

# Grundlagen der Künstlichen Intelligenz

## **13 Maschinelles Lernen**

---

Lernen durch Beobachtung, Entscheidungsbäume

*Volker Steinhage*

# Inhalt

---

- Der lernende Agent
- Induktives Lernen
- Lernen von Entscheidungsbäumen

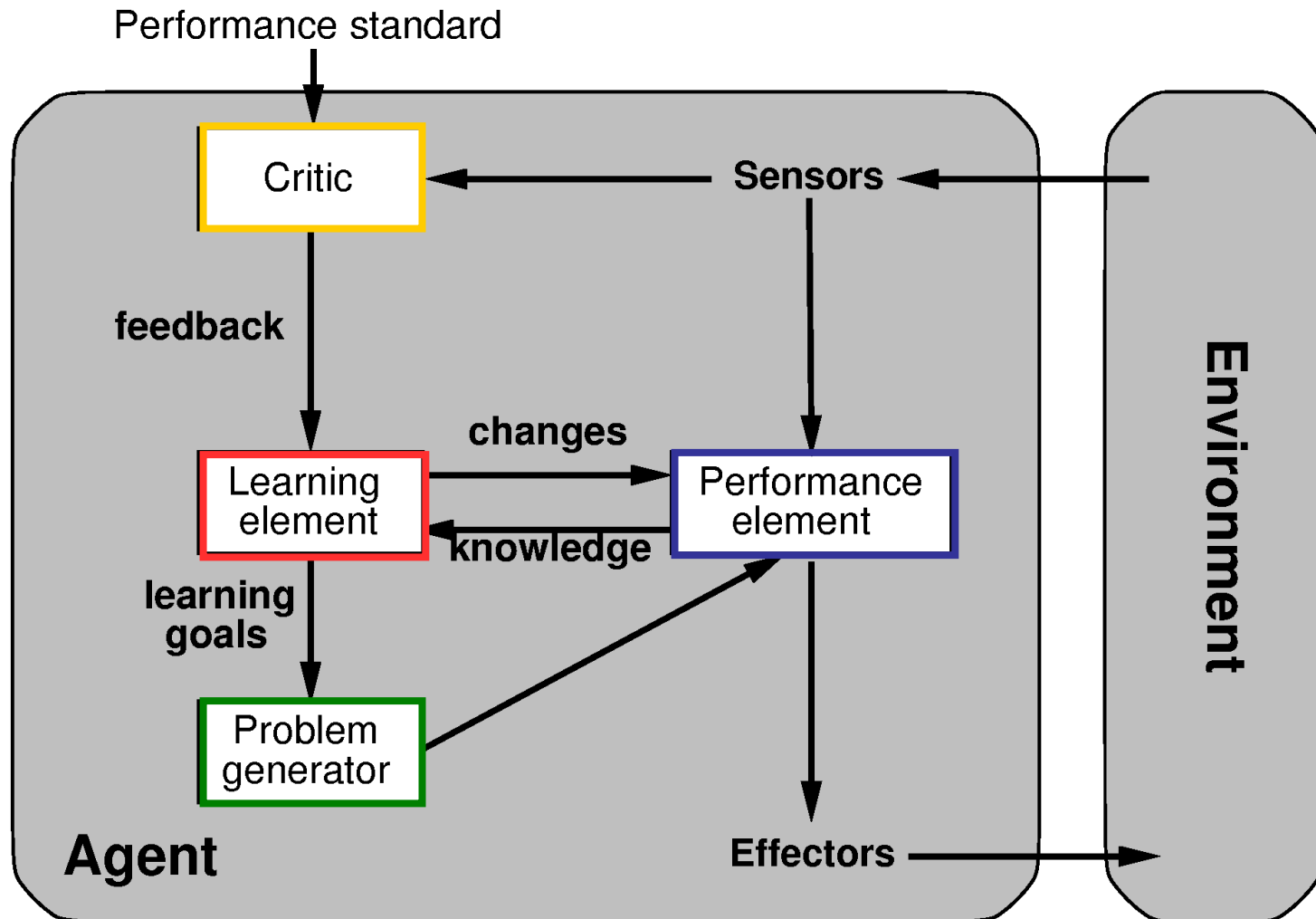
- Was ist *Lernen* im Agentenkontext?

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

- 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

Aus der zweiten Vorlesung ist die Struktur des lernenden Agenten bekannt:



# Bausteine des lernenden Agenten

**Performance-Element:** Verarbeitet Wahrnehmungen und wählt Aktionen aus

→ entspricht einem der bisherigen Agentenmodelle

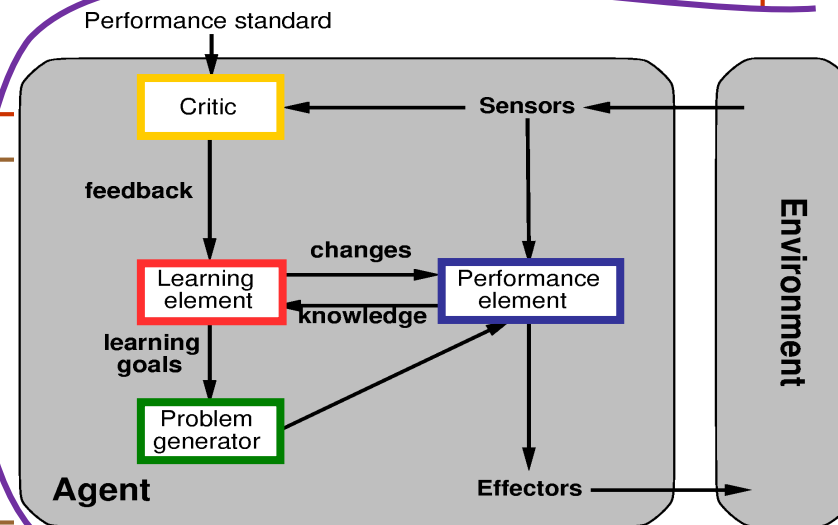
**Learning-Element:** Durchführen von Verbesserungen

→ 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)*

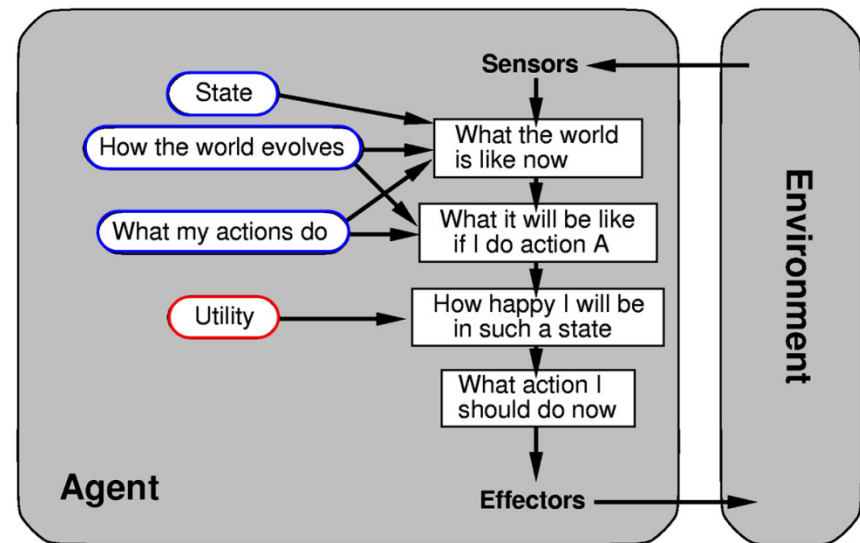
**Problem-Generator:** Vorschlagen von *explorativen* Aktionen, die den Agenten zu neuen Erfahrungen führen



# 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 über Perzepte
- **Ausgabe**: Effekte der Aktionen des Agenten



Betrachtung der **Effekte**:

- Effekte, die der Agent durch sein Handeln erzielen soll (**Ideal**)
- Effekte, die tatsächlich eintreten (**Realität**)

**können sich unterscheiden**



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

# Induktives Lernen

---

In dieser und den folg. Vorlesungseinheiten werden Formen des induktiven Lernens behandelt

- Induktives Lernen:

Eingabe: einzelne Beispielinstanzen (Stichproben)

Ausgabe: allgemeine Regeln, Funktionen, etc.

Kurz: Generalisierung vom Besonderen zum Allgemeinen

- Deduktives Lernen:

Eingabe: allgemeine Regeln, Sätze, ...

Ausgabe: spezialisierte/instanziierte Regeln, Sätze, ...

Kurz: Spezialisierung vom Allgemeinen zum Besonderen



# 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 ob 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
  - Eingabe: Tupel der Art [Wochentag, Uhrzeit, Fahrdistanz, Fahrdauer]
  - Ziel: Der Agent lernt aus Korrelationen Konzepte für Haupt-, Normal- und Schwachverkehrszeiten – ohne jemals explizit als solche bezeichnete Beispiele gesehen zu haben

# Form der Rückkopplung: Verstärkendes Lernen

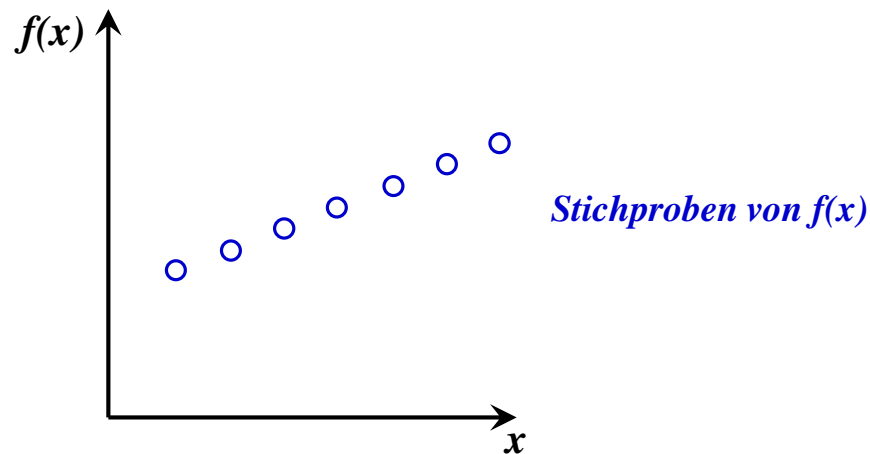
---

## *Verstärkendes Lernen (Reinforcement Learning):*

- Die Trainingsmenge  $T$  enthält Paare aus Eingabe- und Verstärkungswerten
- Die Rückkopplung ist also eine Verstärkung
- Eine Verstärkungen ist eine Belohnung oder ein Bestrafung, die mit dem in der Eingabe kodierten Zustand oder kodierten Aktion einher geht
- Bspl.: Taxifahrer-Agent
  - Die Höhe des Trinkgeldes sei die Verstärkung zu jedem Eingabetupel der Art [Wochentag, Uhrzeit, Fahrdistanz, Fahrdauer, Verhalten] zu einer Taxifahrt.
  - ↗ Der Agent kann so z.B. lernen, Fahrten zu bestimmten Zeiten zu bevorzugen, sein Verhalten gegenüber dem Fahrgast zu ändern, ...

# Start: Überwachtes Lernen (1)

- Ein *überwachtes* Lernverfahren schätzt eine unbekannte Funktion  $f$  aus einer Trainingsmenge  $T$  von Ein-/Ausgabepaaren  $(x_i, f(x_i))$  \*
- Jedes Paar  $(x_i, f(x_i))$  wird als *Beispiel* oder *Stichprobe* bezeichnet
- *Umsetzung*:  
Eingabe: Trainingsmenge  $T$  von Stichproben  $(x_i, f(x_i))$  der unbekannt. Funktion  $f$
- Ausgabe: eine *Hypothese*  $h$ , die  $f$  approximiert

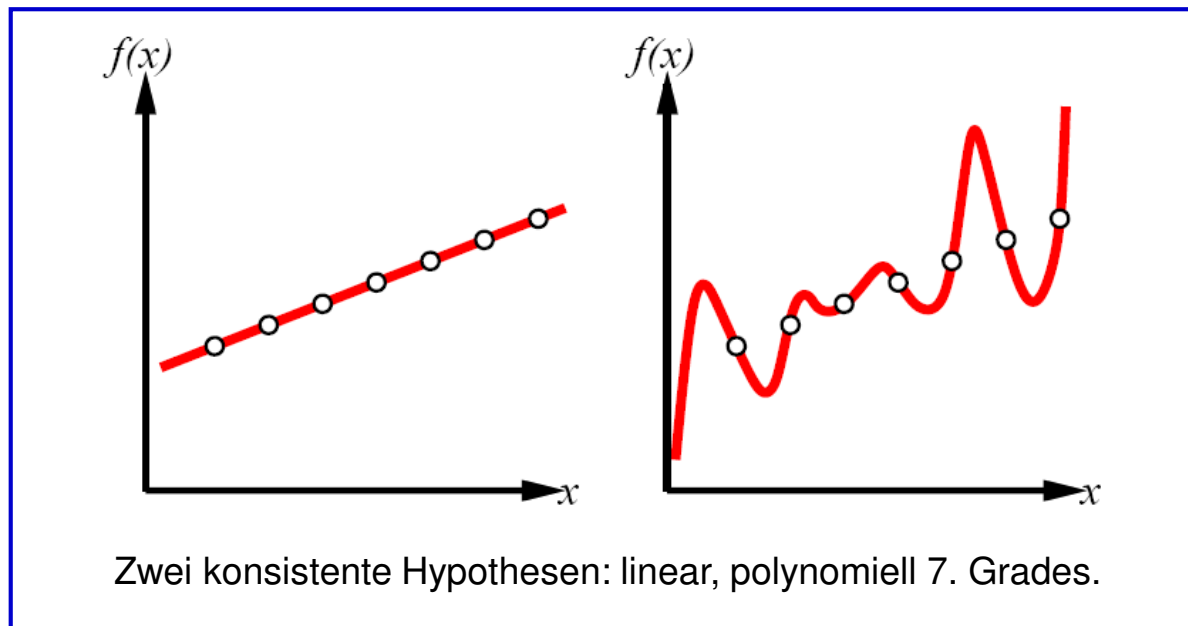


---

\* 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*<sup>(1)</sup> (*Occam's Razor*): von mehreren konsistenten Alternativen ist die *einfachste* Hypothese auszusuchen

<sup>(1)</sup> nach dem Theologen und Logiker *William von Ockham*  
(\* 1285 Ockham (Grafschaft Surrey), † 9.4.1347 München)

# Entscheidungsbäume

---

- **Eingabe:** Beschreibung einer Situation durch eine Menge von Eigenschaften bzw. Attributen (entsprechen *Grundliterals* in FOL)
- **Ausgabe:** Ja/Nein-Entscheidung bezüglich eines *Booleschen Zielprädikats*
- **Repräsentationsmächtigkeit:** *Boolesche Funktionen* über Grundliterals in FOL
- **Aufbau:**
  - jeder **innere 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)

---

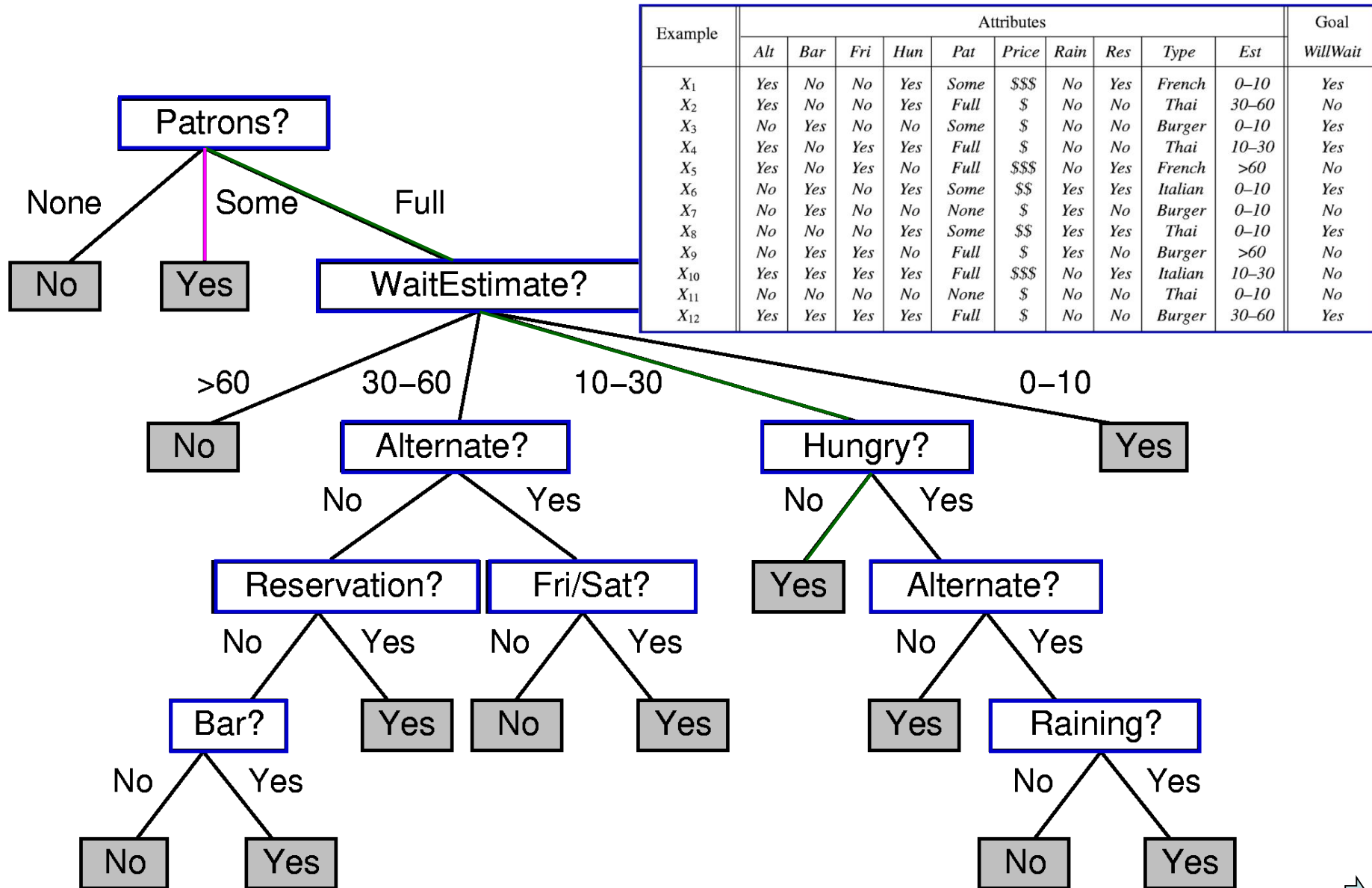
## Beschreibende Attribute:

- *Patrons*: Wieviele Gäste sind da? (None, Some, Full)
- *WaitEstimate*: Wie lange vorauss. 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)



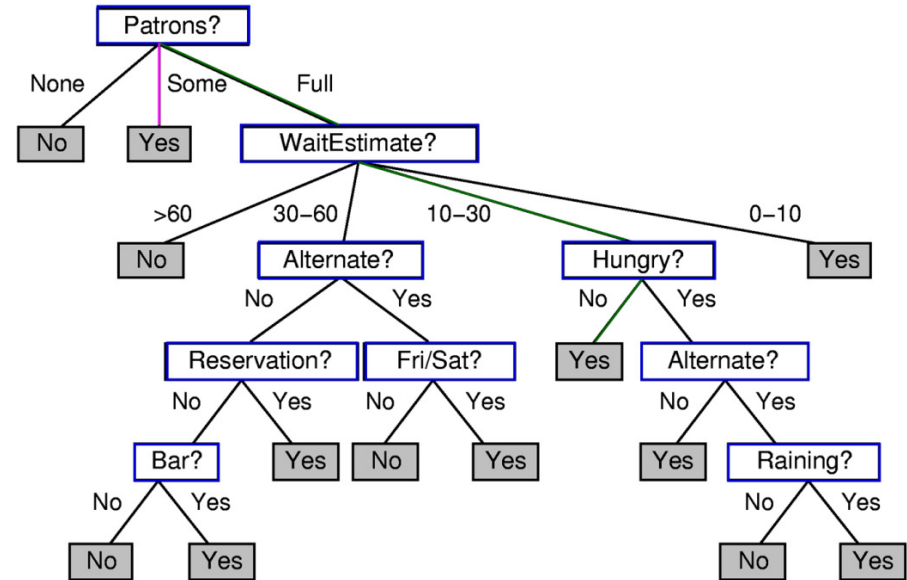
Ein konstruierter Entscheidungsbaum von Stuart Russel



# Repräsentation des Zielprädikats

Ein Entscheidungsbaum ist als **Konjunktion von Implikationen** und damit als KNF darstellbar.

Die Implikationen bzw. Disjunktionen in KNF entsprechen den Pfaden, die in YES-Knoten enden:



$$\forall r \{$$

$$\quad [ \textit{Patrons}(r, \textit{Some}) \Rightarrow \textit{WillWait}(r) ]$$

$$\quad \wedge \dots$$

$$\quad \wedge [ \textit{Patrons}(r, \textit{Full}) \wedge \textit{WaitEstimate}(r, 10-30) \wedge \textit{Hungry}(r, \textit{No}) \Rightarrow \textit{WillWait}(r) ]$$

$$\quad \wedge \dots$$

$$\}$$

$$\{$$

$$\quad [ \overline{\textit{Patrons}(r, \textit{Some})}, \textit{WillWait}(r) ],$$

$$\quad \dots$$

$$\quad [ \overline{\textit{Patrons}(r, \textit{Full})}, \overline{\textit{WaitEstimate}(r, 10-30)}, \overline{\textit{Hungry}(r, \textit{No})}, \textit{WillWait}(r) ],$$

$$\quad \dots$$

$$\}$$

# Ausdruckskraft von Entscheidungsbäumen

---

Die Sprache von klassischen Entscheidungsbäumen ist inhärent aussagenlogisch (bzw. die über Grundliteralen in FOL):

**Theorem 1:** *Alle aussagenlogischen Formeln sind mit Entscheidungsbäumen darstellbar*

# Kompakte Repräsentationen

---

- 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 Entscheidungsbaumlernens ist die Ableitung kompakter Entscheidungsäume
- Aber es gibt Boolesche Funktionen, die einen Baum exponentieller Größe erfordern:
  - **Parity Funktion:** 
$$p(x) = \begin{cases} 1 & \text{geradeAnzahl von Eingaben} \\ 0 & \text{sonst} \end{cases}$$
  - Es gibt keine kompakte Repräsentation für *alle* Booleschen Funktionen

# Restaurantbeispiel: Lernen aus Trainingsmenge

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>&gt;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>&gt;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>

# Der triviale Entscheidungsbaum

---

Der triviale Entscheidungsbaum:

- ein Pfad für jedes Beispiel
- memorisiert lediglich die Beispiele der Trainingsmenge

- ~ keine kompakte Repräsentation
- ~ keine Extraktion eines allgemeinen Musters  
(keine Generalisierung)
- ~ keine Vorhersagekraft

# Kompakte Entscheidungsbäume

---

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

**Aber:** das Erzeugen des kleinsten Entscheidungsbaums ist nicht handhabbar

**Daher:** Einsatz von Heuristiken, die zu einer *kleinen* Menge von Tests führen

# Gewichtung von Attributen (informell)

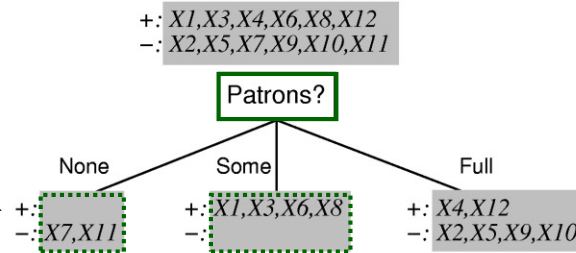
Heuristik: **Wähle** in jedem Aufbauschritt des Baumes das **Attribut**, das den **maximalen Informationsgewinn** für die richtige Klassifikation der Trainingsbeispiele liefert.

Bspl.:

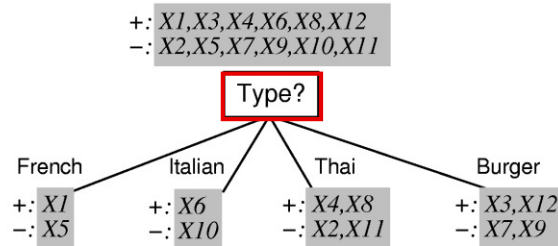
Example	Attributes										Goal	
	Alt	Bar	Fri	Hun	Pat	Price	Rain	Res	Type	Est	WillWait	
X <sub>1</sub>	Yes	No	No	Yes	Some	\$\$\$	No	Yes	French	0-10	Yes	
X <sub>2</sub>	Yes	No	No	Yes	Full	\$	No	No	Thai	30-60	No	
X <sub>3</sub>	No	Yes	No	No	Some	\$	No	No	Burger	0-10	Yes	
X <sub>4</sub>	Yes	No	Yes	Yes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

(a)

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

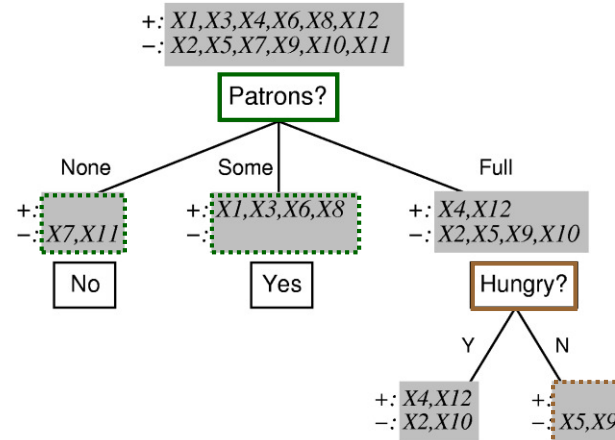


(b)



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

(c)



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

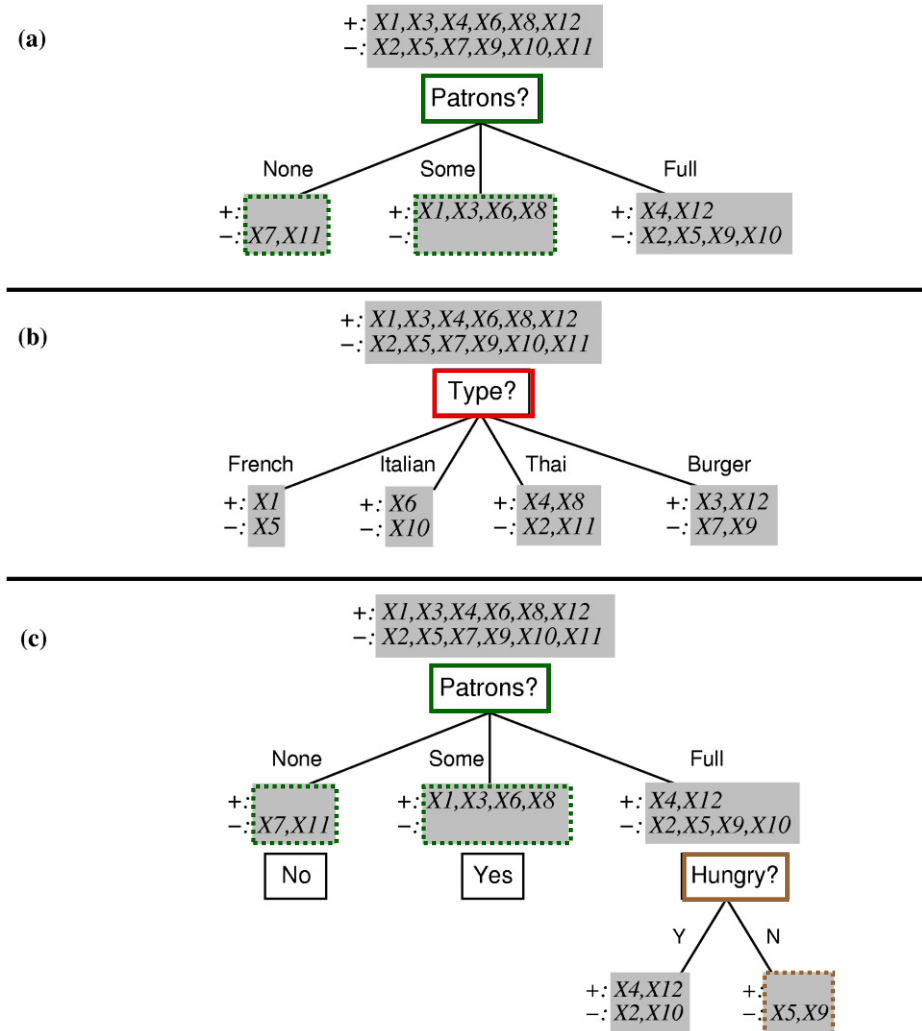


# Rekursives Lernverfahren (1)

Der Aufbau eines Entscheidungsbaumes erfolgt rekursiv:

Nach jeder Auswahl eines inneren Entscheidungsknotens (Testknotens)

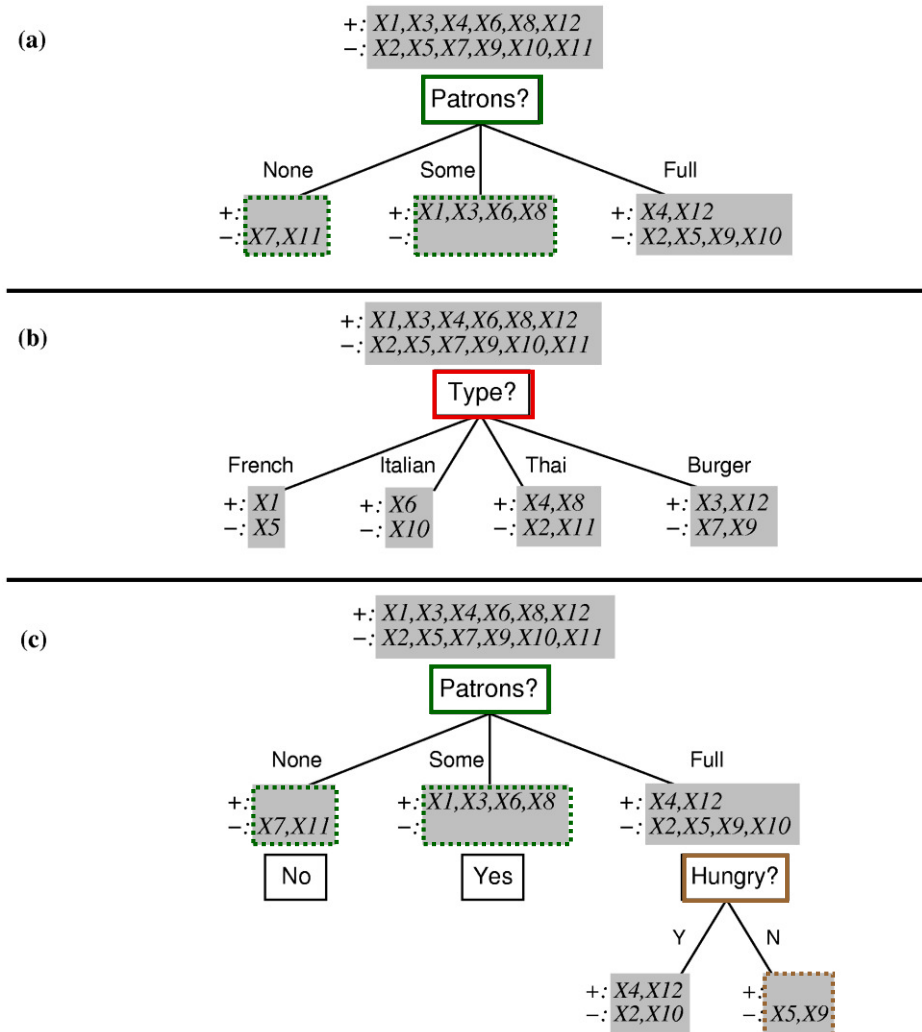
- liegt ein neues Entscheidungsbaum-Lernproblem vor
- mit weniger Beispielen, die noch nicht klassifizierbar sind
- mit reduzierter Zahl von noch nicht verwendeten Testattributen



# Rekursives Lernverfahren (2)

Für jeden Aufbauschritt können die vorläufigen Blattknoten vier Fälle zeigen:

- 1) **Positive und negative Beispiele**: wähle neues Attribut
- 2) **Nur positive oder nur negative Beispiele**: terminaler Blattknoten
- 3) **Keine Attribute** mehr, aber noch Beispiele mit unterschiedlicher 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<sub>i</sub>* ← {elements of *examples* with *best* =  $v_i$ }

*subtree* ← DECISION-TREE-LEARNING(*examples<sub>i</sub>*, *attributes* – *best*,

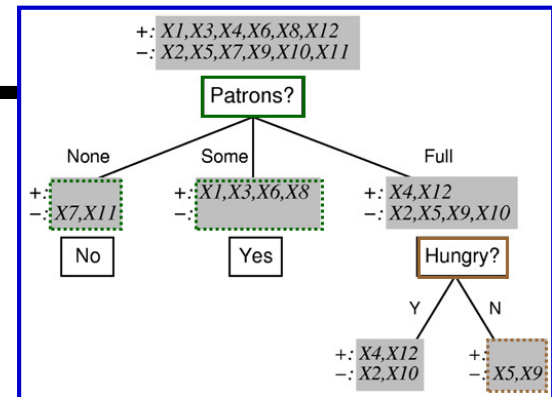
MAJORITY-VALUE(*examples*))

Majoritäts-  
information  
zu Fall 4

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

**end**

**return** *tree*



# Bewertung und Auswahl von Attributen

---

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

„ Wähle in jedem Aufbauschritt des Baumes das Attribut,  
das den maximalen Informationsgewinn  
für die richtige Klassifikation der Trainingsbeispiele liefert.“

Frage: wie ist dieser maximale Informationsgewinn zu ermitteln?

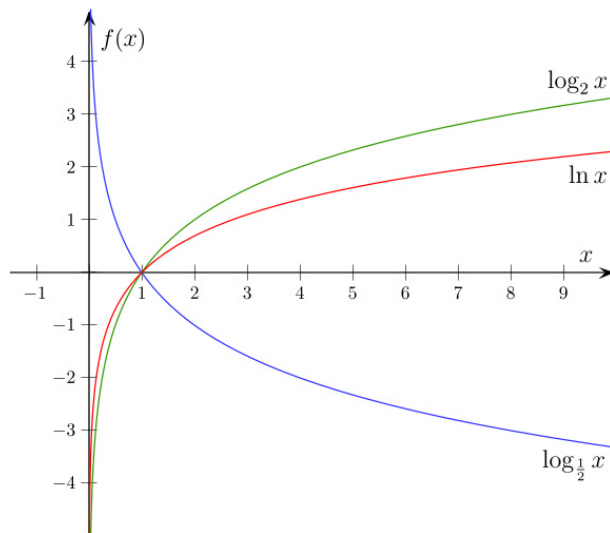
# Quantifizierung durch Informationstheorie (1)

Die Entropie  $H$  ist ein Maß für die Unsicherheit einer Zufallsvariable:

Bei  $n$  möglichen (Ausgangs-)Werten  $v_i$  ( $i = 1, \dots, n$ )  
mit entsprechenden W'keiten  $P(v_i)$  ist die Entropie  $H$ :

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

mit  $0 \cdot \log_2 0 = 0$



Logarithmenkurven von  
<https://de.wikipedia.org/wiki/Logarithmus> (6.6.18)

# Quantifizierung durch Informationstheorie (2)

---

Bspl. Münzwurf:

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

mit  $0 \cdot \log_2 0 = 0$

**Gezinkte Münze** mit  $P(\text{Kopf}) = 100\%$  und  $P(\text{Zahl}) = 0\%$

~ minimale Unsicherheit ~  $H(1,0) = -1 \cdot 0 + 0 \cdot 0 = 0$

**Faire Münze** mit  $P(\text{Kopf}) = 50\%$  und  $P(\text{Zahl}) = 50\%$

~ maximale Unsicherheit ~  $H(1/2, 1/2) = -0.5 \cdot -1 + -0.5 \cdot -1 = 1$

**Gezinkte Münze** mit  $P(\text{Kopf}) = 99\%$  und  $P(\text{Zahl}) = 1\%$

~ geringe Unsicherheit ~  $H(0.99, 0.01) = -0,01 \cdot -6,64 + -0,99 \cdot -0,015 = 0.08$

# Attributselektion für Entscheidungsbäume (1)

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

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

~ Entropie zu Beginn des Ent'-Baum-Lernens:

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

Aufgabe: Auswahl des ersten/nächsten Attributs.

- Annahme: Ein Attribut  $A$  habe  $v$  Werte ~  $A$  unterteilt  $T$  in  $v$  Teilmengen  $T_1, \dots, T_v$ . Jede Teilmenge  $T_i$  habe  $p_i$  positive und  $n_i$  negative Beispiele. Für jede Teilmenge  $T_i$  liegt noch folg. Entropie vor:

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

Zufälliges Bspl. aus  $T$  gehört zu Teilmenge  $T_i$ ,  $i \in \{1, \dots, v\}$ , 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)):

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

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

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

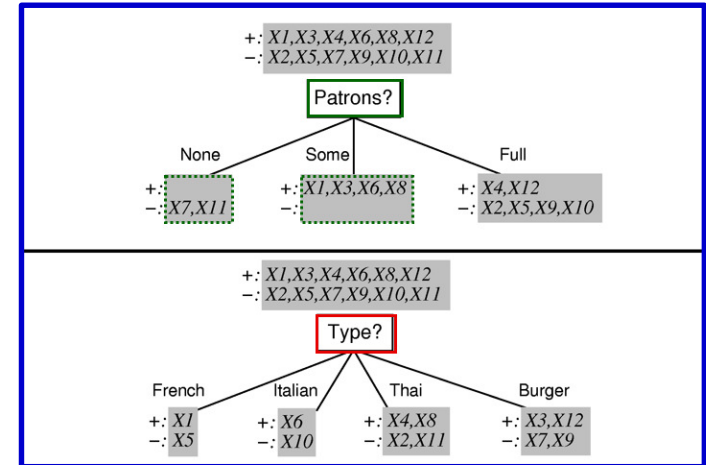
~ Wähle **Attribut  $A$**  aus allen noch nicht im Entscheidungsbaum befindlichen Attributen so, dass der *Informationsgewinn*  $Gain(A)$  maximiert wird.



# 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.



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

Labels above the terms: *none*, *some*, *full*

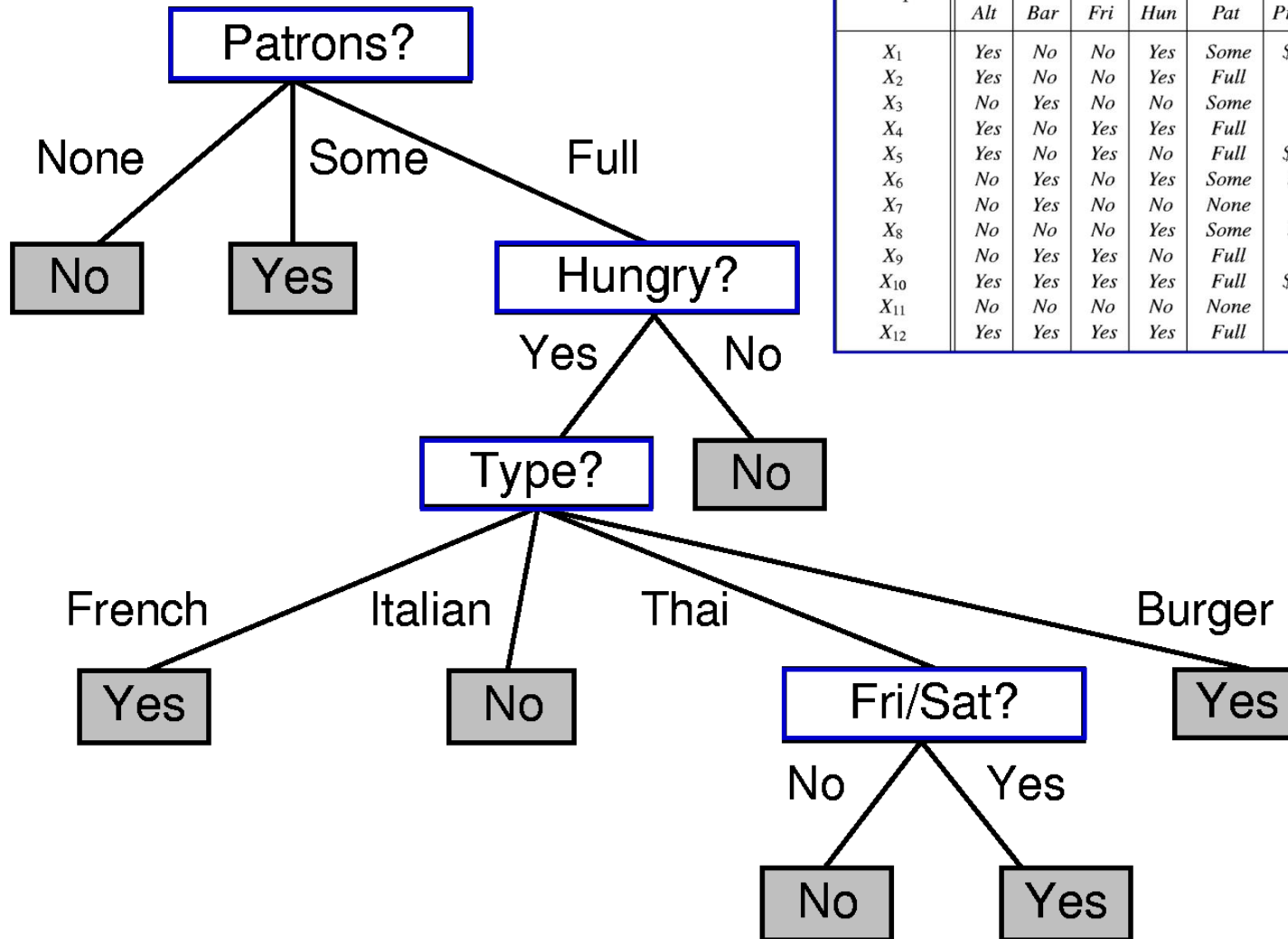
$$Gain(Type?) = 1 - \left[ \frac{2}{12} H\left(\frac{1}{2}, \frac{1}{2}\right) + \frac{2}{12} H\left(\frac{1}{2}, \frac{1}{2}\right) + \frac{4}{12} H\left(\frac{2}{4}, \frac{2}{4}\right) + \frac{4}{12} H\left(\frac{2}{4}, \frac{2}{4}\right) \right] = 0$$

Labels above the terms: *French*, *Italian*, *Burger*, *Thai*

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

# Anwendung auf die Restaurant-Daten

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



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

# Bewertung eines Lernalgorithmus: Trainings- und Testmenge

---

Ansatz zur Beurteilung der Vorhersagekraft:

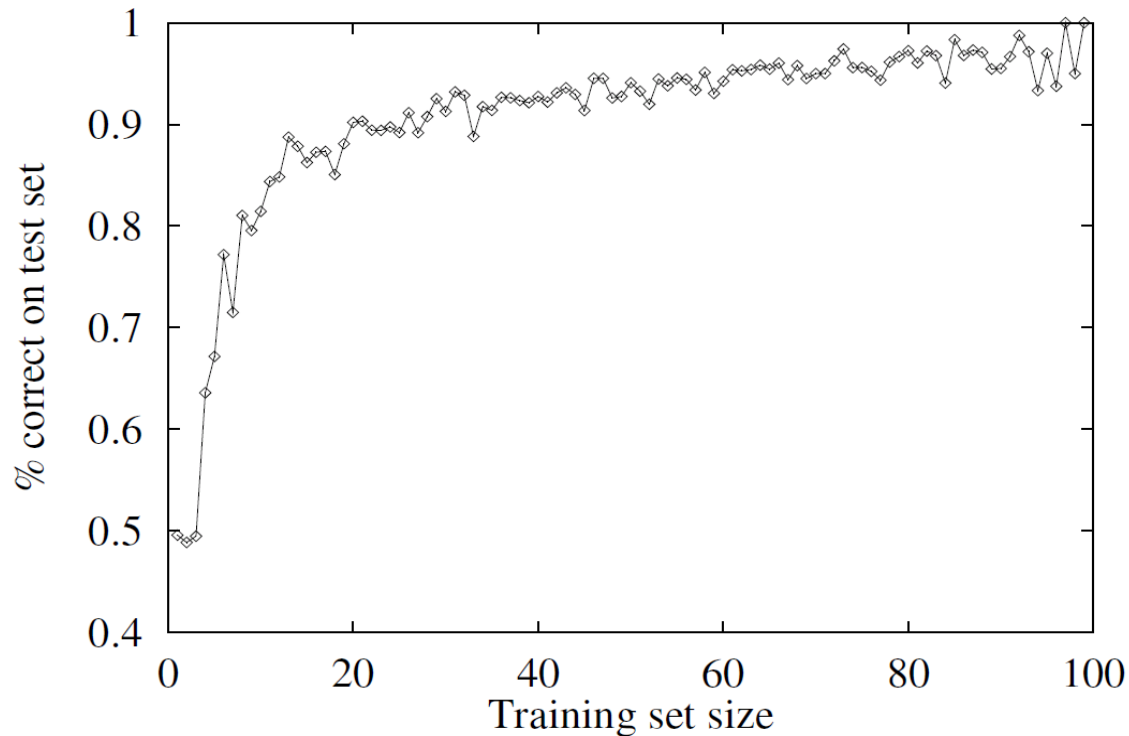
- Sammle eine große Menge  $M$  von Stichproben
- Unterteile  $M$  in zwei *disjunkte* Mengen: Trainingsmenge  $T$  und Testmenge  $E^*$
- Benutze Trainingsmenge  $T$ , um Hypothese  $h$  zu lernen
- Nutze Testmenge  $E$  für Evaluierung um den Anteil korrekt klassifizierter Stichproben zu messen.
- Wiederhole das Verfahren für zufällig gewählte Trainingsmengen unterschiedlicher Größe

Bei überwachtem Lernen  
wie hier: jede Stichprobe  
= Ein-/Ausgabepaar

---

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

# Lernkurve des Restaurantbeispiels



Fragen:

- 1) Die Vorhersagekraft nimmt hier mit wachsender Größe der Trainingsmenge zu
  - ~ Welche Größe der Trainingsmenge ist hinreichend für „gutes Lernen“
  - ~ Nächste Vorlesung
- 2) Optimierung von Lernverfahren? ~ s. Folgefolien
- 3) Gibt es weitere Bewertungsmaße für Lernverfahren? ~ s. Folgefolien

# Bewertung eines Lernalgm.: Trainings-, Validierungs- und Testmenge

---

Ansatz zur Beurteilung der Vorhersagekraft:

- Sammle eine große Menge  $M$  von Stichproben
- Unterteile  $M$  in drei *disjunkte* Mengen: Trainingsmenge  $T$ ,  
Validierungsmenge  $V$  und  
Testmenge  $E$
- Benutze Trainingsmenge  $T$ , um *Hypothese*  $h$  zu lernen
- Nutze Validierungsmenge  $V$  um *Hyperparameter des Lernalgorithmus zu optimieren* (z.B. Anzahl und Größen der Layer in KNNs, Zahl der Entscheidungsbäume in Entscheidungswäldern, etc.)
- Nutze Testmenge  $E$  für Evaluierung um den Anteil korrekt klassifizierter Stichproben zu messen

Bei überwachtem Lernen  
wie hier: jede Stichprobe  
= Ein-/Ausgabepaar

# Bewertung eines Lernalgorithmus: Kreuzvalidierung

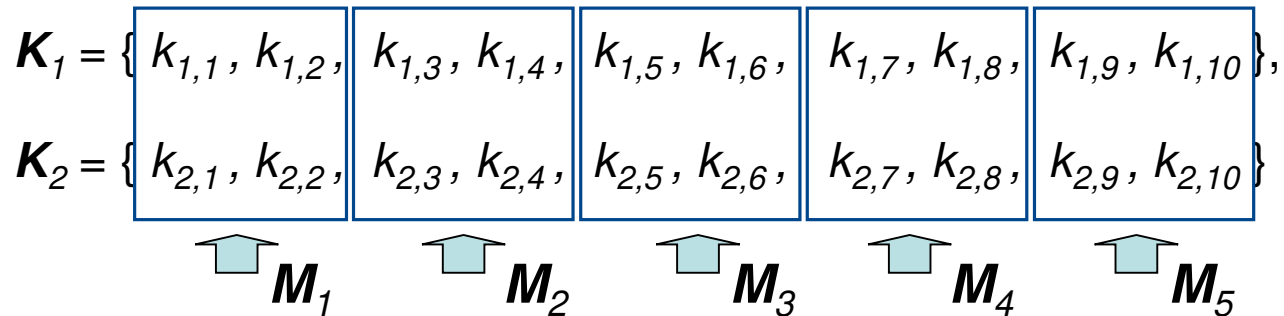
---

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 getestet, nachdem es auf den jeweils restlichen  $k-1$  Teilmengen *neu* 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: Bspl. Kreuzvalidierung

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



- Iteration 1:  $M_1$  für Testen –  $M_2, M_3, M_4, M_5$  für Training
  - Iteration 2:  $M_2$  für Testen –  $M_1, M_3, M_4, M_5$  für Training
  - Iteration 3:  $M_3$  für Testen –  $M_1, M_2, M_4, M_5$  für Training
  - Iteration 4:  $M_4$  für Testen –  $M_1, M_2, M_3, M_5$  für Training
  - Iteration 5:  $M_5$  für Testen –  $M_1, M_2, M_3, M_4$  für Training
- Ergebnis: gemittelter Anteil korrekt klassifizierter Beispiele in jeweil. Testmengen

# Bewertungsmaße für Lernalgorithmen: Konfusionsmatrix (1)

---

Ein trainierter Lernalgorithmus  $L$  kann vier mögliche Ergebnisse für jede Stichprobe  $S$  der Testmenge  $E$  liefern:

- 1)  $S$  ist tatsächlich in Klasse  $C$  und  $L$  ordnet  $S$  auch Klasse  $C$  zu  
    ~ richtig-positiver Fall (*true positive*)
- 2)  $S$  ist tatsächlich nicht in Klasse  $C$ , aber  $L$  ordnet  $S$  der Klasse  $C$  zu  
    ~ falsch-positiver Fall (*false positive*)
- 3)  $S$  ist tatsächlich nicht in Klasse  $C$  und  $L$  ordnet  $S$  auch nicht Klasse  $C$  zu  
    ~ richtig-negativer Fall (*true negative*)
- 4)  $S$  ist tatsächlich in Klasse  $C$ , aber  $L$  ordnet  $S$  nicht der Klasse  $C$  zu  
    ~ falsch-negativer Fall (*false negative*)



# Bewertungsmaße für Lernalgorithmen: Konfusionsmatrix (2)

Die Häufigkeiten der genannten vier Fälle werden in eine sog. **Konfusionsmatrix** eingetragen:

		class $C$ predicted by classifier	
		true	false
actual class is $C$	true	true positive ↓ (correct) hits	false negative ↓ misses
	false	false positive ↓ false hits	true negative ↓ correct rejections

Aus der Konfusionsmatrix lassen sich verschiedene **Bewertungsmaße** ableiten.

# Bewertungsmaße für Lernalgorithmen: Recall

---

**Recall** (Trefferquote) eines Lernalgorithmus  $L$  ist definiert als:

$$\frac{\#true\ positives}{\#true\ positives + \#false\ negatives}$$

- Hoher *Recall* →  $L$  erkennt die meisten Stichproben  $S$  aus Klasse  $C$  richtig
- *Recall* = 1 →  $L$  erkennt alle Stichproben  $S$  aus Klasse  $C$  richtig
- Aber keine Aussage zu den Stichproben  $S$ , die Klasse  $C$  fälschlicherweise zugeordnet werden

# Bewertungsmaße für Lernalgorithmen: Precision

---

**Precision** (Genauigkeit) eines Lernalgorithmus  $L$  ist definiert als:

$$\frac{\#true\ positives}{\#true\ positives + \#false\ positives}$$

- Hohe *Precision* →  $L$  ordnet Klasse  $C$  mehr richtige Stichproben (aus  $C$ ) als falsche Stichproben (nicht aus  $C$ ) zu
- *Precision* = 1 →  $L$  ordnet Klasse  $C$  nur Stichproben  $S$  aus Klasse  $C$  zu
- Aber keine Aussage zu den Stichproben  $S$  aus Klasse  $C$ , die fälschlicherweise  $C$  nicht zugeordnet werden

# Bewertungsmaße für Lernalgorithmen: F-Score und Accuracy

---

**F Score** (F-Maß) ist als Kombination von Precision und Recall mittels des gewichteten harmonischen Mittels definiert:

$$F = 2 \cdot \frac{\textit{precision} \cdot \textit{recall}}{\textit{precision} + \textit{recall}}$$

**Accuracy** (Korrektklassifikationsrate) eines Lernalgorithmus  $L$  wird vielen Evaluierungsgrafiken des Buchs von Russel und Norvig benutzt:

$$\frac{\# \textit{true positives} + \# \textit{true negatives}}{\# \textit{true positives} + \# \textit{true negatives} + \# \textit{false positives} + \# \textit{false negatives}}$$

# Zusammenfassung

---

- Lernverfahren wurden entsprechend der Rückkopplung unterteilt in
  - 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
- Bewertung von Lernalgorithmen