

Grundlagen der Künstlichen Intelligenz

13 Maschinelles Lernen

Lernen durch Beobachtung, Entscheidungsbäume

Volker Steinhage

Inhalt

- Der lernende Agent
- Induktives Lernen
- Lernen von Entscheidungsbäumen

Lernen

- Was ist *Lernen* im Agentenkontext?

Ein Agent lernt, wenn er durch Erfahrung seine Fähigkeit, eine Aufgabe zu lösen, verbessern kann.

→ z.B. Spielprogramme

- Warum *Lernen* für Agenten?

→ Lernen als Voraussetzung für *Autonomie*

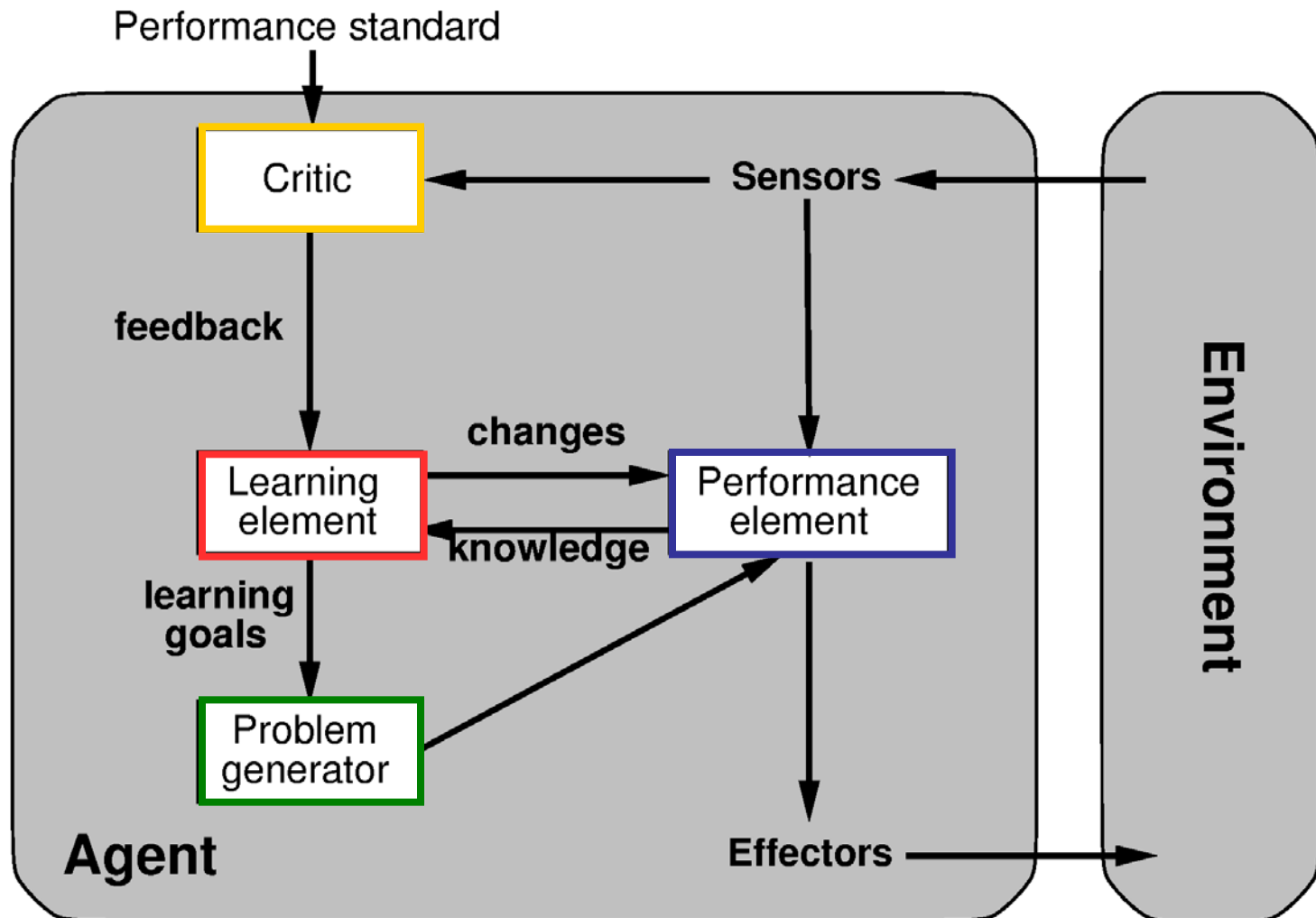
→ Lernen als effizienter Weg für *Wissensakquisition*

→ Lernen als Weg für den Bau von *High-Performance* Systemen

Der lernende Agent

Bisher: die Wahrnehmungen (Perzepte) des Agenten dienten nur dem Handeln.

Jetzt: Perzepte sollen auch der Verbesserung des **zukünftigen Verhaltens** dienen.



Bausteine des lernenden Agenten

Performance-Element: Verarbeitet Wahrnehmungen und wählt Aktionen aus

→ entspricht einem der bisherigen Agentenmodelle

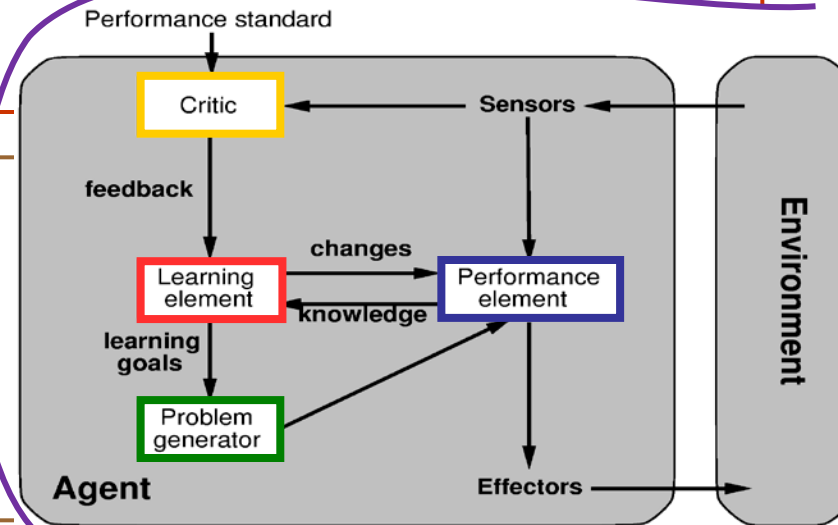
Learning-Element: Durchführen von Verbesserungen

→ Braucht Wissen über sich selbst und wie sich der Agent in der Umwelt bewährt

Critic: Bewertung des Agentenverhaltens auf der Grundlage eines gegebenen Verhaltensmaßstabs

→ *Rückkopplung (feedback)*

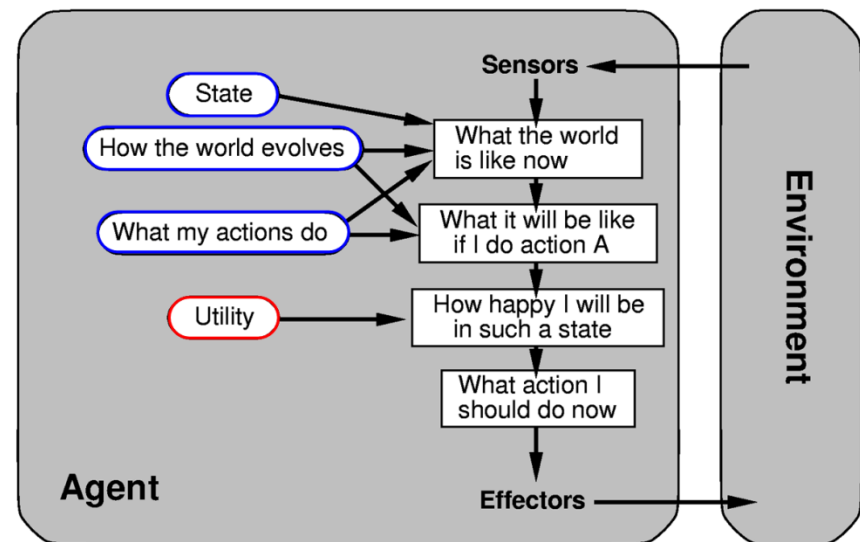
Problem-Generator: Vorschlagen von *explorativen* Aktionen, die den Agenten zu neuen Erfahrungen führen



Das **Learning**-Element

Seine Funktionsweise wird von **drei entscheidenden Fragen** beeinflusst:

1. Welche **Komponenten der Performance-Elements** sollen verbessert werden (Zustandsmodell, Änderungsmodell der Umwelt, Wechselwirkungsmodell mit Umwelt, Nutzenfunktion)?
2. Welche **Repräsentation für die Komponenten** wird gewählt (Logik, BNs, diskret, kontinuierlich,...)?
3. Welche **Form von Rückkopplung** (\approx **Critic**) ist verfügbar?



Form der **Rückkopplung** ist wichtigster Beurteilungsfaktor eines Lernproblems!

Ziel des Lernens

- Ein-/Ausgabekonzept des Agenten:
 - **Eingabe**: Information aus der Umwelt
 - **Ausgabe**: Die Effekte der Aktionen des Agenten
- Betrachtung der **Effekte**:
 - Effekte, die der Agent durch sein Handeln erzielen will (**Ideal**),
 - Effekte, die tatsächlich eintreten (**Realität**),

unterscheiden sich oft erheblich!

~ Ziel der Lernens: **Annähern** des tatsächlichen Handelns an das ideale Handeln.

Repräsentationsformen

Es gibt verschiedene Lernverfahren für alle in der Vorlesung vorgekommenen Formen der Wissensrepräsentation:

1. Numerische Funktionen, z.B. Bewertungsfunktionen bei Spielen:

- Neuronale Netze
- Kernel-Methoden, z.B. Support-Vector-Maschinen

2. Logikbasierte Beschreibungen, z.B. für alle Komponenten logischer Agenten:

- Entscheidungsbäume (für Aussagenlogik)
- Induktive Logische Programmierung (Lernen von Prädikaten)

3. Probabilistische Beschreibungen, z.B. Bayes-Netze:

- Bayessches Lernen
- Unsupervised Clustering

Form der Rückkopplung: Überwachtes Lernen

Überwachtes Lernen (Supervised Learning):

- Es gibt eine Trainingsmenge T mit korrekten Ein-/Ausgabe-Paaren.

↗ Die Rückkopplung für den Agenten ist also:

↗ für jede Eingabe aus T steht die entspr. korrekte Ausgabe (bzw. der Unterschied zw. errechneter und korrekter Ausgabe) zur Verfügung.

↗ Bezogen auf das Handlungslernen des Agenten:

Es ist so, als wenn ein Lehrer dem Agenten die richtige Aktion zu jeder Zustandsbeschreibung aus der Trainingsmenge mitteilt.

Form der Rückkopplung: Unüberwachtes Lernen

Unüberwachtes Lernen (Unsupervised Learning):

- Die Trainingsmenge T enthält nur Eingabewerte bzw. Eingabetupel.

~ keine Rückkopplung!

~ Der Agent kann aus T nur Modelle für das Auftreten von Mustern bzw. Regelmäßigkeiten lernen, aber *nicht*, was er richtigerweise tun müsste.

~ Annahme ist also, dass T inhärente Muster enthält.

— Bspl.: Taxifahrer-Agent

- Eingabetupel seien (Wochentag, Uhrzeit, Fahrdistanz, Fahrdauer).
- Der Agent kann so z.B. Konzepte für Haupt-, Normal- und Schwachverkehrszeiten lernen – ohne jemals explizit als solche bezeichnete Beispiele gesehen zu haben.

Form der Rückkopplung: Verstärkendes Lernen

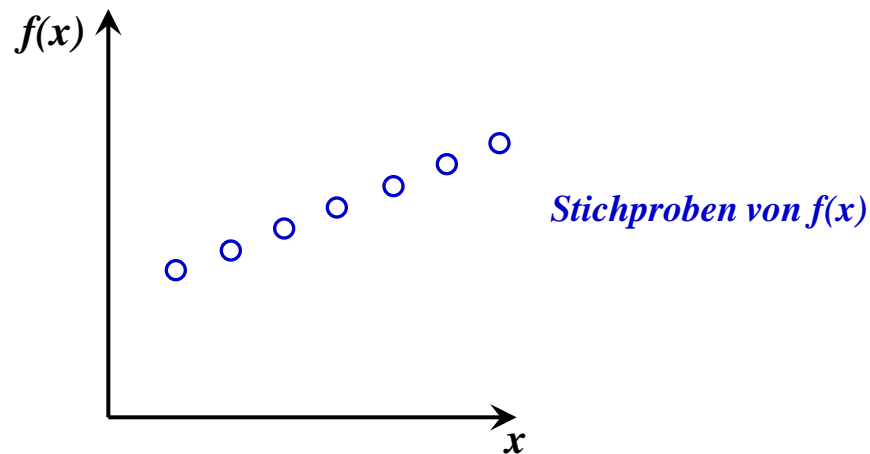
Verstärkendes Lernen (Reinforcement Learning):

- Die Trainingsmenge T enthält Eingabe-/Verstärkungspaare.
- Die Rückkopplung ist also eine Verstärkung.
- Eine Verstärkungen ist ein Gewinn, eine Belohnung oder ein Bestrafung, die mit dem in der Eingabe kodierten Zustand oder kodierten Aktion einher geht.
- Bspl.: Taxifahrer-Agent
 - Die Höhe (oder auch das Fehlen) des Trinkgeldes nach einer Fahrt wäre eine Verstärkung zu jedem Eingabetupel über eine Taxifahrt der Art (Wochentag, Uhrzeit, Fahrdistanz, Fahrdauer, Verhalten).
 - ~ Der Agent kann so lernen, (a) Fahrten zu bestimmten Zeiten zu bevorzugen, (b) sein Verhalten gegenüber dem Fahrgast zu ändern.

Start: Überwachtes Lernen \rightarrow Induktives Lernen (1)

- Induktives Lernen ist ein *überwachtes* Lernverfahren, das eine unbekannte Funktion f aus einer **Trainingsmenge** T von Ein-/Ausgabepaaren $(x_i, f(x_i))$ schätzt.
- Jedes Paar $(x_i, f(x_i))$ wird als **Beispiel** oder **Stichprobe** bezeichnet.
- **Induktive Inferenz**: Für eine Menge $(x_i, f(x_i))$ von Beispielen für f ist eine Funktion* h (**Hypothese**) gesucht, die f approximiert.

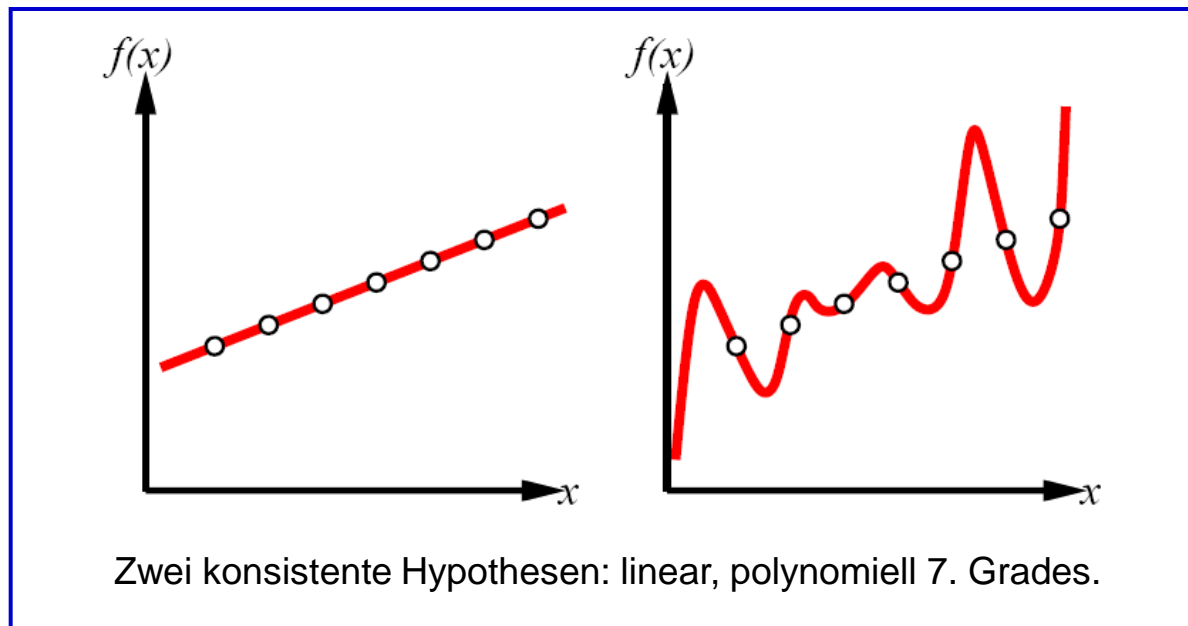
Induktive Inferenz in Abgrenzung gegenüber der deduktiven Inferenz (z.B. in der Logik durch Resolution).



* Das Lernen von Funktionen ist keine starke Einschränkung, da jede Art von Lernen als das Lernen der Repräsentation einer Funktion verstanden werden kann.

Überwachtes Lernen → Induktives Lernen (2)

- Eine Hypothese heißt *konsistent* mit der Trainingsmenge T , wenn sie alle Beispiele aus T erklärt.
- ~ Wie wählen wir aber aus mehreren konsistenten Hypothesen h_i aus?



- Eine Antwort ist *Ockhams Rasiermesser*⁽¹⁾ (*Occam's Razor*): von mehreren konsistenten Alternativen ist die *einfachste* Hypothese auszusuchen.

⁽¹⁾ nach dem Theologen und Logiker *William von Ockham*
(* 1285 Ockham (Grafschaft Surrey), † 9.4.1347 München)

Induktives Lernen → Entscheidungsbäume

- **Eingabe:** *Beschreibung einer Situation* durch eine Menge von Eigenschaften bzw. Attributen (entsprechen *Grundliteralen in FOL*).
- **Ausgabe:** *Ja/Nein-Entscheidung* bezüglich eines *Booleschen Zielprädikats*.
- **Repräsentationsmächtigkeit:** *Boolesche Funktionen* über Grundliteralen in FOL.
- **Aufbau:**
 - ein *interner Knoten* repräsentiert den Test eines Attributs,
 - ausgehende *Kanten* sind mit den möglichen Werten des Tests markiert,
 - jeder *Blattknoten* trägt den Booleschen Wert für das Zielprädikat, der bei Erreichen des Blattes zurückgegeben werden soll.
- **Ziel des Lernprozesses:** Definition eines Zielprädikats als Entscheidungsbaum.

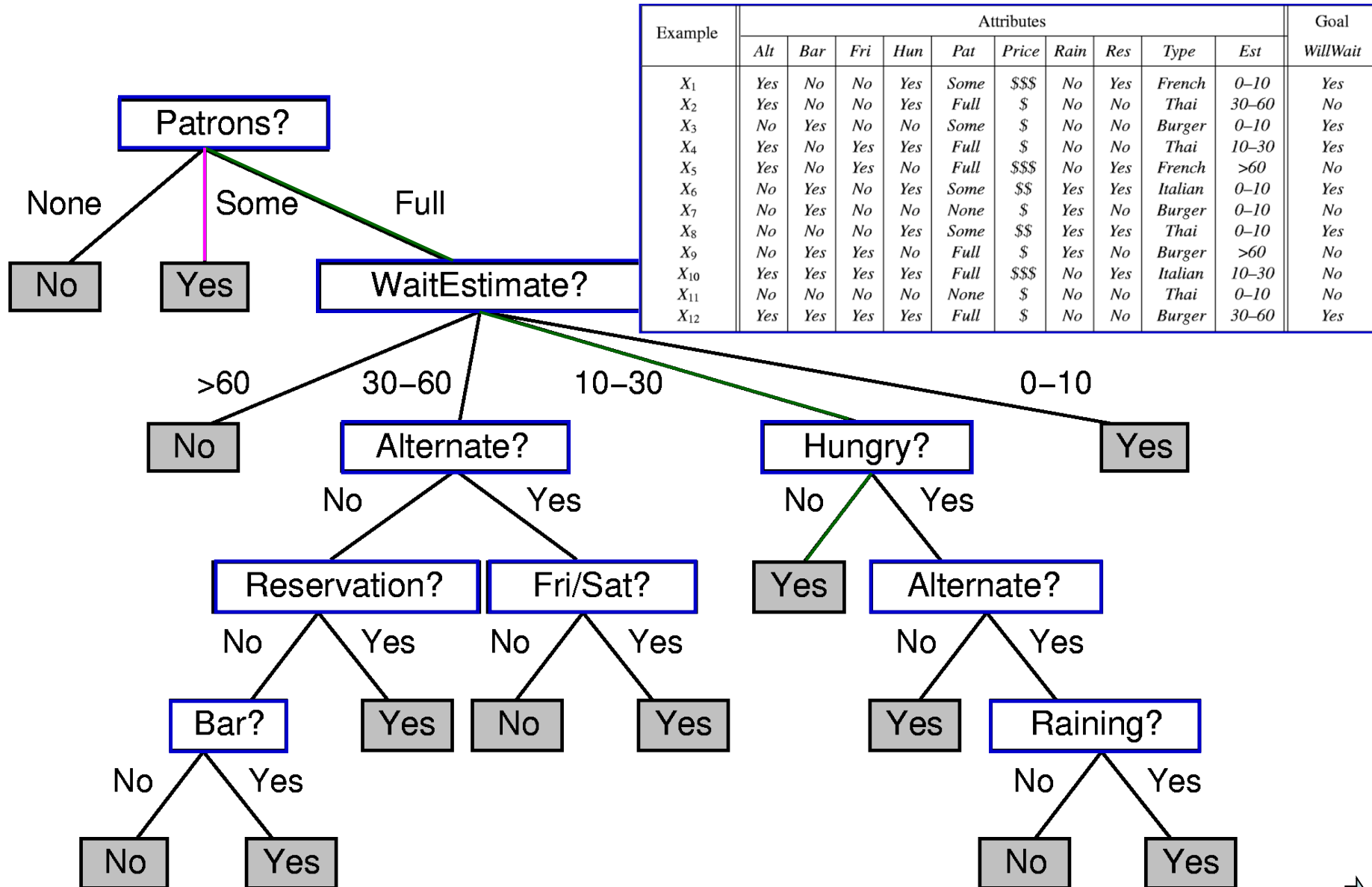
Restaurantbeispiel (Attribute)

Beschreibende Attribute:

- *Patrons*: Wieviele Gäste sind da? (None, Some, Full)
- *WaitEstimate*: Wie lange warten? (0-10, 10-30, 30-60, >60)
- *Alternate*: Gibt es eine Alternative? (T/F)
- *Hungry*: Bin ich hungrig? (T/F)
- *Reservation*: Habe ich reserviert? (T/F)
- *Bar*: Hat das Restaurant eine Bar zum Warten? (T/F)
- *Fri/Sat*: Ist es Freitag oder Samstag? (T/F)
- *Raining*: Regnet es draußen? (T/F)
- *Price*: Wie teuer ist das Essen? (\$, \$\$, \$\$\$)
- *Type*: Art des Restaurants? (French, Italian, Thai, Burger)

Zielprädikat *WillWait*: Soll ich warten? (T/F)

Restaurantbeispiel (Entscheidungsbaum)



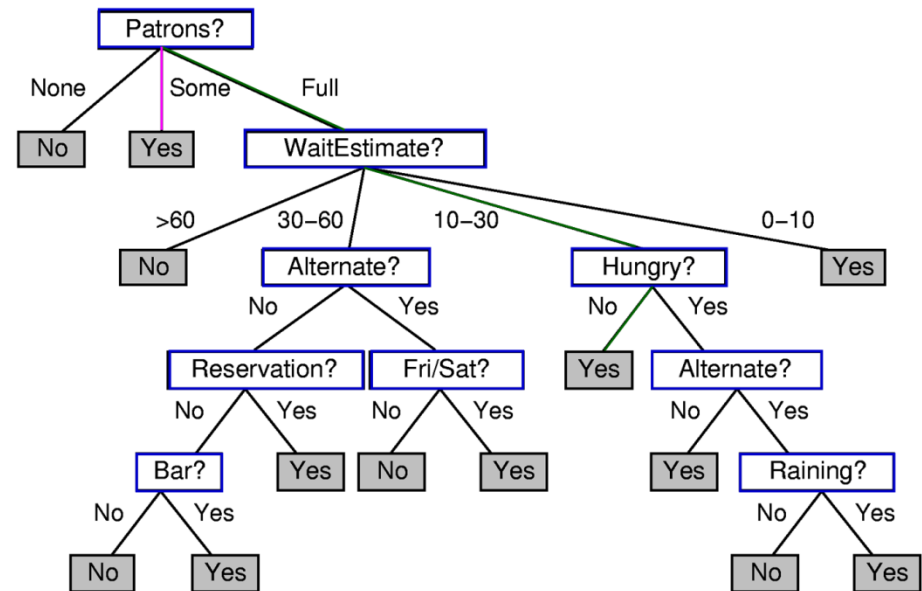
Konstruierter Entscheidungsbaum von Stuart Russel



Repräsentation des Zielprädikats

Ein Entscheidungsbaum kann als
Konjunktion von Implikationen
repräsentiert werden.

Die Implikationen entsprechen den
Pfaden, die in YES-Knoten enden:


$$\begin{aligned} &\forall r \{ \\ &\quad [\textit{Patrons}(r, \textit{Some}) \Rightarrow \textit{WillWait}(r)] \\ &\quad \wedge \dots \\ &\quad \wedge [\textit{Patrons}(r, \textit{Full}) \wedge \textit{WaitEstimate}(r, 10-30) \wedge \textit{Hungry}(r, \textit{No}) \Rightarrow \textit{WillWait}(r)] \\ &\quad \wedge \dots \\ &\} \end{aligned}$$

Ausdruckskraft von Entscheidungsbäumen (1)

Theorem 1: *Alle aussagenlogischen Formeln (Boolesche Fakten) sind mit Entscheidungsbäumen darstellbar.*

Frage: Kann ein Entscheidungsbaum eine beliebige Menge von Objekten darstellen?



Mit klassischen Entscheidungsbäumen *nicht*:

- Alle Tests beziehen sich immer nur auf ein Objekt (im Bspl. das Restaurant r),
- die Sprache von klassischen Entscheidungsbäumen ist inhärent aussagenlogisch.

Ausdruckskraft von Entscheidungsbäumen (2)

Zur Einschränkung von traditionellen Entscheidungsbäumen auf einzelne Objekte:

- Z.B. ist als Test nicht darstellbar:

$$\exists r_2 \text{ Near}(r_2, r) \wedge \text{Price}(r, p) \wedge \text{Price}(r_2, p_2) \wedge \text{Cheaper}(p_2, p)$$

- Möglicher Ausweg: ein neuer Test *CheaperRestaurantNearby*
 - ~ aber ein klassischer Entscheidungsbaum könnte exponentiell mit solchen Attributen anwachsen.

~ Erweiterungen existieren,

z.B. (Blockeel und De Raedt, *Artificial Intelligence*, 1998)*.

* H. Blockeel, L. De Raedt: Top-down induction of first-order logical decision trees. *Artificial intelligence* 101 (1), 285-297, 1998.

Kompakte Repräsentationen (1)

- Theoretisch kann *jede Zeile einer Wahrheitstabelle* in einen Pfad eines Entscheidungsbaums übertragen werden.
 - ~ Allerdings ist die Größe der Tabelle und damit eines so generierten Baums exponentiell in der Anzahl der Attribute.
- Ziel des Lernens von Entscheidungsbäumen sollte also auch sein, *kompakte Entscheidungsbäume* abzuleiten!
 - ~ Sind für alle Boolesche Funktionen über Grundliteralen in FOL auch *kompakte Entscheidungsbäume* ableitbar?

Kompakte Repräsentationen (2)

- Leider nicht: es gibt Boolesche Funktionen, die einen Baum exponentieller Größe erfordern:

- Parity Funktion:

$$p(x) = \begin{cases} 1 & \text{gerade Anzahl von Eingaben} \\ 0 & \text{sonst} \end{cases}$$

- Majority Funktion:

$$m(x) = \begin{cases} 1 & \text{Hälfte der Eingaben ist 1} \\ 0 & \text{sonst} \end{cases}$$

- ~ Es gibt keine kompakte Repräsentation für *alle* möglichen Booleschen Funktionen.

Restaurantbeispiel: Lernen aus Trainingsmenge (1)

Klassifizierung eines Beispiels = Wert des Zielprädikats:

Yes \leadsto positives Beispiel

No \leadsto negatives Beispiel

Trainingsmenge: 6 positive und 6 negative Beispiele

Example	Attributes										Goal
	<i>Alt</i>	<i>Bar</i>	<i>Fri</i>	<i>Hun</i>	<i>Pat</i>	<i>Price</i>	<i>Rain</i>	<i>Res</i>	<i>Type</i>	<i>Est</i>	<i>WillWait</i>
X_1	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>Yes</i>	<i>Some</i>	<i>\$\$\$</i>	<i>No</i>	<i>Yes</i>	<i>French</i>	<i>0–10</i>	<i>Yes</i>
X_2	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>Yes</i>	<i>Full</i>	<i>\$</i>	<i>No</i>	<i>No</i>	<i>Thai</i>	<i>30–60</i>	<i>No</i>
X_3	<i>No</i>	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>Some</i>	<i>\$</i>	<i>No</i>	<i>No</i>	<i>Burger</i>	<i>0–10</i>	<i>Yes</i>
X_4	<i>Yes</i>	<i>No</i>	<i>Yes</i>	<i>Yes</i>	<i>Full</i>	<i>\$</i>	<i>No</i>	<i>No</i>	<i>Thai</i>	<i>10–30</i>	<i>Yes</i>
X_5	<i>Yes</i>	<i>No</i>	<i>Yes</i>	<i>No</i>	<i>Full</i>	<i>\$\$\$</i>	<i>No</i>	<i>Yes</i>	<i>French</i>	<i>>60</i>	<i>No</i>
X_6	<i>No</i>	<i>Yes</i>	<i>No</i>	<i>Yes</i>	<i>Some</i>	<i>\$\$</i>	<i>Yes</i>	<i>Yes</i>	<i>Italian</i>	<i>0–10</i>	<i>Yes</i>
X_7	<i>No</i>	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>None</i>	<i>\$</i>	<i>Yes</i>	<i>No</i>	<i>Burger</i>	<i>0–10</i>	<i>No</i>
X_8	<i>No</i>	<i>No</i>	<i>No</i>	<i>Yes</i>	<i>Some</i>	<i>\$\$</i>	<i>Yes</i>	<i>Yes</i>	<i>Thai</i>	<i>0–10</i>	<i>Yes</i>
X_9	<i>No</i>	<i>Yes</i>	<i>Yes</i>	<i>No</i>	<i>Full</i>	<i>\$</i>	<i>Yes</i>	<i>No</i>	<i>Burger</i>	<i>>60</i>	<i>No</i>
X_{10}	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Full</i>	<i>\$\$\$</i>	<i>No</i>	<i>Yes</i>	<i>Italian</i>	<i>10–30</i>	<i>No</i>
X_{11}	<i>No</i>	<i>No</i>	<i>No</i>	<i>No</i>	<i>None</i>	<i>\$</i>	<i>No</i>	<i>No</i>	<i>Thai</i>	<i>0–10</i>	<i>No</i>
X_{12}	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Full</i>	<i>\$</i>	<i>No</i>	<i>No</i>	<i>Burger</i>	<i>30–60</i>	<i>Yes</i>

Restaurantbeispiel: Lernen aus Trainingsmenge (2)

- Den trivialen Entscheidungsbaum erhält man, indem jedes Beispiel in einem Pfad repräsentiert wird.
- ~ Der triviale Entscheidungsbaum memorisiert lediglich die Beispiele der Trainingsmenge:
 - ~ keine kompakte Repräsentation,
 - ~ keine Extraktion eines allgemeinen Musters
(keine Generalisierung),
 - ~ keine Vorhersagekraft.

Restaurantbeispiel: Lernen aus Trainingsmenge (3)

- Zur Erzielung von Kompaktheit, Generalisierung und Vorhersagekraft ist **Ockham's Razor** anzuwenden:

„Die wahrscheinlichste Hypothese ist die *einfachste*, die alle Beispiele umfasst.“

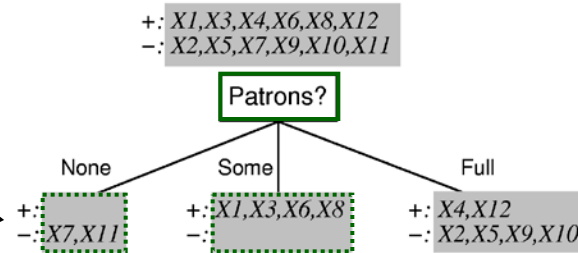
- ~ hier: der Baum mit der *minimalen* Anzahl von Tests!
- ~ Leider ist das Erzeugen des kleinsten Entscheidungsbaums ist nicht handhabbar.
- ~ Heuristiken, die zu einer *kleinen* Menge von Tests führen.

Gewichtung von Attributen (informell)

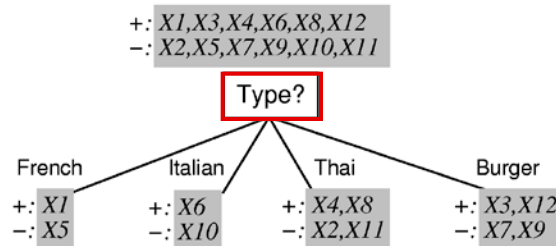
Heuristik: Wähle für jeden Aufbauschritt des Baumes das Attribut, das die größtmögliche Entscheidbarkeit in die Trainingsmenge induziert.

Bspl.: (a)

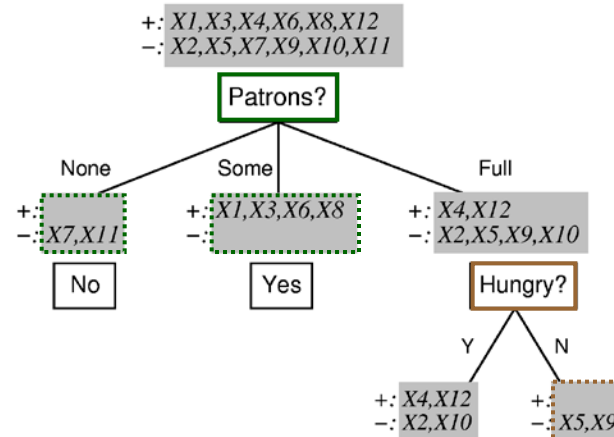
Patrons? ist günstig, weil für die Werte *None* und *Some* keine weiteren Tests benötigt werden.



(b)



(c)



Example	Attributes										Goal	
	Alt	Bar	Fri	Hun	Pat	Price	Rain	Res	Type	Est	WillWait	
X ₁	Yes	No	No	Yes	Some	\$\$\$	No	Yes	French	0-10	Yes	
X ₂	Yes	No	No	Yes	Full	\$	No	No	Thai	30-60	No	
X ₃	No	Yes	No	No	Some	\$	No	No	Burger	0-10	Yes	
X ₄	Yes	No	Yes	Yes	Full	\$	No	No	Thai	10-30	Yes	
X ₅	Yes	No	Yes	No	Full	\$\$\$	No	Yes	French	>60	No	
X ₆	No	Yes	No	Yes	Some	\$\$	Yes	Yes	Italian	0-10	Yes	
X ₇	No	Yes	No	No	None	\$	Yes	No	Burger	0-10	No	
X ₈	No	No	No	Yes	Some	\$\$	Yes	Yes	Thai	0-10	Yes	
X ₉	No	Yes	Yes	No	Full	\$	Yes	No	Burger	>60	No	
X ₁₀	Yes	Yes	Yes	Yes	Full	\$\$\$	No	Yes	Italian	10-30	No	
X ₁₁	No	No	No	No	None	\$	No	No	Thai	0-10	No	
X ₁₂	Yes	Yes	Yes	Yes	Full	\$	No	No	Burger	30-60	Yes	

Type? ist ungünstig, weil für alle Werte weitere Tests benötigt werden.

Also Wahl von *Patrons?*
Und nun weiter mit *Hungry?*
...

Rekursives Lernverfahren (1)

Der Aufbau eines Entscheidungsbaumes erfolgt rekursiv:

- Nach jedem Test liegt ein erneutes Entscheidungsbaum-Lernproblem vor.
- Dabei haben wir aber
 - weniger Beispiele, die noch nicht klassifizierbar sind,
 - die Anzahl der noch nicht verwendeten Attribute um eins reduziert ist.

Rekursives Lernverfahren (2)

Für jeden Schritt können die *aktuellen Blattknoten* vier Fälle zeigen:

- 1) *Positive und negative Beispiele*: Wähle neues Attribut.
- 2) *Nur positive oder nur negative Beispiele*: fertig.
- 3) *Keine Attribute* mehr, aber noch Beispiele mit unterschiedl. Klassifikation: es lagen Fehler in den Daten vor (\sim **NOISE**) oder die Attribute sind unzureichend.
Antworte JA, wenn die Mehrzahl der Beispiele positiv ist, sonst NEIN.
- 4) *Keine Beispiele*: Es gab kein Beispiel mit dieser Eigenschaft. Antworte JA, wenn Mehrzahl der Beispiele *des Elternknotens* positiv ist, sonst NEIN.

Der Algorithmus

function DECISION-TREE-LEARNING(*examples*, *attributes*, *default*) **returns** a decision tree

inputs: *examples*, set of examples

attributes, set of attributes

default, default value for the goal predicate

if *examples* is empty **then return** *default*

Fall 4: Majorität der Elternknoten

else if all *examples* have the same classification **then return** the classification

Fall 2 ~ Blattknoten

else if *attributes* is empty **then return** MAJORITY-VALUE(*examples*)

Fall 3: Majorität der Beispiele

else

best ← CHOOSE-ATTRIBUTE(*attributes*, *examples*)

Fall 1: neues Attribut

tree ← a new decision tree with root test *best*

for each value v_i of *best* **do**

examples_i ← {elements of *examples* with *best* = v_i }

subtree ← DECISION-TREE-LEARNING(*examples_i*, *attributes* – *best*,
MAJORITY-VALUE(*examples*))

Majoritäts-
information
zu Fall 4

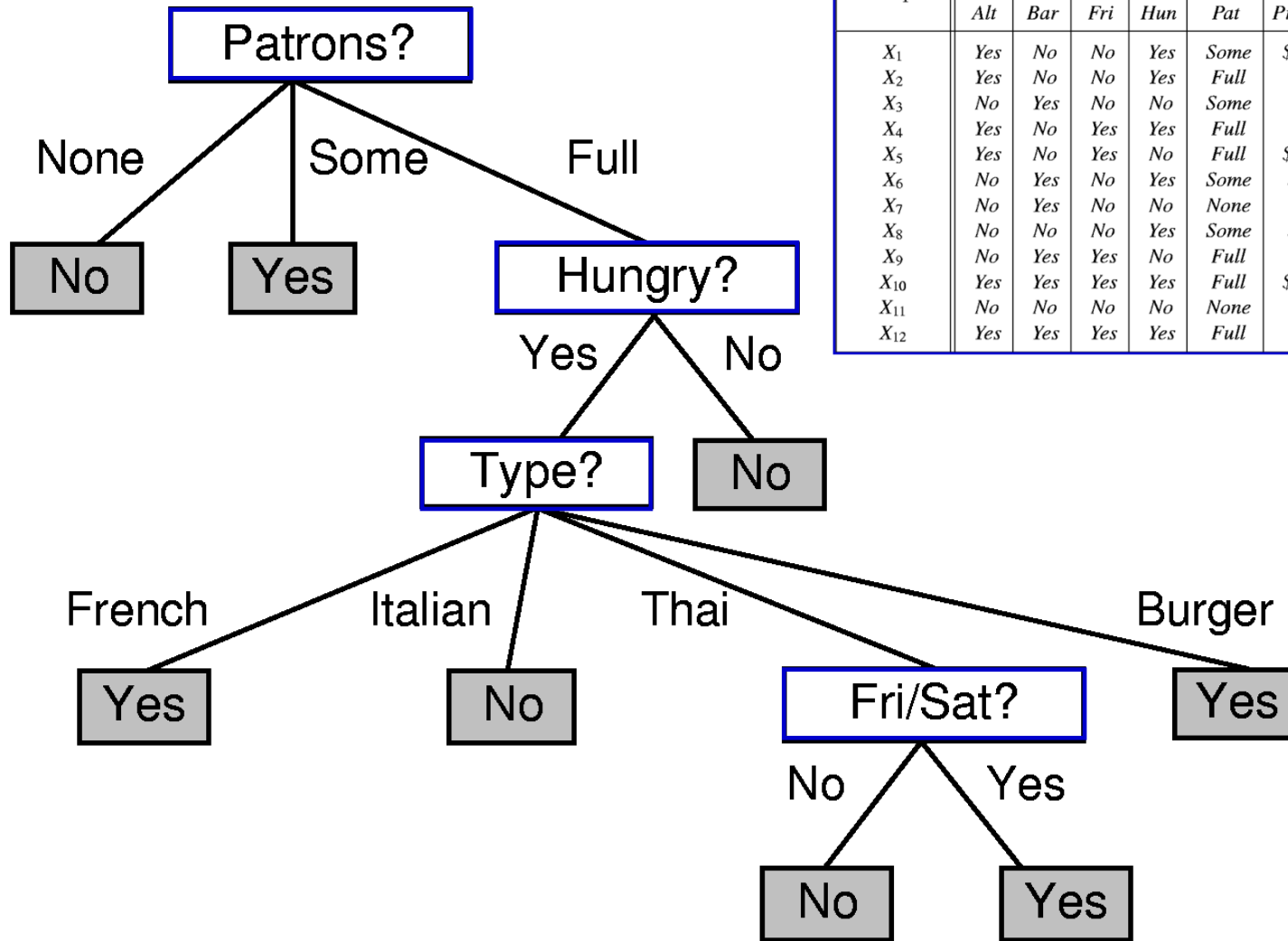
add a branch to *tree* with label v_i and subtree *subtree*

end

return *tree*

Anwendung auf die Restaurant-Daten

Example	Attributes										Goal
	Alt	Bar	Fri	Hun	Pat	Price	Rain	Res	Type	Est	WillWait
X ₁	Yes	No	No	Yes	Some	\$\$\$	No	Yes	French	0-10	Yes
X ₂	Yes	No	No	Yes	Full	\$	No	No	Thai	30-60	No
X ₃	No	Yes	No	No	Some	\$	No	No	Burger	0-10	Yes
X ₄	Yes	No	Yes	Yes	Full	\$	No	No	Thai	10-30	Yes
X ₅	Yes	No	Yes	No	Full	\$\$\$	No	Yes	French	>60	No
X ₆	No	Yes	No	Yes	Some	\$\$	Yes	Yes	Italian	0-10	Yes
X ₇	No	Yes	No	No	None	\$	Yes	No	Burger	0-10	No
X ₈	No	No	No	Yes	Some	\$\$	Yes	Yes	Thai	0-10	Yes
X ₉	No	Yes	Yes	No	Full	\$	Yes	No	Burger	>60	No
X ₁₀	Yes	Yes	Yes	Yes	Full	\$\$\$	No	Yes	Italian	10-30	No
X ₁₁	No	No	No	No	None	\$	No	No	Thai	0-10	No
X ₁₂	Yes	Yes	Yes	Yes	Full	\$	No	No	Burger	30-60	Yes

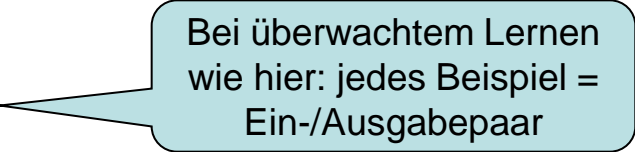


Mit *Decision-Tree-Learning* aus den Trainingsdaten gelernter Entscheidungsbaum



Bewertung eines Lernalgorithmus (1)

Ansatz zur Beurteilung der Vorhersagekraft:

- Sammle eine große Menge **M** von Beispielen. 
- Unterteile **M** in zwei *disjunkte* Mengen: Trainings- und Validierungsmenge.*
- Benutze Trainingsmenge, um *Hypothese* h zu erstellen.
- Benutze Validierungsmenge, um den Anteil korrekt klassifizierter Beispiele zu messen.
- Wiederhole das Verfahren für zufällig gewählte Trainingsmengen unterschiedlicher Größe.

* Trainings- und Validierungsmenge müssen unbedingt getrennt gehalten werden. Beliebter Fehler: Aufgrund der Validierung wird der Lernalgorithmus verändert und danach mit den selben Trainings- und Validierungsmengen getestet. Dadurch wird Wissen über die Validierungsmenge in den Algorithmus gesteckt und es besteht keine Unabhängigkeit zwischen Trainings- und Validierungsmengen mehr.

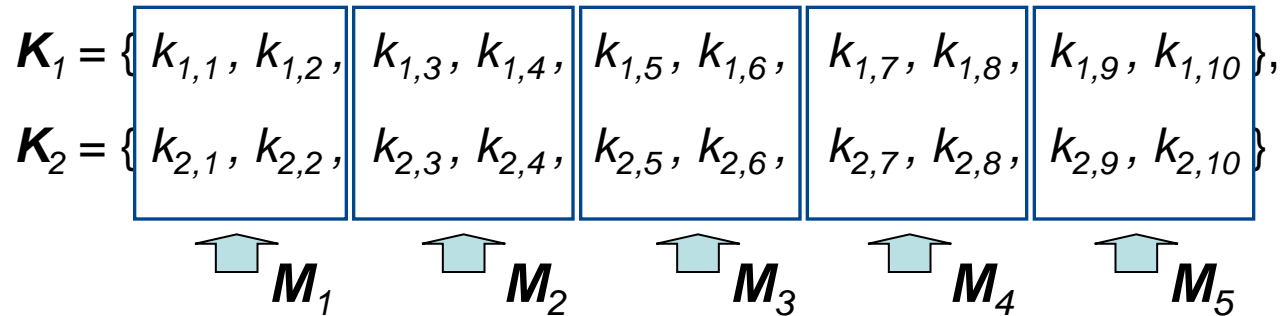
Bewertung eines Lernalgorithmus (2)

Kreuzvalidierung (*cross validation*) setzt den skizzierten Ansatz systematisch um

- Bei k-fachen Kreuzvalidierung (*k-fold cross validation*) wird die Beispielmenge M in k disjunkte Teilmengen unterteilt.
- In k Iterationen wird das Lernverfahren jedes Mal auf einer anderen Teilmenge validiert, nachdem es auf den jeweils restlichen k-1 Teilmengen trainiert wurde.
- Ergebnis der Bewertung ist der über die k Iterationen gemittelte Anteil korrekt klassifizierter Beispiele .
- Bei der k-fachen stratifizierten Kreuzvalidierung (*stratified k-fold cross-validation*) wird darauf geachtet, dass jede der k Teilmengen annähernd die gleiche Verteilung besitzt. Dadurch wird die Varianz der Abschätzung verringert.

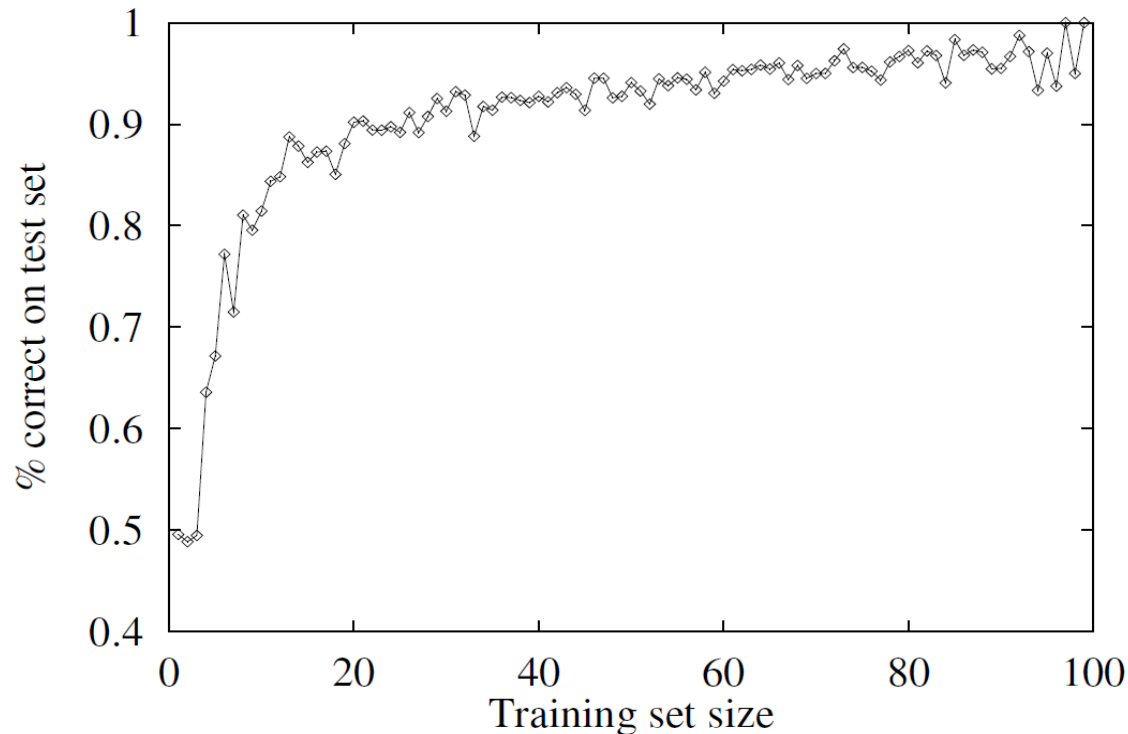
Bewertung eines Lernalgorithmus (3)

Beispiel einer 5-fachen stratifizierten Kreuzvalidierung mit Beispielmenge M , die je 10 Beispiele für zwei Klassen K_1 und K_2 zeigt:



- Iteration 1: M_1 für Validierung – M_2, M_3, M_4, M_5 für Training,
- Iteration 2: M_2 für Validierung – M_1, M_3, M_4, M_5 für Training,
- Iteration 3: M_3 für Validierung – M_1, M_2, M_4, M_5 für Training,
- Iteration 4: M_4 für Validierung – M_1, M_2, M_3, M_5 für Training,
- Iteration 5: M_5 für Validierung – M_1, M_2, M_3, M_4 für Training.

Lernkurve des Restaurantbeispiels



~ Mit wachsender Größe der Trainingsmenge, nimmt die Vorhersagekraft zu.

Anwendungsbeispiele:

- Training von Steuerungssystemen (Flugsimulator)
- Entscheidungsunterstützungssysteme
- Data Mining in Datenbanken (C 4.5)

Bewertung und Auswahl von Attributen

Bislang liegt nur eine informelle Darstellung der Heuristik zur Attributauswahl beim Lernen von Entscheidungsbäumen vor:

„Wähle für jeden Aufbauschritt des Baumes das **Attribut**, das die **größtmögliche Entscheidbarkeit** in die Trainingsmenge induziert.“

Frage: wie ist die Induktion von Entscheidbarkeit in die Trainingsmenge quantifizierbar?

Quantifizierung durch Informationstheorie (1)

Bspl. Münzwurf: Welchen Wert hat die Vorabinformation über den Ausgang eines Münzwurfs mit Einsatz 1€, Gewinn 1€ bei Kopf und Verlust 1€ bei Zahl?

Faire Münze:

⇒ Erwarteter Gewinn pro Wurf ist $= 0.5 \cdot 1€ + 0.5 \cdot (-1€) = 0.0 €$

~ Wert der Information über den Ausgang ist kleiner als 1€.

Gefälschte Münze mit 99% Kopf und 1% Zahl:

~ Erwarteter Gewinn pro Wurf ist $= 0.99 \cdot 1€ + 0.01 \cdot (-1€) = 0.98€$

~ Wert der Information über den Ausgang ist kleiner als 0.02€.

~ Je weniger man über den Ausgang weiß, desto wertvoller ist die Vorabinformation darüber, desto größer ist der Informationsbedarf.

Quantifizierung durch Informationstheorie (2)

Die Informationstheorie bewertet Information ähnlich, jedoch in Bits statt in Euro:

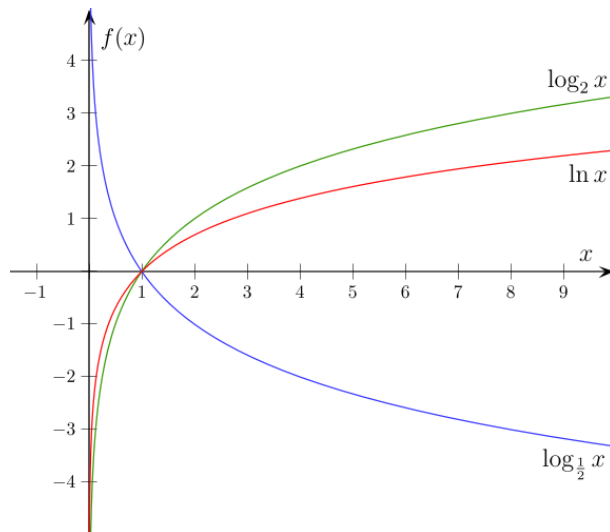
bei n möglichen (Ausgangs-)Werten v_i ($i = 1, \dots, n$)

mit entsprechenden W'keiten $P(v_i)$

ist der Informationsgehalt / bzw. **Entropie** / der aktuellen Antwort:

$$I(P(v_1), \dots, P(v_n)) = \sum_{i=1}^n P(v_i) \log_2 \left(\frac{1}{P(v_i)} \right) = \sum_{i=1}^n -P(v_i) \log_2 P(v_i)$$

mit $0 \cdot \log_2 0 = 0$.



Logarithmenkurven von
<https://de.wikipedia.org/wiki/Logarithmus> (5.6.13)

Quantifizierung durch Informationstheorie (3)

- Gegeben Informationsgehalt I bei n möglichen Ausgangswerten v_i mit entsprechenden Wahrscheinlichkeiten $P(v_i)$:

$$I(P(v_1), \dots, P(v_n)) = \sum_{i=1}^n P(v_i) \log_2 \left(\frac{1}{P(v_i)} \right) = \sum_{i=1}^n -P(v_i) \log_2 P(v_i).$$

~ Beispiel der fairen Münze:

$$I\left(\frac{1}{2}, \frac{1}{2}\right) = -\frac{1}{2} \cdot \log_2 \left(\frac{1}{2}\right) - \frac{1}{2} \cdot \log_2 \left(\frac{1}{2}\right) = \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 1 = 1.$$

~ Beispiel der unfairen Münze: $I(0.99, 0.01) = 0,01 \cdot 6,64 + 0,99 \cdot 0,015 = 0.08$.

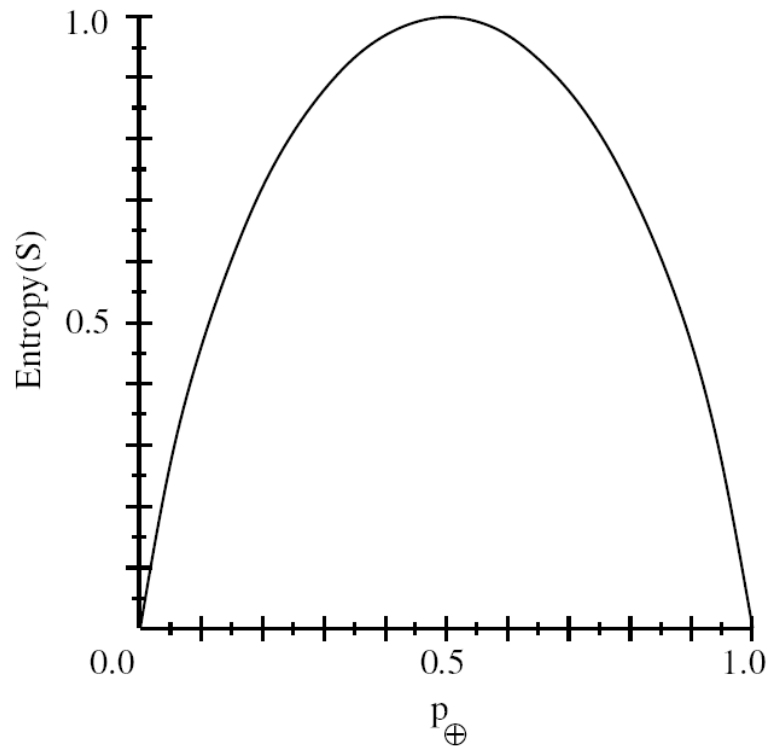
~ Beispiel eines sicheren Ausgangs: $I(1, 0) = 1 \cdot 0 + 0 = 0$.

- Also auch hier: je weniger man über den Ausgang weiß,
desto wertvoller ist die Vorabinformation darüber.

Quantifizierung durch Informationstheorie (4)

Entropieverteilung für ein Ereignis, das zwei mögliche Ausgangswerte hat:

die Entropie ist maximal, nämlich 1, wenn beide Werte gleich wahrscheinlich sind – und damit maximale Unsicherheit über den Ausgang besteht..



Attributselektion für Entscheidungsbäume (1)

Geg.: Trainingsmenge T mit p positiven Beispielen und n negativen Beispielen.

~ Information einer korrekten Antwort zu Beginn des Ent'-Baum-Lernens:

$$I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) = -\frac{p}{p+n} \log_2\left(\frac{p}{p+n}\right) - \frac{n}{p+n} \log_2\left(\frac{n}{p+n}\right). \quad (1)$$

Aufgabe: Auswahl des ersten/nächsten Attributs.

- Annahme: Ein Attribut A habe v Werte ~ A unterteilt T in v Teilmengen E_1, \dots, E_v .
Jede Teilmenge E_i habe p_i positive und n_i negative Beispiele. Also ist noch folg.
Information zur Klassifikation nötig:

$$I\left(\frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i}\right). \quad (2)$$

Zufälliges Bspl. aus T gehört zu Teilmenge E_i , $i \in \{1, \dots, v\}$, mit W'keit $\frac{p_i + n_i}{p + n}$. (3)

Attributselektion für Entscheidungsbäume (2)

~ *Verbleibender Informationsbedarf* (*Remainder*) $R(A)$ nach Auswahl von A ist (aus (2) und (3)):

$$R(A) = \sum_{i=1}^v \frac{p_i + n_i}{p + n} \cdot I\left(\frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i}\right). \quad (4)$$

~ *Informationsgewinn* (*Gain*) durch Auswahl von Attribut A ist (aus (1) und (4)):

$$Gain(A) = I\left(\frac{p}{p + n}, \frac{n}{p + n}\right) - R(A). \quad (5)$$

~ Attribut A ist von allen noch nicht im Entscheidungsbaum befindlichen Attributen so auszuwählen, dass der *Informationsgewinn* $Gain(A)$ maximiert wird.

Restaurantbeispiel: Auswahl des 1. Attributs

Vergleich der Informationsgewinne
für die Attribute *Patrons* und *Type*
als erstes Bewertungsattribut.

Beachte: Trainingsmenge T hat 6
positive und 6 negative Beispiele.

Example	Attributes										Goal	
	Alt	Bar	Fri	Hun	Pat	Price	Rain	Res	Type	Est	WillWait	
X ₁	Yes	No	No	Yes	Some	\$\$\$	No	Yes	French	0-10	Yes	
X ₂	Yes	No	No	Yes	Full	\$	No	No	Thai	30-60	No	
X ₃	No	Yes	No	No	Some	\$	No	No	Burger	0-10	Yes	
X ₄	Yes	No	Yes	Yes	Full	\$	No	No	Thai	10-30	Yes	
X ₅	Yes	No	Yes	No	Full	\$\$\$	No	Yes	French	>60	No	
X ₆	No	Yes	No	Yes	Some	\$\$	Yes	Yes	Italian	0-10	Yes	
X ₇	No	Yes	No	No	None	\$	Yes	No	Burger	0-10	No	
X ₈	No	No	No	Yes	Some	\$\$	Yes	Yes	Thai	0-10	Yes	
X ₉	No	Yes	Yes	No	Full	\$	Yes	No	Burger	>60	No	
X ₁₀	Yes	Yes	Yes	Yes	Full	\$\$\$	No	Yes	Italian	10-30	No	
X ₁₁	No	No	No	No	None	\$	No	No	Thai	0-10	No	
X ₁₂	Yes	Yes	Yes	Yes	Full	\$	No	No	Burger	30-60	Yes	

none
some
full

$$Gain(Patrons?) = 1 - \left[\frac{2}{12} I(0,1) + \frac{4}{12} I(1,0) + \frac{6}{12} I\left(\frac{2}{6}, \frac{4}{6}\right) \right] \approx 0.541.$$

French
Italian
Burger
Thai

$$Gain(Type?) = 1 - \left[\frac{2}{12} I\left(\frac{1}{2}, \frac{1}{2}\right) + \frac{2}{12} I\left(\frac{1}{2}, \frac{1}{2}\right) + \frac{4}{12} I\left(\frac{2}{4}, \frac{2}{4}\right) + \frac{4}{12} I\left(\frac{2}{4}, \frac{2}{4}\right) \right] = 0.$$

~ Also liefert *Patrons* den größeren Informationsgewinn im Vergleich zu *Type*.

Kategorien von Attributen

Folgende Kategorien werden bei Attributen unterschieden:

- **Nominale Attribute:** Reine Aufzählungen bzw. Mengen ohne Ordnung (z.B. Früchte)
- **Ordinale Attribute:** Diskrete, zuweilen endliche Mengen mit einer Ordnung (z.B. natürliche oder ganze Zahlen, Schulnoten)
- **Kontinuierliche Attribute:** Teilmengen der rationalen oder der reellen Zahlen (z. B. für Temperatur, Geschwindigkeit)

Zu den Attributentypen in Entscheidungsbäumen (1)

- Für Attribute mit geordnetem Wertebereich (also für ordinale und kontinuierliche Attribute) sind auch Tests über die Relationen „>“, „<“, „≥“ und „≤“ möglich bzw. z.T. auch nötig, da vollständige Aufzählungen aller möglicher Werte gar nicht möglich sind.
- Beim Restaurant-Beispiel wären bzgl. des ordinalen Attributs *Price* mit dem geordneten Wertebereich \$, \$\$, \$\$\$ auch Entscheidungsknoten möglich wie „*Price* ≤ \$\$“ oder „*Price* > \$“. Prinzipiell sind dann für die Suche nach möglichst kompakten Entscheidungsbäumen aber auch verschiedene binäre Aufteilungen der Trainingsmenge bzgl. solcher Attribute zu testen.

Zu den Attributentypen in Entscheidungsbäumen (2)

- Um bei der Suche nach möglichst kompakten Entscheidungsbäumen das Testen verschiedener binäre Aufteilungen der Trainingsmenge bzgl. ordinaler und kontinuierlichen Attribute mit großen oder unendlichen Wertebereichen einzugrenzen, kann man das sog. *Binning* umsetzen.
- Beim *Binning* wird der Wertebereich der Größe nach aufsteigend in Intervalle – sogenannte *bins* (engl. für *Behälter*) – eingeteilt.
- Beim Restaurant-Beispiel wurde z.B. der prinzipiell kontinuierliche und unendliche Wertebereich des Attributs *WaitEstimate* auf die vier Bins *0-10*, *10-30*, *30-60*, *>60* unterteilt, sodass ein ordinales Attribut mit 4 möglichen Wertebelegungen entstand.*

* Formal korrekter und besser für die Tests wäre natürlich eine Unterteilung in disjunkte Intervalle wie z.B. $[0,10)$, $[10,30)$, $[30,60)$, $[60,\infty)$ gewesen.

Rauschen und Überanpassung

- Was ist *Rauschen* (engl. *Noise*)?
 - ↗ Zufällige Fehler in den Lerndaten wie z.B. Beobachtungsfehler, Messfehler, Übertragungsfehler, ...
 - **Effekt:** Irreführende Stichproben, durch die größere Bäume durch Überanpassung (Overfitting) mehr Fehler auf neuen Daten machen.
- Vermeiden von Overfitting durch sog. *Kürzen des Entscheidungsbaums*:
 - Kürzen verhindert weitere Aufteilung durch Attribute, die irrelevant sind, trotz dessen die Daten im entsprechenden Knoten uneinheitlich klassifiziert sind.
 - Ein Hinweis auf Irrelevanz ist ein sehr geringer Informationsgewinn durch das Attribut.

Entscheidungswälder

- Das Konzept der Entscheidungsbäume lässt sich im Rahmen eines Ensemble-Ansatzes (s. folg. Vorlesung) auf sog. Entscheidungswälder erweitern.
- Diese finden z.B. im Bildverstehen/Computersehen zahlreiche Anwendungen.
- Referenz: A. Criminisi, J. Shotton, and E. Konukoglu: *Decision Forests for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning*, 28 October 2011,
<http://research.microsoft.com/apps/pubs/default.aspx?id=155552> (26.5.14).

Zusammenfassung: Lernen & Entscheidungsbäume

- Lernverfahren lassen sich entsprechend der Rückkopplung unterteilen:
 - überwachte Lernverfahren
 - unüberwachte Lernverfahren
 - verstärkende Lernverfahren
- Entscheidungsbäume
 - sind eine Möglichkeit zum überwachten Lernen,
 - sind eine Möglichkeit, Boolesche Funktionen zu repräsentieren,
 - können in der Größe exponentiell in der Anzahl der Attribute sein.
 - Es ist oft schwierig, den minimalen Entscheidungsbaum zu finden.
 - Eine Methode zur Generierung von möglichst flachen Entscheidungsbäumen beruht auf der Gewichtung der Attribute.