Intelligente Systeme

Zusammenfassung

PER NATZSCHKA

Inhaltsverzeichnis

Author's address: Per Natzschka, per.natzschka1@mailbox.tu-dresden.de.

1 SUCHE

1.1 Klassifizierung

1.1.1 *Parameter.*

- b: max. Verzweigungsfaktor (branching factor)
- d: Tiefe der besten Lösung (depth)
- m: max. Tiefe des Baumes (kann ∞ sein)

1.1.2 *Bewertung.*

- Zeitkomplexität: Zahl expandierter Knoten?
- Speicherkomplexität: Zahl an Knoten im Speicher?
- Vollständigkeit: findet Lösung?
- Optimalität: findet beste Lösung?

1.2 Uninformierte Suche

1.2.1 *Breiten- und Tiefensuche.*

Strategie	Breitensuche BFS	Tiefensuche DFS	Tiefenbeschränkte Suche	Iteratives Vertiefen
Datenstruktur	FIFO	FILO	FILO	FILO
Zeitkomplexität	$O(b^d)$	$O(b^m)$	$O(b^t)$	$O(b^d)$
Speicherkomplexität	$O(b^d)$	$O(b \cdot m)$	$O(b \cdot t)$	$O(b \cdot d)$
Vollständigkeit	ja, wenn b endlich	ja, wenn d endlich und Schleifenüberprüfung	ja, wenn $d \leq t$	ja, wenn $d < \infty$ und $b < \infty$
Optimalität	✓(wenn b endlich)	×	×	✓

Tabelle 1. Vergleich Breiten- und Tiefensuche

1.2.2 *Uniforme Kostensuche.*

- Dijkstra
- expandiere immer Knoten mit geringsten Kosten von der Wurzel
- optimal
- vollständig

1.3 Informierte Suche

Strategie	Greedy-Suche	A*-Suche
Vollständigkeit	ja, bei Schleifenüberprüfung	
Zeitkomplexität	$O(b^m)$	$O(b^m)$
Speicherkomplexität	$O(b^m)$	$O(b^m)$
Optimalität	×	✓(wenn h zulässig)

Tabelle 2. Vergleich informierter Suchstrategien

- Heuristik
  - $h : V \to \mathbb{R}$
  - schätzt Kosten von Knoten zum Ziel
  - $h(Ziel) = 0$
  - $\forall_{n \in V} : h(n) \geq 0$
- Zulässige Heuristik:
  - $\forall_{n \in V} : h(n) \leq h^*(n)$
  - h^* sind wahre Kosten
- Konsistente Heuristik
  - $\forall_{n,n' \in V} : h(n) \leq c(n, n') + h(n')$
  - n' ist Nachfolger von n
  - $c(n, n')$ sind Kosten für Übergang $n \to n'$
  - immer zulässig

1.3.1 Greedy Suche. Knoten mit geringster Distanz zum Ziel wird zuerst expandiert.

$$v_{next} = arg\,min_{v \in V} h(v)$$

1.3.2 A-Suche.*

- neben h auch bisherige Kosten $g : V \to \mathbb{R}$ einberechnet
- Kostenfunktion $f : V \to \mathbb{R}, n \mapsto g(n) + h(n)$.
- gegen Speicherüberlauf: Schwellwert K
  - $K \geq K*$
  - K^* sind wahre Kosten
  - Knoten mit $f(n) > K$ werden nicht besucht
  - Bestimmung von K durch nichtoptimalen Algorithmus (z. B. Greedy)

2 ENTITÄTENERKENNUNG

Strategie	Vorteile	Nachteile
Lexikon	• einfach • schnell	• Mehrdeutigkeiten nicht erkannt • Vollständigkeit nicht garantiert • Pflege (keine Abstraktionen/Muster)
Reguläre Ausdrücke	• kompakte Repräsentation	• Woher kommt Ausdruck? • manuelle Definition/Debugging mühsam • Mehrdeutigkeiten nicht erkannt
Rand-klassifikator	• Mehrdeutigkeiten teilweise aufgelöst (lokaler Kontext)	• Qualität von Trainingsdaten • Auswahl von Merkmalen
Token-Fenster	• Mehrdeutigkeit aufgelöst (lokaler Kontext)	• Qualität von Trainingsdaten • Auswahl von Merkmalen
Hidden Markov Models (HMM)	• Mehrdeutigkeit aufgelöst	• Qualität von Trainingsdaten

Tabelle 3. Vergleich der Strategien zur Entitätenerkennung

- Wissensakquise
  - per Hand
  - Regeln lernen (Entscheidungsbäume)
  - Natürliche Sprachverarbeitung
- Informationsextraktion
  (1) Entitäten lokalisieren
  (2) Entitäten klassifizieren
  (3) Fakten extrahieren
- Probleme
  - Sprachlich
    * lexikalisch (tumor ↔ tumour)
    * orthographisch (α-helix ↔ alpha-helix)
    * strukturell (lung cancer ↔ cancer of the lung)
    * morphologisch (kick ↔ kicked)
  - Mehrdeutigkeit (Siemens, Paris)
  - Anaphora
    * Pronomen
    * Proformen (Er fliegt nach Paris. Er will *dort* Urlaub machen.)
    * Bridging (Der Motor ist kaputt. Der Keilriemen ist gerissen ↔ Der Motor ist kaputt. Der Schnürsenkel ist gerissen.)
  - Metonymie/Vertauschung (Schiller lesen)
  - Synekdoche (Ober-/Überbegriffe)
- Text preprocessing
  - Format

  - Satzgrenzen
  - Tokenisierung
  - Stemming
- direkte Suche linear in
  - Textgröße
  - Zahl der Lexikoneinträge
  - Länge der Lexikoneinträge
- Boyer-Moore (basically KMP)
- Zipf's Law
  - Worte nach Häufigkeit sortieren
  - Wahrscheinlichkeit eines Wortes invers proportional zu Rang
- Trie
  - Baum
  - n-te Verzweigung $\leftrightarrow$ n-ter Buchstabe
- Radix-Tree
  - wie Trie
  - Verzweigungen mit Teilwörtern
- Levenshtein-Distanz
  - Top-Down-Implementierung: $O(2^n)$
  - Bottom-Up-Implementierung: $O(n^2)$ (dynamisches Programmieren, u. a. Speicherung der Zwischenwerte in Matrix)
- Dice-Koeffizient
  - Betrachtung v. Trigrammen
  - $t(Peter) = \{Pet, ete, ter\}$
  - $dice(a, b) = 2 \cdot \frac{|t(a) \cap t(b)|}{|t(a)| + |t(b)|} \in [0, 1]$
  - Reihenfolge geht verloren

2.1 Klassifikatoren

2.1.1 Randklassifikator.

- Klassifiziere Leerstellen zwischen Token
  - Anfang und Ende von Entitäten
  - Satzgrenzen
- Leerstellen als Vektoren von Merkmalen, z. B.:
  - nächstes Token beginnt mit Großbuchstaben
  - vorheriges ist Zahl
  - vorheriges ist "in"

2.1.2 Token-Fenster.

- Klassifikation anhand von Kontextfenster
- Fenstergröße?
- Merkmale?

  - Token, Stämme, POS-Tags
  - n-grams
  - Präfixe/Suffixe
  - Vorher vergebene Klassen
- Lernen anhand von Merkmalsvektoren aus Beispielen

2.1.3 Hidden Markov Models.

- Markov-Modelle
  - probabilistische Modelle
  - modellieren Zustandsübergänge
  - nur von aktuellem Zustand abhängig
- Hidden Markov Models
  - Zustände selbst nicht betrachtet
  - Übergangswahrscheinlichkeiten abhängig von allen vorherigen Zuständen
- $\Theta = (S, \Sigma, A, B, \Pi)$
  - S: Zustände (semantische Klassen/Labels)
  - Σ: Alphabet (Beobachtungen/Tokens)
  - A: Übergangswahrscheinlichkeiten
  - B: Emissionswahrscheinlichkeiten
  - Π: Anfangszustandswahrscheinlichkeiten
- Wahrscheinlichkeiten an Trainingsdaten gelernt
- Wahrscheinlichkeit von Beobachtungssequenz o und Zustandssequenz s: $P(s \cap o)$
  - $P(s \cap o) = P(o|s)P(s)$
  - $P(s \cap o) \propto P(o_0|s_0)P(s_0) \prod_{t=1}^{|o|} P(o_t|s_t)P(s_t|s_{t-1})$
- Bestimmung der besten Lösung durch $arg\,max_s P(s, o)$ $\rightarrow$ ineffizient
- Viterbi-Algorithmus (effizient)
  - dynamische Programmierung
  - Tabelle: $S \times o$
  - $F(i, 0) = \Pi(i) \cdot B(i, 0)$
  - $F(i, j) = max_r(F(r, j-1) \cdot A(r, i)) \cdot B(i, j)$
  - Spaltenweise Berechnung von links nach rechts

2.2 Bewertung

- Precision: $p = \frac{TP}{TP+FP}$
- Recall: $r = \frac{TP}{TP+FN}$
- F-Measure: $2 \cdot \frac{p \cdot r}{p+r}$ (Harmonisches Mittel)
- k-fache Kreuzvalidierung
  - Daten in k Gruppen aufgeteilt
  - Training auf $k-1$ Gruppen
  - Validierung mit k. Gruppe

3 RELATIONSEXTRAKTION

Strategie	Vorteile	Nachteile
Maschinelles Lernen	• einfach	• Qualität von Trainingsdaten • Auswahl von Merkmalen
Kookkurrenz	• einfach • hoher Recall	• kein Relationstyp
Reguläre Ausdrücke	• hohe Precision • Transparenz	• Generierung

Tabelle 4. Vergleich der Strategien zur Relationsextraktion

3.1 Kookkurrenz

- Wie oft treten Begriffe gemeinsam auf?
- Signifikanz
  - steigt mit gemeinsamen Auftreten
  - sinkt mit nicht gemeinsamen Auftreten
- log odds ratio
  - $c(a,b) = \log_2(\frac{n \cdot n_{ab}}{n_a \cdot n_b})$
  - n ist Anzahl aller Dokumente
  - n_{ab} ist Anzahl der Dokumente mit a und b
  - n_a ist Anzahl der Dokumente mit a
  - log optional/Basis frei wählbar (nur scaling)

3.2 Reguläre Ausdrücke

Syntax	Semantik
.	beliebiges $a \in \Sigma$
$a*$	$\{a\}^*$
$a+$	$\{a\}^+$
$a?$	$\{\varepsilon, a\}$
$a \mid b$	$\{a, b\}$

Tabelle 5. Syntax und Semantik von Regulären Ausdrücken

3.2.1 Vorgehen.

- Lernen anhand positiver Beispiele von Entitäten in Relation
- Zeichenketten zwischen Entitäten
- Multiples Sequenzalignement der Zeichenketten
- Ableiten eines regulären Ausdrucks

3.2.2 *Multiple Sequence Alignement (MSA).*

- Dynamische Programmierung
  - bei m Wörtern, Levenshtein über m Dimensionen
  - m Wörter der Länge n
  - n^m Zellen mit jeweils $2^m - 1$ Nachbarn
  - $O(2^m - 1)$
- Optimierung: Greedy (Inkrementelles Positionieren)
  - zwei Strings alignen
  - nächste Strings mit bisherigem Alignement alignen
  - Kosten bei jedem Schritt inkrementieren
  - Pro: $O(m \cdot n^2)$
  - Con: nicht optimal, Reihenfolge der Abarbeitung bestimmt stark das Ergebnis
  - Heuristik: Reihenfolge der Strings
  - Clustering der Zeichenketten nach paarweiser Distanz
  - Reihenfolge basierend auf Clustern (lokales Optimum)
  - lokales Optimum
- Optimierung: A* (Optimales Positionieren)
  - Knoten: Zellen der Matrix
  - Kanten: Verbindung benachbarter Zellen
  - Startknoten: $(0, 0, 0)$
  - Zielknoten: (n, n, n)
  - Kosten h^*: Kosten für multiples Alignement der Reste der Zeichenkette
  - multiple Alignements schlechter als Summe paarweiser Alignements
  - Heuristik h ist Summe der Kosten aller paarweisen Alignements
  - $h(i_1, \ldots, i_m) = \sum_{a,b \in \{l_1[i_1:], \ldots, l_m[i_m:]\}} dist(a, b)$
  - für Heuristik Betrachtung von $\frac{m \cdot (m-1)}{2}$ Alignements

4 ENTSCHEIDUNGSBÄUME

Vorteile	Nachteile
• einfach, da tabellarisches Wissen oft verfügbar • schnell durch Greedy-Ansatz (Entropie) • robust durch Random Forests	• stark von Datenerhebung abhängig • Wahl der Atribute • Wahl der Wertebereiche

Tabelle 6. Vor- und Nachteile von Entscheidungsbäumen

- Entscheidungsbaum
  - Innere Knoten: Attribute
  - Kanten: Wert der Attribute
  - Blätter: Einteilung der Daten
- Ziel: möglichst kleiner Entscheidungsbaum
  - globales Optimum
  - lokales Optimum (greedy)
- Zahl möglicher Bäume mit m Blättern: $T(m)$
  - $T(m) = (2m-3)!!$
  - !! ist Doppelfakultät
    - $1!! := 1$
    - $n!! := n \cdot (n-2)!!$
- Rekursives Teilen der Beispiele
  (1) Teilgruppe ist homogen → Knoten erhält Label der Teilgruppe
  (2) Teilgruppe ist inhomogen → wähle nächstes Attribut
  (3) Es gibt keine Attribute mehr → Knoten erhält Label der Mehrheit der Teilgruppe
  (4) Es gibt keine Beispiele mehr → Knoten erhält Label der Mehrheit der Teilgruppe des Elternknotens
- Entropie H
  - Bewertung der Homogenität einer Menge
  - $H(C) = \sum_{i \in C} -p_i \cdot \log_2(p_i)$
  - T: Menge der Beispiele, p_i: Anteil der Beispiele, die zu Klasse i gehören, C: Klassen
  - Informationsgewinn durch Attribut A: $IG(A) = H(T) - R(T)$
  - Restentropie $R(T) = \sum_{i=1}^{v} H(T_i) \cdot \frac{|T_i|}{|T|}$
  - $H(T_i) = \sum_{j \in C} -p_j \cdot \log_2(p_j)$
  - Werte des Attributs A: $[1, \ldots, v]$
- Bestimmung des besten Attributes durch $arg\,max_{a \in A} IG(a)$
- Random Forests
  - Overfitting vermeiden
  - Bootstrapping: Vergleiche Baum mit Bäumen, die aus variierten Daten erzeugt wurden
  - viele Bäume generieren und aggregieren

5 ARGUMENTATION

5.1 Datenbasis

- Fakten
  - manuell
  - Datenbanken
  - Entitäten-/Relationsextraktion
- Regeln
  - manuell
  - Entscheidungsbäume
  - Formale Konzeptanalyse

5.2 Argumente

Eigenschaft	$\neg$	*not*
Form	explizit	implizit
Aussprache	neg	not
Beweis	gilt bewiesenermaßen nicht	konnte nicht bewiesen werden

Tabelle 7. Vergleich der Negationsformen

- Objektivliterale L
  - Atome: A
  - explizite Negation eines Atoms: $\neg A$
- Defaultliteral $notL$
- Regel $r = L \leftarrow L_1, \ldots, L_m, notL_{m+1}, \ldots, notL_{m+n}$;
  - Kopf von r: L
  - Rumpf von r: $L_1, \ldots, L_m, notL_{m+1}, \ldots, notL_{m+n}$
- Argument für Objektivliteral L_1
  - Sequenz von Regeln: $[r_1, \ldots, r_k]$
  - L_i ist Kopf von r_i
  - $\forall_{L_i \leftarrow \ldots, L_j, \ldots; \in [r_1, \ldots, r_k] \wedge L_j\, Objektivliteral} \exists r_j \in [r_1, \ldots, r_k] : i < j$

5.3 Attacken

5.3.1 *Grundattacken.*

- A **u***ndercuts* B
  - A invalidiert Prämisse von B
  - $A = [\ldots; L \leftarrow Body; \ldots]$
  - $B = [\ldots; L' \leftarrow \ldots, not\ L, \ldots; \ldots]$
- A **r***ebuts* B
  - A widerspricht B
  - $A = [\ldots; L \leftarrow Body; \ldots]$

  - $B = [\ldots; \neg L \leftarrow Body; \ldots]$
  - symmetrisch

5.3.2 *Abgeleitete Attacken.*

- $A\ \mathbf{a}ttacks\ B = A\ u\ B \vee A\ r\ B$
- $A\ \mathbf{d}efeats\ B = A\ u\ B \vee (A\ r\ B \wedge not\ B\ u\ A)$
- $A\ \mathbf{s}trongly\ \mathbf{a}ttacks\ B = A\ a\ B \wedge not\ B\ u\ A$
- $A\ \mathbf{s}trongly\ \mathbf{u}ndercuts\ B = A\ u\ B \wedge not\ B\ u\ A$

5.3.3 *Betrachtung als Mengen.*

- Undercuts: u
- Rebuts: r
- Attacks: $a = u \cup r$
- Defeats: $d = u \cup (r \setminus \overline{u})$
- Strongly attacks: $sa = (u \cup r) \setminus \overline{u}$
- Strongly undercuts: $su = u \setminus \overline{u}$

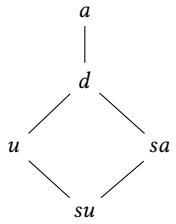

Fig. 1. Obermengenbeziehungen der Attacken

5.4 Semantik

- x/y-Fixpunktsemantik: jede x-Attacke muss durch y-Gegenattacke verteidigt werden
- $F_{x/y} = \{A \mid A \text{ ist } x/y\text{-akzeptierbar bzgl. } S\}$
- Einteilung von Argumenten
  - x/y-gerechtfertigte Argumente $J_{x/y}$: kleinster Fixpunkt von $F_{x/y}$
  - x/y-verworfene Argumente: x-Attackiert durch gerechtfertigtes Argument
  - x/y-verteidigbare Argumente: weder gerechtfertigt, noch verworfen
- Vorgehen (Iterativ)
  - nimm alle Regeln auf, die nicht durch Programmklausel x-attackiert werden
  - wenn Regel x-attackiert wird, nimm sie auf, wenn sie durch bereits aufgenommene Regel y-verteidigt werden kann

Semantiken	Dung	Prakken & Sartor	Well-founded Semantics WFS	WFSX
x/y	a/u	d/su	u/u	u/a

Tabelle 8. Verschiedene Semantiken

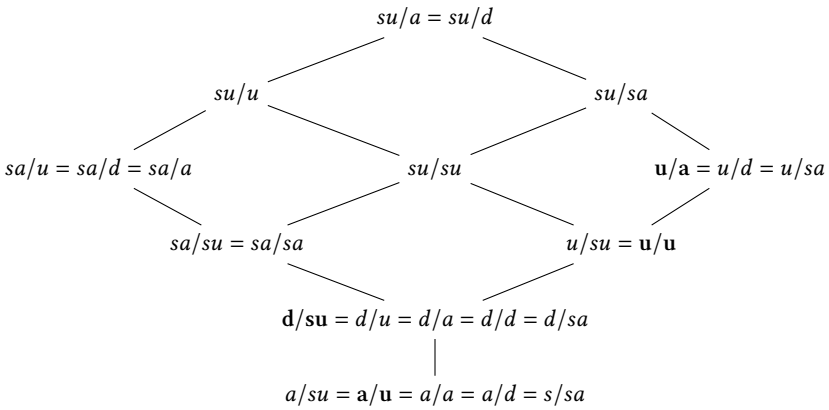

Fig. 2. Obermengenbeziehungen der Semantiken

6 PROBABILISTISCHES SCHLIESSEN

6.1 Definitionen

6.1.1 Bedingte Wahrscheinlichkeit.

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}$$

$$\begin{aligned} P(A \cap B) &= P(A \mid B) \cdot P(B) \\ &= P(B \mid A) \cdot P(A) \\ &= P(A) \cdot P(B) \qquad \text{(wenn } A \text{ und } B \text{ unabhängig)} \end{aligned}$$

$$\begin{aligned} P(A \cup B) &= P(A) + P(B) - P(A \cap B) \\ &= P(A) + P(B) - P(A) \cdot P(B) \qquad \text{(wenn } A \text{ und } B \text{ unabhängig)} \end{aligned}$$

6.1.2 Satz von Bayes.

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)} = \frac{P(A \cap B)}{P(B)}$$

$$P(A \mid B_1 \cap \cdots \cap B_k) = \frac{P(B_1 \cap \cdots \cap B_k \mid A) \cdot P(A)}{P(B_1 \cap \cdots \cap B_k)} \qquad \text{(Verallgemeinerung)}$$

$$= \frac{P(A) \cdot \prod_{i=1}^{k} P(B_i \mid A)}{P(B_1 \cap \cdots \cap B_k)} \qquad (B_1, \ldots, B_k \text{ unabhängig})$$

$$P(A_1 \mid B_1 \cap \cdots \cap B_k) \lesseqgtr P(A_2 \mid B_1 \cap \cdots \cap B_k)$$

$$\Leftrightarrow P(A_1) \cdot \prod_{i=1}^{k} P(B_i \mid A_1) \lesseqgtr P(A_2) \cdot \prod_{i=1}^{k} P(B_i \mid A_2)$$

6.2 Anwendung

- Beispiel Spamfilter
- $P(spam \mid \text{Congratulations ur awarded}) \lesseqgtr P(ham \mid \text{Congratulations ur awarded})$
- Problem, wenn "ur" nicht in Trainingsdaten
  - $P(spam \mid ur) = \frac{\#(\text{Mails mit "ur"})}{\#(\text{Spam-Mails})} = 0$
  - Gesamtwahrscheinlichkeit wird 0
  - Laplace-Smoothing: $P(spam \mid ur) = \frac{\#(\text{Mails mit "ur"})+1}{\#(\text{Spam-Mails})+|\text{Vokabular}|}$

7 NEURONALE NETZE

7.1 Aufbau

- Neuronen
  - Inputs
  - Gewichte
  - Bias
  - Output: $y = f_w(x)$ mit Aktivierungsfunktion f_w
- Topologische Anordnung
  - Input-Layer
  - Hidden-Layers
  - Output-Layer

7.2 Lineare Regression

- $f_{w_0,w_1}(x) = w_1 x + w_0 = y$
- Verlustfunktion
  - $Loss(f_w) = \sum_{j=1}^{N} (y_j - f_w(x_j))^2 = \sum_{j=1}^{N} (y_j - (w_1 x_j + w_0))^2$
  - N: Anzahl Trainingswerte
  - x_j: Argument des j-ten Trainingswertes
  - y_j: j-ter Trainingswert
- Verlust minimieren: Geschlossene Lösung
  - Minimum von *Loss* berechnen
  - $0 = \frac{\partial}{\partial w_0} Loss(f_w) = 2 \cdot \sum_{j=1}^{N} y_j - (w_1 x_j + w_0)$
  - $w_0 = \frac{(\sum_{j=1}^{N} y_j) - w_1 \sum_{j=1}^{N} x_j}{N}$
  - $0 = \frac{\partial}{\partial w_1} Loss(f_w) = 2 \cdot \sum_{j=1}^{N} (y_j - (w_1 x_j + w_0)) x_j$
  - $w_1 = \frac{(N \sum_{j=1}^{N} x_j y_j) - (\sum_{j=1}^{N} x_j)(\sum_{j=1}^{N} y_j)}{N(\sum_{j=1}^{N} x_j^2) - (\sum_{j=1}^{N} x_j)^2}$
  - meist zu kompliziert → *Gradient Descent*
- Gradient Descent
  - Greedy-Ansatz
  - Gradient
    - $\nabla f(x_1, \ldots, x_k) = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_k} \end{pmatrix}$

        * zeigt in Richtung des stärksten Anstieges
        * Betrag gibt Stärke des Anstiegs an
    - $w_0 \leftarrow w_0 - \alpha(\frac{\partial}{\partial w_0} Loss(f_w))$
    - $w_1 \leftarrow w_1 - \alpha(\frac{\partial}{\partial w_1} Loss(f_w))$
    - Schrittgröße α
        * am Anfang groß wählen (~ 0.01), schrittweise verringern
        * Schrittweise verringern
        * zu groß: kein Optimum
        * zu klein: viele Schritte

7.3 Klassifikation

7.3.1 Lineare Klassifikation.

- zwei Klassen nach linearer Regression: C_1, C_2
- $x_2 = w_1 x_1 + w_0 \Rightarrow w_0 x_0 + w_1 x_1 + w_2 x_2 = 0$, für $x_0 = 1$ und $w_2 = -1$
- $\begin{pmatrix} w_0 \\ w_1 \\ w_2 \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} = \vec{w} \cdot \vec{x} = 0$ Schwellwert
- $\vec{w} \cdot \vec{x} < 0 \Rightarrow C_1$
- $\vec{w} \cdot \vec{x} > 0 \Rightarrow C_2$
- Schwellwert als Aktivierungsfunktion f_w
    - $f_w(\vec{x}) = Step(\vec{w} \cdot \vec{x})$
    - $Step(x) = \begin{cases} 1, \text{wenn } x > 0 \\ 0, \text{wenn } x < 0 \end{cases}$
    - Problem: $Loss$ nicht differenzierbar
    - Lösung: $f_w(x) = Logistic(x) = \frac{1}{1+e^{-x}}$
    - $Logistic'(x) = Logistic(x) \cdot (1 - Logistic(x))$
- Gradient Descent
- $f_w(x) = Logistic(x)$
    - $w_i \leftarrow w_i - \alpha \frac{\partial}{\partial w_i} (y - f_w(x))^2$
    - $w_i \leftarrow w_i - \alpha(y - f_w(x)) \cdot f_w(x)(1 - f_w(x)) \cdot (-x_i)$
- Grenze: Nichtlineare Daten

7.3.2 Nichtlineare Klassifikation.

- $L + 1$ vernetzte Schichten, K Klassen
- Input-Layer: $l = 0$
- Output-Layer: $l = L$
- Neuronen definiert durch
    - Layer: l
    - Index: i, j
    - Input für Layer l: $h_i^{(l-1)}$
    - Gewicht für $h_i^{(l-1)}$: $w_{i,j}^{(l)}$

  - Gesamtinput: $a_j^{(l)} = \sum w_{i,j}^{(l)} \cdot h_i^{(l-1)} = \vec{w}_j{}^{(l)} \cdot \vec{h}^{(l-1)}$
  - Aktivierungsfunktion: $h(x)$
  - Ausgabe: $h_j^{(l)} = h(a_j^{(l)})$
- Aktivierungsfunktion Softmax
  - erhält Wahrscheinlichkeitsverteilung f aus Outputlayer
  - $Logistic$ in K Dimensionen
  - $f_k = \frac{e^{a_k^{(L)}}}{\sum e^{a_j^{(L)}}}$ für $k = 1, \ldots, K$
  - $f_k \in [0,1] \wedge \sum_{k=1}^{K} f_k = 1$
  - $f = [f_1, \ldots, f_K]$ (Wahrscheinlichkeitsverteilung)
- Darstellung als Funktion
  - $a_j^{(l)} = \vec{w}_j{}^{(l)} \cdot \vec{h}^{(l-1)}$
  - Gewichte der Neuronen von Ebene l bilden Matrix $W^{(l)}$
  - $a^{(l)} = W^{(l)} \cdot \vec{h}^{(l-1)} = \begin{pmatrix} w_{1,1} & \ldots & w_{K,1} \\ \vdots & \ddots & \vdots \\ w_{K,1} & \ldots & w_{K,K} \end{pmatrix}^{(l)} \begin{pmatrix} \vec{h}_1 \\ \vdots \\ \vec{h}_K \end{pmatrix}^{(l-1)}$
  - $f(\vec{x}) = f[a^{(L)}(h^{(L-1)}(\ldots(h^{(1)}(a^{(1)}(\vec{x})))))]$
- Kreuzentropie (siehe 4)
  - misst Unordnung zwischen p und q
  - $H(T) = \sum_{i=1}^{K} -p_i \cdot \log_2(q_i)$
  - Bewertung der Ähnlichkeit von f und y
  - $L(y, f) = -\sum_{i=1}^{K} y_i \cdot \log_2(f_i)$
  - Durchschnittliche Kreuzentropie bei N Beispielen: $\frac{1}{N} \sum_{j=1}^{N} L(y_j, f(\vec{x_j})$
- Training
  - Trainingsbeispiele (x, y)
  - lerne Gewichte $w_{i,j}^{(l)}$
  - Gradient von Lossfunvtion zu Hidden Layers → Back-propagation
  - Stochastic Gradient Descent
    - Gradienten über alle N Beispiele berechnen → zu viel
    - Gradienten über jeweils ein Beispiel → zu wenig
    - Gradienten über Mini-Batches (z. B. 256)
  - Lernrate α (siehe 7.2)
  - Overfitting
    - zu starke Anpassung auf Trainingsdaten
    - $accuracy_{training} \gg accuracy_{validation}$
    - Auswahl der Trainingsdaten randomisieren (90/10 - Trainingsdaten/Validation)
    - Modellkomplexität (Zahl der Parameter) verringern

7.4 Grenzen

- Netzwerkarchitektur
- Lokale Optima, Heuristik
- Trainingsdaten
- Overfitting
- Intransparente Modelle

8 UNSUPERVISED LEARNING

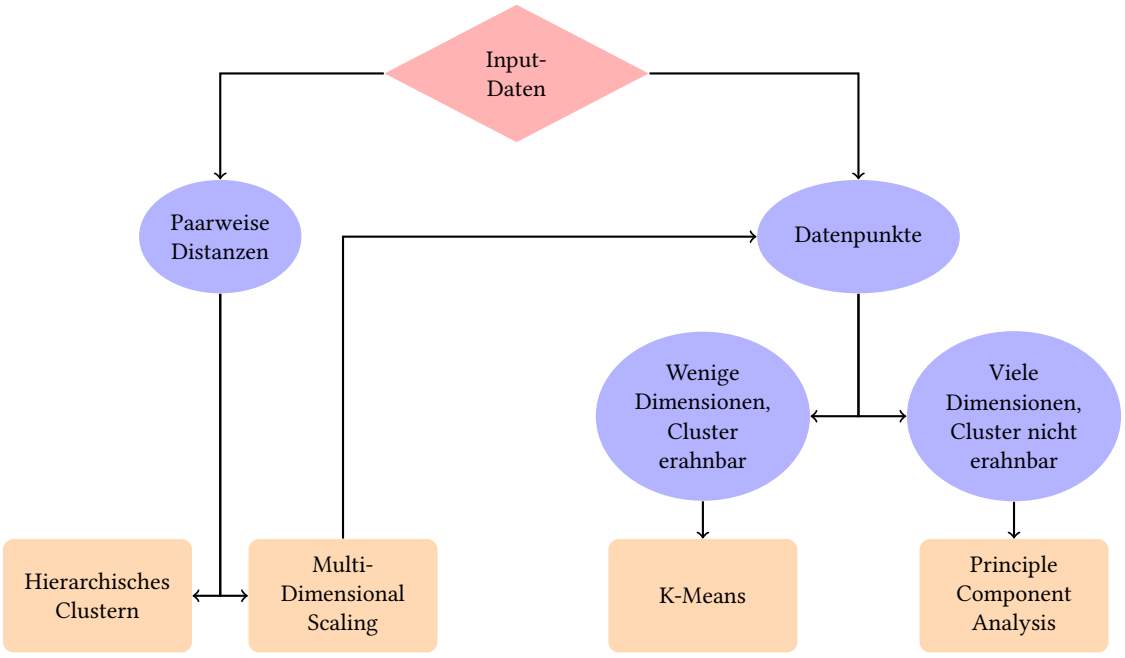

Fig. 3. Übersicht der Unsupervised-Learning-Methoden

- Supervised
  - Beispiele
    - Entscheidungsbäume
    - HMM
    - Bayes
    - Neuronale Netze
  - braucht große Mengen an Trainingsdaten
  - Qualität und Menge reicht oft nicht
- Unsupervised
  - Beispiel siehe Übersicht der Unsupervised-Learning-Methoden
  - braucht keine Trainingsdaten

Methode	Vorteile	Nachteile
K-Means	• einfach • schnell ($O(n)$)	• Greedy, kein globales Optimum → Bootstrapping • Wie viele Cluster k? • Welches Distanzmaß?
Hierarchical Clustering		• Komplexität $O(n^3)$ • Greedy, kein globales Optimum • Welche Linkage-Methode • Welches Distanzmaß?
Principal Component Analysis	• einfach • Visualisierung • Komplexität $O(p^2 \cdot n + p^3)$	• Komplexität $O(p^2 \cdot n + p^3)$ • n Punkte, p Dimensionen
Multi-Dimensional Scaling		• nicht immer möglich z. B. $D(A,B) = 1 = D(B,C)$ $D(A,C) = 5$

Tabelle 9. Vergleich der Methoden des Unsupervised Learnings

8.1 K-Means

- wähle erste k Elemente als Clusterzentren (Repräsentanten)
- weise jedes Element Cluster zu, bei dem sich Varianz am wenigsten erhöht (geringste Distanz zum Clusterzentrum)
- Wahl eines neuen Repräsentanten anhand der Elemente im Cluster
- wiederholen

8.2 Hierarchical Clustering

- Input: Paarweise Distanzen (Heuristik)
- Output: Baum mit Objekten als Knoten
- Iteratives mergen der Objekte mit kleinster Distanz
- Distanz des neuen Clusters $w = (u, v)$ zu anderen Objekten x
  - Single/Complete linkage
    - **Single** linkage: $D(x, w) = \min(D(x, u), D(x, v))$
    - **Complete** linkage: $D(x, w) = \max(D(x, u), D(x, v))$
  - **P**air **G**roup **M**ethod using **A**rithmetic mean (PGMA)
    - **Average** linkage/Weighted PGMA (WPGMA): $D(x, w) = \frac{D(x,u) + D(x,v)}{2}$
    - Unweighted PGMA (**UPGMA**): $D(x, w) = \frac{m_u \cdot D(x,u) + m_v \cdot D(x,v)}{m_u + m_v}$
    - m_u ist Anzahl Knoten in u
  - Linkage hat Einfluss auf Ergebnis

8.3 Principal Component Analysis

- Mehrere Hauptachsen
- Basiswechsel

  - rotiert Achsen
  - x_1-Achse entspricht erster Hauptachse, x_2-Achse zweiter, usw.
- Reduktion der Dimensionen → Suche der Cluster in kleineren Dimensionen
- Voraussetzung: Hoher erklärter Varianzanteil

8.4 Multi-Dimensional Scaling

- Verallgemeinerung der Principal Component Analysis mit messbaren Relationen zwischen Daten
- Relationen → Koordinaten
- Platzierung der ersten Koordinate A
- Platzierung der zweiten Koordinate B auf Kreis um A mit $r = D(A, B)$
- Sukzessives platzieren der weiteren Knoten auf Schnittpunkten der Kreise um Koordinaten