

INHALTSVERZEICHNIS

1 Einführung	4
1.1 Definition Maschinelles Lernen	4
1.2 Komponenten eines lernenden Systems	4
1.3 Einordnungskriterien von Lernverfahren	5
2 Induktives Lernen	5
2.1 Induktion vs Deduktion	5
2.2 Konzeptlernen als Suche im Hypothesenraum	5
2.3 Specific-to-general-Suche	6
2.4 Versionsraum (Version Space)/Candidate-Elimination-Algorithmus	7
2.5 Notwendigkeit von Vorzugskriterien	8
3 Unüberwachte Lernverfahren	9
3.1 k-means Clustering	9
3.1.1 Fuzzy-k-means-Clustering	10
3.1.2 Einordnung k-means-Clustering	10
3.2 Hierarchisches Clustering	11
3.2.1 Agglomerative Hierarchical Clustering	11
3.3 Begriffliche Ballungen & COBWEB	11
3.3.1 COBWEB	12
4 Lerntheorie, Algorithmenunabhängige Verfahren (für überwachtes induktives Lernen)	13
4.1 Definition Lernmaschine	13
4.2 Probleme beim Lernen	13
4.3 Überwachtes Lernen aus Beispielen	13
4.4 Fehler: empirischer & realer Fehler	13
4.5 Overfitting	14
4.6 Modellauswahl (Crossvalidation, Bootstrap, AdaBoost)	14
4.7 Was ist PAC (Lernbarkeit)?	15
4.8 Approximation des realen Fehlers nach Vapnik und Chervonenkis	16
4.9 Wie kann korrektes Lernen erfolgen?	17
4.9.1 Structural Risk Minimization	17
5 Reinforcement Learning	17
5.1 Markov decision process (deterministisch)	18
5.2 Anwendungsbeispiele	18
5.3 Strategielernen - Policy learning	18
5.4 Optimale Strategie	18
5.5 Simple Temporal Difference Learning	19
5.6 Problemdimensionen beim RL	19
5.7 Die Q-Funktion	19
5.8 Suchstrategien/Experimentieren	19
5.9 Exploration vs Exploitation	20
5.10 Optimierungen	20
5.11 Repräsentation, Generalisierung	20
5.12 Lernen von Aktionssequenzen	20
5.12.1 Warum Lernen von Aktionssequenzen	20
5.12.2 Temporal-Difference-Learning	21

6 Support Vektor Methoden	21
6.1 Lineare Support Vektor Methode	21
6.2 Nichtlineare Support Vektor Methode	22
6.2.1 Version Space für SVM	22
6.3 Erweiterungen	23
6.4 Bewertung SVM	23
7 Entscheidungsbäume	23
7.1 Grundlagen	23
7.2 ID3	24
7.2.1 Overfitting	25
7.2.2 Erweiterungen	26
7.2.3 ID3: Zusammenfassung	28
7.3 C4.5	28
7.3.1 Rule Post-Pruning	28
7.3.2 Abschätzung der Güte der Regeln (Pessimistische Abschätzung)	29
7.3.3 Umwandlung in Regeln	29
7.4 ID5R	29
7.4.1 Repräsentation der EB	29
7.4.2 Baum Update Algorithmus	30
7.4.3 Rekonstruierung	30
7.5 Random Forests	31
8 Lernen nah Bayes	31
8.1 Motivation	31
8.1.1 Was ist Lernen nach Bayes?	31
8.1.2 Warum Lernen nach Bayes?	32
8.1.3 Lernen nach Bayes: Schwierigkeiten	32
8.1.4 Wahrscheinlichkeitstheorie	32
8.2 Theorem von Bayes	32
8.3 MAP-/ ML-Hypothesen	33
8.4 Konzeptlernen	33
8.5 Optimaler Bayes-Klassifikator	34
8.6 Naiver Bayes-Klassifikator	35
8.7 Bayes'sche Netze	36
8.7.1 Inferenz	36
8.7.2 Lernen	37
8.8 EM (Expectation Maximization)-Algorithmus	37
8.9 Zusammenfassung	38
9 Hidden Markov Modelle	39
9.1 Motivation	39
9.2 Diskreter Markov Prozess	40
9.3 Hidden Markov Modelle	40
9.4 Die 3 grundlegenden Probleme	40
9.5 Lösungen der Probleme	41
9.5.1 Evaluationsproblem	41
9.5.2 Dekodierungsproblem	42
9.5.3 Lern- oder Optimierungsproblem	42
9.6 Arten von HMMs	43

10 Markov Logik Netze (MLN)	44
10.1 Markov Logik	44
10.1.1 Markov Logik Netz	44
11 Deduktives Lernen	46
11.1 Deduktive Lernverfahren	46
11.2 Erklärungsbasiertes Lernen (EBL)	46
11.2.1 Erklärungsbasierte Generalisierung (EBG)	49
11.3 Deduktives vs Induktives Lernen	50
11.4 Hybride Lernverfahren	50
12 Instanzbasiertes Lernen	52
12.1 Einführung	52
12.2 Der k-NN Algorithmus	52
12.3 Case-Based Reasoning (CBR)	53
13 Neuronale Netze	56
13.1 Perzeptron [Rosenblatt 1960]	56
13.2 Gradientenabstieg	56
14 Evolutionäre Algorithmen	57
14.1 Mutation	58
14.2 Rekombination	58
14.3 Selektion	58
14.4 Genetische Programmierung	59

Maschinelles Lernen I

Manuel Lang

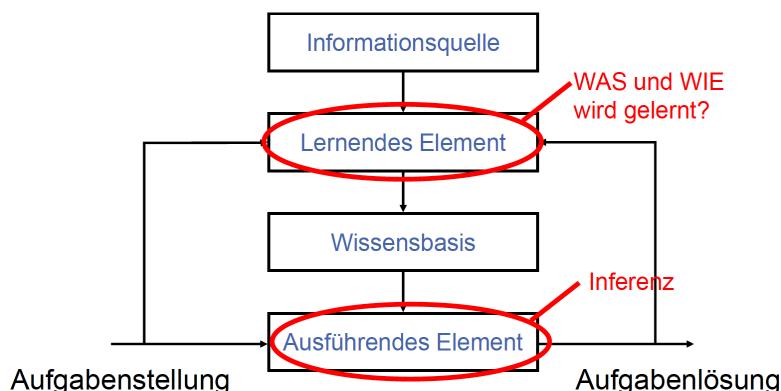
9. Juli 2017

1 EINFÜHRUNG

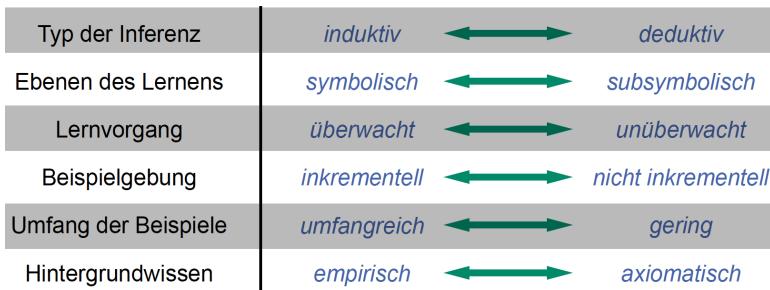
1.1 DEFINITION MASCHINELLES LERNEN

Ein System lernt aus Erfahrung E in Hinblick auf eine Klasse von Aufgaben T und einem Performanzmaß P, wenn seine Leistungen bei Aufgaben aus T gemessen mit P durch Erfahrung aus E steigt.

1.2 KOMPONENTEN EINES LERNENDEN SYSTEMS



1.3 EINORDNUNGSKRITERIEN VON LERNVERFAHREN



2 INDUKTIVES LERNEN

Was ist Induktion? - Prozess des plausiblen Schließens vom Speziellen zum Allgemeinen

2.1 INDUKTION VS DEDUKTION

Induktion

- Wahrheitserweiternd
- Macht Lebewesen überlebensfähig
- Plausibilität

Induktive Lernhypothese: Jede Hypothese, die die Zielfunktion über einer genügend großen Menge von Trainingsbeispielen gut genug approximiert, wird die Zielfunktion auch über unbekannten Beispielen gut approximieren.

Deduktion

- Wahrheitserhaltend
- Logischer Schluss
- Korrektheit

2.2 KONZEPTLERNEN ALS SUCHE IM HYPOTHESENRAUM

Konzept

- Beschreibt Untermenge von Objekten oder Ereignissen definiert auf größerer Menge
- Bool'sche Funktion definiert über größere Menge
- Beispiel: vogel: Tiere → {true,false}, vogel(Storch) = true, vogel(Hase) = false

Lernen von Konzepten

- Gegeben: Beispiele, die als Mitglieder oder Nichtmitglieder eines Konzeptes gekennzeichnet sind
- Gesucht: Automatischer Schluss auf die Definition des zugrundeliegenden Konzepts
- Definition Konzeptlernen: Schließen auf eine Boolean-wertige Funktion aus Trainingsbeispielen ihres Inputs und Outputs

Konsistenz und Vollständigkeit im Hypothesenraum

- Konsistenz: Keine negativen Beispiele werden positiv klassifiziert
- Vollständigkeit: Alle positiven Beispiele werden als positiv klassifiziert

Lernen als Suche im Hypothesenraum

- Repräsentation der Hypothesen legt implizit Hypothesenraum fest (Domänenwissen -> Bias)
- Lernen als Suche im Raum der möglichen Hypothesen: 96 mögl. Instanzen, 973 mögl. Hypothesen
- Jeder Beschreibungsraum für Konzepte ist nach Generalität (halb-)geordnet:
 - $h_1 = \langle \text{sonnig}, ?, ?, \text{stark}, ?, ? \rangle$
 - $h_2 = \langle \text{sonnig}, ?, ?, ?, ?, ? \rangle$
 - h_k spezieller $h_i \Leftrightarrow \forall x \in X : [h_k(x) = 1 \Rightarrow h_i(x) = 1]$
 - wobei $h(x) = 1$ bedeutet: x erfüllt die Hypothese h

2.3 SPECIFIC-TO-GENERAL-SUCHE

Suche vom Speziellen zum Allgemeinen:

- Ausgangspunkt ist die speziellste Hypothese $\langle \#, \dots, \# \rangle$
- Positive Beispiele: (minimale) Verallgemeinerung
- Negative Beispiele: werden nicht betrachtet
- Initialisiere h mit der spezifischen Hypothese in H
- Für jedes positive Trainingsbeispiel x : Für jede Attributeinschränkung a_i in $h = \langle a_0, \dots, a_n \rangle$
 - Wenn a_i von x erfüllt wird dann tue nichts
 - Sonst ersetze a_i durch die nächstallgemeinere Einschränkung, die durch x erfüllt wird
- Gib die Hypothese aus

Beurteilung:

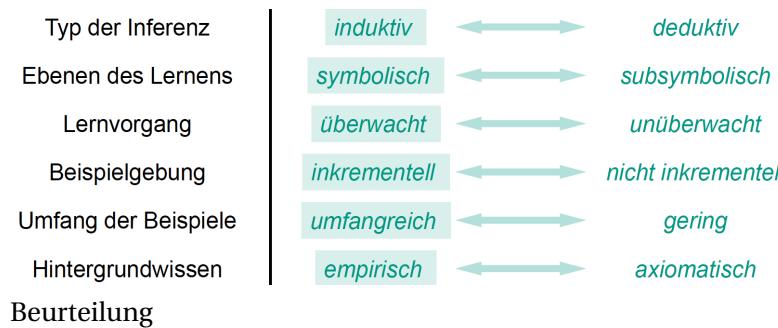
- Wichtiges Prinzip im Konzeptlernen
- Für Hypothesenräume, die durch Konjunktionen von Attributeinschränkungen beschrieben sind garantiert das Verfahren die spezifischste Hypothese, die mit den positiven Trainingsbeispielen vereinbar ist
- Endhypothese ist auch mit negativen Trainingsbeispielen konsistent, solange die Trainingsbeispiele korrekt sind und die Zielhypothese in H enthalten ist
- Offene Fragen: Sind die Trainingsbeispiele konsistent? Endkonzept = korrektes Zielkonzept? Warum spezifischste Hypothese?

2.4 VERSIONSRAUM (VERSION SPACE) / CANDIDATE-ELIMINATION-ALGORITHMUS

Versionsraum (Version Space): Der Versionsraum $VS_{H,D}$ bezüglich des Hypothesenraums H und der Menge von Trainingsbeispielen D ist die Untermenge der Hypothesen von H , die mit den Trainingsbeispielen in D konsistent ist.

Version Space / Candidate-Elimination-Algorithmus

- Lernen ist inkrementell: Menge der konsistenten Hypothesen (Version space) ist ein Intervall in der partiellen Ordnung spezifischer als auf dem Hypothesenraum
- Gespeichert werden: Menge der spezifischen Hypothesen S und Menge der allgemeinsten Hypothesen G , die alle Bereiche abdecken => Hypothesen müssen nicht einzeln gespeichert werden
- Spezifischste und Allgemeinste Hypothesen:
 - $S = \{s \mid s \text{ ist eine Hypothese, die mit den betrachteten Beispielen konsistent ist und es gibt keine Hypothese, die spezifischer als } s \text{ und auch konsistent mit allen Beispielen ist}\}$
 - Initialisierung: $S = \{\#\}$
 - $G = \{g \mid g \text{ ist eine Hypothese, die mit den betrachteten Beispielen konsistent ist, und es gibt keine Hypothese, die allgemeiner als } g \text{ und auch konsistent mit allen Beispielen ist}\}$
 - Initialisierung: $G = \{?\}$
- Ist n ein negatives Beispiel
 - Lösche aus S die Hypothesen, die n abdecken.
 - Spezialisiere die Hypothesen in G soweit, dass sie n nicht abdecken und dass sie allgemeiner als eine Hypothese in S bleiben.
 - Lösche aus G alle Hypothesen, die spezifischer als eine andere Hypothese aus G sind.
- Ist p ein positives Beispiel
 - Lösche aus G die mit p inkonsistenten Hypothesen.
 - Verallgemeinere die Hypothesen in S soweit, dass sie p abdecken und dass sie spezifischer als eine Hypothese in G bleiben.
 - Lösche aus S alle Hypothesen, die allgemeiner als eine andere Hypothese aus S sind.



Beurteilung

- Version Space konvergiert zur korrekten Hypothese (S=G)
 - Voraussetzung: Beispiele konsistent, korrekte Hypothese in Hypothesenraum enthalten
 - Probleme: fehlerbehaftete Trainingsdaten (Rauschen), Zielkonzept nicht von Hypothesenrepräsentation abgedeckt => mögliche Erweiterung: disjunktive Begriffe
- Anfrage des Konzeptlerners möglichst so, dass Hypothesen im Version Space halbiert werden, dann: schnelle Lernrate / geringe Anzahl von Beispielen
- Umgang mit teilweise gelernten Konzepten nötig
- Wenn mehr als eine Hypothese im Versionsraum vorhanden ist dann alle klassifizieren positiv bzw. negativ => Entscheidung klar, ansonsten Mehrheitsentscheidung evtl. mit Angabe von Konfidenz, probabilistische Entscheidung / Wahrscheinlichkeit, Plausibilitätsbetrachtungen
 - - konsistente Beispiele notwendig
 - - Attributgeneralisierungsregeln maßgebend für Lernerfolg
 - + Kein Speichern aller Beispiele notwendig
 - + Stellt fest, wann genügend Beispiele gegeben wurden (S=G)
- Unter Umständen Art noch benötigter Beispiele erkennbar

2.5 NOTWENDIGKEIT VON VORZUGSKRITERIEN

- Problem: Zielkonzept evtl. nicht im Hypothesenraum enthalten
- Lösung: Hypothesenraum, der alle möglichen Hypothesen enthält?
- Grundlegende Eigenschaft von induktiver Inferenz: Ein induktives Lernsystem, das keine a-priori Annahmen über die Identität des Zielkonzeptes macht, hat keine rationale Basis, um unbekannte Instanzen zu klassifizieren.
- Induktives lernen erfordert Vorannahmen (inductive bias)

Vergleich induktiver Lernsysteme anhand ihres inductive bias

- Auswendiglerner: keine Vorannahme
- Version Space: Zielkonzept kann im Hypothesenraum repräsentiert werden
- Specific to General-Lerner: wie Version Space und zusätzlich: alle Instanzen sind negative Instanzen, solange nicht das Gegenteil bekannt ist
- Je strenger die Vorannahmen, desto mehr unbekannte Beispiele können klassifiziert werden

Vorzugskriterien (Bias)

- Bias (Vorzugskriterium): Vorschrift nach der Hypothesen gebildet werden
- Mögliche Vorzugskriterien: Verständlichkeit (für den menschlichen Benutzer), Klassifikationsgenauigkeit, Messaufwand für die verwendeten Deskriptoren, Berechnungs- und Speicheraufwand für die Hypothese
- Hypothesenraumbias: h gehöre zu einem beschränkten Raum von Hypothesen, logische Konjunktionen, lineare Schwellwertfunktionen, Geraden, Polynome, 3-NN (Nearest Neighbor),...
- Präferenzbias: Ordnung auf dem Raum der Hypothesen, wähle h mit der höchsten Präferenz: bevorzuge Hypothesen mit weniger Disjunktionen, bevorzuge kleinere Entscheidungsbäume
- Problem: Es existiert keine Funktion h , die konsistent mit allen Trainingsbeispielen ist, z.B. wegen verrauschter Trainingsdaten
- Unterschiedliche Ansätze möglich -> unterschiedliche Lösungen
- Anpassen des Hypothesenraumbias: Problem, da zwar sehr gute Klassifikation i.a. durch eine komplexe Hypothese erreicht wird aber Overfitting
- Anpassen des Präferenzbias: Wähle das h , das möglichst viele Beispiele richtig klassifiziert, Misklassifikation muss in Kauf genommen werden

3 UNÜBERWACHTE LERNVERFAHREN

3.1 K-MEANS CLUSTERING

- klassifiziert eine Datenmenge in eine a-priori vorgegebene Anzahl von Ballungen
- Grundidee
 - Definieren eines Mittelpunkts für jeden Cluster
 - Iterative Anpassung / Verbesserung

- Optimalitätskriterium: Minimierung der Abstände aller Datenpunkte von ihrem Ballungsmittelpunkt
- Gegeben: Menge X von unklassifizierten Trainingsbeispielen mit jeweils d Attributen $x_i = \langle attr1_i, attr2_i, \dots, attrd_i \rangle$ und Anzahl der gesuchten Ballungen k
- Gesucht: Einteilung der Trainingsmenge in Ballungen X_1, \dots, X_k (mit Zentren c_1, \dots, c_k) unter Minimierung von $\sum_{j=1}^k \sum_{x_i \in X_j} |x_i - c_j|$
- Algorithmus
 - Platziere k Punkte c_j im d -dimensionalen Raum als initiale Mittelpunkte der Ballungen
 - Klassifizierte jedes x_i gemäß der c_j neu bis sich die c_j nicht mehr ändern

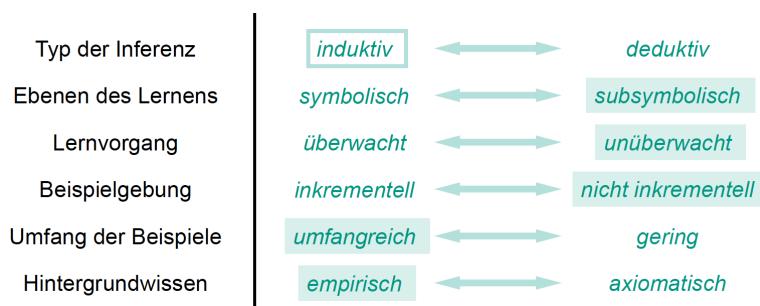
$$l = \arg \min_{j=1}^k |x_i - c_j| \Rightarrow x_i \in X_l$$
 und berechne die Mittelpunkte c_j zu jeder Ballung neu

$$c_j = \frac{\sum_{i=1}^{|X_j|} x_i}{|X_j|}$$
- Resultate hängen stark von der initialen Belegung der c_j ab: mehrere suboptimale Lösungen möglich, mögliche Lösung: Algorithmus mehrfach mit unterschiedlichen Startpunkten
- Resultate abhängig von der Metrik
- Resultate abhängig von Wahl von $k \rightarrow$ Anstoß mit mehreren k , aber Overfitting

3.1.1 FUZZY-K-MEANS-CLUSTERING

- bisher: jeder Datenpunkt in genau einem Cluster
- Abschwächung: jeder Datenpunkt hat eine abgestufte Zugehörigkeit zu jedem Cluster
- Problem: Laufzeit $O(kn)$ je Iteration

3.1.2 EINORDNUNG K-MEANS-CLUSTERING

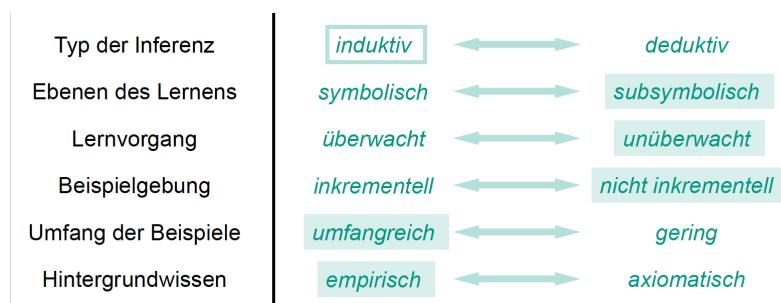


3.2 HIERARCHISCHES CLUSTERING

- k-means: flache Datenbeschreibung
- Ballungen können Subballungen und Subsubballungen haben
- Idee: Iteratives Vereinen von (Sub-)Clustern zu größeren Clustern

3.2.1 AGGLOMERATIVE HIERARCHICAL CLUSTERING

- Suche und verbinde zwei nächste Cluster
- einfaches Verfahren
- besser wenn k nicht bekannt, aber gewisse Aussagen über die Form der Cluster gefunden werden können
- unabhängig von Initialwerten
- finden der richtigen Parameter nötig
- rauschanfällig



3.3 BEGRIFFLICHE BALLUNGEN & COWEB

Klassische Ballungsverfahren

- Definition der Ähnlichkeit auf der Basis einer meist numerischen Ähnlichkeitsfunktion
- Ähnlichkeitsmaß ist kontextfrei, d.h. unabhängig von Umgebung
- keine Ausnutzung konzeptueller Zusammenhänge
- keine Verwendung von Gesalteigenschaften
- Ähnlichkeit hängt nicht von der Einfachheit der resultierenden Beschreibungen ab

Bildung von Begriffshierarchien

- Ziel: Wie können Beispiele in Klassen bezüglich ihrer Ähnlichkeit geordnet werden?
Klasseninformationen sind nicht gegeben.

- Beispielhafte Algorithmen: COBWEB (Lernen von Begriffen für Attribute mit symbolischen Wertebereichen), CLASSIT (Lernen von Begriffen für Attribute mit numerischen Wertebereichen)

3.3.1 COBWEB

- Lernen durch inkrementelles Aufbauen und Anpassen eines Strukturaums
- Repräsentation der Begriffshierarchie als Baum (Verzweigungen stehen für Einteilungen der Unteräume in verschiedene Kategorien, Blätter sind die speziellsten Begriffe)
- Es werden nominale Attributwerte gestattet
- Auswahl geeigneter Kategorien
 - Maß für die Ballungsnützlichkeit (category utility)
 - Eine Ballung c_i besitzt eine hohe Nützlichkeit, wenn man
 - * falls x zu c_i gehört, die Attributwerte von x mit hoher Wahrscheinlichkeit vorhersagen kann ($p(v|c)$) und
 - * falls die Attributwerte v von x gegeben sind, die Zugehörigkeit von x zu c_i mit hoher Wahrscheinlichkeit bestimmt werden kann ($p(c|v)$)
- Maximiere die Ähnlichkeit zwischen den Instanzen einer Klasse und gleichzeitig die Unterschiede zwischen den Klassen
- Man erhält ein Maß für die Vorhersagekraft und die Vorhersagbarkeit jedes Attributes in einem Konzept (Category Utility)

$$CU = \sum_{k=1}^K \sum_{i=1}^I \sum_{j=1}^{J(i)} P(A_i = V_{ij}) \cdot P(A_i = V_{ij}|C_k) \cdot P(C_k|A_i = V_{ij})$$

C_1, \dots, C_k = Partitionierung in Unterkonzepte, K = Anzahl der Klassen (Nachfolgeknoten), I = Anzahl der Attribute, $J(i)$ = Anzahl der Attributwerte des i -ten Attributes, V_{ij} = j -ter möglicher Wert für Attribut i , $P(A_i = V_{ij}|C_k)$ = Vorhersagbarkeit (predictability) eines Attributwertes, $P(C_k|A_i = V_{ij})$ = Vorhersagekraft (predictiveness) eines Attributwertes

Typ der Inferenz	<i>induktiv</i>	\longleftrightarrow	<i>deduktiv</i>
Ebenen des Lernens	<i>symbolisch</i>	\longleftrightarrow	<i>subsymbolisch</i>
Lernvorgang	<i>überwacht</i>	\longleftrightarrow	<i>unüberwacht</i>
Beispielgebung	<i>inkrementell</i>	\longleftrightarrow	<i>nicht inkrementell</i>
Umfang der Beispiele	<i>umfangreich</i>	\longleftrightarrow	<i>gering</i>
Hintergrundwissen	<i>empirisch</i>	\longleftrightarrow	<i>axiomatisch</i>

4 LERNTHEORIE, ALGORITHMENUNABHÄNGIGE VERFAHREN (FÜR ÜBERWACHTES INDUKTIVES LERNEN)

4.1 DEFINITION LERNMASCHINE

- Eine lernende Maschine wird bestimmt durch:
 - Hypothesenraum $h_\alpha : \alpha \in A$
 - Lernverfahren: die Methode um α_{opt} mit Hilfe von Lernbeispielen zu finden (benötigt Fehlerfunktion, Optimierungsmethode)
- Das resultierende (Entscheidungs-) Modell M_{opt} ist gegeben durch die Auswertung der optimalen Hypothese $h_{\alpha_{opt}}$, die durch die lernende(n) Maschine(n) bestimmt wird
- Problem: Welche Maschine ist zu wählen (Welches Lernverfahren?) Welches Modell soll gewählt werden?

4.2 PROBLEME BEIM LERNEN

- Statistisches Problem: Das Verfahren betrachtet einen - gemessen an der Menge von Trainingsdaten - „zu großen“ Hypothesenraum (auf Basis der Trainingsdaten eignen sich mehrere Hypothesen gleichermaßen gut)
- Komplexitätsproblem: Aufgrund der Komplexität des Problems kann das Lernverfahren nicht das Finden einer optimalen Lösung innerhalb des Hypothesenraumes garantieren. (Bei Verwendung von speziellen Verfahren oder Heuristiken besteht die Gefahr einer suboptimalen Lösung)
- Repräsentationsproblem: Der Hypothesenraum enthält keine ausreichend gute Approximationen der Zielfunktion/Konzept etc... (Das Lernverfahren kann einen gewünschten Approximationsgrad nicht liefern.)

4.3 ÜBERWACHTES LERNEN AUS BEISPIELEN

Lernen = Schätzen einer Abbildung (Hypothese) $h_\alpha(\vec{x}) = y$ mit Hilfe von (bekannten) Beispielen (Lernbeispielen) $(\vec{x}_1, y_1), \dots, (\vec{x}_n, y_n) \in X \times Y$ generiert von einem (nicht bekannten) System mit der Wahrscheinlichkeit $P(\vec{x}, y)$

4.4 FEHLER: EMPIRISCHER & REALER FEHLER

Der reale Fehler $E(h) = \int l(h(\vec{x}), y) dP(\vec{x}, y)$ bezüglich aller real vorkommenden Daten ist leider nicht berechenbar -> gute Schätzung nötig.

$$\text{Empirischer Fehler } E_D(h) = \frac{1}{|D|} \sum_{(\vec{x}, y) \in D} l(h(\vec{x}), y)$$

Fehlerminimierung durch Gradientenabstieg

4.5 OVERFITTING

- Definition: Die Tendenz der Maschine sich beim Lernen auf die Lernbeispiele zu spezialisieren (auswendig lernen)
- Formal: $h \in H - overfit = \exists h' \in H.s.d. \forall Testdaten E_{Lerndaten}(h) < E_{Lerndaten}(h') E_{Testdaten}(h) > E_{Testdaten}(h')$
- Lernfehler fällt, Testfehler steigt, Generalisierung fällt
- Erklärung: Lerndatenmenge und Testdatenmenge unterschiedlich (je nach Komplexität der Hypothese) -> unterschiedlicher Verlauf von $E_{Lern}(h)$ und $E_{Verifikation}(h)$
- Lösung: Repräsentative Beispiele (Anzahl und Art/Verteilung), Lernprozess durch den Verifikationsfehler steuern (Verifikationsdatensatz kann anschließend nicht für die Gütebestimmung genutzt werden), richtige Wahl und Suche der optimalen Hypothesen h_a

4.6 MODELLAUSWAHL (CROSSVALIDATION, BOOTSTRAP, ADABOOST)

- Cross-Validierung
 - Teile die Daten wiederholt in Lern- und Validierungsdaten
 - Bestimme darauf verschiedene Hypothesen (bzw. deren Parameter)
 - Berechne jeweils Generalisierung
 - Wiederhole
- n-fold-crossvalidation
 - Zerlege die Menge der Lerndaten in n Mengen
 - Trainiere jew. auf n-1 Mengen, teste auf 1 Menge
 - Wiederhole
- Leave One Out (Spezialfall)
 - Jeweils ein Beispiel für das Lernen weglassen
 - Addiere Fehler für weggelassene Beispiele
 - Wiederhole
- statistische Auswertung (Mittelwert, Varianz, ...) auf verschiedenen Modellen auch mit relativ wenig Lerndaten evaluieren

Bootstrap

- Grundgedanke: Wie kann man mit einfachen Verfahren mehr erreichen?
- In der Literatur oft - allgemeiner anekdotischer Hintergrund

- Baron Münchhausen: am eigenen Schopf aus dem Sumpf gezogen
- (für Informatiker) Computer startet zunächst ein einfaches Programm um dann vereinfachte Programme zu starten -> mächtige Leistung
- und beim Lernen
 - Ziehe zufällig (mit Zurücklegen) aus D jeweils $|D|$ Beispiele m Mal
 - Bestimme jeweils die Modell-Parameter
 - Wiederhole
 - Bestimme den Mittelwert, Varianz,... der Parameter des Modells
- Analyse der Güte/Stabilität
- mit weiteren Ansätzen höhere Güte des Modells erreichen

Bagging (Bootstrap aggregation)

- Variante des Bootstrap
- Verwende mehrere Modelle mit Bootstrap-Prinzip
- ziehe $n < |D|$ Beispiele mit Zurücklegen
- Bestimme die jeweiligen Modelle
- Kombinieren die Modelle z.B. durch gewichtete Summe
- Höhere Güte des Modells
- Aussagen über die Stabilität des Modells (große Abweichungen in den einzelnen Modellen = instabil)

Boosting/AdaBoost aus CV bekannt

4.7 WAS IST PAC (LERNBARKEIT)?

PAC = Probably Approximate Correct

Gegeben:

- eine Menge X von Instanzen der Länge n
- ein Konzept C
- ein Hypothesenraum H
- eine Lerndatenmenge D

Kann eine korrekte Hypothese aus $H, h(\vec{x}) = c(\vec{x}), \forall \vec{x} \in D$ gefunden werden?

- Nein

- aber eine ϵ -genaue: $E_D(h) \leq \epsilon, 0 \leq \epsilon \leq \frac{1}{2}$

- Approximate Correct

Kann diese sicher gefunden werden?

- Nein
- aber mit beliebiger Wahrscheinlichkeit $1 - \delta, 0 < \delta < \frac{1}{2}$
- Probably

Wie ist das Problem (Finden der Hypothese) lösbar?

- in polynomialer Zeit abh. von $\frac{1}{\delta}, \frac{1}{\epsilon}, n$
- mit Speicheraufwand abh. von: C

Und die Anzahl der benötigten Lerndaten (Stichprobenkomplexität) ist: $m \geq \frac{1}{\epsilon}(\ln(\frac{1}{\delta}) + \ln|H|)$

Und was heißt das?

- je größer die gewünschte Sicherheit
- je kleiner der zulässige Fehler
- je größer der Hypothesenraum
- um so größer die Anzahl der benötigten Daten

4.8 APPROXIMATION DES REALEN FEHLERS NACH VAPNIK UND CHERVONENKIS

- Vapnik-Chervonenskis (VC) Dimensionen: Die VC Dimensionen von H^α (Hypothesenraum) ist gleich der maximalen Anzahl von Datenpunkten (aus einer Menge S) die von H^α beliebig separiert werden
- Eine Abbildung (Hypothese) h separiert die Daten aus S wenn durch h zwei Untermen gen definiert werden $x|h(x) = 0$ und $x|h(x) = 1$
- $VC(h) =$ Maß für die Kapazität von lernenden Maschinen

Abschätzung des Testfehlers

- Nach Vapnik gilt mit Wahrscheinlichkeit $1 - \eta$: $E(h_\alpha) \leq E_{emp}(h_\alpha) + \sqrt{\dots \frac{VC(h_\alpha)}{N} \dots}$ mit VC-Dimensionen der lernenden Maschine $VC(h_\alpha)$, Anzahl der Lernbeispiele N , empirischem Fehler abhängig von $VC(h_\alpha)$ und $N E_{emp}(h_\alpha)$ und realem (zu minimierenden) Fehler $E(h_\alpha)$
- Lernerfolg ist abhängig von:
 - Kapazität der lernenden Maschine (so gering wie nötig)
 - Optimierungsmethode (so gut wie möglich)
 - Lernbeispiele (repräsentativ, so viele wie möglich)

4.9 WIE KANN KORREKTES LERNEN ERFOLGEN?

4.9.1 STRUCTURAL RISK MINIMIZATION

Ziel: finde eine Lösung für

$\min_{H_n} E_{emp}(h_\alpha) + \sqrt{\dots \frac{VC(h_\alpha)}{N} \dots}$ → finde $VC(h_\alpha)$ („Maschine“), N („Beispiele“) und α („Minimum des emp. Fehlers“)

Ideale Lösung

- Minimiere Summe (nicht Summanden)
- Strukturiere den Hypothesenraum
- Suche Optimum (Minimum für $E(h_\alpha)$)

Probleme

- Strukturierung
 - Berechnung der VC Dimension schwer, rechenintensiv, für viele Hypothesenräume unmöglich
- Optimierung
 - Finden, definieren der Hypothesenräume
 - große Kapazität → kleiner empirischer Fehler
 - geringe Kapazität → größerer Fehler
 - Minimierung von $E_{emp}(h_\alpha^n)$, $\forall H^n$

5 REINFORCEMENT LEARNING

- Problemstellung: Markov decision process (MDP)
- Lernziel: maximale Bewertung (reward) anhand von gesuchter Aktionssequenz (a_1, a_2, \dots, a_n)
- Problemdimensionen in RL
- Strategielernen (Policy-learning)
 - Optimale Strategie
 - State Value Function
 - Akkumulierte Bewertung
 - TD-Lernmethode (Temporal difference learning)

5.1 MARKOV DECISION PROCESS (DETERMINISTISCH)

- Autonomer Agent & Umwelt
 - Zustandsgetriebener Prozess
 - Sensorik - Erfassung (partiell) von Zuständen $s_t \in S$
 - Aktorik - Einwirkung auf die Umwelt durch Aktionen $a_t \in A$
 - Zustandsänderungen (meist bekannt oder beobachtbar)
 - * $\delta : (S \times A) \rightarrow S$
 - * $\delta(s_t, a_t) = s_{t+1}$
 - Markov-Bedingung: keine Abhängigkeit von der Vergangenheit
- Bewertung von Aktionen (direkt oder indirekt bekannt/messbar)
 - $r : (S \times A) \rightarrow R$
 - $r(s_t, a_t) = r_t$

5.2 ANWENDUNGSBEISPIELE

- Steuerung von Robotern (z.B. mobiler Roboter in Büroumgebung, Post holen, wenn welche da ist, dock-in Station wenn Batterie leer wird...)
- Spiele (Brettspiele: Schach, Back-Gammon)
- Optimierung und Planung (Optimierungen von Fertigungsprozessen, Planung Taxizentrale)

5.3 STRATEGIELERNEN - POLICY LEARNING

Finde die (optimale) Zielfunktion (target function) $\pi : S \rightarrow A$, $\pi(s_t) = a_t$ so dass die akkumulierte Bewertung (zum Ziel hin) $V^\pi(s_t) = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$ maximiert wird.

5.4 OPTIMALE STRATEGIE

- optimale Zielfunktion $\pi^*(s) = \operatorname{argmax}_\pi V^\pi(s)$, $\forall s$
- maximale akkumulierte Bewertung $V^*(s) = V^{\pi^*}(s)$
- rekursive Definition (Bellmann Gleichung): $V^*(s_t) = r_t + \gamma V^*(s_{t+1})$
- Problem: Wie erhält man $V^*(s)$

5.5 SIMPLE TEMPORAL DIFFERENCE LEARNING

- Idee: Lerne $\hat{V}^*(s)$ als Schätzung von $V^*(s)$ und wähle dann $\pi^*(s) = \arg\max_a[r(s, a) + \gamma \hat{V}^*(\delta(s, a))]$
- Problem: sehr langsames Lernen und hartes Ersetzen
- Optimierung: Initialisiere $\hat{V}^*(s)$ zufällig
- Problem: $r(s, a)\delta(s, a)$ müssen bekannt sein, on-policy-learning: $\pi^*(s)$ verwendet → langsames Lernen, nicht realistisch für echte Anwendungen!

5.6 PROBLEMDIMENSIONEN BEIM RL

- Zielfunktion Vorhersage: $V(s_{t+1})$ vs Aktionswahl a_t
- Bewertung $r(s, a)$ direkt vs verzögert
- Zustandsübergänge $\delta(s, a)$ deterministisch vs stochastisch
- Modell (Simulation) des Systems vorhanden vs nicht vorhanden
- Zustandsraum und Aktionsraum eindimensional vs hochdimensional, diskret vs kontinuierlich

5.7 DIE Q-FUNKTION

$Q(s, a)$ - maximale Bewertung, die erreicht werden kann im Zustand s durch die Aktion a

$Q(s, a) = r(s, a) + \gamma V^*(\delta(s, a))$ (Bellmann) mit $V^*(s) = \max_{a'} Q(s, a')$

rekursiv: $Q(s, a) = r(s, a) + \gamma \max_{a'} Q(\delta(s, a), a')$

Idee: Lerne $\hat{Q}(s, a)$, $\forall (s, a) \in S \times A$

Wähle best Aktion anhand einer Strategie z.B. $\pi^*(s) = \arg\max_a Q(s, a)$

5.8 SUCHSTRATEGIEN/EXPERIMENTIEREN

- Problem: Welche Aktion soll in einem Zustand gewählt werden?
- Aktion mit maximalem $\hat{Q}(s, a)$?
 - lokales Lernen nur bestimmter Aktionen
 - Aktionen die anfangs nicht gewählt wurden, werden nicht weiter ausgewertet obwohl sie eventuell bessere Ergebnisse liefern würden
- Probabilistische Auswahl: $P(a_i|s) = \frac{k^{\hat{Q}(s, a_i)}}{\sum_i k^{\hat{Q}(s, a_i)}}, k > 0$
- $\forall a_i \exists P(a_o|s) \neq 0$

5.9 EXPLORATION VS EXPLOITATION

Probabilistische Aktionsauswahl je nach Faktor k:

- klein: Exploration (globales Lernen, neue Aktionen untersuchen)
- groß: Exploitation (lokales Lernen, bekannte Aktionen untersuchen)

5.10 OPTIMIERUNGEN

- In jeder Lernepisode alle $\hat{Q}(s, a)$ vom Zustand s zum Ziel anpassen $\hat{Q}(s, a) \leftarrow \hat{Q}(s, a) + \alpha[r + \gamma \max_{a'} \hat{Q}(s', a') - \hat{Q}(s, a)]$
- Speichern von Bewertung r für jedes Paar (s, a) wenn $\hat{Q}(s, a)$ von $\hat{Q}(s', a')$ abhängig ist und sich $\hat{Q}(s', a')$ ändert, ändere auch $\hat{Q}(s, a)$
 - schnelle Konvergenz $\hat{Q} \rightarrow Q$ durch Anpassung mehrerer Werte
 - Speicheraufwand steigt
- Anwendung immer dann, wenn Aktionen hohen Zeitaufwand haben (z.B. in der Robotik)
- Lernen mit (adaptivem) Modell
 - meisten Lernschritte auf simulierter Umgebung
 - wenig Aktionen in realer Umgebung
 - Anpassung des Modells

5.11 REPRÄSENTATION, GENERALISIERUNG

- Problem: kontinuierlicher Zustandsraum
 - Speichern der Q-Werte in einer lookup-Tabelle unmöglich
 - sehr hohe Anzahl von Lerniterationen nötig
- Lösung: Kombination von RL mit Methoden höherer Generalisierung (z.B. Neuronalen Netzen)

5.12 LERNEN VON AKTIONSSEQUENZEN

5.12.1 WARUM LERNEN VON AKTIONSSEQUENZEN

- Bewertung erst nach einer Sequenz von Aktionen bekannt, z.B. Schach: selten ist nur ein Zug relevant
- Bewertung erst am Ziel, z.B. Spiel gewonnen?
- bei langen Aktionssequenzen kann erst am Ende der Sequenz gelernt werden
- nachfolgende Aktionen können für den schlechten Ausgang verantwortlich sein

5.12.2 TEMPORAL-DIFFERENCE-LEARNING

- Die Differenz folgender Schätzungen als Lernsignal für $\hat{Q}(s, a)$, $\hat{V}^*(s)$
- Vorwärtssicht (theoretisch)
 - Gewichtete Anpassung an direkt nachfolgender Schätzung (1-step) oder
 - Gewichtete Anpassung an n Schritte nachfolgender Schätzung (n-step)
- Rückwärtige Sicht des TD-Lernens (praktisch): Fehlersinglae (temporal differences) in den Schätzungen werden nach hinten weitergegeben
- Eligibility Traces (Verantwortlichkeitsspur)
 - Zustände für die Zustandsbewertung → V-Lernen
 - (Zustand/Aktion) für die Q-Wert Bewertung → Q-Lernen

6 SUPPORT VEKTOR METHODEN

6.1 LINEARE SUPPORT VEKTOR METHODE

- Problem: Klassifikation, Trenne 2 Mengen
- Lösung: Finde die beste Trenn-Gerade
- Intuition: Größe des Randes (margin) führt zu Generalisierung
- Hyperebene: $\vec{x} \in S : \vec{w}\vec{x} + b = 0, (\vec{w}, b) \in S \times R$
- negative Grenze $\vec{w}\vec{x} + b = -1$
- positive Grenze $\vec{w}\vec{x} + b = +1$
- Abstand $\frac{\vec{w} \cdot (\vec{x}_1 - \vec{x}_2)}{\|\vec{w}\|} = \frac{2}{\|\vec{w}\|}$
- Bedingung für die optimale Hyperebene: $\min_{i=1 \dots n} |\vec{w}\vec{x}_i + b| = 1$, \vec{x}_i - Lerndaten
- Der nächste Punkt hat den Abstand $\frac{1}{\|\vec{w}\|}$
- Der Abstand zwischen den 2 Klassen $\frac{2}{\|\vec{w}\|}$
- Entscheidungsfunktion eines Hyperebenen-Klassifikators $f_{\vec{w}, b}(\vec{z}) = \text{sign}(\vec{w}\vec{z} + b)$
- Optimierung: Hyperebene mit maximalem Abstand (margin) → Maximiere $\frac{2}{\|\vec{w}\|}$ → Minimiere $\|\vec{w}\|^2$ unter den Bedingungen $y_i(\vec{w}\vec{x}_i + b) \geq 1$ (d.h. die Daten werden korrekt klassifiziert)
- Laut Vapnik die Lernmaschine mit der kleinsten möglichen VC-Dimension (falls die Klassen linear trennbar sind)

- Minimierung mit Lagrange-Methode (finde den Sattelpunkt der Funktion) Problem: Daten sind oft nicht linear separierbar → Idee: Erlaube eine geringe Zahl von Missklassifikationen für höhere Generalisierung unter Änderung der Randbedingung $y_i(\vec{w}\vec{x}_i + b) \geq 1 - \xi$ mit $i = 1 \dots n, \xi \geq 0$ → reicht oft aber auch nicht → Nichtlineare Kernel-Methoden

6.2 NICHTLINEARE SUPPORT VEKTOR METHODE

- Idee: Transformiere die Daten in einen anderen Raum
- Problem: Transformationen + Rechnen in hoch-dimensionalen Räumen ist rechenintensiv
- Kernel-Trick: Daten kommen nur im Skalarprodukt Form, daher Berechnung mit einfachen Funktionen
 - Skalarprodukt $K(\vec{x}, \vec{y}) = \vec{x} \cdot \vec{y}$
 - Polynom (Vovk) $K(\vec{x}, \vec{y}) = (\vec{x} \cdot \vec{y} + c)^d$
 - Radiale Basis-Funktionen $K(\vec{x}, \vec{y}) = e^{-\frac{\|\vec{x} - \vec{y}\|^2}{2\sigma^2}}$
 - Sigmoid (Ähnlichkeit mit FF-Neuronalen Netzen) $K(\vec{x}, \vec{y}) = \tanh(\kappa(\vec{x} \cdot \vec{y}) + \theta)$

6.2.1 VERSION SPACE FÜR SVM

- Besondere Eigenschaft: Dualität Merkmal. u. Hypothesenraum
- Randbedingung korrekter Klassifikation $y_i(\vec{w}\vec{x}_i + b) \geq 1$
- Ungleichheit kann nach \vec{w} im Merkmalsraum und nach \vec{x} im Hypothesenraum interpretiert werden
- D.h. aber auch, dass jeder Datenpunkt $x_i, i = 1 \dots n$ eine Hyperebene im Hypothesenraum definiert s.d. gültige \vec{w} jeweils in einem Halbraum = reduzierter Hypothesenraum sind
- Heißt gleichzeitig dass wir nach dem \vec{w} suchen, dass den maximalen Abstand zu allen Hyperebenen der Datenpunkte hat → Mittelpunkt der Hyperkugel

Wieso SRM (Structural Risk Minimization) bei SVM?

- Suchraum für die Trennhyperebene wird während der Sattelpunkt-Suche eingeschränkt
- Irrelevante Daten fallen nicht mehr ins Gewicht
- VC-Dimension h fällt stetig: $h \leq D^2 |w|^2$
- Lösung = Mittelpunkt der Hyperkugel im Hypothesenraum

6.3 ERWEITERUNGEN

- Multiple Klassifikation (k Klassen)
 - Einer-gegen-Alle: k SVM's (für jede Klasse eine) Abstimmungsverfahren
 - Einer-gegen-Einen: $k(k - 1)/2$ SVM's Abstimmungsverfahren
 - Mehrfahrzugehörigkeit: k SVM's (für jede Klasse eine) Abstimmungsverfahren
 - k -class SVM von Watkins: kein Abstimmungsverfahren
- Gewichtete SVM - je ein C für positive und negative Klasse
- Dichte-Träger-Schätzung: Eine Funktion f ist gesucht, die für eine kleine Region, welche die meisten Lernbeispiele enthält, den Wert > 0 und sonst den Wert 0 (oder < 0) annimmt.

6.4 BEWERTUNG SVM

- Pro
 - Optimale Hyperebene → gute Lernergebnisse
 - Finden der optimalen VC-Dimension → korrektes Lernen
 - Verarbeitung hochdimensionaler Daten
 - Anwendungsspezifische Kernel (mit Datenverarbeitung)
 - Schnelle Auswertung
 - Viele Anwendungen: Klassifikation, Regression, PCA
 - Probabilistische Sicht!? → siehe ML II
 - Semi-überwachtes Lernen → siehe ML II
 - Aktives Lernen → siehe ML II
- Kontra
 - Vorverarbeitung extern (kein tiefes Lernen)
 - Finden des optimalen Kernels - aktuelle Forschung
 - Parametrisierung des Kernels - aktuelle Forschung
 - Speicher und Rechenaufwand (Trainieren)
 - Anzahl der SV abh. von Problem und Parameter

7 ENTSCHEIDUNGSBÄUME

7.1 GRUNDLAGEN

- Knoten repräsentieren einen Attributtest

- Zweige entsprechen Attributwerten
- Blätter entsprechen einer Aussage i.A. Klassifikation
- Für welche Problemstellungen eignen sich Entscheidungsbäume
 - Instanzen lassen sich als Attribut-Wert Paare beschreiben
 - Zielfunktion besitzt diskrete Ausgabewerte
 - Disjunkte Hypothesen erforderlich
 - Beispieldaten sind möglicherweise verrauscht
 - Beispieldaten enthalten evtl. fehlende Attributwerte
- Verfahren
 - ID3 (Quinlan, 1986): nicht-inkrementelles Verfahren
 - C4.5 (Quinlan, 1993): Verbesserung von ID3 durch generalisierte Regeln (Pruning), kommerzielles System
 - ID5R (Utgoff, 1989): inkrementelles Verfahren

7.2 ID3

1. $A \leftarrow$ Das beste Entscheidungsattribut für den nächsten Knoten.
2. Weise A als Entscheidungsattribut für den nächsten Knoten zu.
3. Füge für jeden möglichen Wert von A einen Nachfolgeknoten ein.
4. Verteile die Trainingsdaten gemäß ihrer Attributwerte auf die Nachfolgeknoten.
5. Terminiere wenn die Trainingsdaten perfekt abgebildet (klassifiziert) sind, sonst iterate über die Nachfolgeknoten.

Entropie

- Die Entropie als Maß der Homogenität der Trainingsdaten: $Entropie(S) = -p_+ \log_2 p_+ - p_- \log_2 p_-$
- Ziel ist es die Daten durch Festhalten eines Attributwertes v möglichst die Klasse + oder - einzuteilen, d.h. sukzessive die Entropie maximal zu reduzieren

Informationsgewinn

- $Gewinn(S, A) =$ Erwartete Reduzierung der Entropie durch die Einsortierung über Attribut A
- $Gewinn(S, A) = Entropie(S) - \sum_{v \in V(A)} \frac{|S_V|}{|S|} Entropie(S_V)$
- Ziel: Maximierung → Baum mit niedriger Tiefe

Suche im Hypothesenraum

- Es gibt typischerweise viele Entscheidungsbäume, die mit den Trainingsbeispielen konsistent sind
- Hypothesenraum ist bei Bäumen vollständig, d.h. Zielfunktion ist enthalten
- Suche der Hypothese: „Simple-to-complex“ oder „hill climbing“ nach Informationsgewinn
- Lokale Minima (wie bei allen hill climbing Algorithmen) möglich

Allgemein: Präferenz-/Restriktions-Bias

- Präferenzbias
 - Ordnung auf dem Raum der Hypothesen
 - Wähle Hypothese h mit der höchsten Präferenz
- Restriktionsbias
 - Einschränkung des Hypothesenraums z.B. auf lineare Schwellwertfunktionen

ID3: Induktiver Bias

- Präferenz für kleine Bäume und für Bäume, deren Attribute nahe der Wurzel einen hohen Informationsgewinn besitzen.
- ID3-Bias ist eine Präferenz für bestimmte Hypothesen, aber keine Restriktion des Hypothesenraums H
- Occam's Razor: Veborzuge die einfachste Hypothese , die mit den Trainingsdaten übereinstimmt.

Ocam's Razor (Warum sollten kurze Hypothesen bevorzugt werden?)

- Es gibt weniger kurze als lange Hypothesen: Eine kurze Hypothese, welche die Daten korrekt beschreibt, ist wahrscheinlich kein Zufall. Eine lange Hypothese, welche die Daten korrekt beschreibt, könnte hingegen Zufall sein.
- Kurze Bäume sind effizienter bezüglich interner Repräsentation und Auswertung

7.2.1 OVERFITTING

- ID3 Basisverfahren
 - Jeder Zweig wächst solange, bis die Trainingsbeispiele perfekt klassifiziert werden.
 - Dies basiert auf dem statistisch approximierten Informationsgewinn (Entropie, etc...)

- Dies kann zu Probleme führen, wenn die Daten verrauscht sind (z.B. Klasseninformationen) oder die Beispiele nicht repräsentativ sind (z.B. zu wenige)
- Fehler der Hypothese H auf
 - den Trainingsdaten: $Fehler_{Training}(h)$
 - der gesamten Verteilung D der Daten: $Fehler_D(h)$
- Definition: Eine Hypothese h overfittet die Daten D , wenn es eine alternative Hypothese h' gibt, so dass $Fehler_{Training}(h) < Fehler_{Training}(h')$ und $Fehler_D(h) > Fehler_D(h')$
- Lösungen zur Vermeidung von Overfitting
 - Frühzeitiges Stoppen des Baumwachstums
 - Nachträgliches „Prunen“ des Baumes (in der Praxis erfolgreicher)
- Kriterium für die Bestimmung der optimalen Baumgröße?
 - Separate Testdatenmenge
 - Statistischer Test auf den Trainingsdaten
 - Maß für die Kodierungskomplexität der Trainingsbeispiele und des Entscheidungsbaums (Prinzip der minimalen Beschreibungslänge)

Reduced Error Pruning

- Aufteilen in Trainings- und Testdaten
- Solange Pruning nicht negativ:
 - Evaluiere Auswirkungen des Entfernen jedes Knotens (+ Nachfolgern) auf Klassifikationsgüte bzgl. Testdaten
 - Entferne Knoten, dessen Entfernen die Klassifikationsrate bzgl. der Testdaten am meisten erhöht
→ liefert die kleinste Variante des akkuratesten Unterbaums.
- Problem: Bei wenigen Daten erhöht das Aufteilen der Daten die Fehleranfälligkeit noch weiter.

7.2.2 ERWEITERUNGEN

Attribute mit vielen Werten

- Problem: Attribute mit vielen Werten werden durch $Gewinn(S, A)$ ggü. solchen mit wenigen Werten bevorzugt
- Lösung: Bestrafung von Attributen, Verwende $GewinnAnteil(S, A) = \frac{Gewinn(S, A)}{SplitInformation(S, A)}$
und $SplitInformation(S, A) = - \sum_{i=1}^c \frac{|S_i|}{S} \log_2 \frac{|S_i|}{S}$

Kontinuierliche Attributwerte

- Gegeben: Attribut A mit kontinuierlichen Werten
- Vorgehen: A_c war, wenn $A > c$
- Problem: Wahl von Schwellwert c
 - Auswahl über Informationsgewinn: Sortierung der Beispiele gemäß ihrer Werte, optimaler Schwellwert liegt in der Mitte zwischen zwei benachbarten Beispielen mit unterschiedlichen Klassenzugehörigkeiten

Unbekannte Attributwerte

- Problem: Fehlende Attributwerte → wie solche Daten verwenden?
- Lösung: Sortiere alle Beispiele wie gewohnt in den EB ein wobei fehlende Attribute bekommen
 - den häufigsten Attributwert der Beispiele
 - den häufigsten Attributwert der Beispiele der gleichen Klasse
 - jedem Wert v_i mit Wahrscheinlichkeit p_i → Verteile das Beispiel gemäß p_i anteilig auf die Nachfolger (Umsetzung komplexer)
- Verfahren bei der Klassifikation entsprechend

Attribute mit Kosten

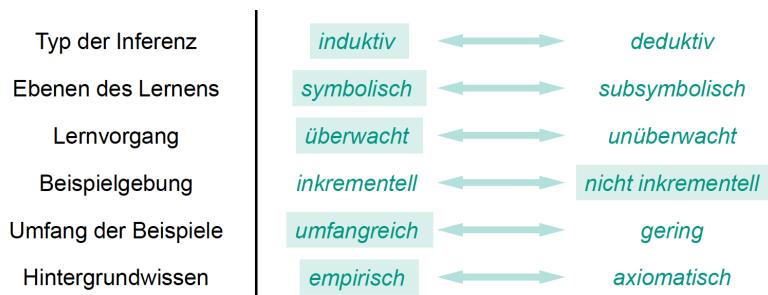
- Problem: Bestimmung der Attributwerte mit unterschiedlichen Kosten verbunden
- Lösung: Finden eines korrekten Entscheidungsbaums mit niedrigen Kosten durch Verwendung von $\frac{Gewinn^2(S,A)}{Kosten(A)}$ oder $\frac{2^{Gewinn(S,A)-1}}{Kosten(A)+1^w}$

ID3: Window (Lernmethode für große Datenmengen)

- Wähle zufällige Untermenge der Trainingsdaten aus (Window)
- Bestimme EB mit diesen Beispielen
- Klassifizierte alle übrigen Beispiele mit gelerntem EB
- Falls nicht alle Daten korrekt klassifiziert, füge einen Teil der falsch klassifizierten zum Fenster hinzu (zufällig ausgewählt) und erstelle neuen EB

7.2.3 ID3: ZUSAMMENFASSUNG

- Top down Wachstum der EB
- Vollständiger Hypothesenraum
- Induktiver Bias: Präferenz für kleine EB und solche, die Attribute mit hohem Informationsgewinn nahe der Wurzel besitzen.
- Wichtiges Problem: Overfitting → nachträgliches Prunen notwendig
- Erweiterungen: Kontinuierliche / fehlende Attributwerte, Einbeziehung von Kosten für Attribute



7.3 C4.5

- Quinlan, 1993
- Weiterentwicklung des ursprünglichen ID3-Algorithmus
- Unterstützt kontinuierliche Attribut-Werte
- Kann mit fehlenden Attributwerten umgehen
- Verwendet Rule Post-Pruning

7.3.1 RULE POST-PRUNING

1. Erstelle Baum wie gewohnt aus Trainingsdaten (Overfitting erlaubt)
2. Konvertiere Baum in äquivalente Menge von IF-THEN Regeln (IF-Teil enthält alle Attributtests eines Pfadds, THEN die Ausgabe/Klasse)
3. Prune (Generalisiere) die Regeln so lange sich ihre Vorhersagegenauigkeit nicht verschlechtert (durch Entfernen von Vorbedingungen)
4. Sortiere alle Regeln nach ihrer Klassifikationsgüte und verwende sie in dieser Reihenfolge

7.3.2 ABSCHÄTZUNG DER GÜTE DER REGELN (PESSIMISTISCHE ABSCHÄTZUNG)

- Bestimmung der Regelgüte auf den Trainingsdaten
- Verwenden von statistischen Verfahren (z.B. Urnenmodell → Binomialverteilung): Berechnung der Standardabweichung, Verwendung der unteren Grenze des gewählten Konfidenzintervalls als Regelgüte
- Diese Heuristik ist statistisch nicht ganz einwandfrei, führt in der Praxis jedoch dennoch zu guten Ergebnissen

7.3.3 UMWANDLUNG IN REGELN

Vorteile

- Unterscheidung zwischen verschiedenen Kontexten, in denen ein Entscheidungsknoten benutzt wird
- Keine Unterscheidung zwischen Attributen näher an der Wurzel und solchen näher an den Blättern (vereinfacht Prunen)
- Lesbarkeit für Menschen

Typ der Inferenz	<i>induktiv</i>	\longleftrightarrow	<i>deduktiv</i>
Ebenen des Lernens	<i>symbolisch</i>	\longleftrightarrow	<i>subsymbolisch</i>
Lernvorgang	<i>überwacht</i>	\longleftrightarrow	<i>unüberwacht</i>
Beispielgebung	<i>inkrementell</i>	\longleftrightarrow	<i>nicht inkrementell</i>
Umfang der Beispiele	<i>umfangreich</i>	\longleftrightarrow	<i>gering</i>
Hintergrundwissen	<i>empirisch</i>	\longleftrightarrow	<i>axiomatisch</i>

7.4 ID5R

- Utgoff, 1989
- inkrementelles Verfahren, d.h. sukzessives Einbringen von Beispielen in den Aufbau des EB
- Ergebnis ist äquivalent zu einem von ID3 erzeugten Baum

7.4.1 REPRÄSENTATION DER EB

- Antwortknoten (Blätter): Klassenbezeichner, Beschreibungen der Instanzen, die zu einer Klasse gehören
- Entscheidungsknoten: Attributtest mit Zweigen für jeden Attributwert, für jeden Attributwert Zähler für die positiven und negativen Beispiele, zusätzlich Zähler für noch ausstehenden Attribut-Tests, → ermöglicht Berechnung des Informationsgewinns auf jeder Ebene ohne die bisherigen Beispiele erneut zu betrachten

7.4.2 BAUM UPDATE ALGORITHMUS

Gegeben: Bestehender Entscheidungsbaum EB, neue Instanz I
Algorithmus

1. Wenn EB leer, dann gib einen Antwortknoten mit der Klasse von I zurück
2. Wenn EB ein Antwortknoten mit er gleichen Klasse wie I, dann füge I zur Menge der Instanzen dieses Knotens hinzu
3. Sonst
 - a) Wenn EB Antwortknoten, dann Umwandlung in Entscheidungsknoten mit beliebigem Testattribut
 - b) Aktualisiere die Zähler des Entscheidungsknoten (für Testattribut und alle anderen Attribute)
 - c) Ist das Testattribut nicht optimal, dann
 - i. Restrukturiere Baum so, dass Attribut mit höchstem Informationsgewinn Testattribut wird
 - ii. Wähle Testattribut mit höchstem Informationsgewinn rekursiv in den Unterbaum
 - d) Aktualisiere rekursiv den gemäßt des Attributwerts von I gewählten Unterbaum

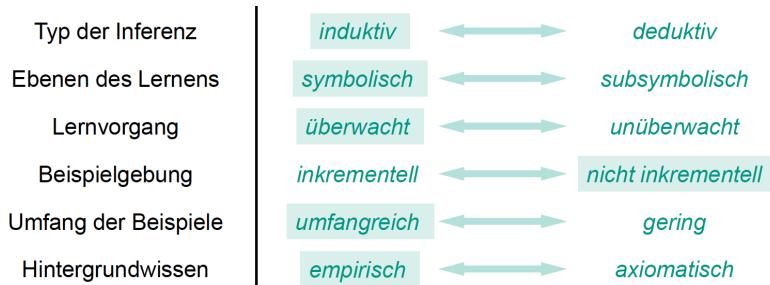
7.4.3 REKONSTRUIERUNG

1. Wenn Attribut A_{neu} mit höchstem Informationsgewinn an der Wurzel terminiere
2. Sonst
 - a) Ziehe A_{neu} rekursiv an die Wurzel jedes direkten Unterbaums. Falls erforderlich wandle jeden Antwortknoten in Entscheidungsknoten mit Testattribut A_{neu} um
 - b) Transponiere den Baum so, dass A_{neu} an der Wurzel des neuen Baums A_{alt} an der Wurzel des neues Baums und A_{alt} an der Wurzel jedes direkten Unterbaumes steht.

Typ der Inferenz	<i>induktiv</i>	\longleftrightarrow	<i>deduktiv</i>
Ebenen des Lernens	<i>symbolisch</i>	\longleftrightarrow	<i>subsymbolisch</i>
Lernvorgang	<i>überwacht</i>	\longleftrightarrow	<i>unüberwacht</i>
Beispielgebung	<i>inkrementell</i>	\longleftrightarrow	<i>nicht inkrementell</i>
Umfang der Beispiele	<i>umfangreich</i>	\longleftrightarrow	<i>gering</i>
Hintergrundwissen	<i>empirisch</i>	\longleftrightarrow	<i>axiomatisch</i>

7.5 RANDOM FORESTS

- Mehrere Entscheidungsbäume (=Wald) erstellen (einfach, unkorreliert, zufällige Wahl von Attributen)
- Für eine Klassifikation darf jeder Baum in diesem Wald eine Entscheidung treffen und die Klasse mit den meisten Stimmen entscheidet die endgültige Klassifikation
- Eigenschaften: Schelles Training da einzelne Entscheidungsbäume kleiner, Trainingszeit steigt linear mit der Anzahl der Bäume, parallelisierbar (sowohl Training als auch Evaluation), effizient für große Datenmengen
- Vergleich zu Standard-Entscheidungsbäumen
 - Kein Abschneiden der Bäume (Pruning) → Overfitting erlaubt
 - Attributwahl auf zufälliger Untermenge aller Attribute (Randomisierung)
- Eigenschaften der erstellten Bäume
 - Jeder Baum sollte für sich ein guter Klassifikator sein
 - Die Bäume sollten untereinander möglichst unkorreliert sein
- Randomisierungsmöglichkeiten
 - Bootstrapping: Aus N Trainingsdaten werden N Trainingsbeispiele mit zurückgelegten gezogen. Baum hat so ca. 63% der Größe verglichen mit allen Trainingsdaten
 - Auswahl des Attributtests aus einer Teilmenge der vorhandenen Attributtests
 - The main secret is to inject the „right kind of randomness“
- Anwendung z.B. Real time head pose estimation
- Ergebnis setzt sich aus den votes der Einzelbäume zusammen



8 LERNEN NAH BAYES

8.1 MOTIVATION

8.1.1 WAS IST LERNEN NACH BAYES?

Statistische Lernverfahren

- Kombiniere vorhandenes Wissen (a priori Wahrscheinlichkeiten) mit beobachteten Daten
- Hypothesen können mit einer Wahrscheinlichkeit angegeben werden
- Jedes Beispiel kann die Glaubwürdigkeit einer bestehenden Hypothese erhöhen oder verringern: kein Ausschluss bestehender Hypothesen
- Mehrere mögliche Hypothesen können gemeinsam ausgewertet werden, um genauere Ergebnisse zu erzielen

8.1.2 WARUM LERNEN NACH BAYES?

Erfolgreiche Lernverfahren:

- Naiver Bayes-Klassifikator
- Bayes'sche Netze

Analyse anderen Lernverfahren: „Gold-Standard“ für die Beurteilung von (nicht statistischen) Lernverfahren

8.1.3 LERNEN NACH BAYES: SCHWIERIGKEITEN

- Initiales Wissen über viele Wahrscheinlichkeiten notwendig (aber: oft Schätzungen basierend auf Hintergrundwissen, vorhandenen Daten, etc. möglich)
- Erheblicher Rechenaufwand für optimale Bayes'sche Hypothese im allgemeinen Fall
 - Linear mit Anzahl der möglichen Hypothesen
 - Aber: In speziellen Fällen deutliche Reduzierung des Rechenaufwands möglich

8.1.4 WAHRSCHEINLICHKEITSTHEORIE

- Produktregel: Konjunktion zweier Ereignisse A und B $P(A \wedge B) = P(A|B)P(B) = P(B|A)P(A)$
- Summenregel: Disjunktion zweier Ergebnisse A und B $P(A \vee B) = P(A) + P(B) - P(A \wedge B)$
- Theorem der totalen Wahrscheinlichkeit: Für sich gegenseitig ausschließende Ereignisse A_1, \dots, A_n mit $\sum_{i=1}^n P(A_i) = 1$ gilt $P(B) = \sum_{i=1}^n P(B|A_i)P(A_i)$

8.2 THEOREM VON BAYES

- $P(h)$ Wahrscheinlichkeit, dass h aus H gültig ist (a priori, d.h. vor Beobachtung von D)
- $P(D)$ Wahrscheinlichkeit, dass D als Ereignisdatensatz auftaucht (ohne Wissen über gültige Hypothese)
- $P(D|h)$ Wahrscheinlichkeit des Auftretens von D in einer Welt, in der h gilt

- $P(h|D)$ Wahrscheinlichkeit, dass h wahr ist gegeben die beobachteten Daten D (a posteriori)

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

Herleitung

$$\begin{aligned} P(A \wedge B) &= P(A \wedge B) \\ P(B|A)P(A) &= P(A|B)P(B) \\ P(B|A) &= \frac{P(A|B)P(B)}{P(A)} \end{aligned}$$

8.3 MAP- / ML-HYPOTHESEN

Ziel: Finden der Hypothese h aus H mit der größten Wahrscheinlichkeit gegeben die beobachteten Daten D . Dies ist die Maximim a posteriori (MAP) Hypothese.

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

Unter der Annahme $P(h_i) = P(h_j)$ lässt sich diese zur Maximum Likelihood (ML) Hypothese vereinfachen: $h_{ML} = \arg \max_{h_i \in H} P(D|h_i)$

Brute Force Lernen von MAP-Hypothesen

1. Berechne für jede Hypothese $h \in H$ die a posteriori Wahrscheinlichkeit $P(h|D) = \frac{P(D|h)P(h)}{P(D)}$
2. Gib die Hypothese h_{MAP} mit der größten a posteriori Wahrscheinlichkeit aus $h_{MAP} = \arg \max_{h \in H} P(D|h)P(h)$

8.4 KONZEPTLERNEN

Problemstellung

- Endlicher Hypothesenraum H über dem Raum der Instanzen X
- Aufgabe: Lernen eines Zielkonzepts $c : X \rightarrow \{0, 1\}$
- Feste Sequenz von Instanzen: $\langle x_1, \dots, x_m \rangle$
- Sequenz der Zielwerte $D = \langle d_1, \dots, d_m \rangle$

Annahmen

- Trainingsdaten D sind nicht verrauscht
- Zielkonzept c ist in H enthalten
- Kein Grund a priori anzunehmen, dass irgendeine Hypothese wahrscheinlicher ist als eine andere

$$\text{Vorwissen } P(D|h) = \begin{cases} 1, & \text{falls } h(x_i) = d_i, \forall d_i \in D \\ 0, & \text{sonst} \end{cases} \quad \text{mit } P(h) = \frac{1}{|H|}$$

Berechnung der a posteriori Wahrscheinlichkeit:

$$1. \text{ Fall (konsistente Hypothesen)}: P(h|D) = \frac{\frac{1}{|H|}}{P(D)} = \frac{1 \cdot \frac{1}{|H|}}{\sum_{h-\text{konsistent}} P(D|h)P(h)} = \frac{1 \cdot \frac{1}{|H|}}{\frac{|VS_{H,D}|}{|H|}} = \frac{1}{|VS_{H,D}|}$$

$$2. \text{ Fall (sonst)}: P(h|D) = \frac{0 \cdot P(h)}{P(D)} = 0$$

$$\text{Ergebnis: } P(h|D) = \begin{cases} \frac{1}{|VS_{H,D}|} & \text{falls } h \text{ konsistent mit } D \\ 0 & \text{sonst} \end{cases}$$

- Definition: Ein Lernverfahren ist ein konsistenter Lerner, wenn es eine Hypothese liefert, die keine Fehler auf den Trainingsdaten macht.
- Unter obigen Voraussetzungen gibt jeder konsistente Lerner eine MAP-Hypothese aus
- Methode um induktiven Bias auszudrücken

Lernen einer reell-wertigen Funktion

- Gesucht: reell-wertige Zielfunktion f
- Gegeben: Beispiele $\langle x_i, d_i \rangle$ mit verrauschten Trainingswerten für d_i :
 - $d_i = f(x_i) + e_i$
 - e_i ist eine Zufallsvariable (Rauschen), die unabhängig für alle x_i entsprechend einer Normalverteilung mit Mittelwert $\mu = 0$ gezogen wird
- Die Maximum Likelihood Hypothese h_{ML} ist diejenige, welche die Summe der Fehlerquadrate minimiert: $h_{ML} = \arg \min_{h \in H} \sum_{i=1}^m (d_i - h(x_i))^2$

Klassifikation neuer Instanzen

- Bisher: Suche nach der Hypothese mit der größten Wahrscheinlichkeit gegeben die Daten D
- Jetzt: Welches ist die wahrscheinlichste Klassifikation v_j einer neuen Instanz x?

8.5 OPTIMALER BAYES-KLASSIFIKATOR

- Optimale Klassifikation nach Bayes: $v_{OB} = \arg \max_{v_j \in V} \sum_{h_i \in H} P(v_j|h_i)P(h_i|D)$
- Vorteil: Kein anderes Klassifikationsverfahren (bei gleichem Hypothesenraum und Vorwissen) schneidet im Durchschnitt besser ab!
- Nachteil: Sehr kostenintensiv bei großer Hypothesenzahl!

Gibbs Algorithmus (Ensenmble Lernen):

- Wähle h aus H zufällig gemäß $P(h|D)$

- Nutze $h(x)$ als Klassifikation von x
- Bestimme Erwartungswert wie vorher
- Eigenschaft: Unter bestimmten Annahmen gilt $E[\text{error}_{\text{Gibbs}}] \leq 2E[\text{error}_{\text{BayesOptimal}}]$

8.6 NAIVER BAYES-KLASSIFIKATOR

- Gegeben
 - Instanz x : Konjunktion von Attributen $\langle a_1, a_2 \dots a_n \rangle$
 - Endliche Menge von Klassen $V = v_y, \dots, v_m$
 - Menge klassifizierter Beispiele
- Gesucht: Wahrscheinlichste Klasse für eine neue Instanz
 $v_{MAP} = \arg \max_{v_j \in V} P(v_j | a_1, a_2 \dots a_n) = \arg \max_{v_j \in V} \frac{P(a_1, a_2 \dots a_n | v_j) P(v_j)}{P(a_1, a_2 \dots a_n)} = \arg \max_{v_j \in V} P(a_1, a_2 \dots a_n | v_j) P(v_j)$
- $P(v_i)$ lässt sich leicht aus dem Auftreten der Klasse v_i in der Trainingsmenge berechnen
 - einfaches Zählen.
- $P(a_1, a_2 \dots a_n | v_j)$ ist schwerer zu berechnen: Auszählen aller Kombinationen über Attributwerte → dazu ist eine riesige Trainingsmenge notwendig
- Vereinfachende Annahme (a_i bedingt unabhängig): $P(a_1, a_2 \dots a_n | v_j) = \prod_i P(a_i | v_j)$
- Naiver Bayes-Klassifikator: $v_{NB} = \arg \max_{v_j \in V} P(v_j) \prod_i P(a_i | v_j)$

Bedingte Unabhängigkeit (Definition): X ist bedingt unabhängig von Y gegeben Z , wenn die Wahrscheinlichkeitsverteilung von X bei gegebenem Wert von Z unabhängig vom Wert von Y ist: $(\forall x_i, y_i, z_k) P(X = x_i | Y = y_i, Z = z_k) = P(X = x_i | Z = z_k)$ oder kompakter $P(X|Y, Z) = P(X|Z)$

Beispiel Donner ist bedingt unabhängig von Regen gegeben Blitz $P(\text{Donner}|\text{Regen}, \text{Blitz}) = P(\text{Donner}|\text{Blitz})$

Zusammenfassung

- $P(v_j)$ und $P(a_i | v_j)$ werden basierend auf den Häufigkeiten in den Trainingsdaten geschätzt
- Wahrscheinlichkeiten für Klassifikation entspricht gelernter Hypothese
- Neue Instanzen werden klassifiziert unter Anwendung obiger MAP Regel
- Wenn Annahme (bedingte Unabhängig der Attribute) erfüllt ist, ist v_{NB} äquivalent zu einer MAP-Klassifikation
- keine explizite Suche im Hypothesenraum!

Schätzen von Wahrscheinlichkeiten

- Problem: Was, wenn für eine Klasse v_j ein Attribut a_i einen bestimmten Wert in den Daten gar nicht annimmt? $\hat{P}(a_i|v_j) = 0 \rightarrow \hat{P}(v_j) \prod_i \hat{P}(a_i|v_j) = 0$
- Lösung $\hat{P}(a_i|v_j) \leftarrow \frac{n_c + mp}{n+m}$ (m-Laplace Schätzer) mit n - Anzahl der Beispiele mit $v = v_j$, n_c - Anzahl der Beispiele mit $v = v_j$ und $a = a_j$, p - A priori Wahrscheinlichkeit für $\hat{P}(a_i|v_j)$ z.B. $p = \frac{1}{|Werte(a_i)|}$ und m - Anzahl der virtuellen Beispiele gewichtet mit a priori Wahrscheinlichkeit p

Klassifikation von Texten → nicht relevant?!

8.7 BAYES'SCHE NETZE

- Naive Bayes-Annahme der bedingten Unabhängigkeit oft zu restriktiv
- Ohne solche, vereinfachenden Annahmen ist Lernen nach Bayes jedoch oft nicht möglich.
- Bayes'sche Netze beschreiben bedingte Abhängigkeiten/Unabhängigkeiten bzgl. Untermengen von Variablen
- Erlauben somit die Kombination von a priori Wissen über bedingte (Un-)Abhängigkeiten von Variablen mit den beobachteten Trainingsdaten.

Repräsentieren eine gemeinsame Wahrscheinlichkeitsverteilung von Zufallsvariablen

- Gerichteter, azyklischer Graph
- Jede Zufallsvariable wird durch einen Knoten im Graph repräsentiert
- Definition: X ist Nachfolger von Y, wenn ein gerichteter Pfad von Y nach X existiert.
- Die Kanten repräsentieren die Zusicherung, dass eine Variable von ihren Nicht-Nachfolgern bedingt unabhängig ist, gegeben ihre direkten Vorgänger
- Lokale Tabellen mit bedingten Wahrscheinlichkeiten für jede Variable gegeben ihre direkten Vorgänger
- Es gilt: $P(y_1, \dots, y_n) = \prod_{i=1}^n P(y_i | \text{Vorgänger}(Y_i))$ wobei $\text{Vorgänger}(Y_i)$ die Menge der direkten Vorgänger von Y_i ist.

8.7.1 INFERENCE

Wie lassen sich die Werte einer oder mehrerer Netzvariablen bestimmen, gegeben die beobachteten Werte von anderen?

- Netz enthält alle benötigten Informationen
- Ableitung einer einzigen Variable ist einfach

- Aber: Der allgemeine Fall ist NP-vollständig

In der Praxis:

- Einige Netztopologien erlauben exakte Inferenz
- Verwendung von Monte Carlo Methoden zur Zufallssimulation von Netzen: Berechnung von approximierten Lösungen

8.7.2 LERNEN

- Aufgabenstellungen: Struktur des Netzes bekannt oder unbekannt, alle Variablen direkt beobachtbar oder nur teilweise
- Struktur, alle Variablen beobachtbar: Lernen wie für Naiven Bayes-Klassifikator
- Struktur bekannt, nur einige Variablen beobachtbar: Gradientenanstieg, EM
- Struktur unbekannt: Heuristische Verfahren

8.8 EM (EXPECTATION MAXIMIZATION)-ALGORITHMUS

Problemstellungen:

- Daten sind nur partiell beobachtbar
- Unüberwachtes Clustering (Zielwert ist nicht beobachtbar)
- Überwachtes Lernen (einige Attribute der Instanzen sind nicht beobachtbar)

Anwendungen:

- Trainieren von Bayes'schen Netzen
- Lernen von Hidden Markov Modellen

EM-Mixtur aus k Gaußverteilungen

- Generierung jeder Instanz x wie folgt
 - Wähle eine der k Gaußverteilungen mit gleichmäßiger Wahrscheinlichkeit
 - Generiere eine Instanz zufällig entsprechend der gewählten Gaußverteilung

EM für die Bestimmung der k Mittelwerte \mathbf{I}

- Gegeben
 - Instanzen aus X generiert entsprechend einer Mixtur aus k Gaußverteilungen
 - Unbekannte Mittelwerte $\langle u_1, \dots, u_k \rangle$
 - Es ist unbekannt welche Instanz x_i entsprechend welcher Gaußverteilung generiert wurde

- Gesucht: Maximum Likelihood Schätzung für $h = \langle \mu_1, \dots, \mu_k \rangle$
- Erweiterte Sicht: Beschreibung der Instanzen durch: $y_i = \langle x_i, z_{i1}, z_{i2} \rangle$
 z_{ij} ist 1, falls x_i entsprechend der j-ten Gaußverteilung gezogen wurde, sonst 0
- x_i beobachtbar, z_{ij} nicht beobachtbar

Eigenschaften des EM-Algorithmus

- Konvergiert gegen eine lokale Maximum Likelihood Hypothese und liefert Schätzungen für die versteckten Variablen z_{ij} .
- Sucht die Maximum Likelihood Hypothese h' , welche $E[\ln P(Y|h')]$ maximiert, wobei Y die vollständigen Daten sind (beobachtbare plus versteckte Variablen) und der Erwartungswert über die möglichen Werte der versteckten Variablen berechnet wird

Allgemeines EM-Problem

- Gegeben
 - Beobachtbare Daten $X = x_1, \dots, x_m$
 - Nicht beobachtbare Daten $Z = z_1, \dots, z_m$
 - Parametrisierte Wahrscheinlichkeitsverteilung $P(Y|h)$, wobei $Y = y_1, \dots, y_m$ die vollständigen Daten $Y = X \cup Z$ sind und h die Parameter sind
- Gesucht: Hypothese h , welche $E[\ln P(Y|h)]$ (lokal) maximiert

Allgemeines EM-Verfahren

- Definiere Likelihood Funktion Q , welche $Y = X \cup Z$ unter Verwendung der beobachtbaren Daten X und der aktuellen Parameter h berechnet, um Z zu schätzen. $Q \leftarrow E[\ln P(Y|h^*)|h, X]$
- E-Schritt: Berechne $P(Z|X, h)$ unter Verwendung der aktuellen Hypothese h und der beobachtbaren Daten X
- M-Schritt: Ersetze Hypothese h mit Hypothese h' , welche die Q -Funktion maximiert

$$h' \leftarrow \arg \max_{h'} Q$$

8.9 ZUSAMMENFASSUNG

- Bayes-Methoden ermitteln a posteriori Wahrscheinlichkeiten für Hypothesen basierend auf angenommenen a priori Wahrscheinlichkeiten und beobachteten Daten.
- Mit Bayes-Methoden kann wahrscheinlichste Hypothese (MAP-Hypothese) bestimmt werden („optimale“ Hypothese).
- Der Optimale Bayes-Klassifikator bestimmt die wahrscheinlichste Klassifikation einer neuen Instanz aus den gewichteten Vorhersagen aller Hypothesen.

- Der Naive Bayes-Klassifikator ist ein erfolgreiches Lernverfahren. Annahme: bedingte Unabhängigkeit der Attributwerte.
- Bayes-Methoden erlauben die Analyse anderer Lernalgorithmen, die nicht direkt das Bayes-Theorem anwenden.
- Bayes'sche Netze beschreiben gemeinsame Wahrscheinlichkeitsverteilungen mit Hilfe eines gerichteten Graphen und lokalen Wahrscheinlichkeitstabellen.
- Bayes'sche Netze modellieren bedingte Unabhängigkeiten in Untermengen von Variablen. Weniger restriktiv als der Naive Bayes-Klassifikator.
- Der EM-Algorithmus erlaubt den Umgang mit nicht beobachtbaren Zufallsvariablen.

Typ der Inferenz	<i>induktiv</i>	\longleftrightarrow	<i>deduktiv</i>
Ebenen des Lernens	<i>symbolisch</i>	\longleftrightarrow	<i>subsymbolisch</i>
Lernvorgang	<i>überwacht</i>	\longleftrightarrow	<i>unüberwacht</i>
Beispielgebung	<i>inkrementell</i>	\longleftrightarrow	<i>nicht inkrementell</i>
Umfang der Beispiele	<i>umfangreich</i>	\longleftrightarrow	<i>gering</i>
Hintergrundwissen	<i>empirisch</i>	\longleftrightarrow	<i>axiomatisch</i>

9 HIDDEN MARKOV MODELLE

9.1 MOTIVATION

Signale in der realen Welt sind

- verrauscht (erfassbar)
- besitzen nicht-deterministische Eigenschaften

Stochastische Modelle von Signalen

- z.B. Markov Prozess

Nutzen von stochastischen Signalmodellen

- Erkennung der Signale (Sprache, Gesten)
- Theoretische Beschreibung von signalverarbeitenden Systemen
- Simulationen
- Arbeiten in der Praxis oft außerordentlich gut

9.2 DISKRETER MARKOV PROZESS

- N disrete Zustände S
- Zeitpunkte der Zustandsübergänge t
- Aktueller Zustand zur Zeit t q_t

Markov-Bedingung: Die Wahrscheinlichkeit einen Zustand zu erreichen ist nur von seinem direkten Vorgängerzustand abhängig.

9.3 HIDDEN MARKOV MODELLE

- Beobachtung ist eine stochastische Funktion des Zustandes (Zustände können nur indirekt beobachtet werden)
- HMMs sind ein doppelt stochastischer Prozess: Der zugrunde liegende stochastische Prozess kann nur indirekt durch eine andere Menge von stochastischen Prozessen beobachtet werden, die eine Beobachtungssequenz produziert.
- Definition: Ein HMM ist ein Fünf-Tupel $\lambda = S, V, A, B, \Pi$ mit S - Menge der Zustände $S = S_1, S_2, \dots, S_N$, q_t - Zustand zur Zeit t , V - Menge der Ausgabezeichen $V = v_1, \dots, v_m$, A - Matrix der Übergangswahrscheinlichkeiten $A = (a_{ij})$, a_{ij} ist die Wahrscheinlichkeit, dass S_j nach S_i kommt, B - Menge der Emissionswahrscheinlichkeiten, $b_i(k)$ ist die Wahrscheinlichkeit, v_k im Zustand S_i zu beobachten und Π - die Verteilung der Anfangswahrscheinlichkeiten, π_i ist die Wahrscheinlichkeit, dass S_i der erste Zustand ist

9.4 DIE 3 GRUNDLIEGENDEN PROBLEME

1. Evaluationsproblem: Wie gut erklärt ein Modell eine Beobachtungssequenz
 - Gegebenen: Modell $\lambda = S, V, A, B, \Pi$
 - Gesucht: Wahrscheinlichkeit $P(O|\lambda)$ d.h. für die Ausgabe $O = O_1 O_2 \dots O_T$
2. Dekodierungsproblem: Finden der korrekte verborgenen Zustandssequenz (Optimalitätskriterium?)
 - Gegeben: Ausgabesequenz O und Modell λ
 - Gesucht: wahrscheinlichste Zustandsfolge Q , die O erklärt
3. Lern- oder Optimierungsproblem (Optimierung der Modellparameter)
 - Gegeben: Ausgabesequenz O und Suchraum für Modelle
 - Gesucht: Anpassung der Parameter $\lambda = S, V, A, B, \Pi$ so dass O besser erklärt wird

Beispiel: Worterkennner

- Aufbau von HMMs für jedes zu erkennende Wort (P3)

- Verstehen der aufgebauten Modelle (P2) und damit sinnvolle Verbesserungen (kann auch für Erkennung genutzt werden)
- Erkennen unbekannter Wörter durch Finden des besten Modells dafür (P1)

9.5 LÖSUNGEN DER PROBLEME

9.5.1 EVALUATIONSPROBLEM

Vorwärts-Algorithmus

1. Initialisierung: $\alpha_1(i) = \pi_i b_i(O_1), 1 \leq i \leq N$
2. Induktion: $\alpha_t(i) \rightarrow \alpha_{t+1}(j)$

$$\alpha_{t+1}(j) = [\sum_{i=1}^N \alpha_t(i) a_{ij}] b_j(O_{t+1}), 1 \leq t \leq T-1 \text{ und } 1 \leq j \leq N$$
3. Terminierung $P(O|\lambda) = \sum_{i=1}^N \alpha_T(i)$

Rückwärts-Algorithmus

- Definiere $\beta_t(i) = P(O_{t+1}O_{t+2}\dots O_T | q_t = S_i, \lambda)$ als Wahrscheinlichkeit, dass $O_{t+1}O_{t+2}\dots O_T$ beobachtet wird und S_i = Zustand zum Zeitpunkt t , Wert wird später für Lernalgorithmus benötigt
- Induktion
 1. Initialisierung: $\beta_T(i) = 1, 1 \leq i \leq N$
 2. Induktionsschritt: $\beta_t(i) = \sum_{j=1}^N a_{ij} b_j(O_{t+1}) \beta_{t+1}(j), t = T-1, T-2, \dots, 1, 1 \leq i \leq N$

Definition der optimalen Zustandsfolge?

- Ein Ansatz: Wahl der Zustände q_t , die unabhängig voneinander am wahrscheinlichsten sind
- Definiere: Wahrscheinlichkeit, zum Zeitpunkt t im Zustand S_i zu sein

$$\gamma_t(i) = P(q_t = S_i | O, \lambda)$$

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{P(O|\lambda)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^N \alpha_t(i)\beta_t(i)}$$
- Lösung: $q_t = \arg \max_{1 \leq i \leq N} \gamma_t(i)$ für alle $1 \leq t \leq T$
- Problem: Bei nicht vollständig vernetztem HMM ergibt dies unter Umständen keinen gültigen Pfad (z.B. bestes $q_t = S_i$ und $q_{t+1} = S_j$ aber $a_{ij} = 0$)
- Besser: Wahl der insgesamt besten Zustandsfolge über Maximierung von $P(Q|O, \lambda)$ (entspricht der Maximierung von $P(Q, O|\lambda)$ durch Produktregel)

9.5.2 DEKODIERUNGSPROBLEM

Viterbi-Algorithmus (Idee ähnlich zum Vorwärts-Algorithmus)

- Vorwärtsvariable speichert maximale Wahrscheinlichkeit mit der ein Zustand erreicht wird $\delta_t(i) = \max_{q_1, q_2, \dots, q_{t-1}} P(q_1 q_2 \dots q_t = S_i, O_1 O_2 \dots O_t | \lambda)$
- Induktionsschritt $\delta_{t+1}(j) = [\max_i \delta_t(i) a_{ij}] b_j(O_{t+1})$
- Rückwärtsvariable speichert wahrscheinlichsten Vorgängerknoten $\psi_t(j)$

1. Initialisierung: $\delta_1(i) = \pi_i b_i(O_1), 1 \leq i \leq N, \psi_1(i) = 0$
2. Rekursion: $\delta_t(j) = \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}] b_j(O_t), 2 \leq t \leq T, 1 \leq j \leq N$
 $\psi_t(j) = \arg \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}], 2 \leq t \leq T, 1 \leq j \leq N$
3. Terminierung: Wähle Zielknoten mit maximaler Wahrscheinlichkeit $P^* = \max_{1 \leq i \leq N} [\delta_T(i)], q_T^* = \arg \max_{1 \leq i \leq N} [\delta_T(i)]$
4. Backtracking (Bestimmen) der Zustandsfolge $q_t^* = \psi_{t+1}(q_{t+1}^*), t = T-1, T-2, \dots, 1$

9.5.3 LERN- ODER OPTIMIERUNGSPROBLEM

- Schwierigstes der drei Probleme
- Kein analytischer Lösungsweg bekannt
- Bei gegebener endlicher Beobachtungssequenz O gibt es keinen optimalen Weg, um die Modellparameter zu schätzen
- Lokale Maximierung von $P(O|\lambda)$ mit iterativer Prozedur, z.B. Baum-Welche-Algorithmus

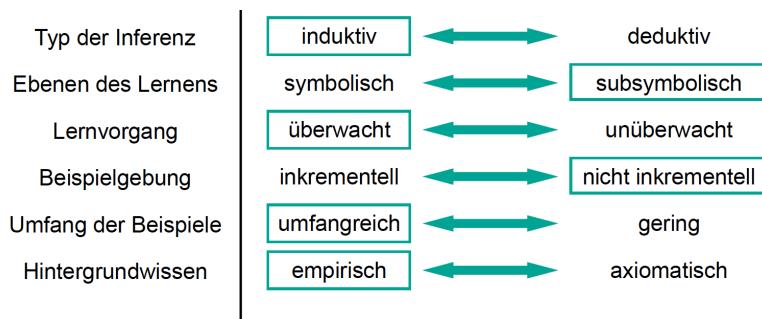
Baum-Welche-Algorithmus

- Gegeben: Trainingssequenz $O_{training}$ und Hypothesenraum für Modelle $\lambda = S, V, A, B, \Pi$
- Sucht: Modell, das die Daten am besten erklärt: $\bar{\lambda} = \arg \max_{\lambda} P(O|\lambda)$
- Ansatz: Hypothesenraum wird so gewählt, dass die Anzahl der Zustände vorgegeben wird
- Lediglich die stochastischen Modellparameter werden angepasst $\bar{\lambda} = \bar{A}, \bar{B}, \bar{\Pi}$
- Beginne mit zufälligem Modell λ , berechne $P(O_{training}|\lambda)$
- Bestimme die erwartete Anzahl von Zustandsübergängen (aus und zwischen Zuständen) und Symbolausgaben

- Neuschätzung der Übergangs- und Emissionswahrscheinlichkeiten: Berechnung eines neuen Modells
- Iteriere so lange bis (lokales) Maximum erreicht ist
- Definiere $\xi_t(i, j)$ als die Wahrscheinlichkeit eines Zustandsübergangs von Zustand i nach j zum Zeitpunkt t , bei gegebenem O und λ :
$$\begin{aligned}\xi_t(i, j) &= P(q_t = S_i, q_{t+1} = S_j | O, \lambda) = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{P(O|\lambda)} \\ &= \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^N \sum_{j=1}^N \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}\end{aligned}$$
- Definition von $\gamma_t(i)$ als die Wahrscheinlichkeit, zum Zeitpunkt t im Zustand S_i zu sein, bei gegebenem O und λ : $\gamma_t(i) = \sum_{j=1}^N \xi_t(i, j)$
- Bewertung
 - Es gilt: $P(O|\bar{\lambda}) \geq P(O|\lambda)$
 - Damit verbessert sich das Modell auf der Menge der Trainingsdaten
 - Realisiert Expectation Maximization Ansatz
 - Training wird abgebrochen, wenn nur noch minimale Verbesserungen
 - Es kann kein globales Maximum garantiert werden

9.6 ARTEN VON HMMs

- Ergodisches Modell: Jeder Zustand kann von jedem anderen Zustand in einer endlichen Anzahl an Schritten erreicht werden
- Links-nach-rechts-Modell (Bakis-Modell): Der Zustandsindex wird mit der Zeit größer oder bleibt gleich
- Links-nach-rechts-Modell mit parallelen Pfaden



10 MARKOV LOGIK NETZE (MLN)

10.1 MARKOV LOGIK

- Grundidee MLN: Vereint Prädikatenlogik erster Ordnung und probabilistische (grafische) Modelle: Prädikatenlogik erster Ordnung -> gut wenn strukturierte Information, Probabilistische Modelle -> gut für unsichere vertrauliche Information (ggf. unstrukturiert)
- Benötigt werden effiziente Algorithmen für Inferenz (für die Auswertung von Anfragen) und für das Lernen von Modellen
- Syntax: Gewichtete logische Formeln (Prädikatenlogik -PL- erster Ordnung)
- Semantik: Template für Erzeugung von Markov Netzen
- Probabilistische Inferenz (WalkSAT, MCMC, KBMC)
- Lernen für Gewichte und Formeln (Voted perceptron, Pseudo-likelihood, Induktive Logische Programmierung (ILP))
- Software: Alchemy, Tuffy,...
- Anwendungen: Informationsgewinnung, Web mining, Ontologieverfeinerung, Klassifikation, Assistenzfunktionen

10.1.1 MARKOV LOGIK NETZ

- Syntax: Ein Markov Logik Netz (MLN) ist eine Menge von Tupeln (F_i, w_i) wobei F_i eine Formel in Prädikatenlogik erster Ordnung und w_i eine reelle Zahl (Gewicht) ist.
- Semantik: Mit einer Menge von Konstanten wird daraus ein Markov Netz definiert mit je einem Knoten für jede mögliche Belegung, jedes Prädikates des MLN und je einem Feature für jede Belegung (grounding) jeder Formel F_i des MLN mit dem entsprechenden Gewicht w_i (Formeln sind dabei Disjunktionsterme/Klauseln -> ggf. Umformung nötig)
- MLN ist eine Schablone (Template) für den Aufbau eines Markov Netzes
- Wahrscheinlichkeit einer Welt X : $P(X) = \frac{1}{Z} \exp(\sum_i w_i n_i(X))$
- Typisierte Variablen und Konstanten reduzieren sehr stark die Größe des aufgebauten Markov Netzes (reduziert Belegung)
- Möglichkeit der Nutzung von Endlichen und kontinuierlichen Domänen, Funktionen, Quantoren, etc.

Vergleich zu Prädikatenlogik erster Ordnung

- Wenn undendliche Gewichte: MLN = reine Prädikatenlogik erster Ordnung (beweisbar)
- Wenn erfüllbare Wissensbasis, d.h. Welten s.d. alle Formeln wahr sind
 - erfüllende Interpretationen (Welten) sind die Modalwerte der Verteilung
 - d.h. dass auch im Falle von verrauschten Daten die Welten, die durch die PL ausgedrückt werden enthalten sind
- Vorteil: Markov Logik erlaubt Widersprüche unter den Formeln -> wichtig für reale verrauschte Daten oder unvollständiges Modell

MAP/MPE Inferenz

- Problem: Finden des wahrscheinlichsten Zustands der Welt (Variablen y), gegeben Evidenzen x (wahrscheinlichste Erklärung für y)
- Bekannter Ansatz: Maximum a-posteriori Ansatz bzw. most probable estimate $\arg \max_y P(y|x)$
- Entsprechend MLN ersetzen: $\arg \max_y \frac{1}{Z} \exp(\sum_i w_i n_i(x, y)) \rightarrow \arg \max_y \sum_i w_i n_i(x, y)$
- Dies ist bekannt als weighted MaxSAT Problem
 - SAT = satisfied belief / satisfiability-solver -> alle Formeln erfüllen
 - MaxSAT -> so viele Formeln wie möglich erfüllen
 - Weighted MaxSAT -> max. gewichtete Anzahl von Formeln erfüllen
- Vorteile: Standard-Verfahren -> SAT solver, potentiell schneller als rein logische Inferenz

WalkSAT Algorithmus

- Ziel: Finden einer Interpretation (Wahrheitswert - Zuordnung, truth assignments) der Variablen um alle Klauseln (gleichzeitig) zu erfüllen
- Klauseln = Disjunktion von Literalen (Verallgemeinerung: jede Formel in disjunktive Form konvertierbar)
- Problem: Wenn es n Konstanten gibt (z.B. jede Person oder jedes Wort einer Sprache) und Klausel mit (max.) Anzahl c unterschiedlicher Variablen, dann benötigt das belegte Netz $O(n^c)$ Speicher -> kann schnell, sehr stark explodieren
- Praktikable Lösung: Nutzung der dünnen Belegung (sparseness): Klauseln werden nur dünn belegt (ground clauses lazily -> LazySAT)

Anfragen und Wahrscheinlichkeit

- Häufige Fragestellung: Anfragen bzgl. Erfüllung einer Formel (gegeben MLN und Konstanten C): $P(\text{Formel}|\text{MLN}, C) = ?$

- entspricht einfach der Summe der Wahrscheinlichkeiten der Welten in denen die Formel erfüllt ist
- Aber: große (exponentielle) Anzahl von Welten
- Lösung: MCMC (markov chain monte carlo) Ansatz: stochastisches Abtasten der Welten (Wiederholtes sampeln von Belegungen für eine/jede Variable in abh. von MAP), Test ob Formel erfüllbar ist, Anteil (Faktor) entspricht der gesuchten Wahrscheinlichkeit P

11 DEDUKTIVES LERNEN

11.1 DEDUKTIVE LERNVERFAHREN

- Deduktion = Ableitung
- Eigenschaften der Lernverfahren
 - Nutzung vorhandenen Wissens
 - Neuformulierung vorhandenen Wissens
 - Explizite Darstellung implizit vorhandenen Wissens
- Beispiele
 - Modus Ponens: $\frac{A, A \rightarrow B}{B}$
 - Reduktion abstrakter auf berechenbare Größen
- Deduktive/Analytische Lernverfahren benutzen hauptsächlich vorhandenes Hintergrundwissen und benötigen wenige oder gar keine Lernbeispiele
- Gegensatz: Induktive/Empirische Lernverfahren

11.2 ERKLÄRUNGSBASIERTES LERNEN (EBL)

- „The key insight behind explanation-based generalization is that it is possible to form a justified generalization of a single positive training example provided the learning system is endowed with some explanatory capabilities. In particular, the system must be able to explain to itself why the training example is an example of the concept under study. Thus, the generalizer is presumed to possess a definition of the concept under study as well as domain knowledge for constructing the required explanation.“ [Mitchell et al.]
- Gegeben
 - Zielkonzept: Eine Beschreibung des zu lernenden Konzepts (die nicht das Operationalitätskriterium erfüllt)
 - Trainingsbeispiel: Ein Beispiel für das Zielkonzept

- Bereichstheorie: Regeln und Fakten, die erklären, warum das Trainingsbeispiel ein Beispiel für das Zielkonzept ist
- Operationalitäts- (Anwendbarkeits-) Kriterium: Ein Prädikat über Konzeptbeschreibungen, das die Forum spezifiziert, in der erlernte Beschreibungen vorliegen müssen
- Gesucht: Eine Generalisierung des Trainingsbeispiels, die eine hinreichende Definition des Zielkonzepts darstellt und das Operationalitätskriterium erfüllt
- Prozess, der implizites Wissen in explizites umwandelt
- Für jedes positive Trainingsbeispiel wird eine Generalisierung erstellt.
- Warum EBL?
 - Analogien zur menschlichen Wissensverarbeitung
 - Biasproblematik bei induktivem Lernen
- Wesentlicher Aspekt: Speedup-Learning
- Voraussetzungen:
 - Vollständigkeit
 - Korrektheit/Konsistenz
 - Anwendbarkeit
- Problemfälle
 - Erklärung nicht möglich oder nicht berechenbar
 - Inkonsistente (Beispiel und Gegenbeispiel) oder multiple Erklärungen
- Wie soll mit Hintergrundwissen / Bereichstheorien umgegangen werden, die die Anforderungen nicht erfüllen? Approximierungen / Explorierung
- EBL-System kann Beispiel erklären. Kann es damit auch Beispiele selbst erzeugen? In manchen Bereichen gilt nicht Erklärungsfähigkeit => Generalisierungsfähigkeit (z.B. NP-vollständige Probleme)
- Beispiele dienen als Fingerzeige, ohne Beispiele ist Suche im Bereichswissen und Aufbau der Erläuterungsstruktur komplexer
- Vorgegebene Beispiele können als typisch ausgewählt werden. Diese Auswahl erfordert Wissen über das Zielkonzept, das i.a. nicht im System enthalten ist.

Wird tatsächlich gelernt?

- Definition: Ein System lernt aus Erfahrung E in Hinblick auf eine Klasse von Aufgaben T und einem Performanzmaß P, wenn seine Leistungen bei Aufgaben aus T gemessen mit P durch Erfahrung aus E steigt.

- Wird Wissen erlernt, das nicht bereits im Hintergrundwissen enthalten ist?
- EBG/EBL verändert die deduktive Hülle der Wissensbasis nicht => unbegrenzte Res-sourcen vorausgesetzt wird nichts neues gelernt
- Aber: In der Praxis sind Zeit/Rechenleistung begrenzt
- Erst durch Makros/Generalisierung können manche Ziele erreicht/Aufgaben durchge-führt werden -> unter dieser Voraussetzung wird tatsächlich gelernt (sog. Speedup-Learning)

Ressourcenaufwand für das Lernen neuer Regeln

- Hardware (Parallelrechner, paralleles Matchen) -> keine echte Lösung, Lösung wird vertagt
- Indizierung von Regeln/Makrooperatoren: keine Reduzierung der Regelmenge
- Vermutlich bester Ansatz: Beschränkung der Aufnahme neuer Regeln bzw. Makroope-ratoren, Messen der Nützlichkeit neuer Regeln/Operatoren (bzgl. neu zu lösender Pro-bleme), ggf. Lösung von Beispielproblemen zur Unterstützung dieser Messungen, Er-halten oder Verwerfen der Regel, je nach Ergebnis

Nützlichkeit von Regeln

- Nützlichkeit einer einzelnen gelernten Regel bzw. eines Makrooperators
- Nützlichkeit der Kombination aller Regeln (MOs) (gelernt & vorgegeben)

Vorteile von EBL

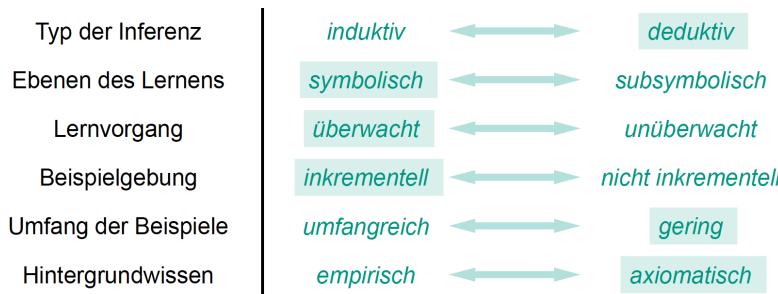
- Gelerntes Wissen ist korrekt: Korrektheit des ursprünglichen Wissens vorausgesetzt
- Kein (impliziter) induktiver Bias
- Stattdessen explizite Formulierung der Domänentheorie: Ausnutzung vorhandenen do-mänenpezifischen Wissens, bessere Überprüfbarkeit, domänenabhängiger, da Be-reichswissen separat repräsentiert

Anwendungen von EBL

- Erzeugung von Makrooperatoren (automatisches Planen): STRIPS (STanford Rese-rearch Institute Problem Solver), SOAR (Symbolic Cognitive Architecture, CMU)
- Lernen Suche zu kontrollieren: Effektivitätssteigerung beim Planen, besonders bei großen Zustandsräumen, PRODIGY (Lernen von Planungsstrategien): Bewertung von Regeln, Einbeziehung von negativen Beispielen

Erzeugung von Makrooperatoren

- Gegeben: Wissen über (Elementar-) Operatoren in einer Problemlösungsdomäne, Erfahrung mit gültigen Problemlösungen in der Domäne
- Gesucht: Zusammengefasste Operatorsequenzen, welche die Kosten für das Finden von Problemlösungswissen reduzieren



11.2.1 ERKLÄRUNGSBASIERTE GENERALISIERUNG (EBG)

- Ziel: Trainingsbeispiel \leftrightarrow Generalisierung
- Zweischritt-Verfahren für jedes positive Beispiel: Explain (Finden einer Erklärung) und Generalize (Generalisierung der Erklärung)
 - Explain
 - * Konstruiere in Termen der Bereichstheorie eine Erklärung, die zeigt, wie das Trainingsbeispiel die Definition des Zielkonzeptes erfüllt.
 - * Konstruiere diese Erklärung so, dass jeder Ast der Erklärungsstruktur in einem Ausdruck endet, der das Operationalitätskriterium erfüllt.
 - Generalize
 - * Bestimme hinreichende Bedingungen, unter denen die oben gefundene Erklärungsstruktur gültig ist und formuliere diese Kriterien in Termen, die das Operationalitätskriterium erfüllen.
- Voraussetzungen:
 - Vollständigkeit
 - Korrektheit/Konsistenz
 - Anwendbarkeit
- Problemfälle
 - Erklärung nicht möglich oder nicht berechenbar
 - Inkonsistente (Beispiel und Gegenbeispiel) oder multiple Erklärungen
- Wie soll mit Hintergrundwissen / Bereichstheorien umgegangen werden, die die Anforderungen nicht erfüllen? Approximierungen / Explorierung

11.3 DEDUKTIVES VS INDUKTIVES LERNEN

	Induktives Lernen	Deduktives Lernen
Ziel	Hypothese passt zu den Daten	Hypothese passt zur Bereichstheorie
Vorteile	wenig a-priori Wissen	wenig Beispiele notwendig
Nachteile	schlecht bei geringen Datenmengen schlecht bei inkorrekttem Bias	schlecht, falls imperfekte Bereichstheorie

Deduktive Methoden: Durch Logik gerechtfertigte Hypothesen
 Induktive Methoden: Statistisch gerechtfertigte Hypothesen

11.4 HYBRIDE LERNVERFAHREN

- Reale Lernprobleme: zwischen induktiv und deduktiv
 - Hintergrundwissen vorhanden, aber nicht beliebig viel
 - Hintergrundwissen korrekt?
 - Trainingsdaten in begrenzter Menge
 - Trainingsdaten korrekt?
- Hybride Verfahren sollten ...
 - ohne Hintergrundwissen so effektiv lernen wie rein induktive Verfahren
 - mit perfektem Hintergrundwissen so effektiv lernen wie rein deduktive Verfahren
 - mit unvollständigem Hintergrundwissen und Trainingsdaten besser als rein induktive und rein deduktive Verfahren sein
 - mit unbekannter Menge von Fehlern in Trainingsdaten zurechtkommen
 - mit unbekannter Menge von Fehlern in Hintergrundwissen zurechtkommen
- Realität: Wunschliste nur teilweise realisierbar

Definition Lernproblem

- Gegeben: Menge von Trainingsbeispielen D , möglicherweise mit Fehlern, Bereichstheorie B , möglicherweise fehlerhaft, Hypothesenraum H
- Ziel: Finde Hypothese h , die am besten sowohl zu Trainingsbeispielen als auch zu Bereichstheorie passt
- Entscheidung über beste Hypothese z.B. mittels $\arg \min_{h \in H} k_D E_D(h) + k_B E_B(h)$
- Frage: Wie k_D, k_B (relative Gewichtungen) wählen?
- Verwinde Vorwissen um initiale Hypothese h_0 abzuleiten -> Bsp. KBANN

KBANN

- Initialisiere Neuronales Netz mittels Bereichstheorie
- Verfeinere initiales Netz durch Backpropagation und Trainingsbeispiele
- Verwendung: Bereichstheorie korrekt -> alle Beispiele korrekt klassifiziert, Beispiele inkorrekt klassifiziert -> Fehler in Bereichstheorie, induktive Verfeinerung durch Backpropagation
- Intuition: Sogar wenn Bereichstheorie nur annähernd korrekt, Starthypothese besser als zufällige Initialisierung
- Gegeben: Menge von Trainingsbeispielen, Bereichstheorie aus nicht-rekursiven Prologähnlichen logischen Ausdrücken
- Gesucht: NN das zu Trainingsdaten passt, angelehnt an Bereichstheorie
- Verfahren in 2 Schritten
 1. Aufbau eines initialen NNs äquivalent zur Bereichstheorie
 2. Verfeinerung des initialen NNs

KBANN: Algorithmus

1. Pro Instanzattribut ein Netz-Input
2. Für jede Klausel der Bereichstheorie ein Neuron wie folgt einfügen: Eingang mit geprüften Attributen verknüpfen, verwendet Gewicht W für nicht-negierte Attribute und -W für negierte, setze Schwellwert auf -(n-0.5)W (mit n = Anzahl der nicht-negierten Bedingungssteile)
3. Zusätzliche Verbindungen, um jedes Element auf Schicht i mit jedem auf Schicht i+1 zu verbinden, zufällige kleine Gewichte zuweisen
4. Backpropagation-Algorithmus mit den Trainingsbeispielen auf initiales Netz anwenden

KBANN: Anwendungen

- Lernen von physikalischen Objektklassen
 - Instanzen beschrieben durch Angaben über Material
 - Beispielaufgabe: Zielkonzept Tasse lernen
 - signifikante Verfeinerung des initialen Netzes durch Backpropagation-Algorithmus
- Erkennung biologischer Konzepte in DNS-Sequenzen
 - Gen-Sequenzen repräsentiert als Strings
 - gesucht: bestimmte genetische Regionen (Promotoren)
 - Bereichswissen aus biol. Forschung extrahiert
 - KBANNs zeigen gute Resultate

12 INSTANZBASIERTES LERNEN

12.1 EINFÜHRUNG

Lazy learning vs Eager learning

- Instanzen-basiertes Lernen: Delayed/Lazy Learning
- Lernen = (einfaches) Abspeichern der Trainingsbeispiele
- Weniger Rechenzeit während des Trainings, dafür mehr bei Anfragen zur Klassifikation
- Unterschiedliche Hypothesen/Lokale Approximation der Zielfunktion für jede Anfrage
- Fleißig (eager) Lernalgorithmen mit dem gleichen Hypothesenraum sind eingeschränkter

Instanzenbasiertes Lernen

- Bildet für jede neue Anfrage eine andere Annäherung an die Zielfunktion
 - Lokale Approximation der Targetfunktion
 - Komplexere symbolische Repräsentation für Instanzen
- Generalisierungsentscheidung wird bis zu einer konkreten Suchanfragen aufgeschoben
- Neue Instanzen werden analog zu ähnlichen Instanzen klassifiziert

Beurteilung

- + Komplexe Targetfunktionen können modelliert werden
- + Informationen aus den Trainingsbeispielen geht nicht verloren
- - evtl. komplexe Berechnungen bei Klassifizierung neuer Instanzen
- - Schwierigkeit: Welche Instanzen sind sich ähnlich?
- - Problem irrelevanter Eigenschaften

12.2 DER K-NN ALGORITHMUS

- Instanzen $x = \langle a_1(x), a_2(x), \dots, a_n(x) \rangle$ korrespondieren mit Punkten im n-dimensionalen Raum \mathbf{R}^n
- Nachbarschaftsbeziehungen sind definiert durch die Euklidische Distanz
- Lernen einer Funktion $f: \mathbf{R}^n \rightarrow V$ aus einer Menge von Trainingsbeispielen $\langle x_i, c(x_i) \rangle$
- V endliche (diskrete) Menge

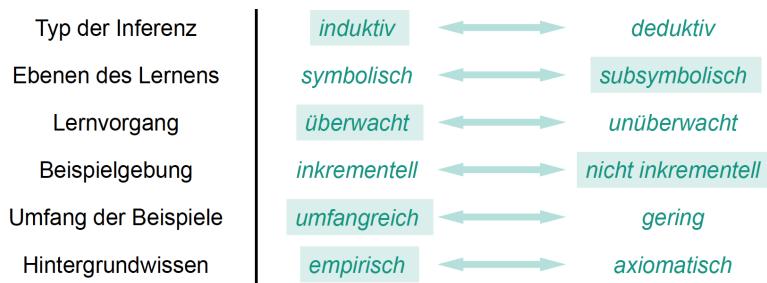
- Trainingsalgorithmus: Für jedes Trainingsbeispiel $< x_i, c(x_i) >$ mit $c(x_i)$ aus V: Füge das Trainingsbeispiel zu der Liste training_examples hinzu
- Klassifikationsalgorithmus (Anfrage x_q): x_1, \dots, x_k seien die k Instanzen von training_examples, die am nächsten zu x_q liegen
- Decision-boundary für $k = 1$ entspricht Voronoi-Diagramm
- Normalisierung der Eingabevektoren oft sinnvoll (Verzerrung bei stark ungleichen Ein-gabedimensionen)

Abstandsgewichteter k-NN Algorithmus

- Nahe Nachbarn gehen genauso stark in die Klassifikation ein wie weiter entfernte, ha-ben aber u.U. viel größere Ähnlichkeiten mit der neu zu klassifizierenden Instanz
- Distanzbasierte Gewichtung: $f(x_q) \leftarrow \arg\max_{v \in V} \sum_{i=1}^k w_i \delta(v, c(x_i))$ mit $w_i = \frac{1}{d(x_q, x_i)^2}$

Bewertung k-NN Algorithmus

- Induktiver Bias: Klassifikation einer Instanz x_q ähnlich zu Klassifikation anderer, be-nachbarter Instanzen
- + Robust in Bezug auf verfälschte Trainingsdaten: insbesondere mit Gewichtung
- - Distanzmaß basiert auf allen Attributen: evtl. nur eine kleine Untermenge relevant, curse of dimensionality
- - Speicherorganisation
- - Haupt-Zeitaufwand in Klassifikation statt Training



12.3 CASE-BASED REASONING (CBR)

Analogien schließen: Ähnlichkeit von Größen hinsichtlich mehrerer Eigenschaften (Relatio-nen) erlaubt Schluss auf Ähnlichkeit weiterer Eigenschaften (Bsp: Sonnensystem, Bohrsches Atommodell)

- allgemeines (abstraktes) Framework

- kein direkt anwendbarer Algorithmus
- Wiederverwendung alter Fälle
- Suche nach Lösungen ähnlicher Probleme

Was ist ein Fall?

- Kognitionswissenschaften: Fälle sind Abstraktionen von Ereignissen, die in Zeit und Raum begrenzt sind
- Technische CBR Sichtweise: Ein Fall ist die Beschreibung einer bereits real aufgetretenen Problemsituation zusammen mit den Erfahrungen, die während der Bearbeitung des Problems gewonnen werden konnten

Ein Fall enthält...

- mindestens:
 - Problembeschreibung
 - Lösung (-versuch)
 - Ergebnis
- zusätzlich:
 - Erklärung, warum das Ergebnis auftrat
 - Lösungsmethode
 - Pointer auf andere Fälle
 - Güteinformation

CBR-Zyklus

- Retrieve (Auffinden)
 - Aufgabe: Finde ähnliche Fälle
 - Ähnlichkeitsmaß
 - * Euklidische Distanz
 - * Syntaktische Ähnlichkeiten (knowledge-poor)
 - * Semantische Ähnlichkeiten (knowledge-intensive)
 - Organisation der Fallbasis
 - * Lineare Liste
 - * Baumstruktur
 - * Graphen, Netze, Indexstrukturen
 - * Datenbanken

- Reuse (Wiederverwendung)
 - Arten der Lösungsübertragung
 - * Keine Adaption, einfaches Übertragen (Copy)
 - * Durch Benutzer
 - * (semi-)automatische Adaption: transforamtional reuse, derivational reuse
 - Eingesetzte Methoden
 - * Benutzerinteraktion
 - * Regelbasiertes Schließen
 - * Modellbasiertes Schließen
 - * Planer
- Revise (Überarbeiten, Anpassen)
 - Überprüfung, Verbesserung der Lösung
 - Evaluierung der Lösung (Überprüfen durch Simulation, Überprüfung in realer Welt)
 - Verbessern bzw. Reparieren der Lösung (Fehler erkennen und erklären, beseitigen unter Berücksichtigung der Fehlererklärungen)
 - potentiell iterativ
- Retain (Zurück behalten)
 - Bewahrung der gemachten Erfahrung
 - Was wird gelernt?
 - * Neue Erfahrung (neuer Fall)
 - * Alten Fall generalisieren
 - * Neue Merkmale (Indizes)
 - * Organisation der Fallbasis (Effizienz)
 - Methoden
 - * Auswendiglernen (Speichern neuer Fälle)
 - * Induktive/Deduktive Lernverfahren

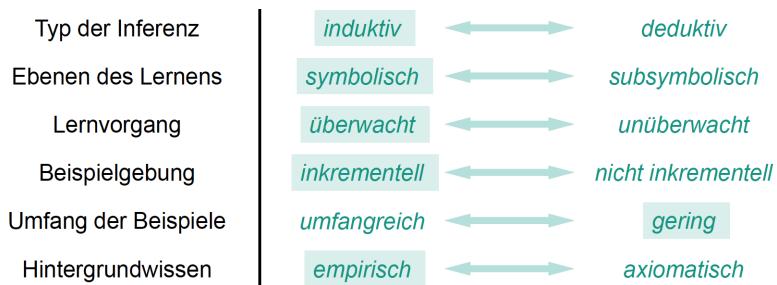
CBR: Vorteile

- Konzeptuell einfach, aber trotzdem können komplexe Entscheidungsgrenzen gebildet werden
- Kann mit relativ wenig Informationen arbeiten
- Analogie zu menschlichem Problemlösen

- Lernen ist einfach (one-shot learning)
- Günstig für mit Regeln schlecht erfassbare Probleme

CBR: Probleme

- Gedächtniskosten
- Klassifikation kann lange dauern
- hängt stark von Repräsentation ab
- Problematisch bei komplexen Repräsentation
- Problematisch: irrelevante Eigenschaften



13 NEURONALE NETZE

13.1 PERZEPTRON [ROSENBLATT 1960]

- Grundidee: Anlehnung an das Funktionsprinzip der natürlichen Wahrnehmung - Reaktion im Tierbereich
- Lernen = Anpassen der Gewichte
- Gesucht: beste Trennhyperebene

13.2 GRADIENTENABSTIEG

Fehlerfunktion

- Fehlerfunktion $E(\vec{w}) = \frac{1}{2|D|} \sum_{d \in D} (t_d - o_d)^2$ mit D - Lerndaten
- Lernen: Minimieren von E
- Gradient $\Delta E(w) = [\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \dots, \frac{\partial E}{\partial w_n}]$

- Anpassung des Gewichtsvektors $\vec{w} \leftarrow \vec{w} + \delta \vec{w}$ mit $\delta \vec{w} = -\eta \Delta E(\vec{w})$

$$\text{Deltaregel } \frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i} \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2 = \frac{1}{2} \sum_{d \in D} \frac{\partial}{\partial w_i} (t_d - o_d)^2 = \frac{1}{2} \sum_{d \in D} 2(t_d - o_d) \frac{\partial}{\partial w_i} (t_d - o_d) = \\ \sum_{d \in D} (t_d - o_d) \frac{\partial}{\partial w_i} (t_d - \vec{w} \cdot \vec{x}_d) \\ \delta w_i = \eta \sum_{t_d = o_d} x_{id}$$

14 EVOLUTIONÄRE ALGORITHMEN

Biologisches Evolutionsmodell nach Darwin: Selektion = Treibende Kraft der Evolution („Man kann die natürliche Selektion genauso gut in Formeln packen, wie es mit natürlichen Neuro-nalen Netzen geht.“)

Technisch: Evolution als Optimierung komplexer, künstlicher Systeme

Nomenklatur

- Individuum: eine mögliche Lösung, Hypothese
- Population und Generation: Lösungs- bzw. Hypothesenmenge
- Erzeugen von Nachkommen: Generierung neuer Hypothesen (z.B. durch Rekombination und Mutation)
- Veränderter Nachfolger, Kind, Nachkomme: neue Hypothese
- Fitness(-funktion): Hypothesengüte
- Selektion der Besten: Auswahl der Hypothesen welche die beste Problemlösung erzeugen

Kodierung der Individuen

- Wissen wird meistens strukturiert repräsentiert
- Kodieren durch Gene
 - k-Alphabet ($k=2$: Binärcodierung): Genetische Algorithmen
 - Reelle Zahlen, Vektoren: Evolutionäre Strategien
 - baumartige Strukturen: Genetisches Programmieren
- Wie viel von dieser Strukturinformation soll genutzt werden?
 - kein Einsatz, ausschließliches Anwenden der Algorithmen z.B. auf den Binärsequenzen
 - volle Ausnutzung der Strukturinformation: Spezielles Anpassen der Optimierungs-algorithmen

Generierung von Nachkommen

- Erfolgt durch s.g. genetische Operatoren

- Zwei Strategien: Exploration (Erforschen des Hypothesenraumes) vs Exploitation (lokale Optimierung)
- Je stärker und zufälliger Änderungen sind, um so geringer ist die Wahrscheinlichkeit, bessere Nachkommen zu erzeugen
- Bei lokalen Verbesserungsmethoden ist die Gefahr der lokalen Minima gegeben
- Explorationsgrad muss gemäß der aktuellen Fitness der Generation ausgewählt werden (z.B. anfangs hoch dann fallend)

14.1 MUTATION

- Alle Bits einer Sequenz werden unabhängig voneinander mit einer bestimmten Wahrscheinlichkeit invertiert
- Für eine bestimmte (oder zufällige) Anzahl von Bits werden die Indizes zufällig ausgewählt
- Stochastisch bei kontinuierlicher Repräsentation: $x_i := x_i + z$, wobei z eine Zufallsvariable ist
- bei Sequenzen werden Teilsequenzen herausgenommen und an anderer Stelle eingefügt

14.2 REKOMBINATION

- Eigenschaften von zwei oder mehreren Eltern sollten gemischt werden
- Kombiniere bspw. die ersten $x\%$ von Gen A und die letzten $1-x\%$ von Gen B -> liefert 2 Nachkommen

14.3 SELEKTION

- 2 Arten der Selektion: Eltern für jeweilige Erzeugung der Nachkommen, Population für die nächste Iteration
- Probleme: Genetische Drift (Individuen vermehren sich zufällig mehr als andere), Crowding (fitte Individuen und ähnliche Nachkommen dominieren die Population)
- Lösung: Verschiedene Populationsmodelle und Selektionsmethoden, Populationsgröße optimieren

14.4 GENETISCHE PROGRAMMIERUNG

- Ziel: Erzeugung optimierter Programme
- Individuen sind Programme
- Repräsentation z.B. als Baum
- Selektion, Mutation und Rekombination auf Baumstrukturen
- Fitness: Programmtest auf einer Menge von Testdaten

Induktives Lernen: Specific to General

Specific-to-general Beispiel III



Himmel	Lufttemp.	Luftfeuch.	Wind	Wassert.	Vorhers.	$c(x_i)$
sonnig	warm	normal	stark	warm	gleichbl.	true
sonnig	warm	hoch	stark	warm	gleichbl.	true
regnerisch	kalt	hoch	stark	warm	wechselh.	false
sonnig	warm	hoch	stark	kalt	wechselh.	true

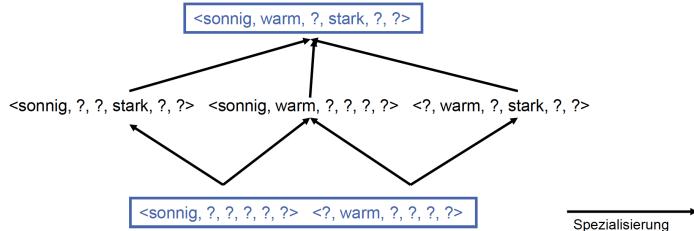
- $h = \langle \text{sonnig}, \text{warm}, ?, \text{stark}, \text{warm}, \text{gleichbl.} \rangle$
- 3. Beispiel: negativ
 - Wird ignoriert
- 4. Beispiel
 - $x_4 = \langle \text{sonnig}, \text{warm}, \text{hoch}, \text{stark}, \text{kalt}, \text{wechselhaft} \rangle$
 - $c(x_4) = \text{true}$
 - Aber nicht von h abgedeckt, $h(x_4) = \text{false} \Rightarrow$ minimale Verallgemeinerung, so dass $h(x_4) = \text{true}$
 - $h = \langle \text{sonnig}, \text{warm}, ?, \text{stark}, ?, ? \rangle$

Induktives Lernen: Version Space

Version Space = alle konsistenten Hypothesen

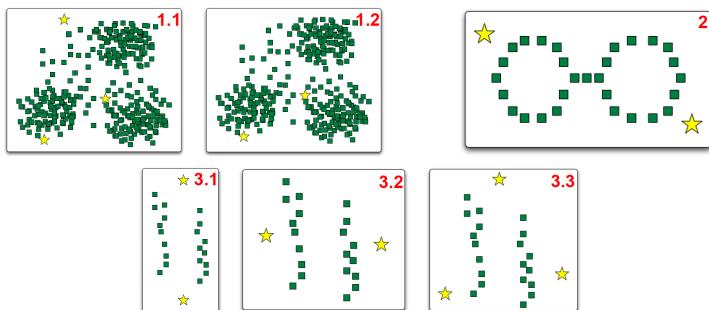


Himmel	Lufttemp.	Luftfeuch.	Wind	Wassert.	Vorhers.	$c(x_i)$
sonnig	warm	normal	stark	warm	gleichbl.	true
sonnig	warm	hoch	stark	warm	gleichbl.	true
regnerisch	kalt	hoch	stark	warm	wechselh.	false
sonnig	warm	hoch	stark	kalt	wechselh.	true

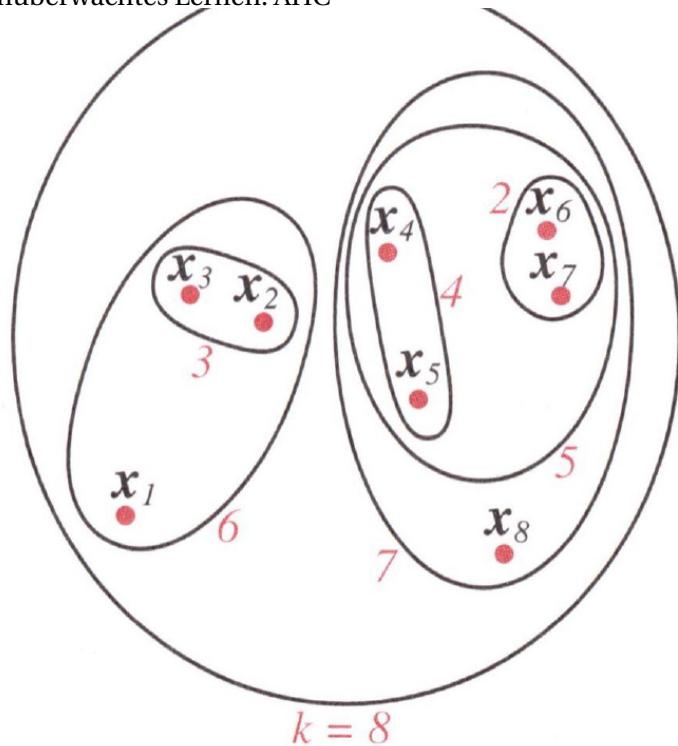


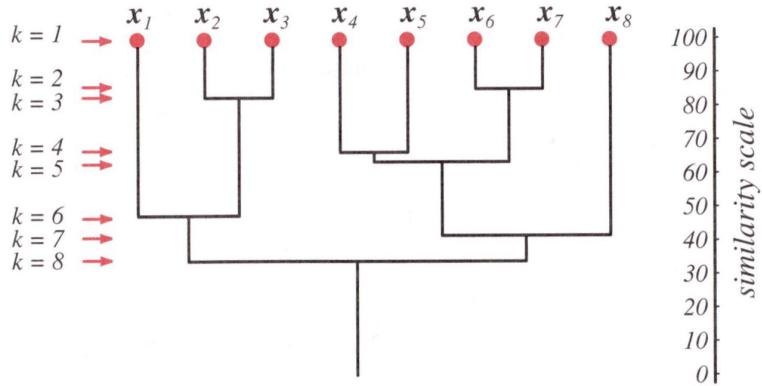
Unüberwachtes Lernen: k means

k-means-Clustering Beispiel(e)



Unüberwachtes Lernen: AHC

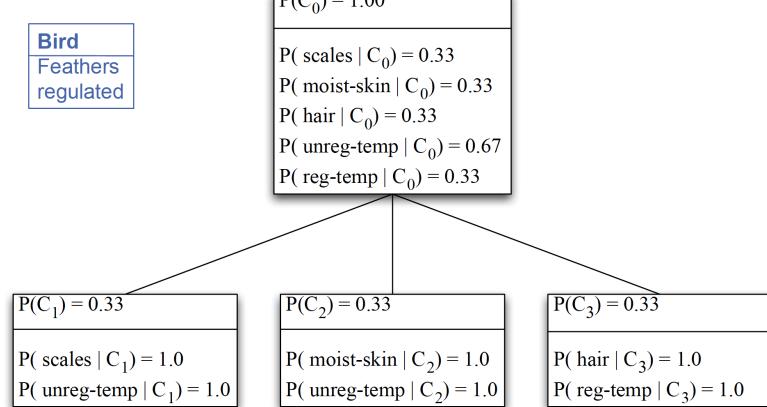


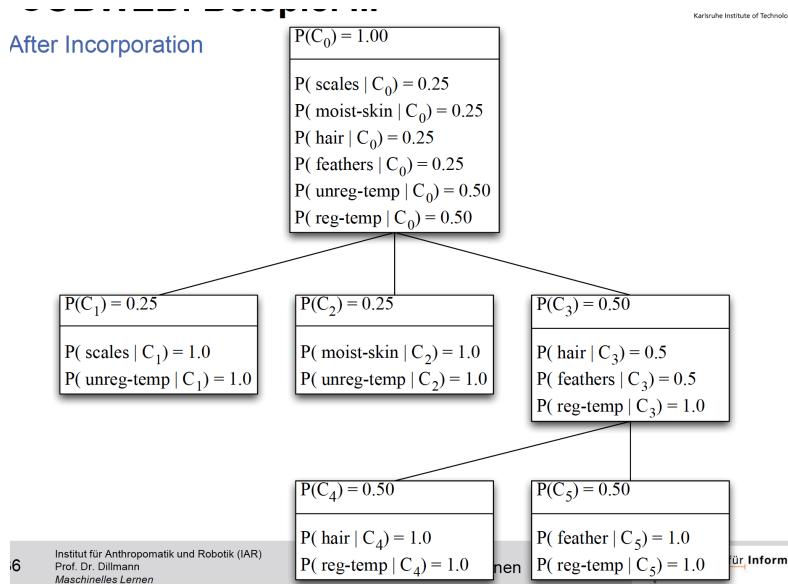


Unüberwachtes Lernen: COBWEB

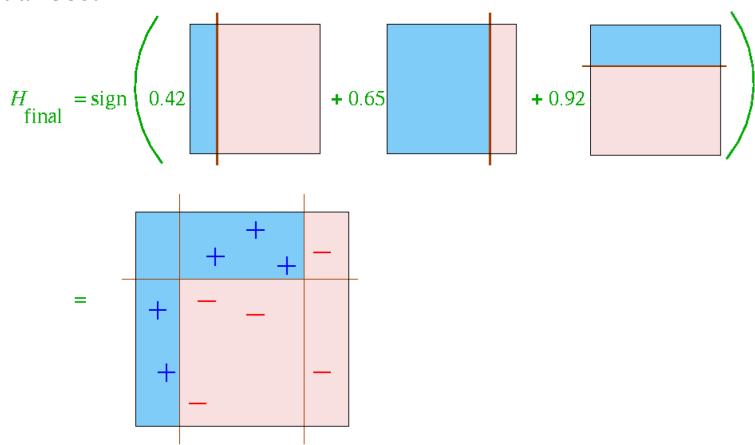
Name	Body cover	Body temp
Fish	scales	unregulated
Amphibian	moist-skin	unregulated
Mammal	hair	regulated
Bird	feathers	regulated

Before Incorporation

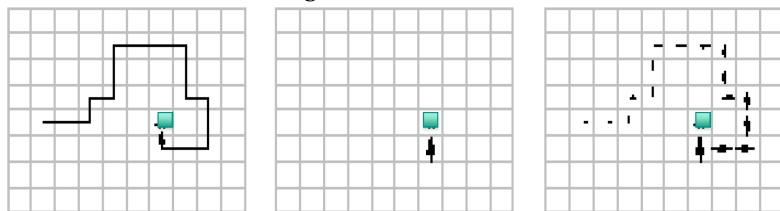




AdaBoost

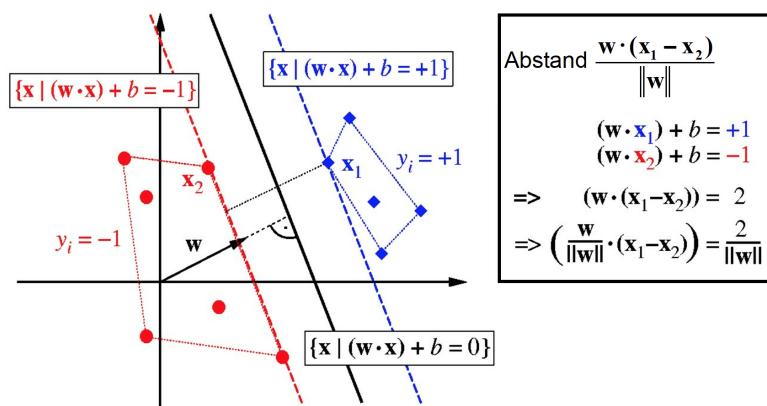


Reinforcement Learning - SARSA

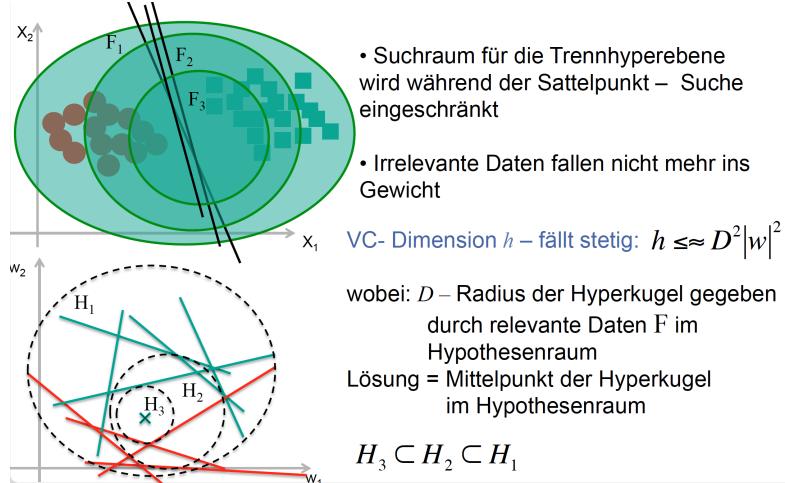


- Ziel – Pfad zum Ziel finden
- Anfangs alle Q = 0
- Reward nur im Ziel
- Mitte – Lernen ohne Eligibility – Trace
 - Nur ein Q- Wert wird angepasst
- Rechts – Einfluss der Anpassung mit Eligibility

SVM

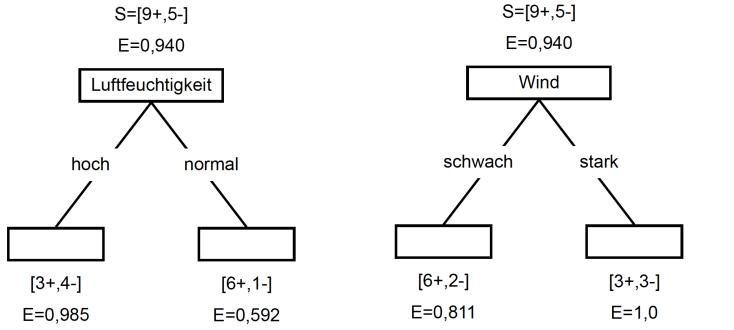


SVM - Wieso SRM?



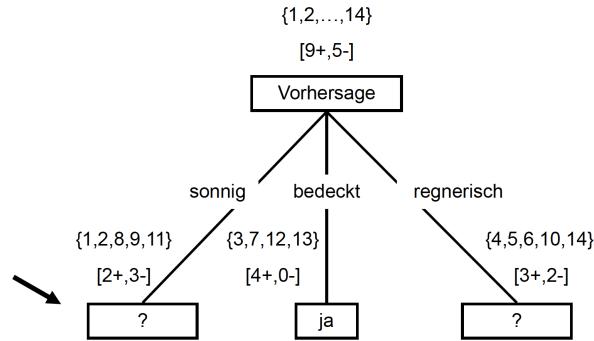
ID3

Nr.	Vorhersage	Temperatur	Luftfeuchtigkeit	Wind	Tennis?
1	sonnig	heiß	hoch	schwach	nein
2	sonnig	heiß	hoch	stark	nein
3	bedeckt	heiß	hoch	schwach	ja
4	regnerisch	warm	hoch	schwach	ja
5	regnerisch	Kalt	normal	schwach	ja
6	regnerisch	Kalt	normal	stark	nein
7	bedeckt	Kalt	normal	stark	ja
8	sonnig	Warm	hoch	schwach	nein
9	sonnig	Kalt	normal	schwach	ja
10	regnerisch	Warm	normal	schwach	ja
11	sonnig	Warm	normal	stark	ja
12	bedeckt	Warm	hoch	stark	ja
13	bedeckt	Heiß	normal	schwach	ja
14	regnerisch	Warm	hoch	stark	nein



$$\begin{aligned} \text{Gewinn}(S, \text{Luftfeuchtigkeit}) \\ = 0,940 - (7/14)0,985 - (7/14)0,592 \\ = 0,151 \end{aligned}$$

$$\begin{aligned} \text{Gewinn}(S, \text{Wind}) \\ = 0,940 - (8/14)0,811 - (6/14)1,0 \\ = 0,048 \end{aligned}$$



$$\begin{aligned} \text{Gewinn}(S_{\text{sonnig}}, \text{Luftfeucht.}) &= 0,970 - (3/5)0,0 - (2/5)0,0 = 0,970 \\ \text{Gewinn}(S_{\text{sonnig}}, \text{Temp.}) &= 0,970 - (2/5)0,0 - (2/5)1,0 - (1/5)0,0 = 0,570 \\ \text{Gewinn}(S_{\text{sonnig}}, \text{Wind}) &= 0,970 - (2/5)1,0 - (3/5)0,918 = 0,19 \end{aligned}$$

alle in einer Klasse oder gleichverteilt

Bayes

Vorwissen über spezielle Krebserkrankung / Labortest:

$$\begin{array}{ll} P(\text{Krebs}) = 0.008 & P(\neg \text{Krebs}) = 0.992 \\ P(\oplus | \text{Krebs}) = 0.98 & P(\ominus | \text{Krebs}) = 0.02 \\ P(\oplus | \neg \text{Krebs}) = 0.03 & P(\ominus | \neg \text{Krebs}) = 0.97 \end{array}$$

Gesucht: $P(\text{Krebs} | \oplus)$

$$\begin{aligned} P(\text{Krebs} | \oplus) &= \frac{P(\oplus | \text{Krebs})P(\text{Krebs})}{P(\oplus)} \\ &= \frac{P(\oplus | \text{Krebs})P(\text{Krebs})}{P(\oplus | \text{Krebs})P(\text{Krebs}) + P(\oplus | \neg \text{Krebs})P(\neg \text{Krebs})} \\ &= 0.98 \cdot 0.008 / (0.98 \cdot 0.008 + 0.03 \cdot 0.992) \\ &= 0.21 \end{aligned}$$

Naive Bayes

Gesucht: $P(v_j)$ und $P(a_i|v_j)$

Vorhersage	Temperatur	Luftfeuchtigkeit	Wind	Tennis?
sonnig	heiß	hoch	schwach	nein
sonnig	heiß	hoch	stark	nein
bedeckt	heiß	hoch	schwach	ja
regnerisch	warm	hoch	schwach	ja
regnerisch	kalt	normal	schwach	ja
regnerisch	kalt	normal	stark	nein
bedeckt	kalt	normal	stark	ja
sonnig	warm	hoch	schwach	nein
sonnig	kalt	normal	schwach	ja
regnerisch	warm	normal	schwach	ja
sonnig	warm	normal	stark	ja
bedeckt	warm	hoch	stark	ja
bedeckt	heiß	normal	schwach	ja
regnerisch	warm	hoch	stark	nein

Neue Instanz: < sonnig, kalt, hoch, stark >

$$v_{NB} = \arg \max_{v_j \in V} P(v_j) \prod_i P(a_i|v_j)$$

$$P(\text{Tennis}=ja) = \frac{9}{14} = 0.64$$

$$P(\text{Wind}=stark|\text{Tennis}=ja) = \frac{3}{9} = 0.33$$

$$P(\text{Tennis}=nein) = \frac{5}{14} = 0.36$$

$$P(\text{Wind}=stark|\text{Tennis}=nein) = \frac{3}{5} = 0.60$$

:

$$P(ja)P(\text{sonnig}|ja)P(\text{kalt}|ja)P(\text{hoch}|ja)P(\text{stark}|ja) = 0.0053$$

$$P(nein)P(\text{sonnig}|nein)P(\text{kalt}|nein)P(\text{hoch}|nein)P(\text{stark}|nein) = 0.0206$$

→ Klassifikation: Tennis = nein

$$\text{Normierte Wahrscheinlichkeit: } \frac{0.0206}{0.0206 + 0.0053} = 0.795$$

EBG

■ Zielkonzept: robust (x)

■ Trainingsbeispiele:

- robot(Num5), r2d2 (Num5), age(Num5, 5), manufacturer(Num5, GR) ...

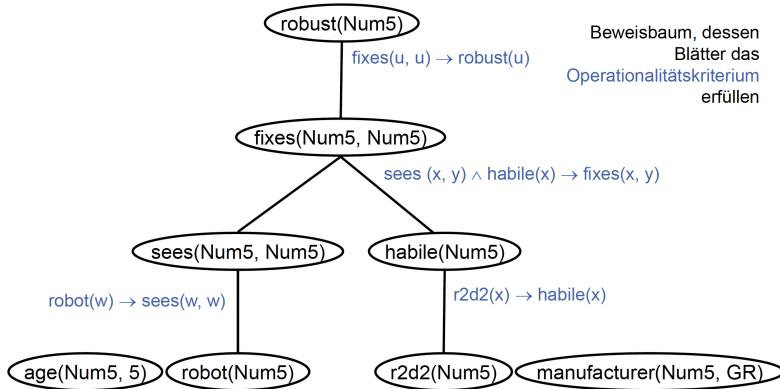
■ Bereichstheorie:

- fixes(u, u) → robust(u)
- sees (x, y) ∧ habile(x) → fixes(x, y)
- robot(w) → sees(w, w)
- r2d2 (x) → habile (x), c3po (x) → habile(x)

■ Operationalitätskriterium:

- Das Zielkonzept ist beschrieben in Termen der Beispielbeschreibung oder einfach auswertbarer Prädikate aus der Bereichstheorie (z.B. less (x, y))

EBG: Explain



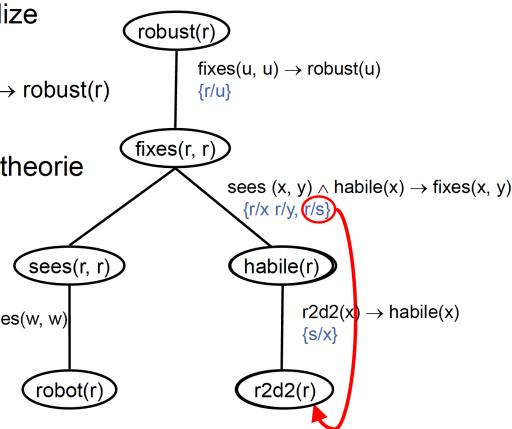
EBG: Generalize

■ 2. Schritt: Generalize

■ Neue Regel:

■ $\text{robot}(r) \wedge \text{r2d2}(r) \rightarrow \text{robust}(r)$

■ Wird der Bereichstheorie hinzugefügt



KBANN

■ Algorithmus

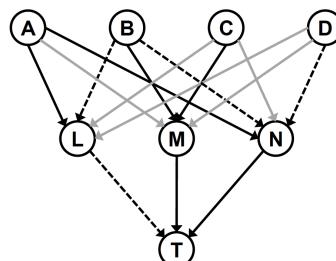
1. Pro Instanzattribut ein Netz-Input
2. Für jede Klausel der Bereichstheorie ein Neuron wie folgt einfügen:
 - Eingang mit geprüften Attributen verknüpfen, verwende Gewicht W für nicht-negierte Attribute und $-W$ für negierte
 - Setze Schwellwert auf $-(n - 0.5)W$ (mit $n = \text{Anzahl der nicht-negierten Bedingungsteile}$)
3. Zusätzliche Verbindungen, um jedes Element auf Schicht i mit jedem auf Schicht $i+1$ zu verbinden, zufällige kleine Gewichte zuweisen
4. Backpropagation-Algorithmus mit den Trainingsbeispielen auf initiales Netz anwenden

■ Beispiel:

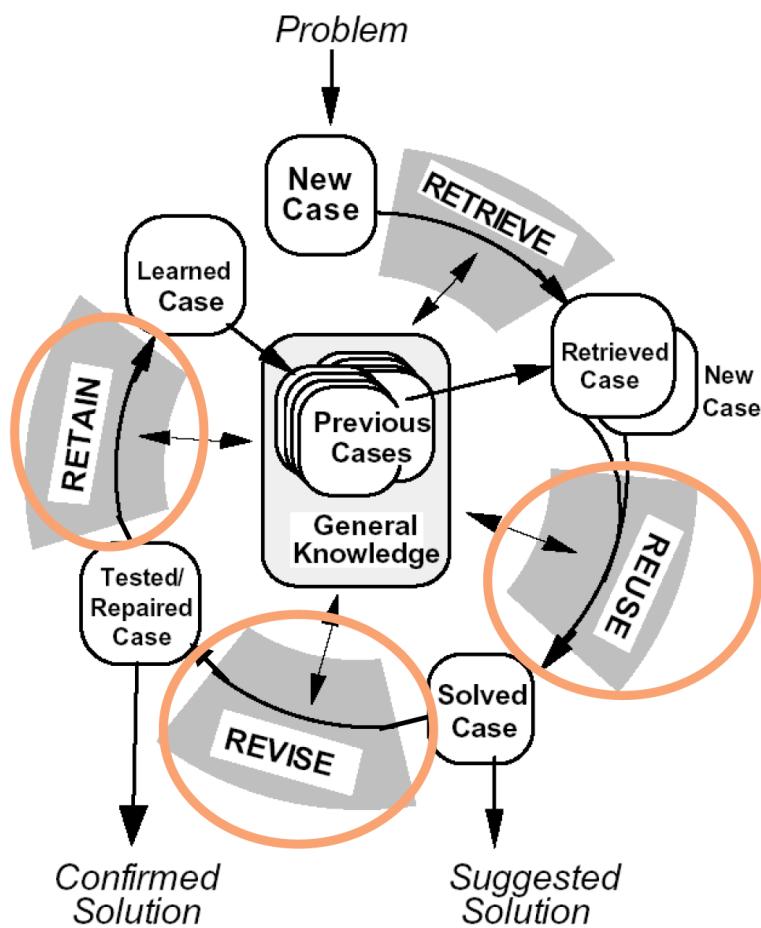
```

 $T \leftarrow \neg L, M, N$ 
 $L \leftarrow A, \neg B$ 
 $M \leftarrow B, C$ 
 $N \leftarrow A, \neg B, \neg D$ 

```



CBR Zyklus



Perzepron

x – Eingabevektor

t – Target (Soll-Ausgabe)

