

Bayes'sches Lernen: Übersicht

- Bayes'sches Theorem
- MAP, ML Hypothesen
- MAP Lernen
- Minimum Description Length Principle
- Bayes'sche Klassifikation
- Naive Bayes Lernalgorithmus

2 Zielrichtungen der Bayes'schen Methoden

Bereitstellen von praktischen Lernalgorithmen:

- Naive Bayes
- Bayes'sche Netze
- Kombiniere Wissen (a priori-Wahrscheinlichkeiten) und beobachtete Daten
- Erfordert a priori-Wahrscheinlichkeiten

Bereitstellen eines konzeptuellen Modells

- „Standard“ zum Vergleich mit anderen Lernalgorithmen
- Zusätzliche Einsichten in Occam's Razor

Bayes'sches Theorem

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

- $P(h)$ = a priori Wahrscheinlichkeit der Hypothese h
- $P(D)$ = a priori Wahrscheinlichkeit der Trainingsdaten D
- $P(h|D)$ = Wahrscheinlichkeit von h gegeben D
- $P(D|h)$ = Wahrscheinlichkeit von D gegeben h

Auswahl von Hypothesen

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

Suchen wahrscheinlichste Hypothese gegeben die Trainingsdaten

Maximum a posteriori Hypothese h_{MAP} :

$$\begin{aligned} h_{MAP} = \arg \max_{h \in H} P(h|D) &= \arg \max_{h \in H} \frac{P(D|h)P(h)}{P(D)} \\ &= \arg \max_{h \in H} P(D|h)P(h) \end{aligned}$$

Unter der Annahme $P(h_i) = P(h_j)$ kann man weiter vereinfachen und wählt die *Maximum likelihood* (ML)-Hypothese:

$$h_{ML} = \arg \max_{h_i \in H} P(D|h_i)$$

Grundlegende Formeln für Wahrscheinlichkeiten

- *Produktregel*: Wahrscheinlichkeit $P(A \wedge B)$ der Konjunktion zweier Ereignisse A und B :

$$P(A \wedge B) = P(A|B)P(B) = P(B|A)P(A)$$

- *Summenregel*: Wahrscheinlichkeit $P(A \vee B)$ der Disjunktion zweier Ereignisse A und B :

$$P(A \vee B) = P(A) + P(B) - P(A \wedge B)$$

- *Theorem der totalen Wahrscheinlichkeiten*: Wenn die Ereignisse A_1, \dots, A_n sich gegenseitig ausschließen und $\sum_{i=1}^n P(A_i) = 1$, dann

$$P(B) = \sum_{i=1}^n P(B|A_i)P(A_i)$$

Brute Force MAP-Hypothesen-Lerner

1. Für jede Hypothese h in H , berechne a posteriori Wahrscheinlichkeit

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

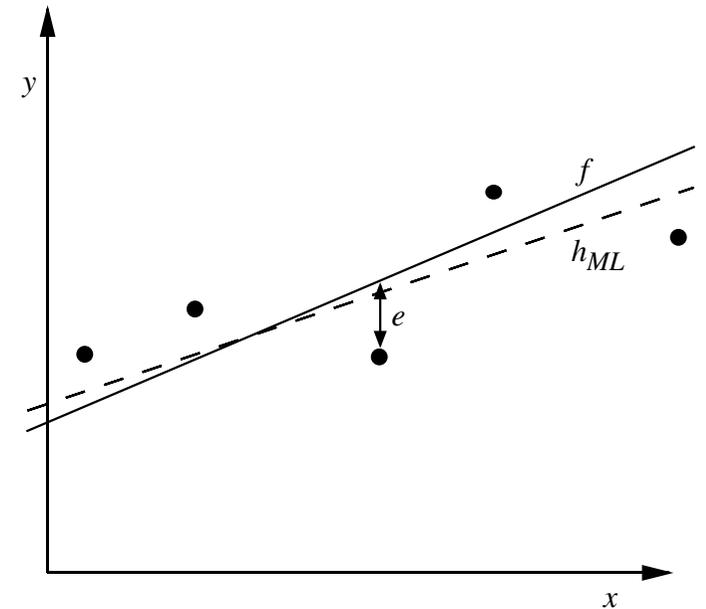
2. Gib Hypothese h_{MAP} mit höchster a posteriori Wahrscheinlichkeit aus

$$h_{MAP} = \operatorname{argmax}_{h \in H} P(h|D)$$

Beispielanwendung: Lernen einer reelwertigen Funktion

Betrachte reelwertige Zielfunktion f
Trainingsbeispiele sind $\langle x_i, d_i \rangle$,
wobei die d_i verrauscht sind

- $d_i = f(x_i) + e_i$
- e_i ist Zufallsvariable (Noise) die unabhängig voneinander für jedes x_i bezüglich einer Normalverteilung mit Mittelwert=0 gezogen werden



Die **Maximum-Likelihood-Hypothese** h_{ML} ist nun genau diejenige, die die **Summe der Quadrate der Fehler** minimiert:

$$h_{ML} = \arg \min_{h \in H} \sum_{i=1}^n (d_i - h(x_i))^2$$

Warum?

$$\begin{aligned}h_{ML} &= \operatorname{argmax}_{h \in H} p(D|h) \\&= \operatorname{argmax}_{h \in H} \prod_{i=1}^n p(d_i|h) \\&= \operatorname{argmax}_{h \in H} \prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{d_i - h(x_i)}{\sigma}\right)^2} \\&= \operatorname{argmax}_{h \in H} \sum_{i=1}^n \ln \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2} \left(\frac{d_i - h(x_i)}{\sigma} \right)^2 \\&= \operatorname{argmax}_{h \in H} \sum_{i=1}^n -\frac{1}{2} \left(\frac{d_i - h(x_i)}{\sigma} \right)^2 \\&= \operatorname{argmax}_{h \in H} \sum_{i=1}^n -(d_i - h(x_i))^2 \\&= \operatorname{argmin}_{h \in H} \sum_{i=1}^n (d_i - h(x_i))^2\end{aligned}$$

Minimum Description Length Principle

Occam's Razor: wähle kleinste Hypothese

MDL: bevorzuge Hypothese h , die folgendes minimiert:

$$h_{MDL} = \operatorname{argmin}_{h \in H} L_{C_1}(h) + L_{C_2}(D|h)$$

wobei $L_C(x)$ die Beschreibungslänge von x unter Kodierung C ist

Beispiel: H = Entscheidungsbäume, D = Labels der Trainingsdaten

- $L_{C_1}(h)$ ist # Bits zum Beschreiben des Baums h
- $L_{C_2}(D|h)$ ist # Bits zum Beschreiben von D gegeben h
 - Anmerkung: $L_{C_2}(D|h) = 0$ falls alle Beispiele korrekt von h klassifiziert werden. Es müssen nur die Ausnahmen kodiert werden.
- h_{MDL} wägt Baumgröße gegen Trainingsfehler ab

Minimum Description Length Principle

$$\begin{aligned}h_{MAP} &= \arg \max_{h \in H} P(D|h)P(h) \\ &= \arg \max_{h \in H} \log_2 P(D|h) + \log_2 P(h) \\ &= \arg \min_{h \in H} -\log_2 P(D|h) - \log_2 P(h)\end{aligned}\tag{1}$$

Interessanter Fakt aus der Kodierungstheorie:

Die optimale (kürzeste) Kodierung für ein Ereignis mit Wahrscheinlichkeit p benötigt $-\log_2 p$ Bits.

Interpretiere (1):

- $-\log_2 P(h)$: Größe von h bei optimaler Kodierung
- $-\log_2 P(D|h)$: Größe von D gegeben h bei optimaler Kodierung

→ wähle Hypothese die folgendes minimiert:

$$\textit{length}(h) + \textit{length}(\textit{misclassifications})$$

Klassifikation neuer Instanzen

Bis jetzt haben wir die *wahrscheinlichste Hypothese* für gegebene Daten D gesucht (d.h., h_{MAP})

Gegeben neue Instanz x , was ist die wahrscheinlichste *Klassifikation*?

- $h_{MAP}(x)$ ist es nicht unbedingt!!!

Beispiel:

- Betrachte 3 Hypothesen und gegebene Daten D :

$$P(h_1|D) = 0,4; P(h_2|D) = 0,3; P(h_3|D) = 0,3$$

- Gegeben sei neue Instanz x ,

$$h_1(x) = +, h_2(x) = -, h_3(x) = -$$

- Was ist $h_{MAP}(x)$, was ist wahrscheinlichste Klassifikation von x ?

Bayes'sche optimale Klassifikation

Bayes'sche optimale Klassifikation:

$$\arg \max_{v_j \in V} \sum_{h_i \in H} P(v_j | h_i) P(h_i | D)$$

Für unser Beispiel:

$$P(h_1 | D) = 0,4; \quad P(- | h_1) = 0; \quad P(+ | h_1) = 1$$

$$P(h_2 | D) = 0,3; \quad P(- | h_2) = 1; \quad P(+ | h_2) = 0$$

$$P(h_3 | D) = 0,3; \quad P(- | h_3) = 1; \quad P(+ | h_3) = 0$$

Deshalb:

$$\sum_{h_i \in H} P(+ | h_i) P(h_i | D) = 0,4$$

$$\sum_{h_i \in H} P(- | h_i) P(h_i | D) = 0,6$$

Gibbs Klassifikation

Bayes'sche Klassifikation optimal, aber teuer bei vielen Hypothesen

Gibbs Algorithmus:

1. Wähle zufällig eine Hypothese h bezüglich $P(h|D)$
2. Benutze h zur Klassifikation

Überraschung: Sei ein Zielkonzept zufällig bezüglich \mathcal{D} aus H gewählt. Dann:

$$E[\text{error}_{Gibbs}] \leq 2 \cdot E[\text{error}_{BayesOptimal}]$$

Naive Bayes Klassifikation

Neben Entscheidungsbäumen, Neuronalen Netzen, Nearest Neighbour eine der am meisten eingesetzten Lernmethoden.

Wann anwendbar:

- Mittlere oder große Trainingsmengen
- Attribute sind bedingt unabhängig gegeben die Klassifikation

Erfolgreiche Anwendungsgebiete:

- Diagnose
- Klassifikation von Textdokumenten

Naive Bayes Klassifikation

Ziel $f : X \rightarrow V$, jede Instanz durch Attribute $\langle a_1, a_2 \dots a_n \rangle$ beschrieben

Wahrscheinlichster Wert von $f(x)$:

$$\begin{aligned} v_{MAP} &= \operatorname{argmax}_{v_j \in V} P(v_j | a_1, a_2 \dots a_n) \\ &= \operatorname{argmax}_{v_j \in V} \frac{P(a_1, a_2 \dots a_n | v_j) P(v_j)}{P(a_1, a_2 \dots a_n)} \\ &= \operatorname{argmax}_{v_j \in V} P(a_1, a_2 \dots a_n | v_j) P(v_j) \end{aligned}$$

Annahme von Naive Bayes: $P(a_1, a_2 \dots a_n | v_j) = \prod_i P(a_i | v_j)$

$$\text{Naive Bayes Klassifikation: } v_{NB} = \operatorname{argmax}_{v_j \in V} P(v_j) \prod_i P(a_i | v_j)$$

Naive Bayes Algorithmus

Naive_Bayes_Learn(examples):

Für jeden Klassifikationswert v_j

$$\hat{P}(v_j) \leftarrow \text{schätze } P(v_j)$$

Für jeden Attributwert a_i jedes Attributs a

$$\hat{P}(a_i|v_j) \leftarrow \text{schätze } P(a_i|v_j)$$

Ergebnis: Tabelle mit geschätzten WKen

Classify_New_Instance(x):

$$v_{NB} = \operatorname{argmax}_{v_j \in V} \hat{P}(v_j) \prod_{a_i \in x} \hat{P}(a_i|v_j)$$

Naive Bayes: Beispiel

Betrachte *PlayTennis* mit neuer Instanz

$\langle \text{Outlk} = \text{sun}, \text{Temp} = \text{cool}, \text{Humid} = \text{high}, \text{Wind} = \text{strong} \rangle$

Wollen berechnen:

$$v_{NB} = \operatorname{argmax}_{v_j \in V} P(v_j) \prod_i P(a_i | v_j)$$

$$P(\text{yes}) P(\text{sun}|\text{yes}) P(\text{cool}|\text{yes}) P(\text{high}|\text{yes}) P(\text{strong}|\text{yes}) = .005$$

$$P(\text{no}) P(\text{sun}|\text{no}) P(\text{cool}|\text{no}) P(\text{high}|\text{no}) P(\text{strong}|\text{no}) = .021$$

$$\rightarrow v_{NB} = \text{no}$$

Naive Bayes: Diskussion (1)

Annahme der bedingten Unabhängigkeit ist oft nicht erfüllt

$$P(a_1, a_2 \dots a_n | v_j) = \prod_i P(a_i | v_j)$$

...aber es funktioniert trotzdem erstaunlich gut. Warum? Abschätzungen für $\hat{P}(v_j | x)$ müssen nicht notwendig korrekt sein, sondern nur

$$\operatorname{argmax}_{v_j \in V} \hat{P}(v_j) \prod_i \hat{P}(a_i | v_j) = \operatorname{argmax}_{v_j \in V} P(v_j) P(a_1 \dots, a_n | v_j)$$

Naive Bayes: Diskussion (2)

Was, wenn aufgrund kleiner Trainingsmengen keines der Trainingsbeispiele mit Klassifikation v_j den Attributwert a_i hat? Dann

$$\hat{P}(a_i|v_j) = 0, \text{ und...}$$

$$\hat{P}(v_j) \prod_i \hat{P}(a_i|v_j) = 0$$

Typische Lösung: **m -Abschätzung**: $\hat{P}(a_i|v_j) \leftarrow \frac{n_c + mp}{n + m}$

wobei

n ... Anzahl der Trainingsbeispiele mit $v = v_j$,

n_c ... Anzahl der Beispiele mit $v = v_j$ und $a = a_i$

p ... a priori Schätzung für $\hat{P}(a_i|v_j)$

(z.B. durch Annahme uniformer Verteilung der Attributwerte \rightarrow

$$p = \frac{1}{|\text{values}(a_i)|})$$

m ... Gewicht für a priori-Abschätzung p (Anzahl "virtueller" Beispiele)

Werbung

Wir suchen ab sofort einen [HiWi](#) für die [Administration, Wartung und Ausbau des Linux-Rechnernetzwerkes](#).

Aufgaben:

- Aufbau eines Linux-Clusters aus ca. 10 Rechnern
- Pflege und Wartung des vorhandenen Systems (Account-Verwaltung, Backups, Updates...)
- Unterstützung bei kleineren Arbeiten (wie z.B. der Gestaltung des Web-Auftritts des Fachbereichs)

vorhergesehene durchschnittliche Arbeitszeit: 10 Wochenstunden (VHS)

- (in der Anfangsphase etwas mehr, in weiterer Folge eher weniger)

Instanzenbasiertes Lernen: Übersicht

- k -Nearest Neighbor
- Lokal gewichtete Regression
- Fallbasiertes Schließen
- Lernen: Lazy oder Eager

Instanzenbasiertes Lernen

Idee: speichere einfach alle Trainingsbeispiele $\langle x_i, f(x_i) \rangle$

Nearest Neighbor:

- Gegeben eine Instanz x_q , suche Trainingsbeispiel x_n , das am nächsten an x_q liegt und setze $\hat{f}(x_q) \leftarrow f(x_n)$

k -Nearest Neighbor: Gegeben x_q ,

Diskreter Fall wähle Mehrheit der Werte der k nächsten Nachbarn

Reellwertiger Fall wähle Mittelwerte der Werte der k nächsten Nachbarn

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^k f(x_i)}{k}$$

Wann ist Nearest Neighbor geeignet?

- Instanzen bilden Punkte im \mathcal{R}^n
- Weniger als 20 Attribute pro Instanz
- jede Menge Trainingsdaten

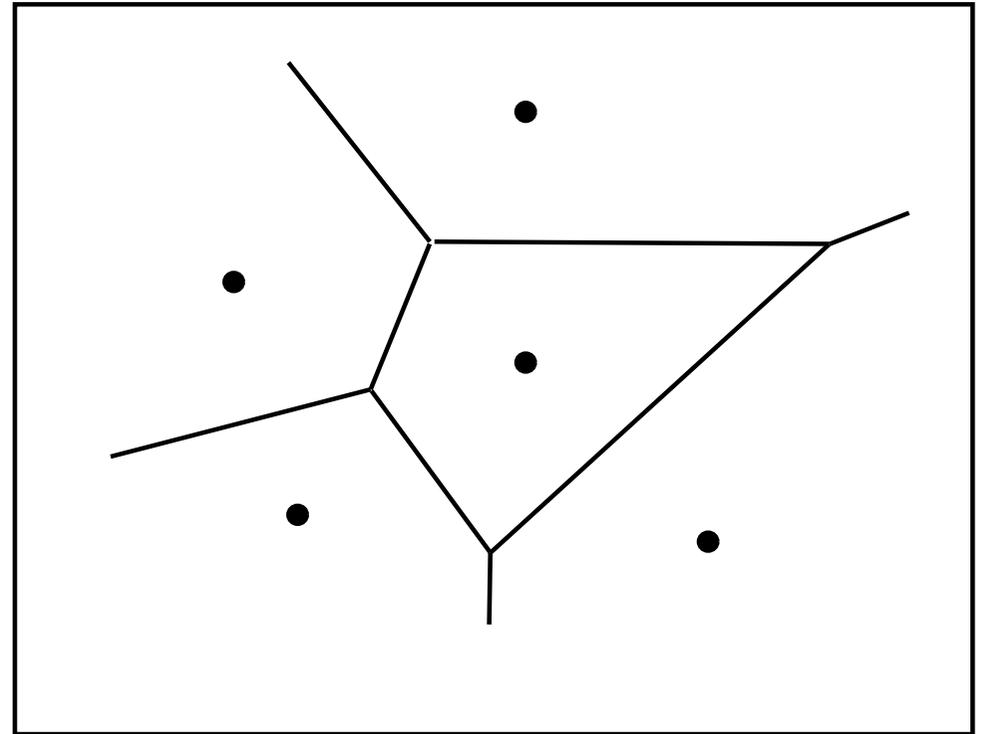
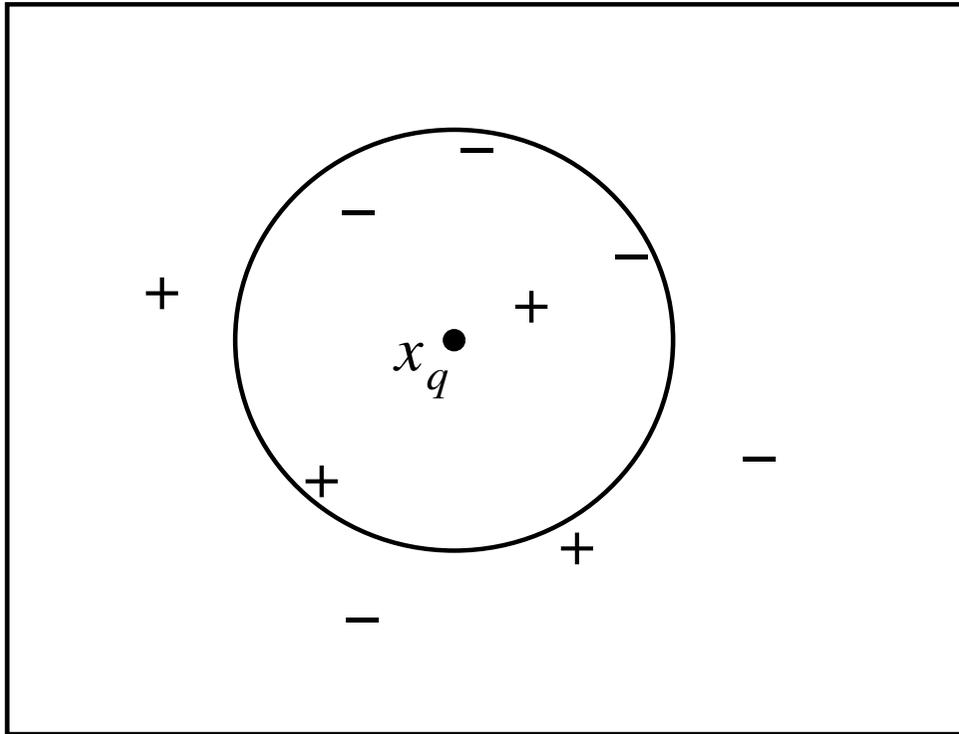
Vorteile:

- Training ist sehr schnell
- auch komplexeste Zielfunktionen lernbar
- kein Generalisierungsmechanismus nötig
- Kein Informationsverlust

Nachteile:

- Abstandsmaß muß angegeben werden
- Langsam zur Anwendungszeit
- sehr sensitiv gegenüber irrelevanten Attributen
- nicht vom Menschen interpretierbar/kommunizierbar

Voronoi-Diagramm



Verhalten im Limes

$p(x)$: WK, daß Instanz x mit 1 (gegenüber 0) bewertet wird

Nearest Neighbor:

- Wenn Zahl der Trainingsbeispiele $\rightarrow \infty$ ergibt sich Gibbs Algorithmus

Gibbs: mit WK $p(x)$ sage 1 voraus, sonst 0

k -Nearest neighbor:

- Wenn Zahl der Trainingsbeispiele $\rightarrow \infty$ und k groß genug ist, dann wird Bayes'sche Optimalklassifikation angenähert

Bayes'sche Optimalklassifikation: wenn $p(x) > .5$ dann sage 1 voraus, sonst 0

Bem.: Gibbs hat höchstens doppelten erwarteten Fehler wie Bayes'sche Optimalklassifikation

Abstandsgewichtetes k -NN

Möchten möglicherweise nähere Nachbarn stärker gewichten

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^k w_i f(x_i)}{\sum_{i=1}^k w_i}$$

wobei

$$w_i \equiv \frac{1}{d(x_q, x_i)^2}$$

und $d(x_q, x_i)$ ist Abstand von x_q und x_i

Jetzt können *alle* Trainingsbeispiele (statt bisher k) einbezogen werden

→ **Shepard's Methode**

Anmerkungen zur Dimensionalität

Beispiel: Instanzen durch 20 Attribute beschrieben, aber nur 2 davon sind relevant

Fluch der Dimensionalität: Nearest Neighbor wird durch hochdimensionale Räume in die Irre geführt

Ansatz:

- Dehne j te Achse durch Gewicht z_j , wobei z_1, \dots, z_n so gewählt werden, daß erwarteter Fehler minimiert wird
- Benutze Cross-Validation zur automatischen Bestimmung von z_1, \dots, z_n
- ($z_j = 0$ eliminiert diese Dimension vollständig)

Lokal gewichtete Regression

k -NN bildet lokale Approximation für f zu jedem Punkt x_q

Warum Approximation $\hat{f}(x)$ für Region um x_q nicht explizit angeben?

- Passe lineare (quadratische, ...) Fkt. den k nächsten Nachbarn an
- Resultiert in "stückweiser Annäherung" an f

Möglichkeiten der zu minimierenden Zielfehlern:

- Quadratischer Fehler über k nächsten Nachbarn

$$E_1(x_q) \equiv \frac{1}{2} \sum_{x \in \text{den } k \text{ nächsten Nachbarn von } x_q} (f(x) - \hat{f}(x))^2$$

- Abstandsgewichteter Quadratischer Fehler über allen Instanzen

$$E_2(x_q) \equiv \frac{1}{2} \sum_{x \in D} (f(x) - \hat{f}(x))^2 K(d(x_q, x))$$

Strategien zur Beispielauswahl

Fallstudie: Aha, Kibler and Albert: Instance-based learning. MLJ 6 (1991).

IB1: Speichere alle Beispiele

- Gute Noisetoleranz, hoher Speicherbedarf

IB2: Speichere nur solche Beispiele, die von bisheriger Fallbasis falsch klassifiziert werden

- Geringe Noisetoleranz, geringer Speicherbedarf

IB3: Wie IB2, aber halte Zähler zu jedem Beispiel, wie oft es an richtiger bzw. falscher Vorhersage beteiligt war. Benutze Signifikanztest um herauszufinden, welche Beispiele vermutlich verrauscht sind (diese werden gelöscht)

- erhöhte Noisetoleranz bei geringem Speicherbedarf

Fallbasiertes Schließen

Können Instanz-basiertes Lernen auch anwenden, wenn $X \neq \mathbb{R}^n$

→ brauchen anderes “Abstands”-Maß: Ähnlichkeit

Verschiedene Möglichkeiten:

- Farben: Abstand im Farbkreis
- Attributvektoren
 - Hammingabstand
- Strings:
 - Anzahl der unterschiedlichen Buchstaben
 - Differenz der Längen
 - Anzahl der Editoperationen

Im allgemeinen: Abstand/Ähnlichkeit domainabhängig, frei wählbar

Fallbasiertes Schließen

Warum eigentlich Ähnlichkeit über Zahlen definieren?

- Terme: Abstand mittels Antiunifikator
 - Abstand von $f(g(a, f(b, b)), c, d)$ und $f(c, c, h(a, a))$ ist $f(X, c, Y)$
- Formeln: lgg (least general generalization)
- Graphen: Größter gemeinsamer Teilgraph
- Bilder: Menge gemeinsamer Bildteile

Allgemein: Ähnlichkeit definiert *Halbordnung* über den Instanzen
→ bestimmte Instanzen sind mglw. unvergleichbar

Was ist mit Symmetrie?

- $\text{sim}(\text{ICE}, \text{Zug}) = \text{sim}(\text{Bummelzug}, \text{Zug})$, aber
 $\text{sim}(\text{Zug}, \text{ICE}) > \text{sim}(\text{Zug}, \text{Bummelzug})$

Lernen: Lazy vs. Eager

Lazy: warte auf Anfrage, bevor generalisiert wird

- k -NEAREST NEIGHBOR, Fallbasiertes Schließen

Eager: Generalisiere, bevor Anfrage kommt

- ID3, NaiveBayes, . . .

Was ist besser?

- Eager Learning muß globale Approximation finden
- Lazy Learner kann viele lokale Approximationen kombinieren
- für den gleichen Hypothesenraum können Lazy Learner komplexere Funktionen repräsentieren (Beispiel: lineare Funktionen)