(hoffentlich kurze) Einführung:

Neuronale Netze

Dipl.-Inform. Martin Lösch

martin.loesch@kit.edu

(0721) – 608 45944

Überblick

- Einführung
- Perzeptron
- Multi-layer Feedforward Neural Network
- MLNN in der Anwendung

EINFÜHRUNG

Vorbild Gehirn

- Gehirn des Menschen
  - Neuron Schaltzeit: > 0.001 sec
  - Anzahl Neuronen: 1010
  - Verbindungen (Synapsen) pro Neuron: 104-105
  - Szenenerkennung: 0.1 sec

- Auffallende Eigenschaften
  - hochparallele Berechnung
  - verteilte Repräsentation von Wissen

Vergleich: Gehirn ←→ serieller Rechner

Eigenschaft	Parallelität	Präzision	Fehler-toleranz	Speicher-zugriff	Erkennen v. Mustern u. Ähnlichkeiten
Gehirn	hoch	mäßig	hoch	global	gut
ser. Rechner	noch mäßig	hoch	niedrig	lokal	mäßig

Eigenschaft	Numerische präzise Be-rechnungen	Fehlerloses Speichern v. Daten	Rekonstrukt. teilw. zerst. Daten	Verallgem. v. Bsp. auf implizite Regeln	Selbst-organisation
Gehirn	schlecht	schlecht	gut	gut	ja
ser. Rechner	gut	gut	schlecht	schlecht	bisher nicht

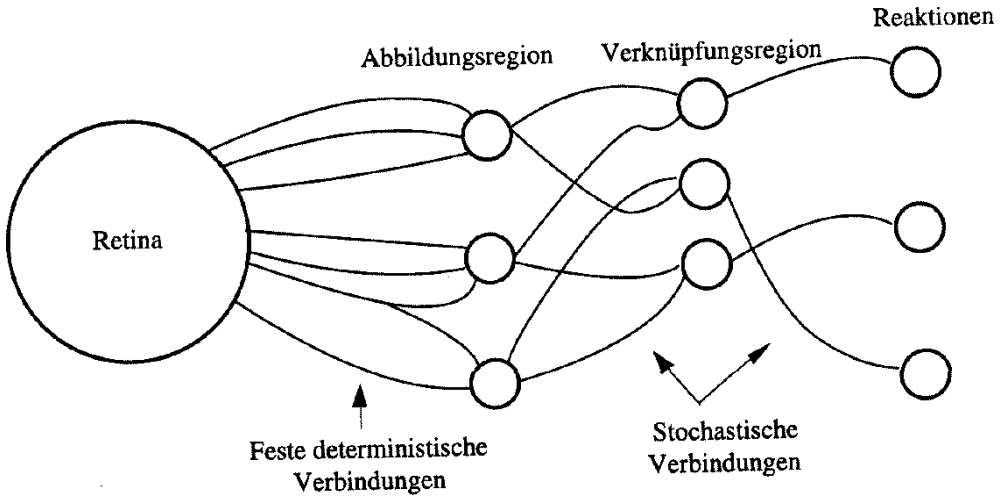
Was ist „konnektionistisches Rechnen“?

- Rechnerarchitekturen, Rechenmodelle und Lernmodelle, die in Anlehnung an natürliche Neuronenmodelle entwickelt werden.
- kennzeichnende Eigenschaften solcher Systeme
  - Große Anzahl einfacher Recheneinheiten (Künstliche Neuronen)
  - Durch gewichtete Kanäle verbunden (Netz)
  - Kein Rechnen mit symbolisch kodierten Nachrichten
  - Wissen wird in der Struktur der Verbindungen repräsentiert
  - Massiver Parallelismus

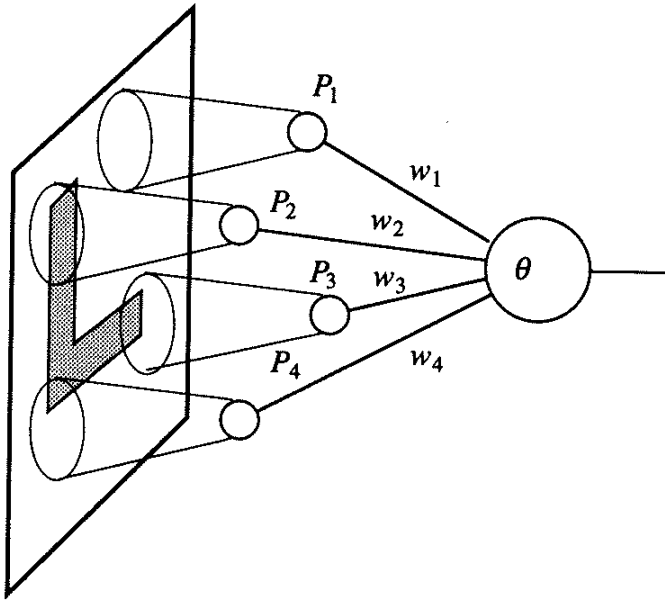
PERZEPTRON

Perzeptron: Idee [Rosenblatt 1960]

Grundidee:
Anlehnung an das Funktionsprinzip der natürlichen Wahrnehmung/Reaktion im Tierreich



Biologie



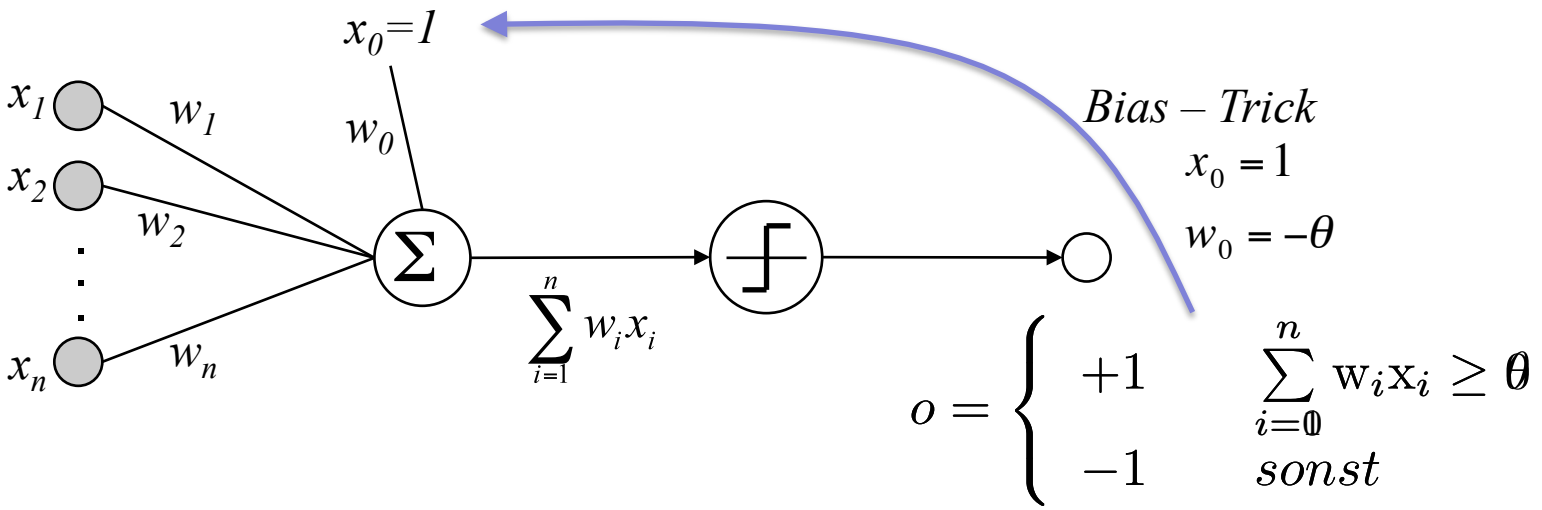
Computer

Perzeptron: Aufbau [Rosenblatt 1960]

- Aufbau eines Perzeptrons

x – Eingabevektor

t – Target (Soll-Ausgabe)

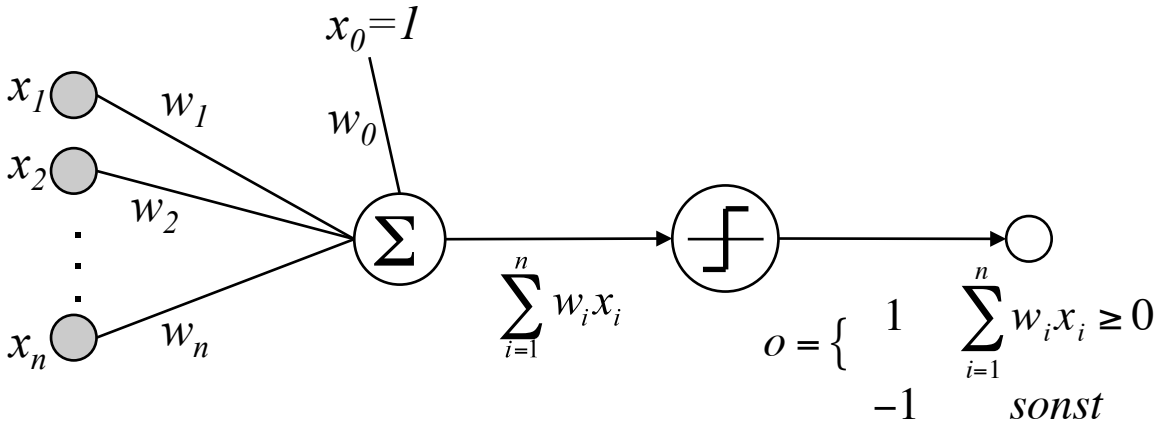


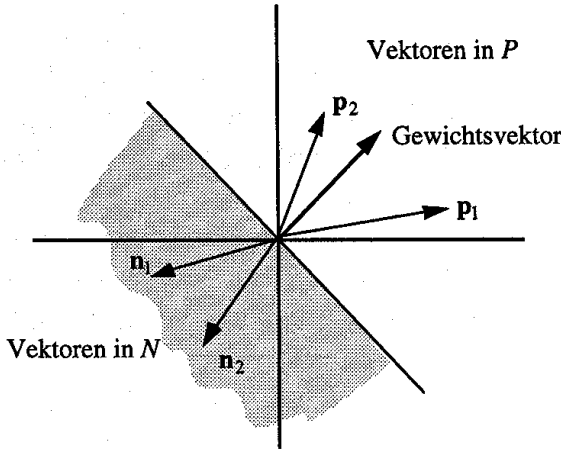
w – Gewichtsvektor

o – Output (Ist-Ausgabe)

Perzeptron: Geometrische Interpretation

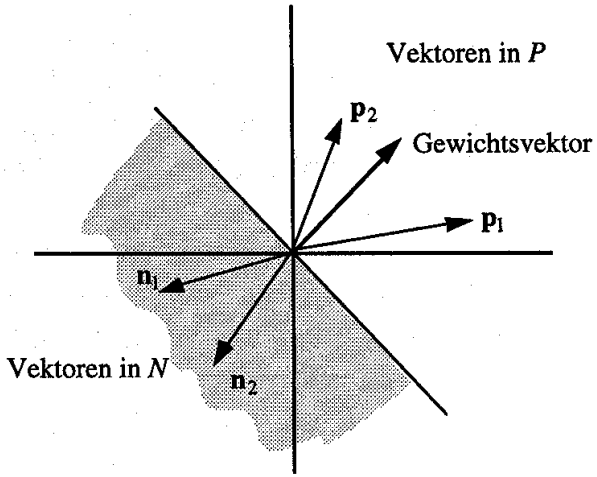
- „Positive und Negative“ Daten (P,N)
- Erweiterung der Dimension durch x_0
- Trennhyperebene (in R^2: Gerade), definiert durch Gewichte (Normalen der Ebene)
- Gewichtete Summe = Skalarprodukt

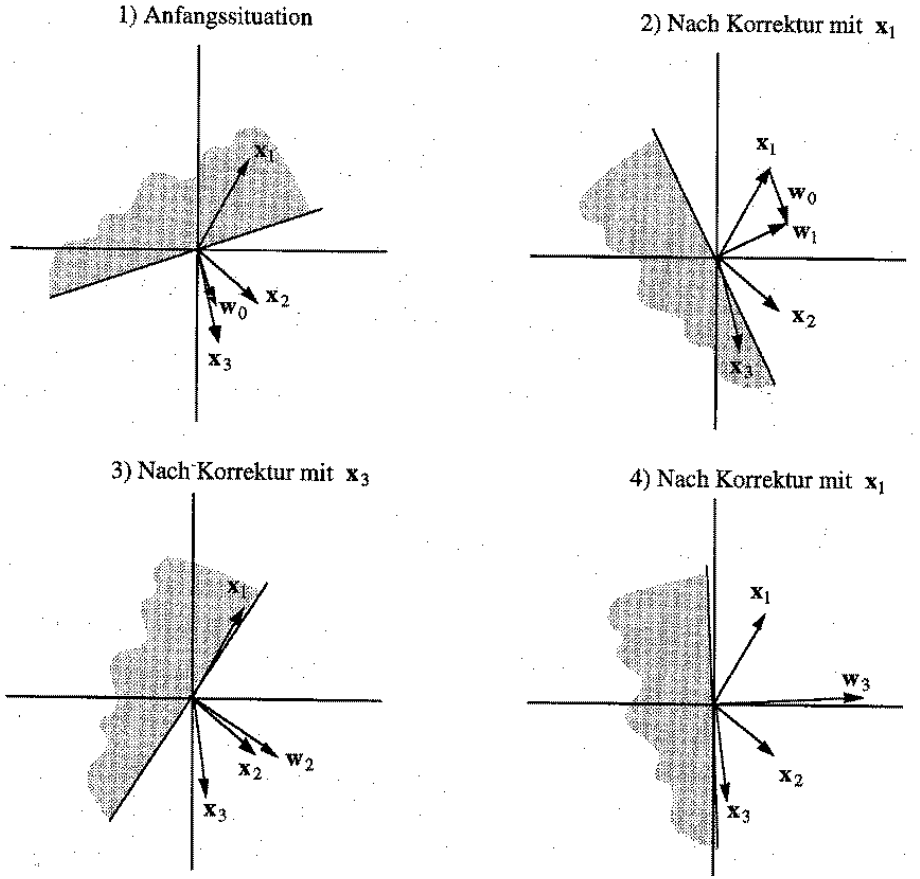




Lernen = Anpassen der Gewichte ➔ Gesucht wird die beste Trennebene

Lernen - Geometrische Interpretation





Hilfsmenge

$$N' = \{x' \mid x' = -x, \forall x \in N\}$$

Neues Lernproblem

$$xw > 0 \quad , \forall x \in N' \cup P$$

➔ Im Beispiel: alle x_i aus P

Perzeptron – Lernalgorithmus

Start: Gegeben Lerndatenmenge $P \cup N$

Der Gewichtsvektor $w(0)$ wird zufällig generiert.

Setze $t:=0$.

Testen: Ein Punkt x in $P \cup N$ wird zufällig gewählt.

Falls $x \in P$ und $w(t) \cdot x > 0$ gehe zu *Testen*

Falls $x \in P$ und $w(t) \cdot x \leq 0$ gehe zu *Addieren*

Falls $x \in N$ und $w(t) \cdot x < 0$ gehe zu *Testen*

Falls $x \in N$ und $w(t) \cdot x \geq 0$ gehe zu *Subtrahieren*

Addieren: Setze $w(t+1) = w(t)+x$.

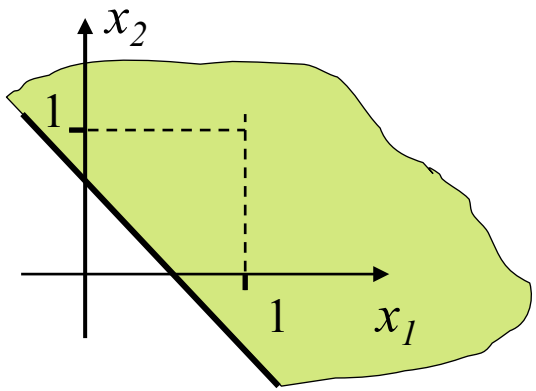
Setze $t:= t+1$. Gehe zu *Testen.*

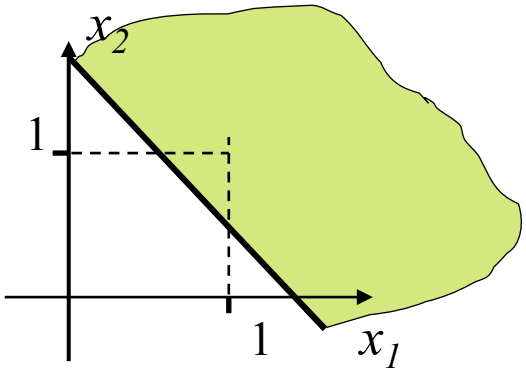
Subtrahieren: Setze $w(t+1) = w(t)-x$.

Setze $t:=t+1$. Gehe zu *Testen*.

Perzeptron: Kapazität

- Bsp. Logik:



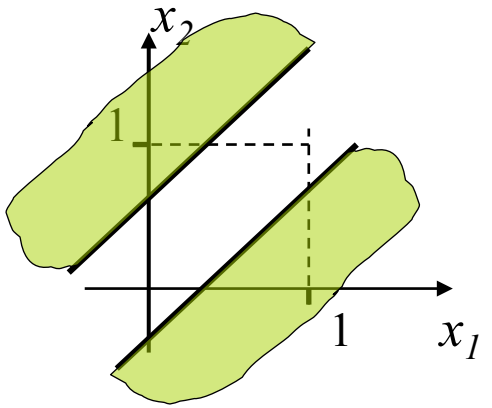


x_1 OR x_2: $0.5x_1 + 0.5\,x_2 > 0.3$ x_1 AND x_2 : $0.5x_1 + 0.5\,x_2 > 0.8$

➔ Durch Kombination von Perzeptronen sind viele Funktionen möglich

XOR: ???

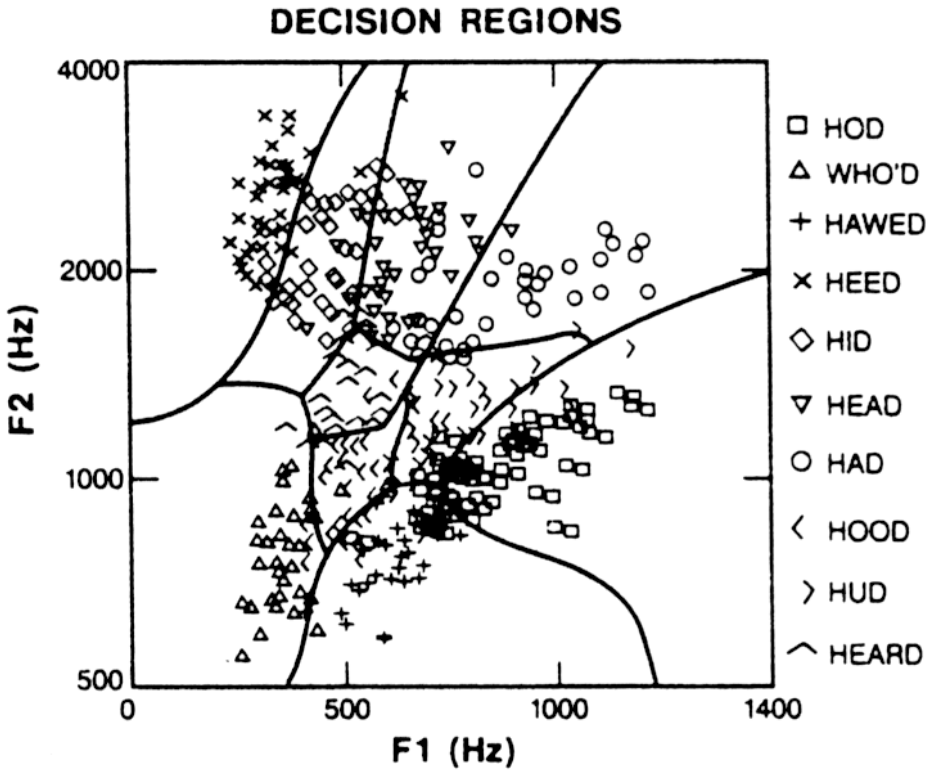
NICHT MÖGLICH!

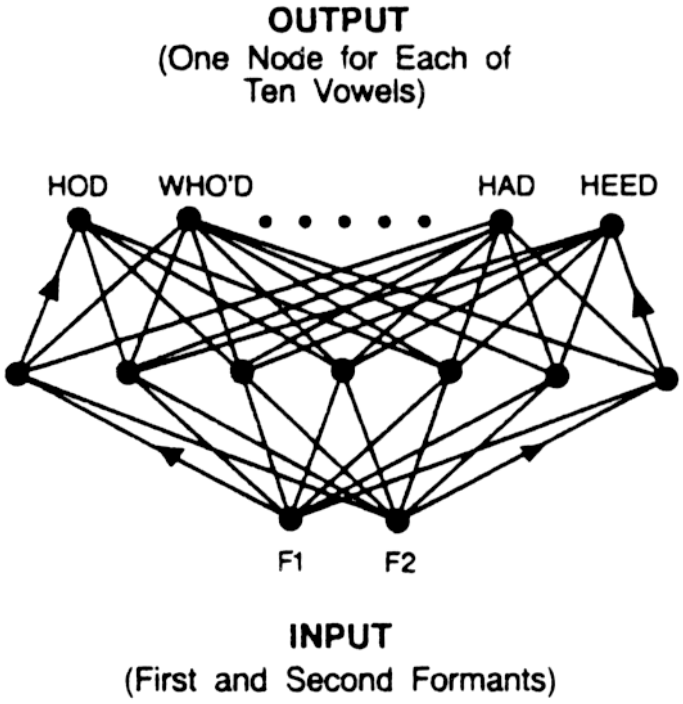


MULTI LAYER FEEDFORWARD NEURAL NETWORK

Nichtlineare Entscheidungsregionen

- Wie kann man nichtlineare Entscheidungsregionen mit KNN lernen?
- Beispiel: Erkennung von Lauten anhand von 2 Formanten (Teiltönen)

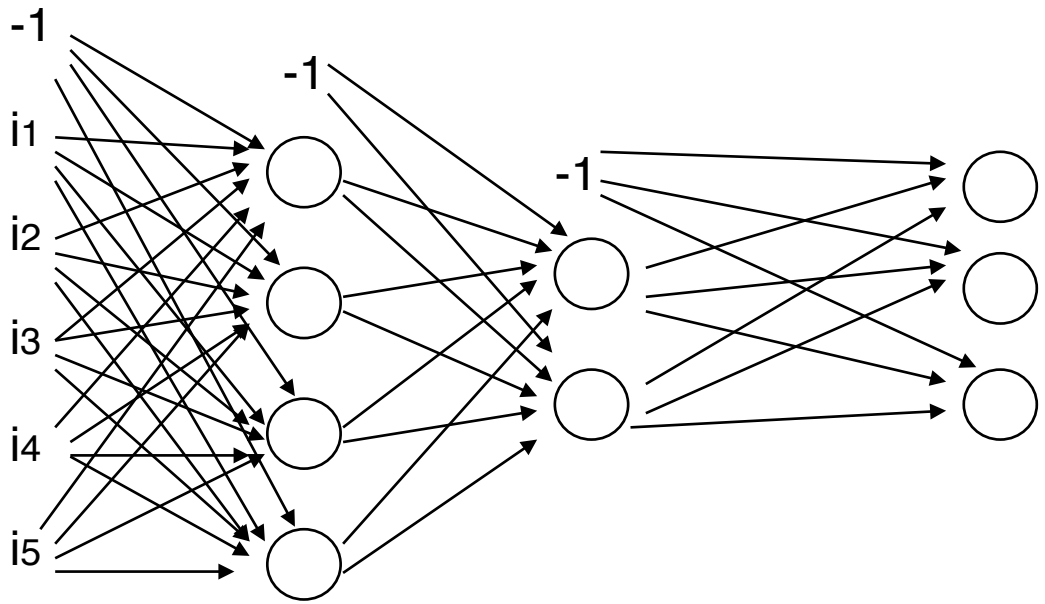




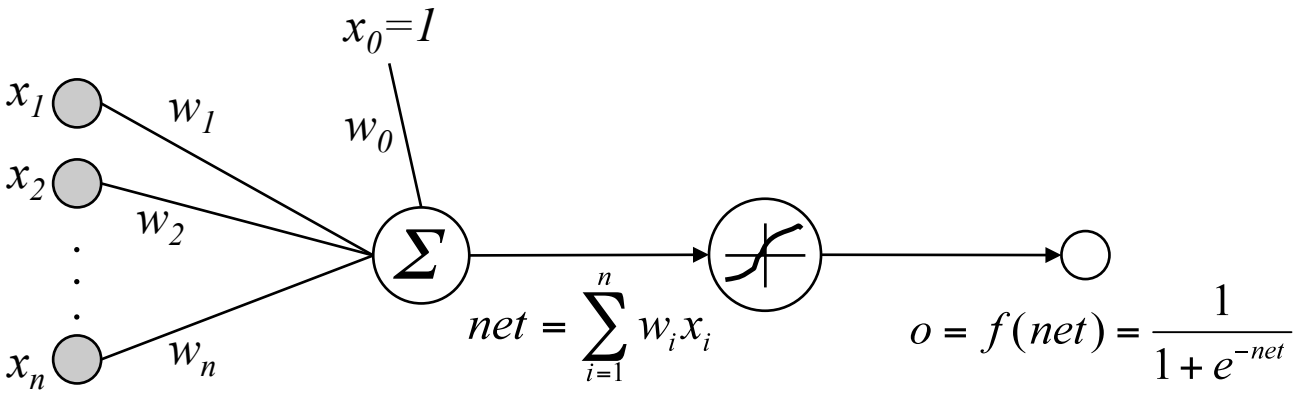
[Lippmann90]

Multi Layer Neural Network (MLNN)

- Netzaufbau: mehrere versteckte (innere) Schichten
- Lernverfahren: Backpropagation-Algorithmus [Rumelhart86, Werbos74]
- Neuronenaufbau: nichtlineare Aktivierungsfunktion



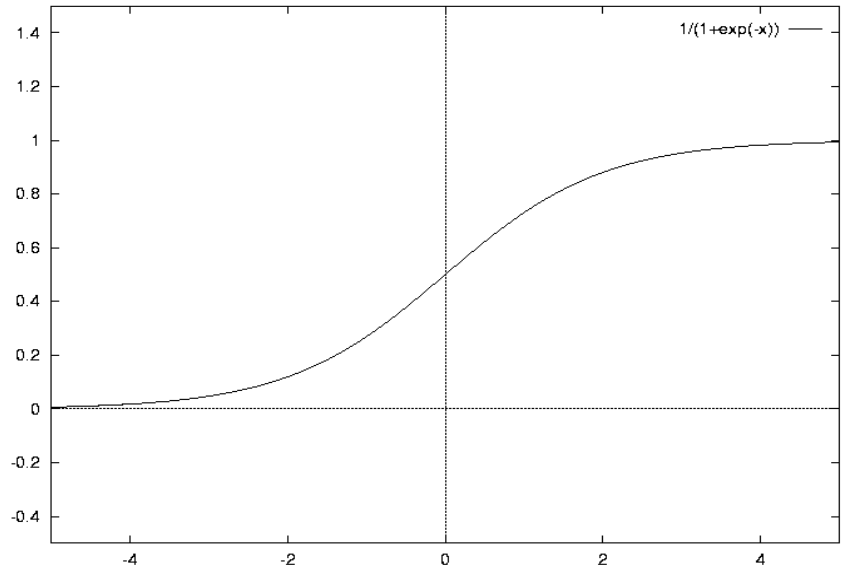
Aufbau der Neuronen

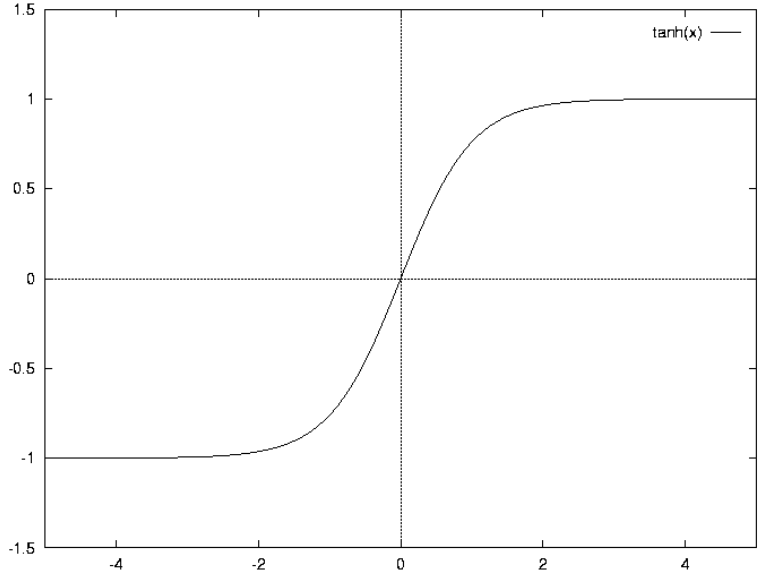


- x_{ij} = i-te Eingabe des Neurons j
- w_{ij} = das Gewicht zwischen Neuron i und Neuron j
- net_j = $\sum_i w_{ij} x_{ij}$ Propagierungsfunktion
- o_j = Ausgabe des Neurons j
- t_j = Zielausgabe (target) des Ausgabeneurons j
- $f(x)$ = Aktivierungsfunktion

- $output$ = Menge der Ausgabeneuronen
- $Downstream\ (j)$ = direkte Nachfolger des Neurons j

Nichtlineare Aktivierungsfunktionen





Sigmoid: $f(x) = \frac{1}{1 + e^{-x}}$

$$\frac{\partial f}{\partial x} = f(x)\,(1 - f(x))$$

$f(x) = \tanh(x)$

$$\frac{\partial f}{\partial x} = (1 + f(x))\,(1 - f(x))$$

Backpropagation Algorithmus I

- Vorgaben
  - Menge T von Trainingsbeispielen (Eingabevektor/ Ausgabevektor)
  - Lernrate η
  - Netztopologie
    - Anzahl und Ausmaße der Zwischenschichten
    - Schichten sind vollständig vorwärts gerichtet verbunden
- Lernziel
  - Finden einer Gewichtsbelegung $\mathbf{W}$, die T korrekt wiedergibt

Backpropagation Algorithmus II

- Initialisieren der Gewichte mit kleinen zufälligen Werten
- Wiederhole…
  - Auswahl eines Beispielmusters d
  - Bestimmen der Netzausgabe
  - Bestimmen des Ausgabefehlers (bzgl. Sollausgabe)
  - Sukzessives Rückpropagieren des Fehlers auf die einzelnen Neuronen

$$\delta_j = \begin{cases} o_j(1-o_j) \sum_{k \in Downstream(j)} \delta_k w_{jk}, & j \notin output \\ o_j(1-o_j)(t_j - o_j), & j \in output \end{cases}$$

  - Anpassen der Gewichtsbelegung um $\Delta w_{ij} = \eta \delta_j x_{ij}$
- … solange ein gewähltes Abbruchkriterium nicht erfüllt ist!

MLNN IN DER ANWENDUNG

Entwurf von Neuronalen Netzen

- Subsymbolische Repräsentation der Ein- und Ausgabe
- Auswahl der Topologie
- Auswahl des Lernverfahrens
- Parametereinstellung
- Implementierung / Realisierung
- Training & Verifikation (Test)

Topologieauswahl

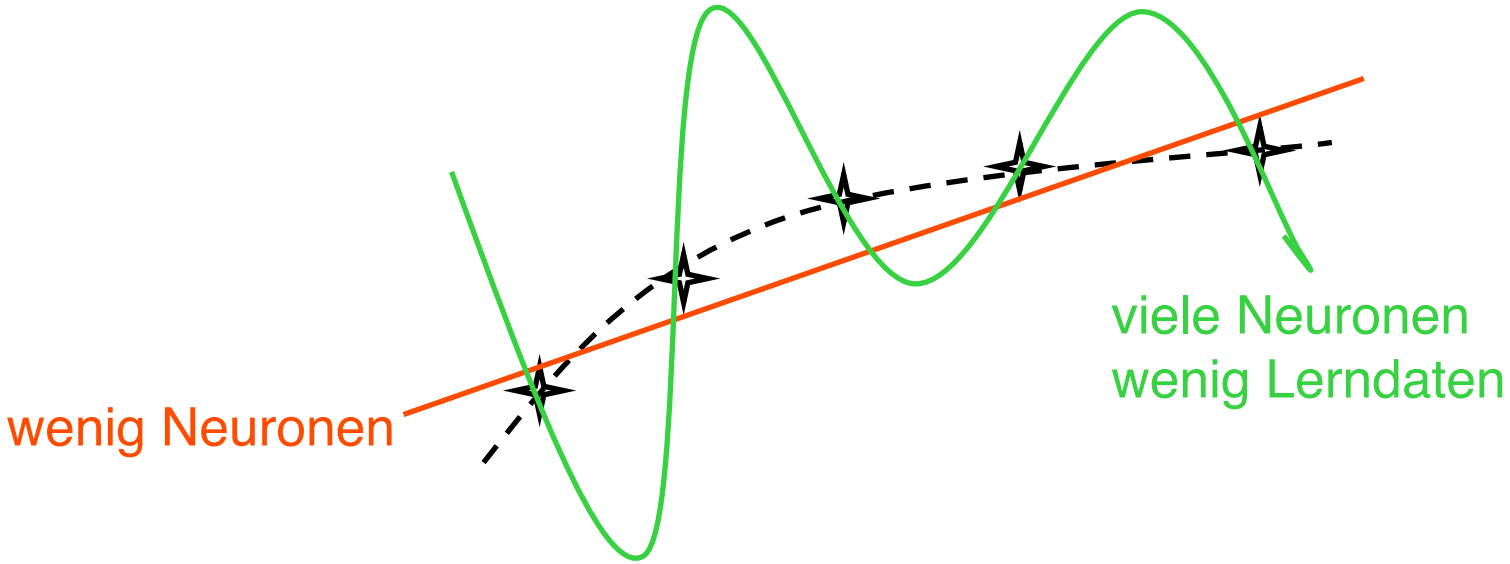
- Zusammenhang zwischen Anzahl der (hidden) layer und Zielfunktion?
  - 3 Layer (1 hidden Layer - sigmoid):
    - jede Boolsche Funktion
    - jede kontinuierliche beschränkte Funktion
      [Cybenko 1989, Hornik et al. 1989]
  - 4 Layer (2 hidden Layer -sigmoid)
    - beliebige Funktionen mit beliebiger Genauigkeit
      [Cybenko 1988]

- ➔ Schon eine geringe Tiefe ist ausreichend

Lernverhalten - Topologieauswahl

- Anzahl der Neuronen pro Schicht im Bezug zu der Anzahl von (stochastisch unabhängigen) Lerndaten ist wichtig
- Aber: allgemeine Aussage nicht möglich

- Beispiel: gestrichelte Kurve soll eingelernt werden



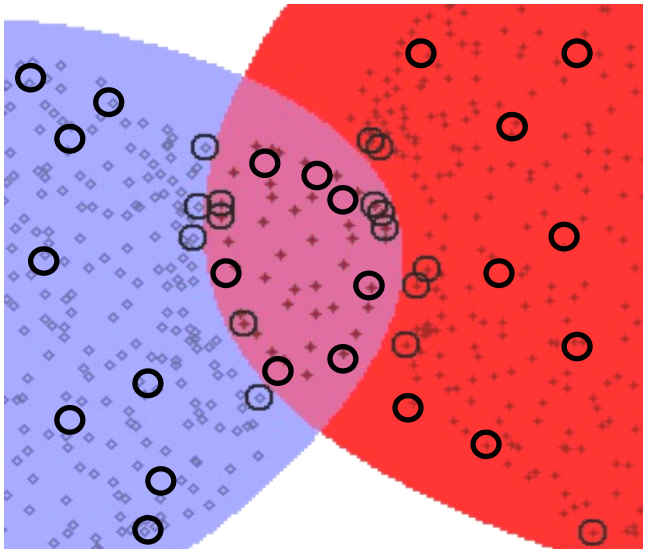
Initialisierung der Gewichte

- Gewichte verschieden wählen
  - sonst funktionsgleiche Neuronen

- zufällig, gleichverteilt und klein
  - $\Rightarrow$ keine anfängliche Ausrichtung

Auswahl repräsentativer Trainingsbeispiele

- Lerndaten
  - für die Anpassung der Gewichte
- Verifikationsdaten
  - für das Testen der Generalisierung

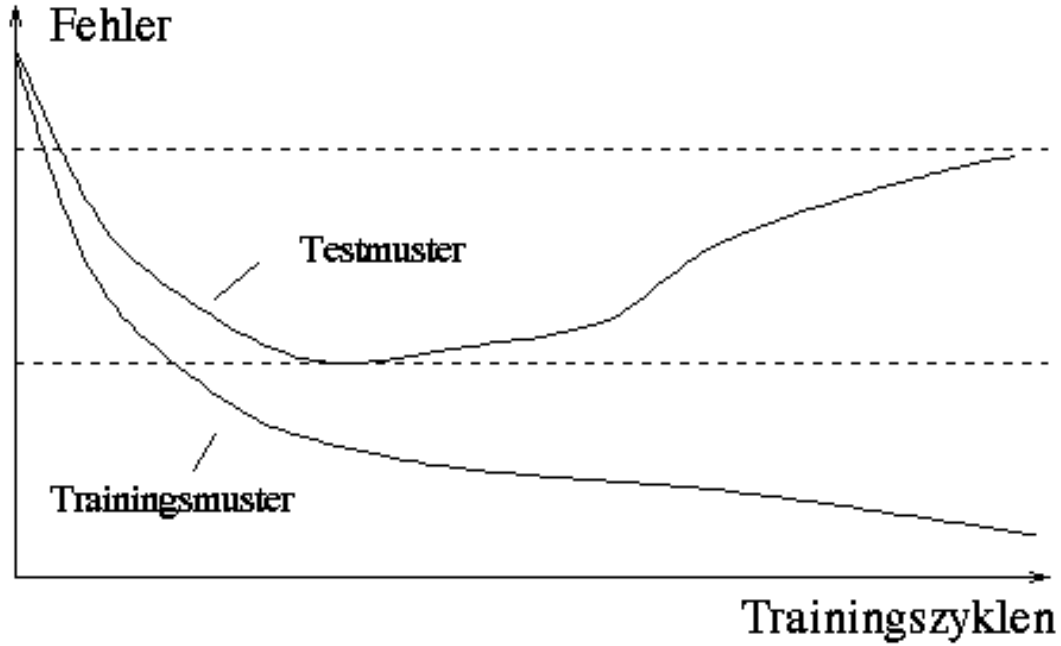


- gute Verteilung der Beispiele
  - Klassifikation: Daten aus allen Klassen
  - Regression: gesamter Definitionsbereich

- Beispiele insbesondere aus komplexen Regionen
  - Klassifikation: Randregionen zwischen Klassen
  - Regression: Verlaufsänderungen

Overfitting

- Fehler auf Verifikationsdaten steigt ab einer Anzahl von Lernzyklen



- Mögliches Abbruchkriterium für Lernvorgang

Entwurfs- und Optimierungskriterien

- Wiedererkennungs-Fehlerrate
- Trainingszeit
- Wiedererkennungszeit
- Speicherbedarf
- Komplexität der Trainingsalgorithmen
- Leichte Implementierbarkeit
- Gute Anpassungsfähigkeit

- Trade-off zwischen Anforderungen nötig

Literatur

- *Tom Mitchell:* ***Machine Learning****.* McGraw-Hill, New York, 1997.
- *M. Berthold, D.J. Hand:* ***Intelligent Data Analysis****.*
- *P. Rojas:* ***Theorie der Neuronalen Netze – Eine systematische Einführung****.* Springer Verlag, 1993.
- *C. Bishop:* ***Neural Networks for Pattern Recognition****.* Oxford University Press, 1995.
- *Vorlesung „****Neuronale Netze 2006****“*: http://isl.ira.uka.de/
- siehe auch *Skriptum „****Ein kleiner Überblick über Neuronale Netze****“*: http://www.dkriesel.com/