ausdrückbar ist

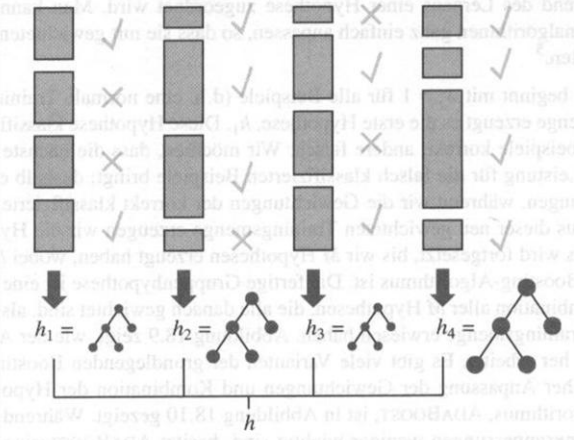


- 18.8 zeigt, wie dies zu einem ausdrucksstärkeren Hypothesenraum führe kann
- Wenn der ursprüngliche Hypothesenraum einen einfachen und effizienten Lernalgorithmus unterstützt, bietet Gruppenlernen eine Möglichkeit, eine sehr viel ausdrucksstärkere Klasse von Hypothesen zu lernen, ohne dass dazu sehr viel mehr rechentechnischer oder algorithmischer Aufwand erforderlich wäre



- Beginnt mit $w_j = 1$ für alle Beispiele
- Aus dieser Menge Erzeugung der ersten Hypothese h_1
- Diese Hypothese klassifiziert einige der Trainingsbeispiele korrekt andere falsch
- Nächste Hypothese soll eine bessere Leistung für die falsch klassifizierten Hypothesen bringen
- Deshalb erhöht man ihre Gewichtung, während die Gewichtung der korrekt klassifizierten Beispiele verringert wird
- Aus der neu gewichteten Trainingsmenge erzeugt man die Hypothese h_2
- Prozess wird fortgesetzt, bis man M Hypothesen erzeugt hat
- Fertige Gruppenhypothese ist eine gewichtete Mehrheitskombination aller M Hypothesen, die danach gewichtet sind, als wie gut sie sich für die Traningsmenge erwiesen haben

Boosting



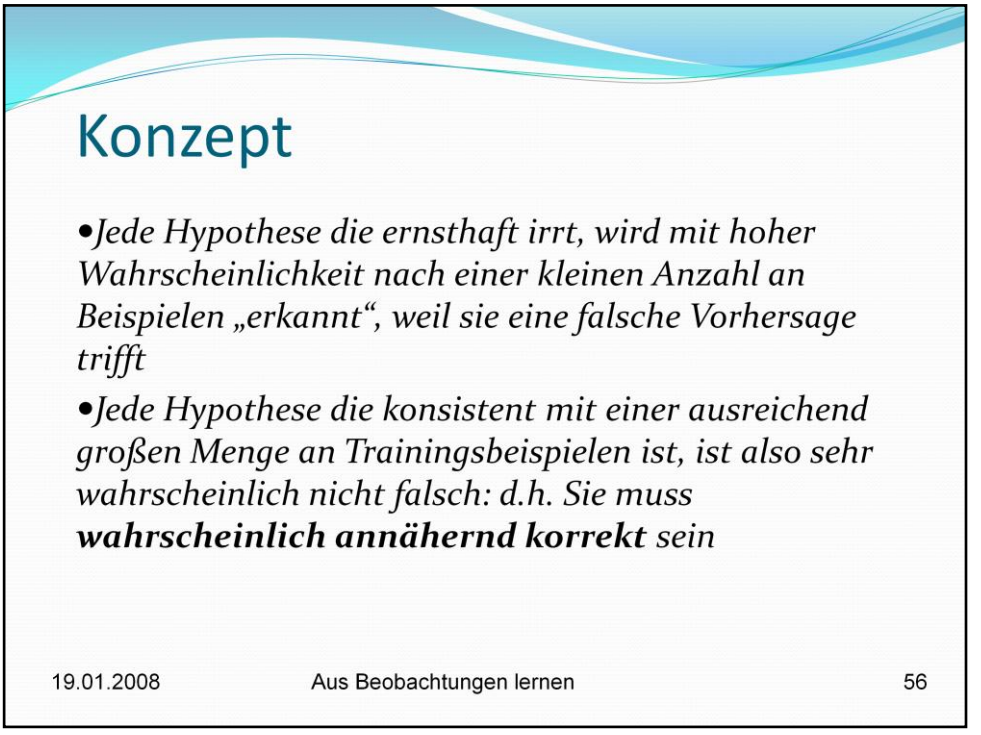
19.01.2008 Aus Beobachtungen lernen 54

- Jedes schattierte Rechteck entspricht einem Beispiel, die Höhe des Rechtecks entspricht der Gewichtung
- Haken und Kreuze geben an ob das Beispiel von der aktuellen Hypothese korrekt klassifiziert wurde
- Größe des Entscheidungsbaums gibt die Gewichtung dieser Hypothese in der fertigen Gruppe an

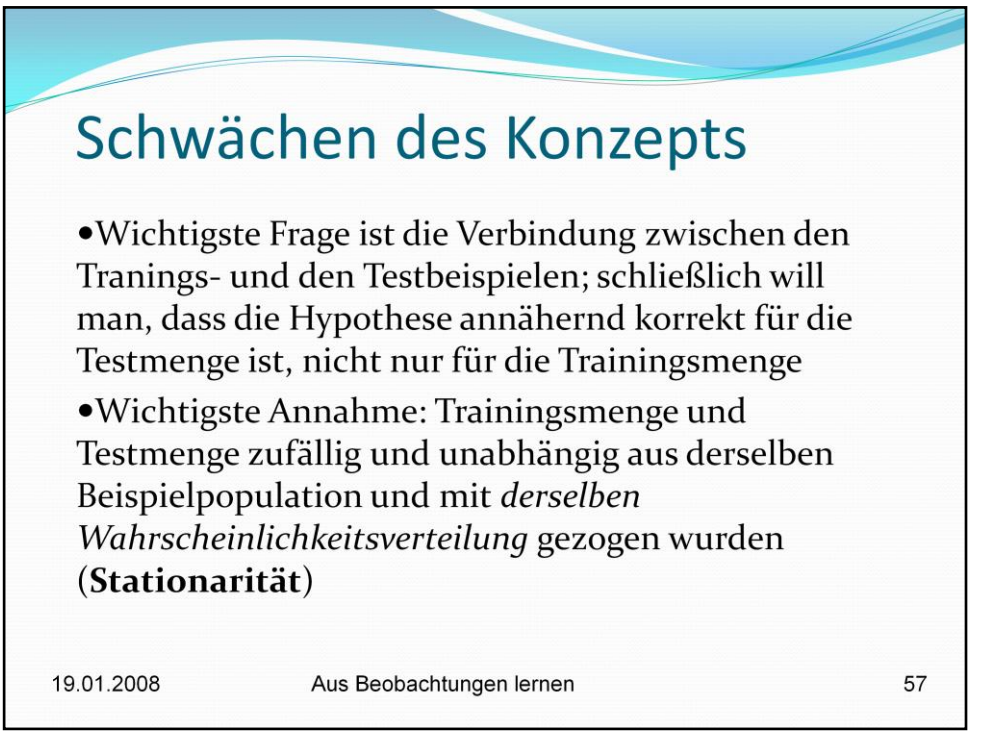
Warum das Lernen funktioniert: Computer-Lerntheorie

- Wichtigste Frage war:
  - Wie kann man sicher sein, dass ein Lernalgorithmus eine Theorie produziert hat, die korrekte Vorhersagen für die Zukunft trifft?
- Formal ausgedrückt:

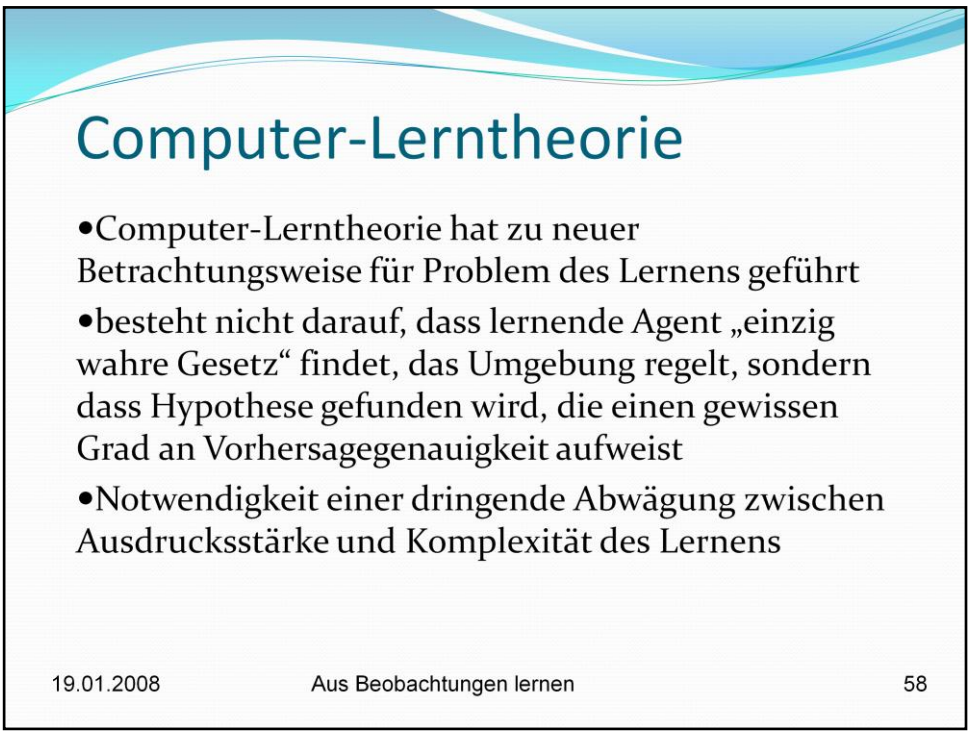
wie weiß man, dass die Hypothese *h* nahe der Zielfunktion *f* liegt, wenn man nicht weiß, was *f* ist?



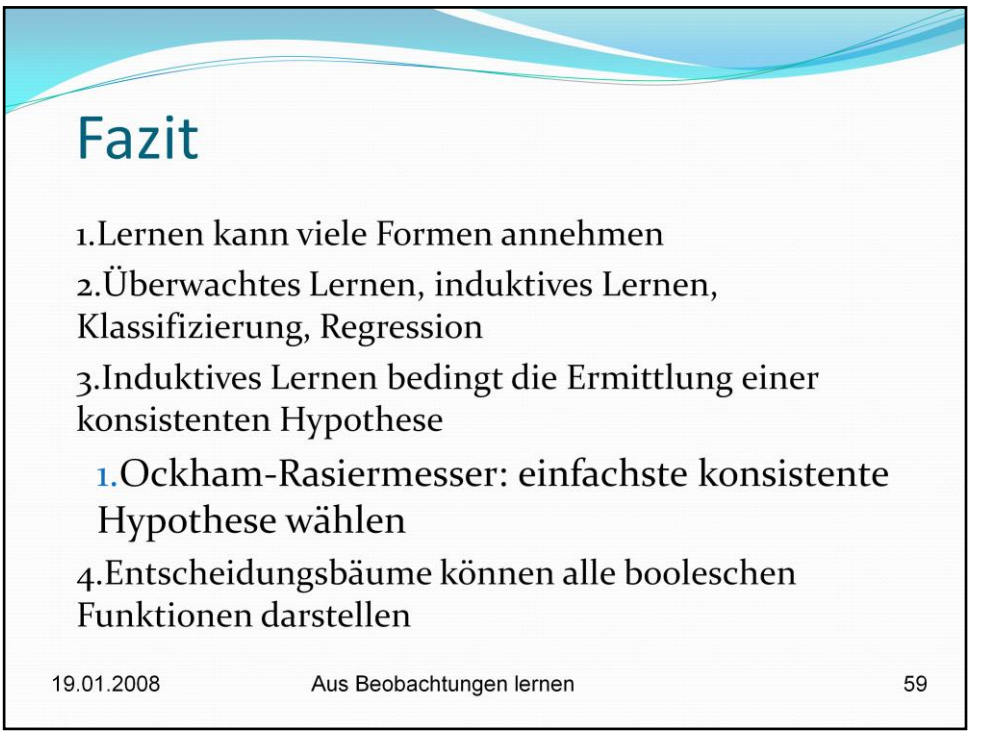
- Jeder Lernalgorithmus der Hypothesen zurückgibt, die wahrscheinlich annähernd korrekt sind, wird als PAC-Alrogithmus bezeichnet (probably approximately correct)



- Ohne diese Annahme kann die Theorie keinerlei Behauptungen über die Zukunft treffen, weil es keine notwendige Verbindung zwischen Zukunft und Vergangenheit gäbe
- Annahme geht davon aus, dass der Prozess, der die Beispiele auswählt, nicht böswillig handelt
- Wenn die Trainingsmenge aus lauter seltsamen Beispielen besteht – z.B. zweiköpfigen Hunden -dann kann der Lernalrogithmus offensichtlich nicht anders, als wenig erfolgreiche Verallgemeinerungen zur Erkennung von Hunden zu treffen



- Anfang der 60er Jahre Konzentration auf das Problem der **Identifikation innerhalb der Grenzen**
- Identifikationsalgorithmus muss gemäß dieses Konzepts eine Hypothese zurückgeben, die genau mit der wahren Funktion übereinstimmt
- Ordnung aller Hypothesen in **H** gemäß irgendeinem Maß
- Wähle die einfachste Hypothese, die mit allen bisher gezeigten Beispielen konsistent ist
- Wenn neue Beispiele eintreffen, verwerfe alte Hypothese wenn ungültig und übernehme stattdessen eine komplexere
- Nachdem wahre Funktion erreicht nicht mehr verwerfen
- Leider sind in vielen Hypothesenräumen die Anzahl der Beispiele und die Rechenzeit bis zum Erreichen der wahren Funktion enorm groß
- (führte zu einer wichtigen Klasse von Lernalrogithmen, Support-Vektor-Maschinen)



1. abhängig von der Natur des Leistungselements, der zu verbessernden Komponenten und dem verfügbaren Feedback
2. Verfügbare Feedback (von einem Lehrer oder aus der Umgebung) den korrekten Wert für Beispiele bereitstellt, Lernproblem **überwachtes Lernen**
    1. Aufgabe ist es dann eine Funktion aus Beispielen für ihre Eingaben und Ausgaben zu lernen (**induktives Lernen**)
    2. Lernen ist diskretwertige Funktion (**Klassifizierung**)
    3. Lernen aus einer stetigwertigen Funktion (Regression)
3. die mit den Beispielen übereinstimmt
    1. Schwierigkeit der Aufgabe ist von der gewählter Repräsentation abhängig

Fazit

1. Heuristik des **Informationgewinns** effiziente Methode für Ermittlung eines einfachen, konsistenten Entscheidungsbaums
2. Leistung eines Lernalgorithmus anhand einer **Lernkurve** messen
3. Vorhersagegenauigkeit für Testmenge als Funktion der Trainingsmengengröße
4. Gruppenmethoden (z.B. Boosting) häufig bessere Leistung als Einzelmethoden