Überblick

- Klassifikation (weiter) und
- Clustering

Klassifikation/Distanzfunktionen

- Exkurs: Metrik

$d(i, j) \geq 0$: Distanz ist eine postitive Zahl

$d(i, i) = 0$: Distanz zu sich selbst ist 0

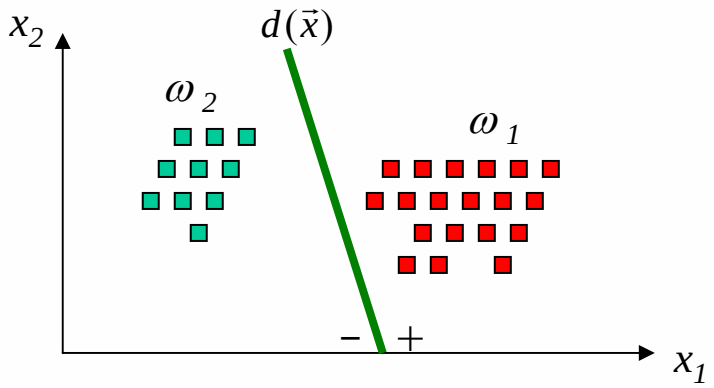
$d(i, j) = d(j, i)$: Distanz ist symmetrisch

$d(i, j) \leq d(i, h) + d(h, j)$: Dreiecksungleichung- > Ein direkter Weg von Objekt i nach j ist nicht mehr als ein Umweg über ein anderes Objekt h

Klassifikation/Diskriminantenfunktionen

[Tou & Gonzales 1974]

- Linear



$$d(\vec{x}) = w_1 x_1 + w_2 x_2 + w_3 = 0$$

Entscheidungsregel

$\vec{x} \rightarrow \omega_1$ falls $d(\vec{x}) > 0$

$\vec{x} \rightarrow \omega_2$ falls $d(\vec{x}) < 0$

(für $d(\vec{x}) = 0$ unbestimmt!)

- Probleme:
  - Geometrische Eigenschaften der Klassen
  - U.U. nichtlineare Form von d()
  - Bestimmung der Parameter w_i aus Klassenstichprobe

Klassifikation/Diskriminantenfunktionen

- Linear
  - Erweiterung n-dimensionaler Fall (für *n* Merkmale)

$$d(\vec{x}) = w_1 x_1 + w_2 x_2 + \ldots w_n x_n + w_{n+1} = \vec{W_0}^T \vec{x} + w_{n+1}$$

mit $\vec{W_0} = [w_1, w_2, \ldots, w_n]$ als Gewichts/Parameter – Vektor

  - Allgemein (2 Klassen Fall):

$$d(\vec{x}) = \vec{W}^T \vec{x} = \begin{cases} > 0 & \text{falls} \quad \vec{x} \to \omega_1 \\ < 0 & \text{falls} \quad \vec{x} \to \omega_2 \end{cases}$$

mit

$$\vec{W} = [w_1, \ldots, w_{n+1}]^T$$

$$\vec{x} = [x_1, \ldots, x_n, 1]^T$$

Erweiterte Vektornotation

Klassifikation/Diskriminantenfunktionen

- Linear

Allgemein (Multi-Klassen-Fall):

„Jede Klasse ist von den anderen Klassen durch *eine* Trennfunktion separierbar“

=> M Funktionen mit

$$d_i(\vec{x}) = \vec{W}_i^T \vec{x} = \begin{cases} > 0 & \text{falls} \quad \vec{x} \rightarrow \omega_i \\ < 0 & \text{sonst} \end{cases}$$

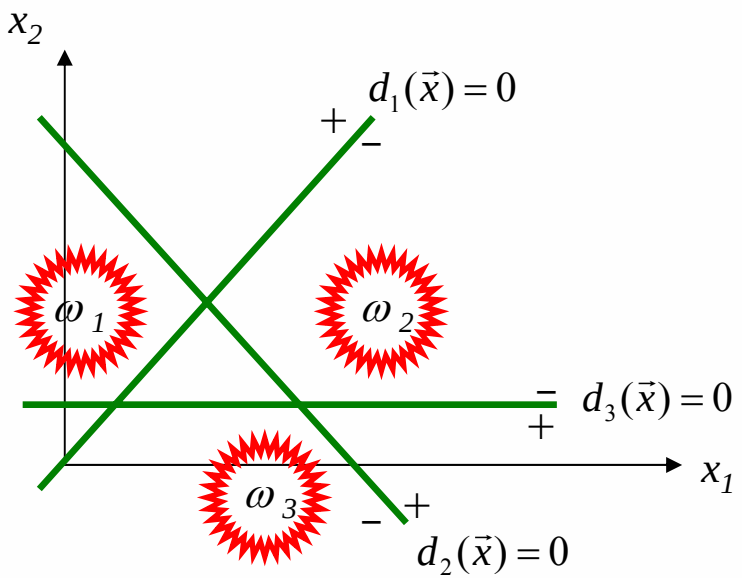
mit $i = 1,..,M$

$$\vec{W}_i = \left[w_{i_1}, ..., w_{i_n}, w_{i_{n+1}} \right]^T$$

Klassifikation/Diskriminantenfunktionen

- Linear

  Beispiel mit 3 Klassen



$$d_1(\vec{x}) = -x_1 + x_2 = 0$$
$$d_2(\vec{x}) = x_1 + x_2 - 5 = 0$$
$$d_3(\vec{x}) = -x_2 + 1 = 0$$

z.B.

$\vec{x} \rightarrow \omega_1$, wenn

$$d_1(\vec{x}) > 0,$$
$$d_2(\vec{x}) < 0$$
$$d_3(\vec{x}) < 0$$

oder wenn $\vec{x}' = (6,5)^T$ gemessen wurde, dann

$$\left.\begin{aligned} d_1(\vec{x}) &= -1 \\ d_2(\vec{x}) &= 6 \\ d_3(\vec{x}) &= -4 \end{aligned}\right\} \Rightarrow \vec{x}' \rightarrow \omega_2$$

Klassifikation/Diskriminantenfunktionen

- Linear

Allgemein (multi Klassen Fall):

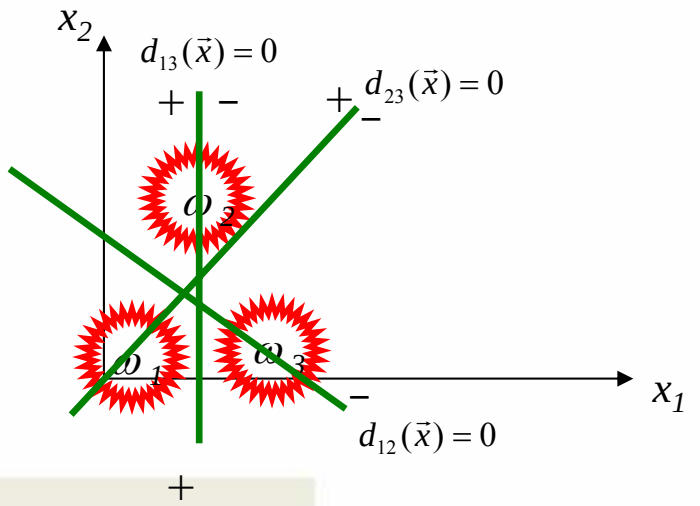
„Jede Klasse ist *nur paarweise* separierbar“

=> M(M-1)/2 Trennfunktionen mit

$$d_{ij}(\vec{x}) = \vec{W}_{ij}{}^{T}\vec{x} \quad \text{mit} \quad d_{ij}(\vec{x}) = -d_{ji}(\vec{x})$$

Entscheidungsregel

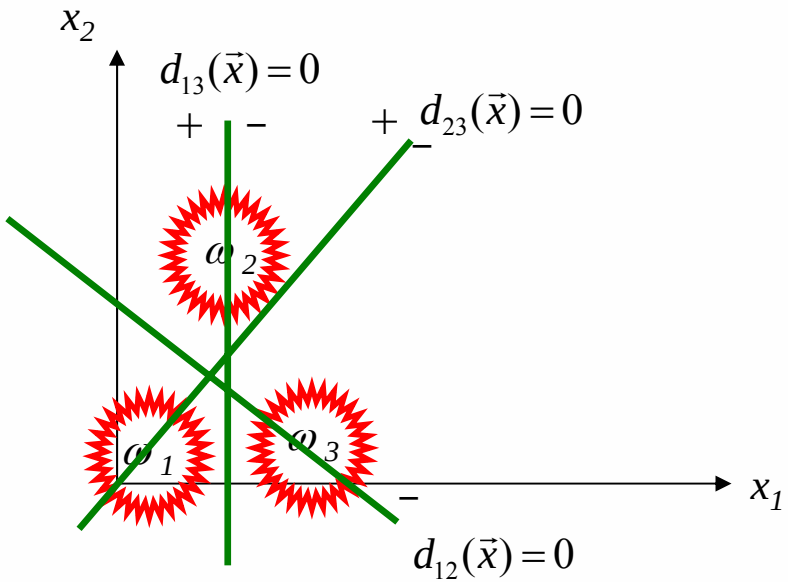
$$\vec{x} \rightarrow \omega_i \quad \text{wenn} \quad d_{ij}(\vec{x}) > 0, \quad \forall i \neq j$$



Klassifikation/Diskriminantenfunktionen

- Linear

  Beispiel mit 3 Klassen



$$d_{12}(\vec{x}) = -x_1 - x_2 + 5 = 0$$
$$d_{13}(\vec{x}) = -x_1 + 3 = 0$$
$$d_{23}(\vec{x}) = -x_1 + x_2 = 0$$

z.B.

$\vec{x} \rightarrow \omega_1$, wenn

$$d_{12}(\vec{x}) > 0,$$
$$d_{13}(\vec{x}) > 0$$

oder wenn $\vec{x}' = (4,3)^T$ gemessen wurde, dann

$$d_{12}(\vec{x}) = -2 \Leftrightarrow d_{21}(\vec{x}) = 2$$
$$d_{13}(\vec{x}) = -1 \Leftrightarrow d_{31}(\vec{x}) = 1$$
$$d_{23}(\vec{x}) = -1 \Leftrightarrow d_{32}(\vec{x}) = 1$$

$\Rightarrow \vec{x}' \rightarrow \omega_3$, da $d_{3j}(\vec{x}) > 0$ für $\forall i \neq j$

Klassifikation/Diskriminantenfunktionen

- Linear

  Generalisierung

$$d(\vec{x}) = w_1 f_1(\vec{x}) + w_2 f_2(\vec{x}) + \ldots w_k f_k(\vec{x}) + w_{k+1}$$
$$= \sum_{i=1}^{k+1} w_i f_i(\vec{x})$$

  mit

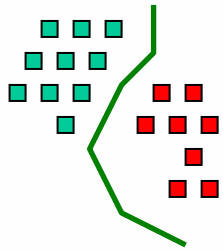
  $f_i(\vec{x})$: reelle Funktion

  $f_{k+1}(\vec{x}) = 1$

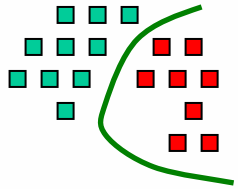
  $k+1 =$ Anzahl der Terme i.d. Entwicklung

  =>Diskriminantenfkt. ist abhängig von Wahl von $f_i()$ und $k+1$

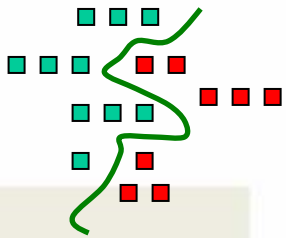
a) f_i ist lineares Polynom



b) f_i ist quadratisches Polynom

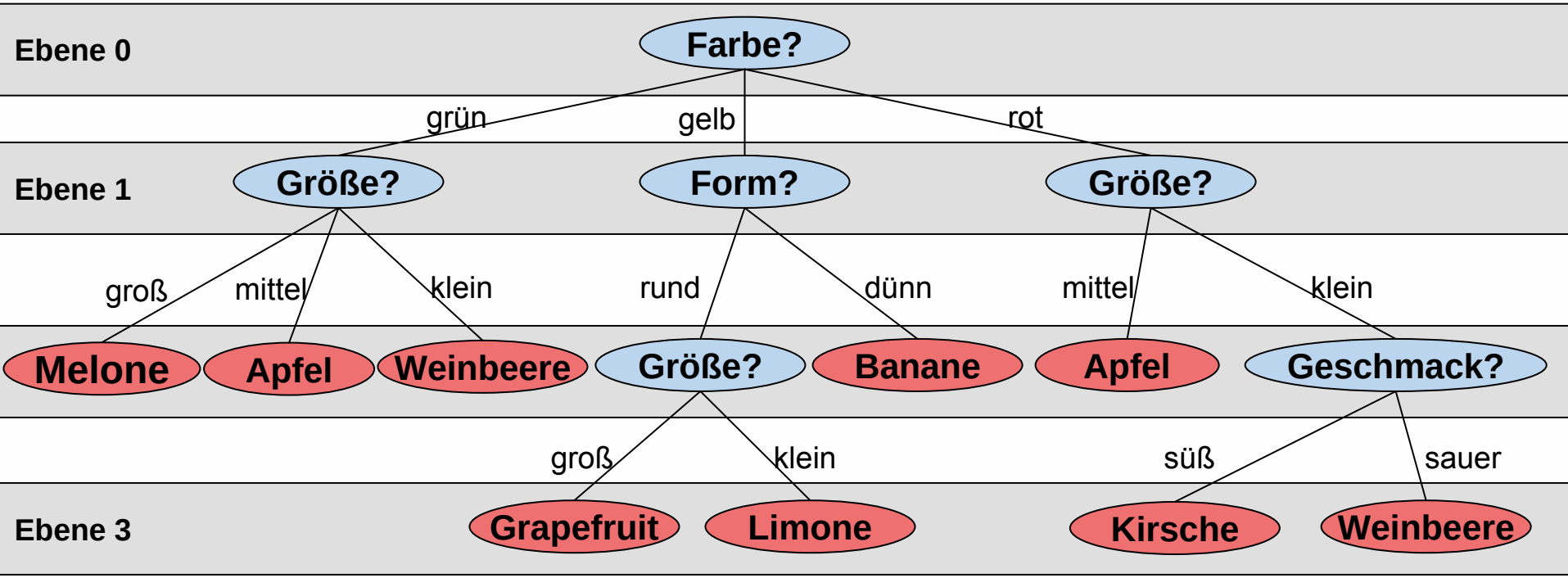


c) f_i ist Polynom r-ter Ordnung



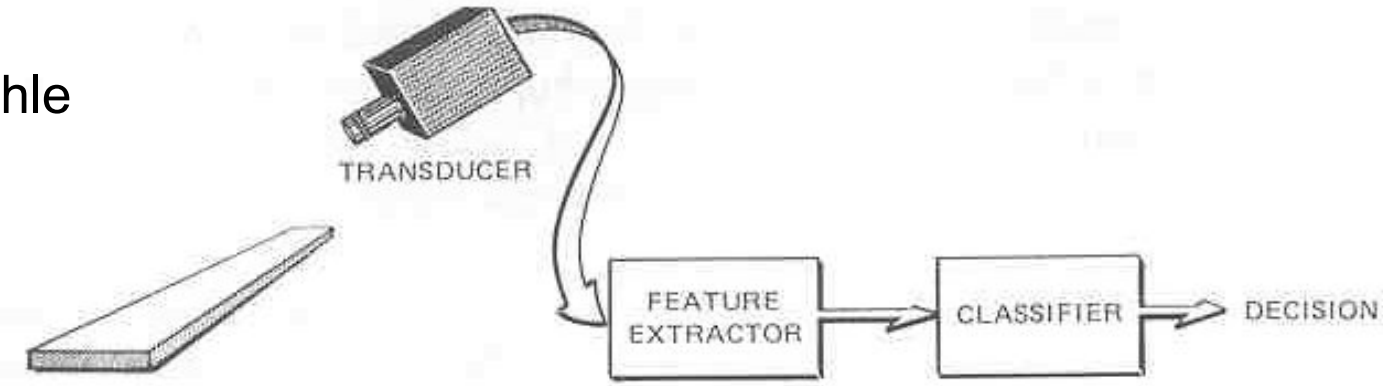
Entscheidungsbaum

- Muster wird mittels einer Reihe von Fragen klassifiziert
- Die nächste zu stellende Frage ist abhängig von der Position im Baum
- Entscheidungen: ja/nein, wahr/falsch oder Werte (Eigenschaften) ...



Klassifikation (Beispiel)

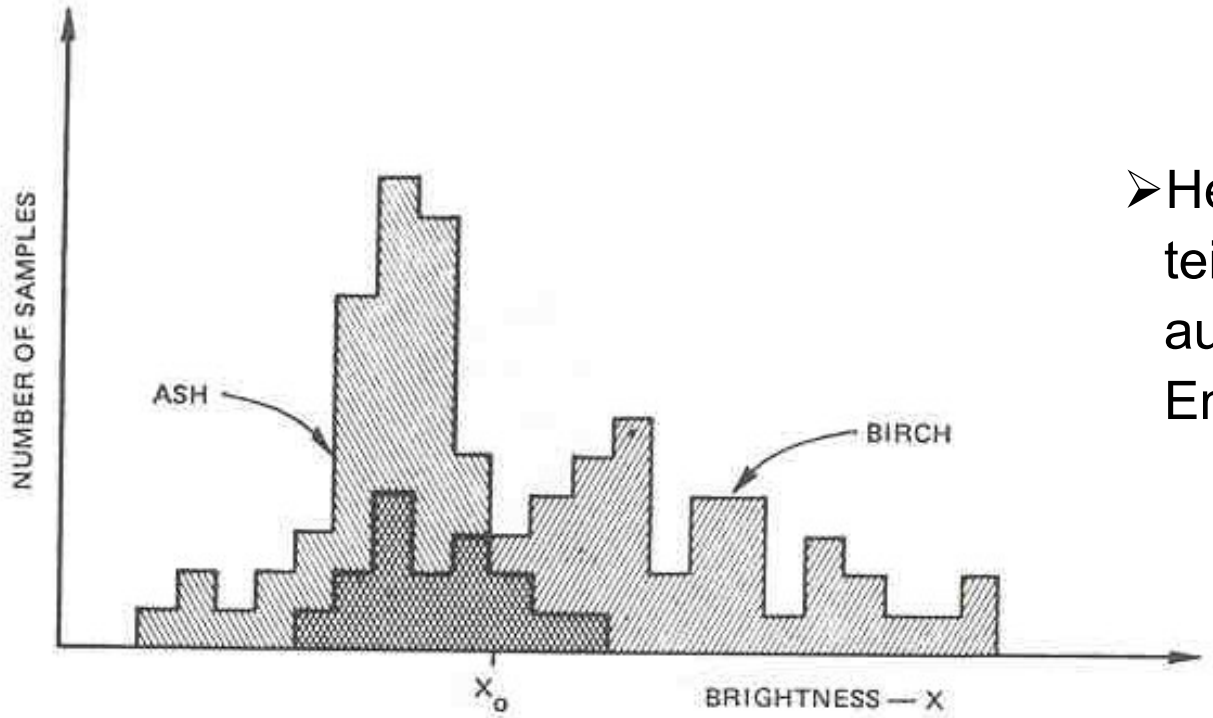
Sägemühle



- 2 Klassen: Esche & Birke
- Annahme: Birke heller als Esche (-> Expertenwissen vom Sägemühlenarbeiter)
- Also (zunächst) ein Merkmal -> Helligkeit

Klassifikation (Beispiel)

Beispiel: Sägemühle



- Helligkeitsverteilung reicht nicht aus für „sichere“ Entscheidung

Klassifikation (Beispiel)

Beispiel: Sägemühle

- Es bedarf eines weiteren Merkmals – Maserung (Esche weist dominante körnige Maserung auf)
- => zwei Merkmale: Helligkeit (x_1) & Maserung (x_2)

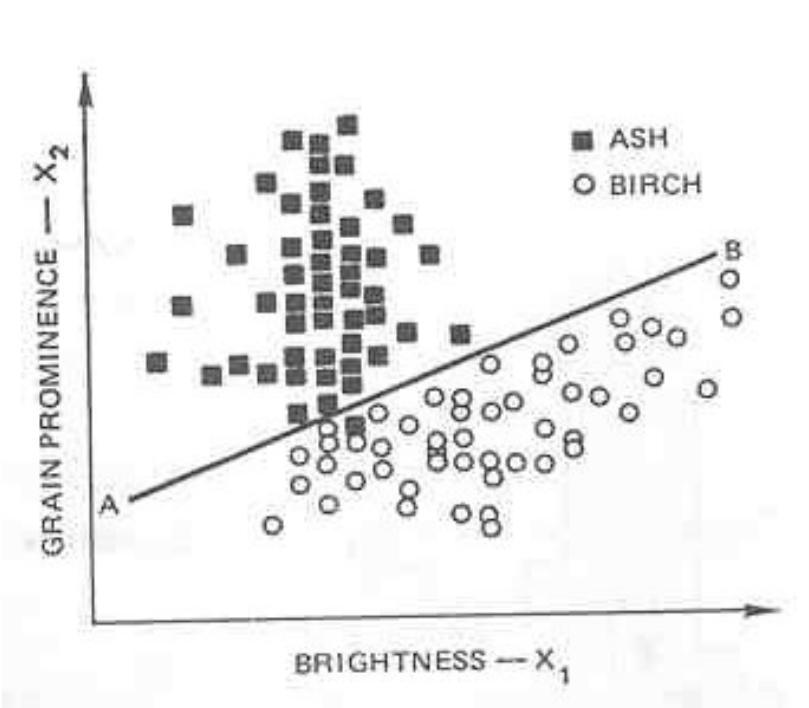
$$\vec{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

- Frage: Wie kann der Merkmalsraum in 2 Regionen/Bereiche unterteilt werden?

Klassifikation (Beispiel)

Beispiel: Sägemühle

- Es sei folgender Stichproben- & Merkmalsraum angenommen



- Mögliche Lösung
  - Alle Vektoren oberhalb von $\overline{AB}$ werden der Klasse „Esche“ zugeordnet und alle unterhalb AB der Klasse „Birke“
- Aber weiterhin Fehlklassifikationen möglich!

Klassifikation (Beispiel)

- Annahmen (allg.) für Bayes Kalkül:
  - Es existiert eine wahrscheinlichkeitstheoretische Formulierung des (jeweiligen) Problems
  - Alle (!) a priori Wahrscheinlichkeitswerte aller Parameter sind vollständig bekannt
- Für Beispiel Sägemühle („2 Klassenproblem“)
  - Klassenvektor = Zustandsvektor $\vec{\omega} = (\omega_1, \omega_2)^T$
  - A priori Wahrscheinlichkeiten eines Zustandes $P(\omega_j)$ bekannt, mit $P(\omega_1)$, $P(\omega_2) \geq 0$ und $P(\omega_1) + P(\omega_2) = 1$ (d.h. Wahrscheinlichkeit dafür, dass sich das System im Zustand ω_j befindet)

Klassifikation/Bayes

- Bayes Kalkül:
  - Merkmalsvektor $\vec{x}$ wird als kontinuierliche Zufallsvariable angenommen
  - Klassenbedingte Wahrscheinlichkeitsdichtefunktion $p(\vec{x} \mid \omega_j)$
- Spezielle Annahmen (Sägemühle)
  - A-priori-Wkt. $P(\omega_j)$ und Wahrscheinlichkeitsdichtefunktion $p(\vec{x} \mid \omega_j)$ seien bekannt,
  - $\vec{x}$ wurde gemessen
  - A posteriori Wahrscheinlichkeit nach Bayes'sche Regel:

$$P(\omega_j \mid \vec{x}) = \frac{p(\vec{x} \mid \omega_j) \cdot P(\omega_j)}{p(\vec{x})}, \text{ mit } \quad p(\vec{x}) = \sum_{j=1}^{2} p(\vec{x} \mid \omega_j) \cdot P(\omega_j)$$

Klassifikation/KNN

- (künstliche) Neuronale Netze
  - Auch Konnektionismus
  - Informationsverarbeitende Systeme
  - Bestehen aus meist großer Anzahl einfacher Einheiten („Neuronen“, Zellen)
  - Einfache Einheiten senden Information durch Aktivierung über gerichtete Verbindungen („connections“, „links“) an andere einfache Einheiten
  - (teilweise) Ähnlichkeit mit erfolgreichen biologischen Systemen
  - Massiv parallele Verarbeitung und Lernfähigkeit

Klassifikation/KNN

- Je nach Gesichtspunkten, die den Einsatz von kNNs bestimmen, sind kNNs interessant für z.B.:
  - Bio- und Neurobiologen, Neurophysiologen, Medizinern
    - Ähnlichkeit des Modells mit der Realität
    - durch Simulation der Modelle evtl. Rückschlüsse auf ungeklärte Eigenschaften/Prozesse des biologischen Systems
  - Psychologen
    - Modellieren psychologischer Phänomene des menschlichen Gehirns
  - Informatiker (auch Mathematiker)
    - Eigenschaft der parallelen Verarbeitung $\rightarrow$ parallele Algorithmen
    - Eigenschaft der Lernfähigkeit sowie deren Effizienz
    - Robustheit rekurrenter Netze, Speicherkapazität, theoretisches Verhalten von Lernalgorithmen

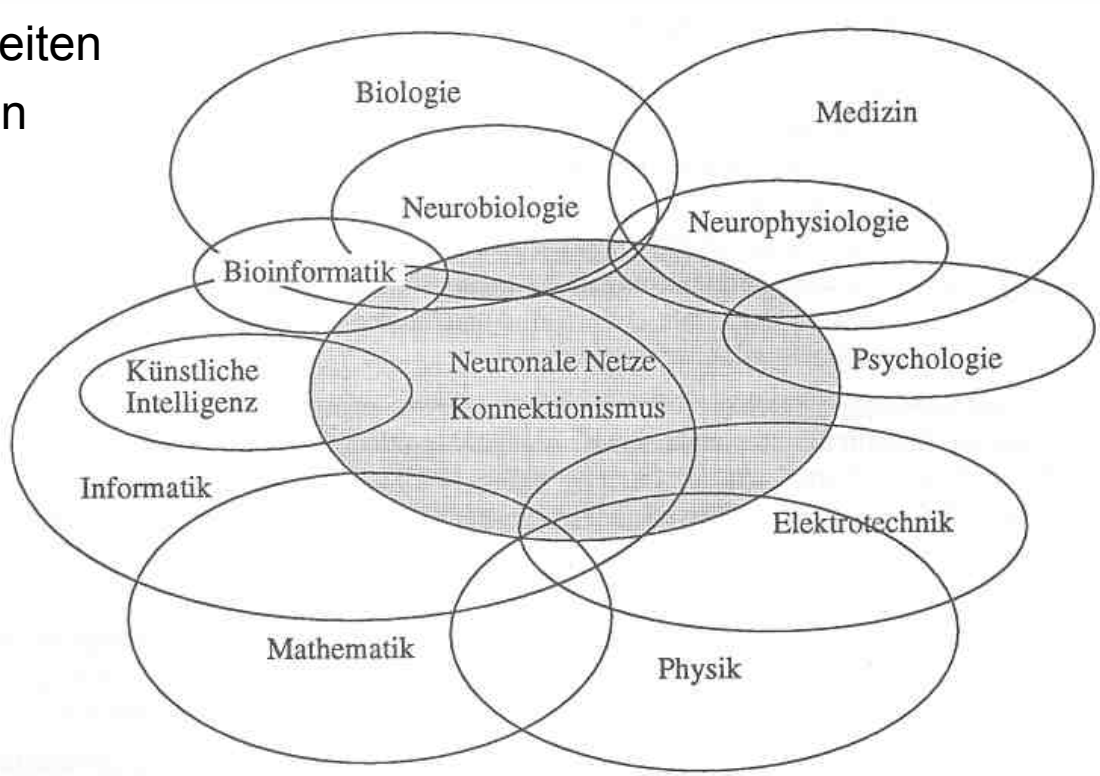
Klassifikation/KNN

- Je nach Gesichtspunkten, die den Einsatz von kNNs bestimmen, sind kNNs interessant für z.B.:
  - Elektrotechniker
    - Bauen/Stellen zusammen mit Informatikern spezialisierte Hardware her/zur Verfügung

Symbolischer vs.(?) Subsymbolischer Ansatz der KI zur Erstellung „intelligenter“ Systeme

Klassifikation/KNN

- Wissenschaftliche Disziplinen, die die kNNs mit ihren Arbeiten berühren bzw. beeinflussen



[Zell 1994], S. 24

Klassifikation/KNN

- Leistungsvergleich Gehirn vs. Rechner (Zahlen nach [Zell 1994])

	Gehirn	Rechner
Anzahl der Verarbeitungselemente	ca. 10^{11} Neuronen	ca. 10^{9} Transistoren
Art der Verarbeitung	massiv parallel	i.a. seriell
Speicherung	assoziativ	adressbezogen
Schaltzeit eines Elementes	ca. 1ms (10^{-3}s)	ca. 1ns (10^{-9}s)
Schaltvorgänge pro s	ca. 10^{3}	ca. 10^{9}
Schaltvorgänge insg. (theoretisch)	ca. 10^{13} /s	ca. 10^{18} /s
Schaltvorgänge insg. (praktisch)	ca. 10^{12} /s	ca. 10^{10} /s

Klassifikation/KNN

- 100-Schritt-Regel

  (Als Argument für die Notwendigkeit massiver Parallelverarbeitung)

  Mensch erkennt eine/n (ihm bekannte/n) Person/Gegenstand in ca. 0,1 s (nach Messungen von Psychologen). Bei einer Schaltzeit von ca. 1ms entspricht das maximal 100 Verarbeitungsschritten.

  Keine Aussage über die Anzahl der Verarbeitungselemente!

  Was kann wohl ein von-Neumann-Rechner in 100 Schritten (Assemblerbefehlen) tun?

Klassifikation/KNN

Vor- und Nachteile von kNNs

+ Lernfähigkeit:

kNNs werden (meist) nicht programmiert, sondern trainiert

+ Parallelität:

vom Ansatz her schon parallel, daher auch Eignung zur Implementierung auf Parallelrechner

+ verteilte Wissensrepräsentation:

Wissen ist in den Gewichten gespeichert → parallele Verarbeitung & Fehlertoleranz

+ Fehlertoleranz:

durch verteilte Repräsentation kann kNN bei Ausfall einzelner Neuronen eine höhere Fehlertoleranz besitzen als herkömmliche Systeme (Wichtig hierbei: Planung des Netzes – Topologie, Werte)

Klassifikation/KNN

Vor- und Nachteile von kNNs

+ assoziative Speicherung:
  es können auch ähnliche Muster abgerufen werden

+ Robustheit gegen Störungen und Rauschen:
  bei richtigem Training reagieren kNNs (meist) weniger empfindlich

+ Default-Wert & spontane Generalisierung:
  kNNs bilden oft automatisch Prototypen von Eingabemustern → auch wenn Eingabe unvollständig kann kNN „etwas“ erkennen...

+ aktive Repräsentation:
  Wissen ist aktiv (durch die Gewichte) repräsentiert, d.h. kein Programm greift von außen auf Wissen zu und Gewichte sind an der Verarbeitung beteiligt

Klassifikation/KNN

Vor- und Nachteile von kNNs

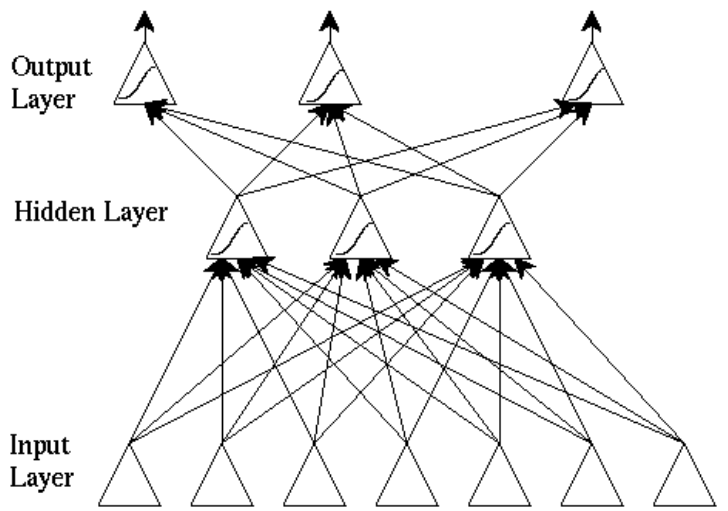
- Wissenserwerb nur durch Lernen:
  es kann kein Basiswissen mitgegeben werden

- keine Introspektion:
  keine Analyse des „eigenen“ Wissens im Gegensatz zu Erklärungskomponenten von XPS

- logisches (serielles) Schließen fast unmöglich:
  kaum Inferenzketten möglich

- Lernen ist relativ langsam:
  alle (populären) Lernverfahren lernen sehr langsam

Klassifikation/KNN



http://www.cs.stir.ac.uk/~lss/NNIntro/InvSlides.html

- Historischer Abriß
  - Die Anfänge
    (Anfang der 40er bis Mitte 50er)
  - Das erste Hoch
    (Mitte der 50er bis Ende der 60er)
  - Die ruhige Phase
    (Ende der 60er bis Anfang/Mitte der 80er)
  - Wiederauflebung
    (Anfang/Mitte der 80er bis heute)

Klassifikation/KNN

- Die Anfänge (Anfang der 40er bis Mitte 50er)
  - McCulloch-Pitts-Neuron (1943)
    zeigten, dass auch einfache Klassen kNNs prinzipiell jede arithmetische oder logische Funktion berechnen können
  - McCulloch-Pitts-Neuron (1947)
    Erkennung räumlicher Muster, insbesondere wurde das Problem der Lageinvarianz betrachtet
  - Hebb‘sche Lernregel (1949)
    einfaches universelles Lernkonzept individueller Neuronen (in der allgemeinen Form bis heute noch Basis fast aller neuronalen Lernverfahren)
  - Lashley (1950)
    durch Versuche an Ratten wurde die These über eine verteilte Wissens-repräsentation im Gehirn aufgestellt (Umfang und nicht Ort der Läsion bestimmte Leistung der Ratten beim Durchlaufen des Labyrinths)

Klassifikation/KNN

- Das erste Hoch (Mitte der 50er bis Ende der 60er)
  - Rosenblatt, Wigthman und Kollegen (1957-58)
    erste erfolgreiche Entwicklung eines Neurocomputers – Mark I Perceptron – am MIT für ME-Aufgaben (20x20 Pixel Bildsensor für einfache Ziffernerkennung); funktionierte mit 512 motorgetriebenen Potis – je ein Poti für ein variables Gewicht
  - Minsky‘s Dissertation (1954)
    bereits 1951 Snark entwickelt für seine Dissertation, allerdings „nur“ auf Papier
  - Rosenblatt (1959)
    Konvergenz-Theorem für Perzeptrons: „... Alles, was ein Perzeptron repräsentieren kann, kann es auch – durch spezielles Lernverfahren von R. - lernen

Klassifikation/KNN

- Das erste Hoch (Mitte der 50er bis Ende der 60er)
  - Selfridge‘s Pandämonium (1958)
    dynamische, interaktive Mechanismen zur Morse-Code-Übersetzung (← mit Modellen menschlicher Informationsverarbeitung – Pandämonium – und Ingenieurtechniken – hill-climbing Verfahren)
  - Steinbuch (1961)
    Lernmatrix, einfache und technische Realisierung von Assoziativspeicher (waren Vorläufer heitiger neuronaler Assoziativspeicher); Realisierung für bedingte Reflexe → Pawlow‘sche Reflexe
  - Widrow & Hoff (1960)
    Adaline - adaptives System für schnelles und genaues Lernen (ähnlich dem Perzeptron: binäres Schwellwert-Neuron); später gründete Widrow Memistor Corporation → Memistoren – transistorenähnliche Bauelemente mit einstell-baren Gewichten (zur Realisierung von kNNs)

Klassifikation/KNN

- Die ruhige Phase (Ende der 60er bis Anfang/Mitte der 80er)
  - Minsky & Papert (1969)
    Perceptrons, genaue mathematische Analyse von Perzeptrons; zeigten, dass z.B. XOR-Problem so nicht lösbar, da es nicht repräsentiert werden kann; Aussage: kNN - „research dead-end“ führte dazu, dass fast keine Forschungsgelder mehr flossen (für ca. die nächsten 15 Jahre!!!) Auch DARPA förderte nicht mehr...
  - Kohonen (1972)
    (unabhängig auch Anderson) Correlation Matrix Memories – Modell eines linearen Assoziators (spezieller Assoziativspeicher); lineare Aktivierungsfunktion und kontinuierliche Gewichte
  - Von der Malsburg (1973)
    Self-organization of orientation sensitiv cells in the striata cortex – komplexes nichtlineares Neuronenmodell zur Computersimulation von RFs

Klassifikation/KNN

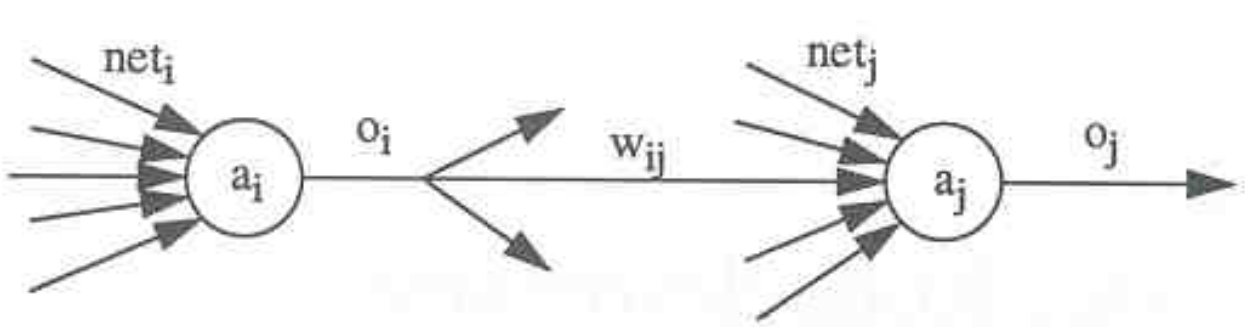
- Die ruhige Phase (Ende der 60er bis Anfang/Mitte der 80er)
  - Grossberg (1976-80)
    einer der ersetn, die sigmoide Aktivierungsfunktionen und nichtlineare laterale Hemmungen verwendeten → ART-Theory (adaptive resonance theory) ART-n, ARTMAP,...
  - McClelland & Rumelhart (1981)
    strukturierte parallele Netze; viele Randbedingen in einer Art constraint satisfaction network bestimmen Erkennung von Teilmustern zu Buchstaben und dann zu ganzen Wörtern
  - Hopfield (1984)
    zunächst binäre (später auch kontinuierliche) Netze als Äquivalent zu Ising-Modellen in der Physik (Untersuchung von Netzen mit Energiefunktionen)
  - Kohonen (1982-89)
    SOM

Klassifikation/KNN

- Wiederauflebung (Anfang/Mitte der 80er bis heute)
  - „Renaissannce“ der kNNs u.a. durch Arbeiten von Hopfield (Hopfield 85: Neural Computation of Decisions in Optimization Problems)
  - ca. 1986 „Entdeckung“ des Backpropagation Lernverfahrens (Rumelhart, Hinton & Williams)
  - Dt. Professoren
    Ritter (Bielfeld), von der Malsburg (Bochum), Haken (Stuttgart), Rojas (Berlin?), von Seelen (Dortmund), Palm (Ulm),...
  - SOMs in der Robotik-Anwendung

Klassifikation/KNN

- Bestandteile (in Analogie zu biologischen Neuronen)
  - Zellkörper: Knoten a_i
  - Dendriten: summieren die Eingabe(-werte) net_i
  - Axon: leitet die Ausgabe eines Neurons nach Außen weiter o_i
  - Synapsen: wenn Axon sich verzweigt dann statt $o_i \rightarrow w_{ij}$



[Zell 1994], S. 72

Klassifikation/KNN

- Bestandteile kNNs
  - Zellen (Neuronen, Elemente, units):
    - Aktivierungszustand (activation) $a_i(t)$ gibt Grad der Aktivierung der Zelle an
    - Aktivierungsfunktion f_{act} gibt an, wie neuer Aktivierungszustand $a_j(t+1)$ des Neurons j aus alter Aktivierung $a_i(t)$ und Netzeingabe $net_j(t)$ berechnet wird

$$a_j(t+1) = f_{act}\left(a_j(t), net_j(t), \theta_j\right)$$

$$\text{mit } \theta_j \text{ als Schwellenwert des Neurons } j$$

    - Ausgabefunktion f_{out} wird aus der Aktivierung des Neuronsbestimmt

$$o_j = f_{out}(a_j)$$

Klassifikation/KNN

- Bestandteile kNNs
  - Verbindungsnetzwerk:
    - kNN als gerichteter, gewichteter Graph
    - Kanten als gewichtete Verbindung
    - Gewichte werden (meist) mit w_{ij} bezeichnet
    - Matrix aller Verbindung W
  - Propagierungsfunktion
    - Gibt an, wie sich die Netzeingabe eines Neurons aus den Netzausgaben der anderen Neurone und den entsprechenden Verbindungsgewichtet berechnet

$$net_j(t) = \sum_i o_i(t) w_{ij}$$

Klassifikation/KNN

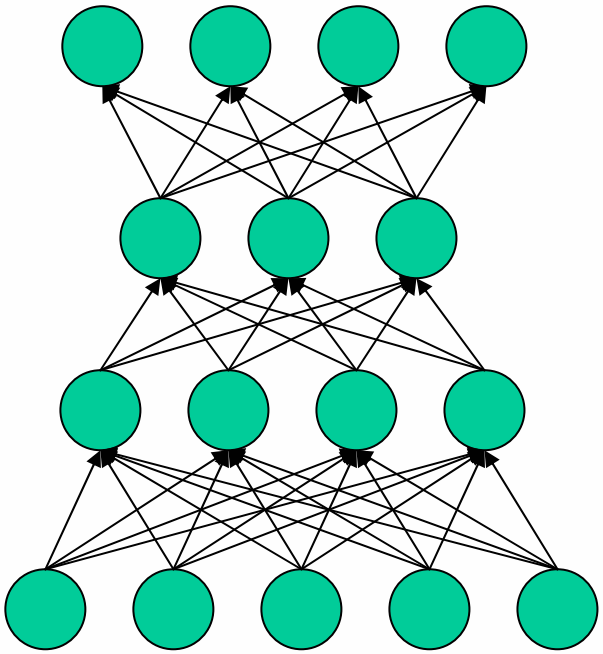
- Bestandteile kNNs
  - Lernregel:
    - ... ist ein Algorithmus nach dem das Netz lernt. Lernen ist (meist) Modifikation der Stärke der Verbindungen (Gewichte)

      Lernen durch wiederholtes anlegen der Trainingsmuster

      Versuch der Fehlerminimierung (zwischen erwarteter Ausgabe und tatsächlicher Ausgabe)

Klassifikation/KNN

Zelltypen



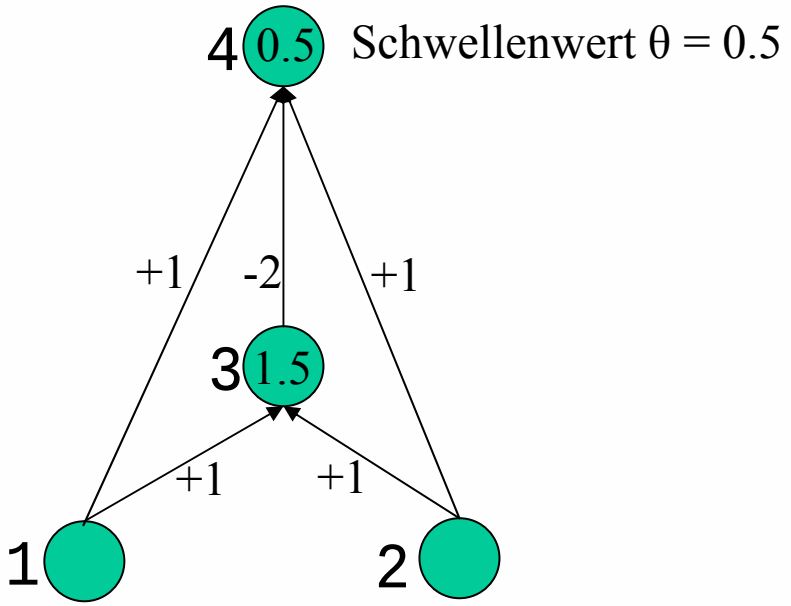
Feedworward Netz

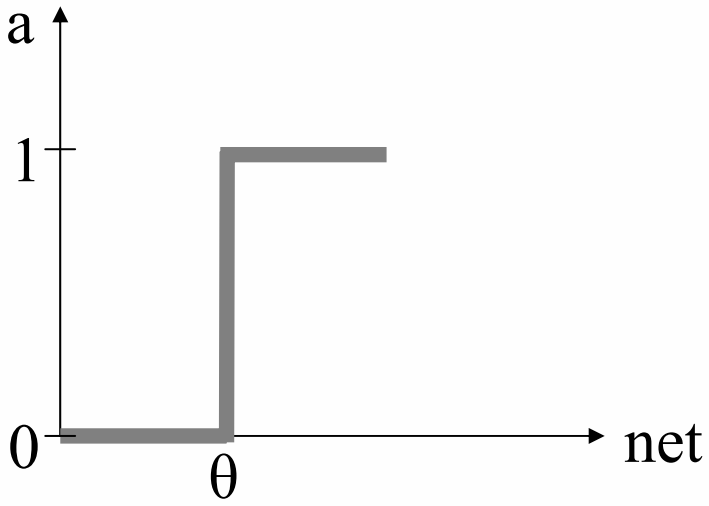
Ausgabeschicht
(output layer)

0 – n verdeckte Schichte(n)
(hidden layer)

Eingabeschicht
(input layer)

Kleines Beispiel „XOR“





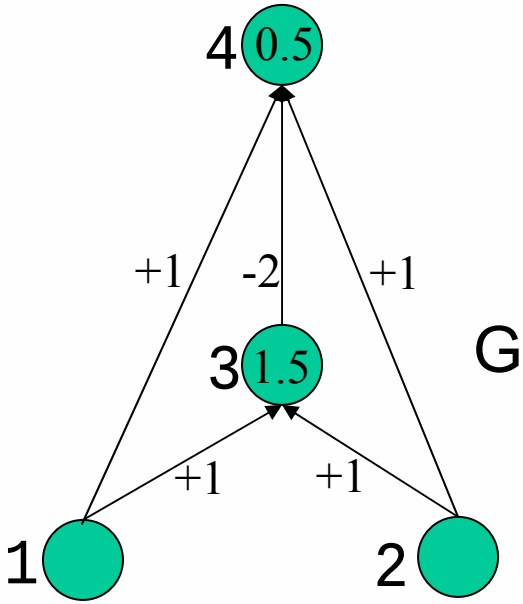
Kleines Beispiel „XOR“

Netzeingabe: $$net_j(t) = \sum_i o_i(t) w_{ij}$$

Schwellenwertfunktion als Aktivierungsfunktion:

$$a_j(t) = \begin{cases} 1 & \text{falls } net_j(t) \geq \theta_j \\ 0 & \text{sonst} \end{cases}$$

Ausgabefunktion ist Identität, d.h. $o_j(t) = a_j(t)$



Kleines Beispiel „XOR“

Gewichtsmatrix: $W = \begin{bmatrix} w_{11} & \cdots & w_{14} \\ \cdots & \cdots & \cdots \\ w_{41} & \cdots & w_{44} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & -2 \\ 0 & 0 & 0 & 0 \end{bmatrix}$

o_1	o_2	net_3	θ_3	o_3	net_4	θ_4	o_4
0	0	$net_3 = 0*1+0*1=0$	1.5	0	$net_4 = 0*1+0*1+0(-2)=0$	0.5	0
0	1	$net_3 = 0*1+1*1=1$	1.5	0	$net_4 = 0*1+1*1+0(-2)=1$	0.5	1
1	0	$net_3 = 1*1+0*1=1$	1.5	0	$net_4 = 1*1+1*1+0(-2)=1$	0.5	1
1	1	$net_3 = 1*1+1*1=2$	1.5	1	$net_4 = 1*1+1*1+1(-2)=0$	0.5	0

Zusammenfassung

- Klassifikation
  - Distanzfunktionen
  - Diskriminantenfunktionen
  - Wahrscheinlichkeiten
  - KNN

Automatisches Clustering

- automatische Bestimmung von Klassen (Clustern)
- vorzugsweise unüberwachtes Lernen
- große Anzahl von (fast) unüberwachten Ansätzen
- ...

Automatisches Clustering

- Partitioning Methoden (z.B. Simple-K-Means)
- Hierarchische Verfahren (z.B. Single Linkage)
- Density-based (z.B. DBSCAN)
- Wahrscheinlickeitsbasierte Verfahren (z.B. EM)
- SOMs
- Fuzzy
- u.a.: Meanshift, inkrementelle Verfahren,...

Automatisches Clustering

- Partitioning:
  - Menge von Objekten n
  - Menge von Cluster k
  - mindestens 1 Objekt im Cluster oder
  - mindestens so viele Objekte wie Cluster ($k \leq n$)
  - Güte der Cluster wird durch eine *Kriteriumsfunktion* bestimmt

Partioning

- Simple-K-Means:
  - mittels Parameter *K* wird Anzahl der Cluster vorgegeben
  - => *K* Clusterzentren $\{cc_1,...,cc_K\}$, per Zufall aus Menge der Objekte $\Omega = \{o_1,...,o_m\}$ ausgewählt
  - Euklidische Distanz zwischen Objekt und Clusterzentrum
  - Objekt wird dem Cluster mit geringster Distanz zugewiesen
  - Anschließend: Mittelwert der Objekt im Cluster
  - Mittelwert ist dann neues Clusterzentrum
  - dann wieder Mittelwertberechnung usw.
  - Abbruch, wenn Kriteriumsfunktion „konvergiert“

Partitioning

- Simple-K-Means (weiter):
  - Kriteriumsfunktion meist Quadratischer Fehler

- Wähle zufällig K Objekte als initiale Clusterzentren
- wiederhole:
  - mit ED werden Objekte den Clusterzentren zugefügt, denen sie am ähnlichsten sind
  - berechne Clusterzentren neu (Mittelwert)
  - Abbruch: wenn keine Veränderung (Quadratischer Fehler)

Partitioning

- Simple-K-Means (weiter):
  - Vorteil:
    - eignet sich für besonders große Datenbestände
    - Komplexität ist O(*ntk*) (nach Han & Kamber 2001)
      - n = Anzahl der Objekt
      - t = Anzahl der Iterationen
      - k = Anzahl der Clusterzentren
  - Nachteile:
    - anfällig gegenüber „Fehlerobjekten“: schon geringen Anzahl beeinflusst Anzahl der Clusterzentren

Hierarchische Verfahren

- Erzeugung einer Clusterhierarchie (bzw. Baum von Clustern)
- agglomerative und divise Ansätze
- agglomerativ:
  - bottom-up Strategie:
  - jedes Objekt ist Cluster
  - verschmelze Cluster solange bis nur noch ein Cluster übrig bleibt oder vorgegebene Anzahl von Cluster erreicht ist
  - Ähnlichkeitsmaß!

Hierarchische Verfahren

- divise:
  - Top-down Strategie:
  - Start: ein großer Cluster
  - zerteile Cluster solange bis jedes Objekt ein Cluster ist oder vorgegebene Anzahl von Cluster erreicht oder Distance-Threshold zwischen zwei Cluster erfüllt ist
- Vorteil:
  - lassen sich auf jeden Merkmalstyp anwenden
  - jedes Ähnlichkeitsmaß
- Nachteil:
  - Terminierungskriterien
  - keine nachträgliche Änderung mehr möglich:
    - agglomerativ: verschmolzene Objekte können nicht mehr getrennt werden
    - divise: getrennte Cluster können nicht mehr vereinigt werden

Hierarchische Verfahren

- Single-Linkage (nach [Anderberg 1973] bzw. [Everitt 1974]):
  - startet mit n Objekten als ein Cluster
  - Cluster mit geringster Distanz zu einander werden verschmolzen
  - Mit Minimumsfunktion (über die verschmolzenen Cluster) werden die Werte (Unähnlichkeitsmatrix) für die restlichen Cluster bestimmt
  - wenn neuer Cluster mit einem der bestehenden Cluster verschmilzt, dann wird für ein beliebiges Objekt (innerhalb des resultierenden Cluster) der Abstand zum nächsten Nachbarn max. den Wert des neu hinzugekommenden Clusters haben

Hierarchische Verfahren

- Single-Linkage (nach [Anderberg 1973] bzw. [Everitt 1974]):
  - Zeile/Spalte (Ubähnlichkeitsmatrix) eines der vorher verschmolzenen Clusters wird gelöscht
  - Anzahl der Cluster wird um Eins verringert
  - Ergebnis: Dendrogramm
  - Vorteil:
    - eignet sich gut für konzentrische Cluster
  - Nachteil:
    - Ansatz „leidet“ unter *Chaining-Effect*

Automatisches Clustering

- Mean-Shift (nach [Cheng 1995]):
  - iterativer Ansatz
  - jeder Datenpunkt wird zum Mittelpunkt innerhalb seiner Nachbarschaft „geshiftet“
  - generalisierte Form:
    - endliche Datenmenge S
    - endliche Menge von Clusterzentren T
    - Kernel K und
    - Gewichtungsfunktion $\omega : S \mapsto (0,\infty)$

Automatisches Clustering

- Mean-Shift (nach [Cheng 1995]):
  - Funktion *m(x)* berechnet Clusterzentren (sample-mean)

$$m(x) = \frac{\sum_{s \in S} K(s-x)\omega(s)s}{\sum_{s \in S} K(s-x)\omega(s)}$$

  - Kernel K kann z.B. Gauss oder Flat sein
  - Entwicklung von T (durch Iteration) wird als Mean-Shift bezeichnet

Automatisches Clustering

- Mean-Shift (nach [Cheng 1995]):
  - T (im blurring-process) gleich S gewählt: T = S
  - nach jeder Iteration werden die neu berechneten Werte für s in vorheriger Gleichung verwendet
  - für die Gewichtung der Daten können fix Gewichte angenommen werden (oder bei jeder Iteration neu berechnet werden)
  - Verfahren Terminiert, sobald ein fixer Punkt erreicht wird
  - Vorteil:
    - prinzipiell paralleles verfahren (alle t aus T werden gleichzeitig berechnet)
  - Nachteil
    - $O(n)^2$, daher für große n ungeeignet

Zusammenfassung

- Klassifikation
  - Distanzfunktionen
  - Diskriminantenfunktionen
  - Wahrscheinlichkeiten
  - KNN
- (fast) automatisches Clustering
  - Partitioning Methoden (z.B. Simple-K-Means)
  - Hierarchische Verfahren (z.B. Single Linkage)
  - Fuzzy
  - u.a.: Meanshift, inkrementelle Verfahren,...