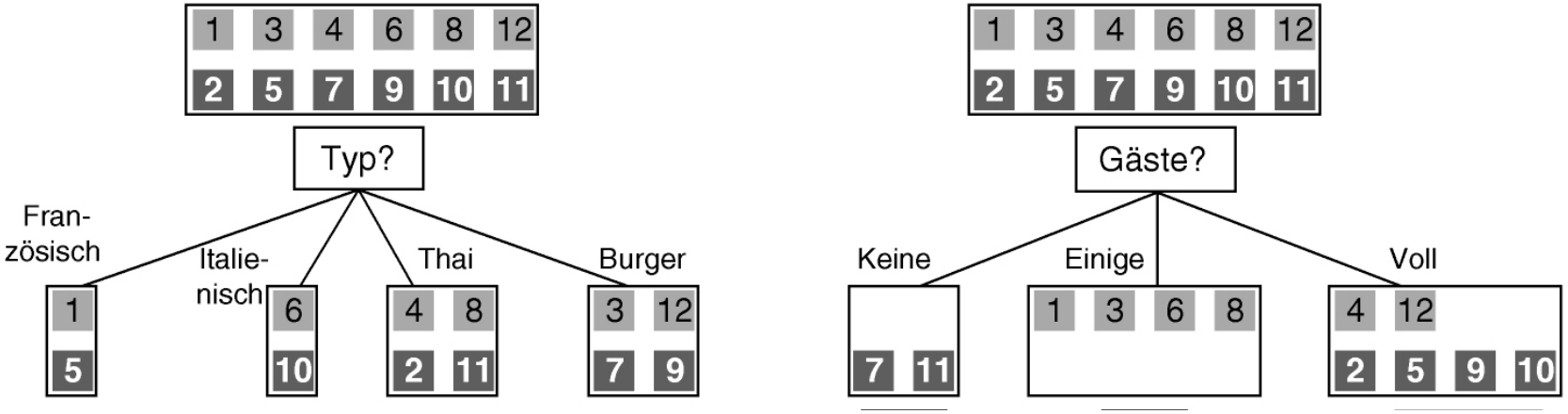
Information

Grundidee: Wähle das Attribut, das die danach erforderliche **Information** minimiert!



Für Typ? noch 12, für Gäste? nur 6 Lernbeispiele „offen“

Informationsmaß (Shannon/Weaver, 1949):

1 **bit** ist das Maß an Information, das dem Ergebnis eines binären Zufallsexperiments entspricht (W‘keitsvert. ⟨0.5, 0.5⟩)

Informationsgehalt (intuitiv: „Maß an Überraschung“)

Gegeben eine W‘keitsverteilung $\mathbf{P}(V)=\langle P(v_1),\ldots,P(v_n)\rangle$.

Informationsgehalt (auch: **Entropie**) $I(\mathbf{P}(V)) := \sum_{i=1}^{n} -P(v_i)\log_2 P(v_i)$

Beispiel: Information aus einem Münzwurf:

fair: $I(\langle 0.5, 0.5\rangle) = -0.5\log_2(1/2) - 0.5\log_2(1/2) = 1$ [bit]

unfair: $I(\langle 0.25, 0.75\rangle) = -0.25\log_2(1/4) - 0.75\log_2(0.75)$

$\approx 0.5 + 0.31 = 0.81$ [bit]

„doof“: $I(\langle 0,1\rangle) = -0\cdot\log_2(0) - 1\cdot\log_2(1) = 0$ [bit]

Beispiel: Information aus Würfeln:

fair: $I(\langle 1/6, \ldots, 1/6\rangle) = 6*(-1/6*\log_2(1/6) = -\log_2(1/6) \approx 2.585$ [bit]

Information pro Attribut (in DTL)

Für Entscheidungsbaumlernen (z.B. binär):

p Beispiele „Ja“, n Beispiele „Nein“

Die Entscheidung entspricht dem Informationsgehalt von

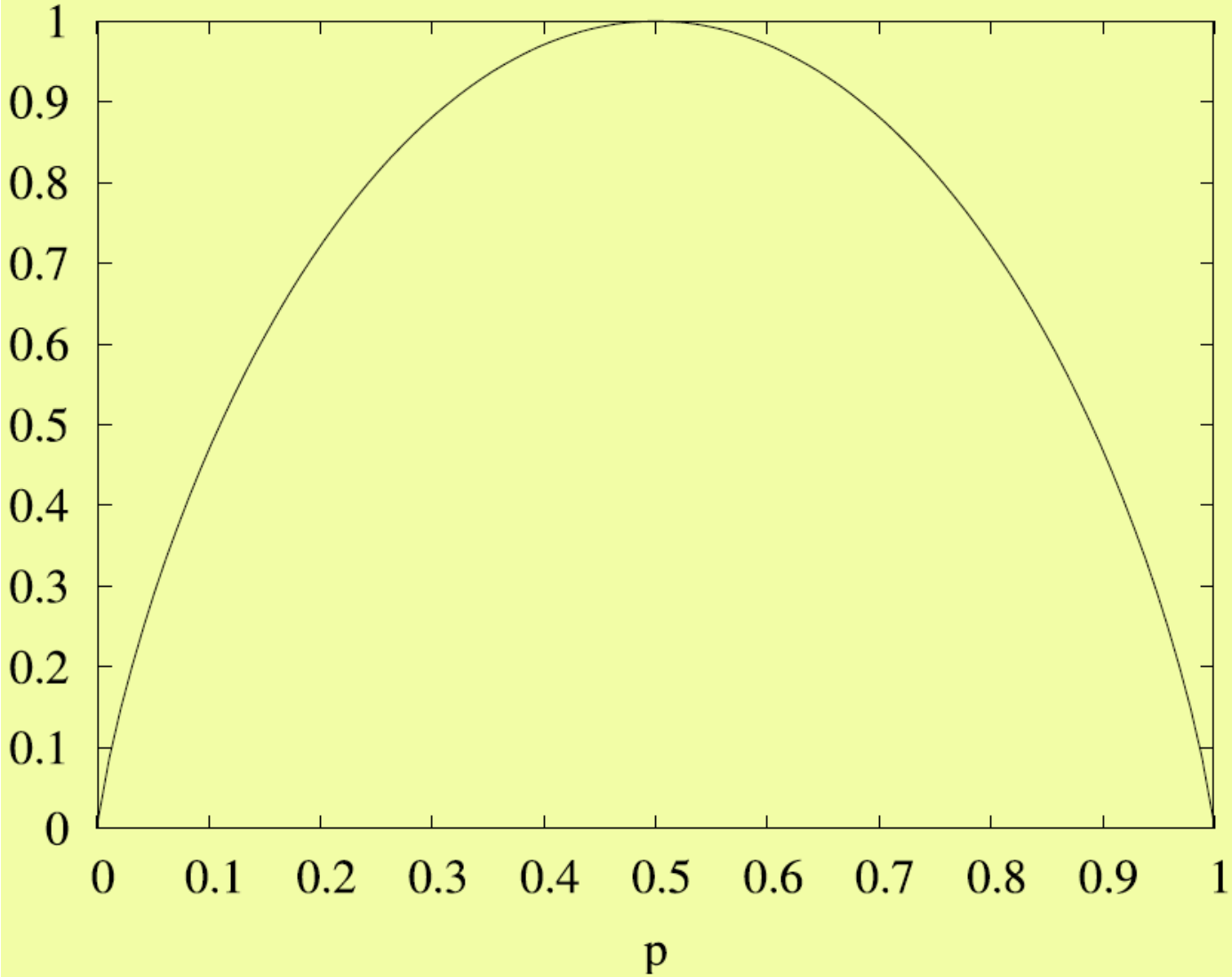
$$I(\langle p/(p+n), n/(p+n)\rangle) = -p/(p+n)\log_2(p/(p+n)) - n/(p+n)\log_2(n/(p+n)) \text{ bit!}$$

Restaurantbeispiel: $p=n=6$, also $I(\langle 0.5, 0.5\rangle)=1$ bit

Jedes (beliebige) Attribut ergibt hier maximal 1 bit Information!

Informationsgehalt für den binären Fall

$$I(\langle p, 1-p \rangle)$$



Maximal informative Attributauswahl

Attribut A mit v Werten zerlegt die Lernbeispiele in $E_1, \ldots, E_v$, jedes E_i hat p_i positive und n_i negative Lernbeispiele

Beliebig gewähltes Lernbeispiel trägt i-ten Wert von A mit Häufigkeit $(p_i+n_i)/(p+n)$

Erwarteter Rest von erforderlicher Information nach A :

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

Informationsgewinn durch Auswahl von A:

$$Gain(A) = I(\langle p/(p+n), n/(p+n)\rangle) - Remainder(A)$$

Beispiel: Maximal informative Auswahl des ersten Attributs

$$Gain(\text{Gäste?}) = 1 - [2/12 I(\langle 0,1\rangle) + 4/12 I(\langle 1,0\rangle) + 6/12 I(\langle 2/6, 4/6\rangle)] \approx 0.541$$

$$Gain(\text{Typ?}) = 1 - [2\cdot 2/12\ I(\langle 1/2,1/2\rangle) + 2\cdot 4/12\ I(\langle 2/4,2/4\rangle)\ ] = 0$$

DTL-artige Lernsysteme

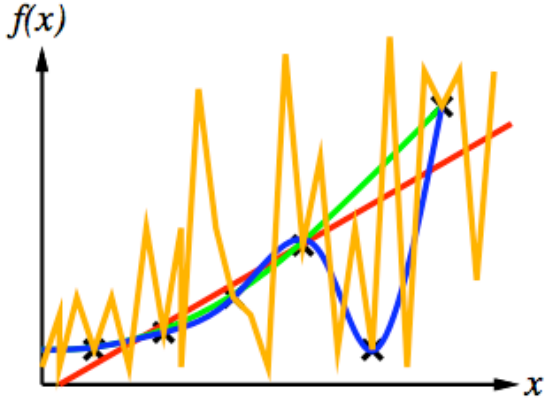
- ID3 (*Iterative Dichotomizer* 3, 1986), J. Ross Quinlan.
  Nachfolger C4.5 (↦ Ertel); kommerzielle Version C5.0
- CART (*Classification and Regression Trees*, urspr. 1984),
  Leo Breiman, kommerzielle Version
- KNIME (Konstanz Information Miner), s. Ertel 8.8.1

KNIME verwendet Bibliothek (Java) von *Machine Learning* Algorithmen, die über Entscheidungsbäume hinausgeht:

- WEKA www.cs.waikato.ac.nz/ml/weka/
  (seit den 1990er Jahren, wird permanent weiter gepflegt)

DTL lässt offen …

- Wieviele Lernbeispiele nimmt man?
- Welche Lernbeispiele nimmt man?
- Welche Attribute gibt es? (bei versteckten nie 100% Erfolg!)
- Welche/wieviele Messfehler sind in den Daten?
- Wie vermeidet man Überanpassung/***overfitting***,
  also unnötig spezielle Lernfunktionen?
- Welche Einschränkungen der Lernfunktion macht man?
  (***Ockham's razor***: wähle einfachste konsistente Hypothese!
  Aber was ist die einfachste?)



➥ Unter allen E-Bäumen mit gleicher Fehlerrate nimm einen kleinsten!

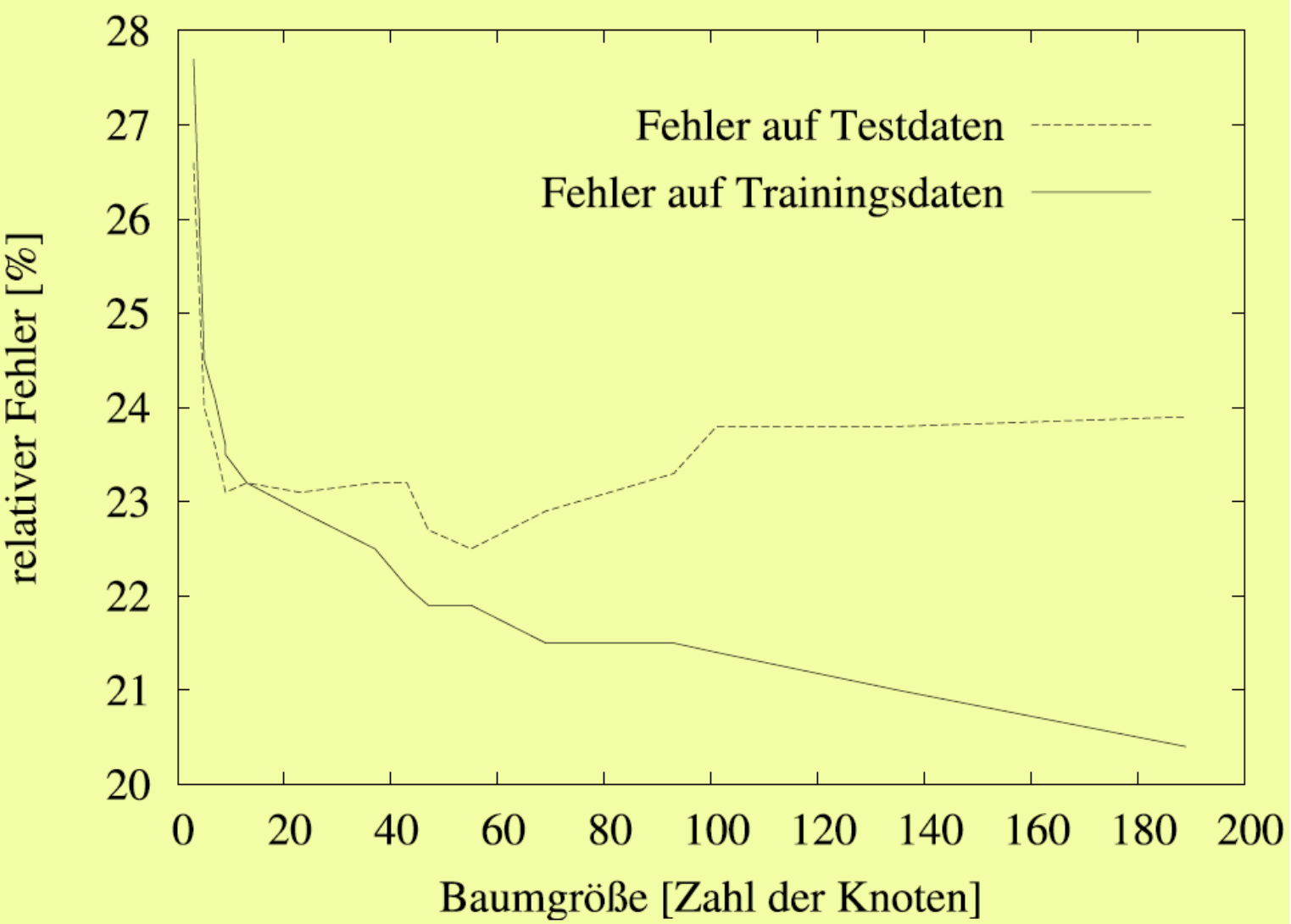
Überanpassung – jetzt formal

Definition 8.7 *Sei ein bestimmtes Lernverfahren, das heißt eine Klasse lernender Agenten, gegeben. Man nennt einen Agenten A überangepasst an die Trainingsdaten, wenn es einen anderen Agenten A' gibt, dessen Fehler auf den Trainingsdaten größer ist als der von von A, aber auf der gesamten Verteilung von Daten ist der Fehler von A' kleiner als der von A.*

Und wie stellt man das fest?

- Unterteile Trainingsdaten (z.B. 2/3 : 1/3 oder 3/4 : 1/4) in
  - zum Lernen verwendete Trainingsdaten
  - Testdaten
- Miss den Trainings-Fehler auf den Testdaten (**Kreuzvalidierung**, ***cross validation***)
- Beschneide E-Baum, um Trainings-Fehler zu reduzieren; ggf. Backtracking auf andere Attribute

Beispiel Kreuzvalidierung (qualitativ)



Fazit Entscheidungsbaumlernen

- Viel verwendetes Klassifikationsverfahren
- Einfach in der Anwendung
- Schnelles Lernen, schnelle Klassifikation
- Irrelevante Attribute fallen beim Lernen weg
- Entscheidungsbaum kann nachträglich inspiziert und verändert werden (E-Baum für Menschen interpretierbar!)
- Gelernte E-Bäume nicht notwendig korrekt und optimal

Naive Bayes-Klassifikation

vgl. Folie 212!

- Bayes-Netze (zur Klassifikation oder wozu auch immer) kann man lernen: Lerne Struktur und lerne W'verteilungen (Ertel 8.5)
- Vereinfachtes Problem **hier**: Lerne **W'verteilungen** für **Naives Bayessches Modell** (Folie 212)

$$\mathbf{P}(Ursache, Effekt_1, \ldots, Effekt_n) \approx \alpha\, \mathbf{P}(Ursache) \prod_i \mathbf{P}(Effekt_i \mid Ursache)$$

- Bei **Klassifikation** hat die Zufallsvariable $Ursache$ mehrere mögliche Ausprägungen $\{Ursache=u_k \mid 1 \le k \le m\}$, also für alle k :

$$\mathbf{P}(U= u_k, Effekt_1, \ldots, Effekt_n) \approx \alpha\, P(U=u_k) \prod_i \mathbf{P}(Effekt_i \mid U=u_k)$$

- Gesuchte Klasse also: $u_{\text{Naive-Bayes}} = \underset{k \in \{1,\ldots,m\}}{\operatorname{argmax}} P(U = u_k) \prod_{i=1}^{n} \mathbf{P}(Effekt_i \mid u_k)$
- **Lernaufgabe**: Lerne aus (vielen) Daten $\mathbf{P}(U)$ und $\mathbf{P}(Effekt_i \mid u_k)$

W‘keiten aus Daten schätzen

$$u_{\text{Naive-Bayes}} = \underset{k\in\{1,\ldots,m\}}{\operatorname{argmax}} P(U=u_k)\prod_{i=1}^{n}\mathbf{P}(Effekt_i|u_k)$$

- a priori W‘keiten sind einfach: Für N Datensätze schätze

$$P(U=u_k) \approx \frac{|\text{Datensätze mit } u_k|}{N}$$

- Bedingte Verteilung $\mathbf{P}(Effekt_i \mid u_k)$: Nach Definition schätze

$$P(E_i=x|u_k) \approx \frac{|E_i=x \wedge U=u_k|}{|U=u_k|} =: \frac{n_{x,uk}}{n_{uk}}$$

- Aber **Achtung**: bei kleiner Einzelw‘keit $P(E_i{=}x \mid u_k)$ akute Gefahr, dass bei begrenztem N kein einziger Datensatz mit $E_i{=}x \wedge U{=}u_k$ vorkommt! (aber dennoch $P(E_i{=}x|u_k){\neq}0$!)

Abhilfe durch a priori-W‘keit

Statt direktem Wert aus Datensatz $P(E_i = x|u_k) \approx \frac{|E_i = x \wedge U = u_k|}{|U = u_k|} =: \frac{n_{x,uk}}{n_{uk}}$

verwende $P(E_i = x|u_k) \approx \frac{n_{x,uk} + m \cdot P(E_i = x)}{n_{uk} + m}$

Dabei …

- schätze $P(E_i{=}x)$ aus d. Daten: $P(E_i{=}x) \approx$ |Datensätze mit $E_i{=}x$| /N
- wähle Parameter m „geeignet“

↪Somit jetzt alles bereit für Naiven Bayes-Klassifikator!

$$u_{\text{Naive-Bayes}} = \underset{k \in \{1,\ldots,m\}}{\text{argmax}} P(U = u_k) \prod_{i=1}^{n} \mathbf{P}(Effekt_i|u_k)$$

Beispiel: Textklassifikation mit Naive-Bayes

Anwendungen

- Ausfiltern von Email
  (Spam-Filter, z.B. CRM114 en.wikipedia.org/wiki/CRM114_(program))
- Ausfiltern von unerwünschten Beiträgen in Diskussionsforen
- Suche relevanter Dokumente in Volltext-Datenbanken

Struktur des Einsatzes (z.B. Spam-Filter)

- Nutzer klassifiziert „mittelgroße“ Zahl von Emails manuell als „gut“ oder „schlecht“
- Trainiere Naive-Bayes-Klassifikator auf „gut“ oder „schlecht“ nach Wörtern, die in den Emails vorkommen

Naive-Bayes-Textklassifikation: Ansatz

- Textklassen I (interessant) und U (uninteressant, Ertel: $\neg I$)
- Trainingsmenge schon klassifizierter Texte
- Attribute: für n Wortpositionen im Text je ein s_i (Wort an Position i)
- Attributwerte: alle im Text vorkommenden Wörter
- Berechne: $P(I|s_1,\ldots,s_n) \approx \alpha \cdot P(I) \prod_i P(s_i|I)$ (analog für U)
- Liege folgender Text vor (erste 104 Wörter):

Eher konservative Universitätsvertreter – betrachten wir *pars pro toto* den Deutschen Hochschulverband – und eher progressive Studierendenvertretende – sagen wir, von einem typischen AStA – sind sich seit Jahren verblüffend einig: Der Untergang der Deutschen Universität steht unmittelbar bevor. Die Begründungen sind gruppen- und interessenspezifisch unterschiedlich, doch kommen die Begriffe Bologna, Verschulung, Studiengebühren und Prüfungsflut zuverlässig auf beiden Seiten vor.

Über den Untergang der Universität bin ich mir nicht sicher. Doch dass sich das Studium in den letzten Jahren für alle Beteiligten verändert hat, das ist gewiss. Die Fächer sind dabei, ihren Lehrkanon durchzusehen und hinsichtlich der Strukturierung in Bachelor- und Masterstudium neu zu sortieren

Klassifizierung auf ersten 100 Wörtern

$$P(I|s_1,\ldots,s_n) = \alpha \cdot P(I) \cdot P(s_1 = \text{"Eher"}|I) \cdot P(s_2 = \text{"konservative"}|I) \cdot \ldots \cdot P(s_{100} = \text{"und"}|I)$$

… und entsprechend für U.

Um das auszurechnen, brauchen wir

- a priori-W'keiten $P(I), P(U)$ ↪
  durch Auszählen der Trainingsmengen-Klassen, s. Folie 276
- bedingte W'keiten für Wortpositionen ↪
  - **Annahme**: $P(s_i|I)$ hängt nicht von Position im Text ab, also
    $$P(s_1 = \text{"und"}|I) = P(s_2 = \text{"und"}|I) = \ldots = P(s_{100} = \text{"und"}|I) =: P(\text{"und"}|I)$$
  - Ermittle $P(w|I), P(w|U)$ für alle Wörter w (Folie 276)
  - Für jedes w im Text zähle Häufigkeit n. Für l Wörter:

$$P(I|s_1,\ldots,s_n) = \alpha \cdot P(I) \cdot \prod_{i=1}^{l} P(s_i|I)^{n_i} \qquad P(U|s_1,\ldots,s_n) = \alpha \cdot P(U) \cdot \prod_{i=1}^{l} P(s_i|U)^{n_i}$$

… und das soll funktionieren?!

- Naive-Bayes liefert gute Ergebnisse!
- Für Spam-Filter „Fehlerraten weit unter 1%“
- Für CRM114 korrekte Erkennung bis über 99,9%, falsch Positive unter 5,3% (😐) (Wikipedia)