Neuronale Netzwerke

Gliederung

- Biologische Motivation
- Künstliche neuronale Netzwerke
  - Das Perzeptron
    - Aufbau
    - Lernen und Verallgemeinern
    - Anwendung
    - Testergebnis
- Anwendungsbeispiele
- Zusammenfassung

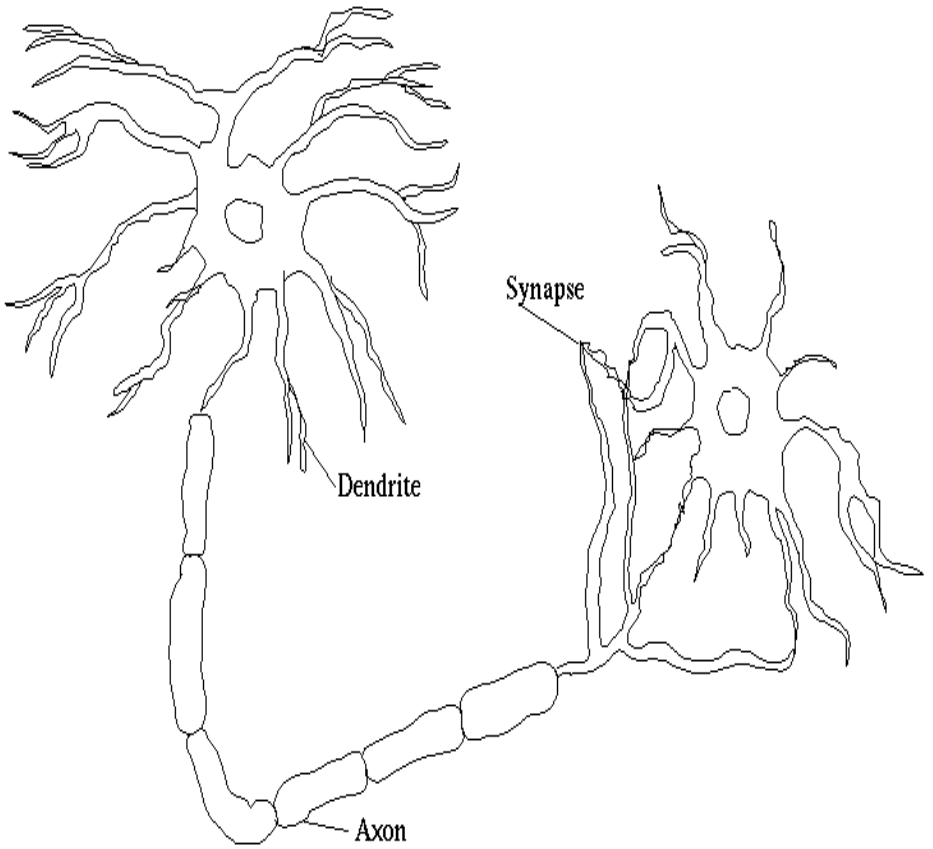
Biologische Motivation

„Neuronale Netze bilden die Struktur und Informationsarchitektur von Gehirn und Nervensystem“

Was ist ein Neuron?

3 Bestandteile:

- neuronale Zellkörper (alle ankommenden Signale werden aussummiert, SW, Aktionspotential)
- Dendriten (verzweigte Erweiterungen, empfangen ankommende Signale)
- Axon (transportiert die Ausgangssignale des Neurons zu Dendriten anderer Neuronen)



Synapsen

- Kontaktstellen
- Elektrische Synapsen
- Chemische Synapsen: können hemmend oder verstärkend wirken

Können Computer unser Gehirn simulieren?

Künstliche neuronale Netzwerke

- Künstliche neuronale Netzwerke sind Computerprogramme, die die Architektur und die Funktion des Gehirns als Vorbild haben.
- Lernen anhand von Beispielen

  (Anpassung der Synapsenstärke)
- Verallgemeinern durchs Lernen

  (konnte aus Beispielen Regeln erkennen)
- Einfachste neuronale Netz: Perzeptron

Das Perzeptron

- Frank Rosenblatt 1958 auf Grundlage von Neuronenmodell von McCulloch und Pitts
- Lernt Tastatureingabe 0 oder 1 vorherzusagen
- Selbst bei vollkommen zufälliger Eingabe von 0 und 1, Perzeptron wird im Mittel besser als 50% vorhersagen

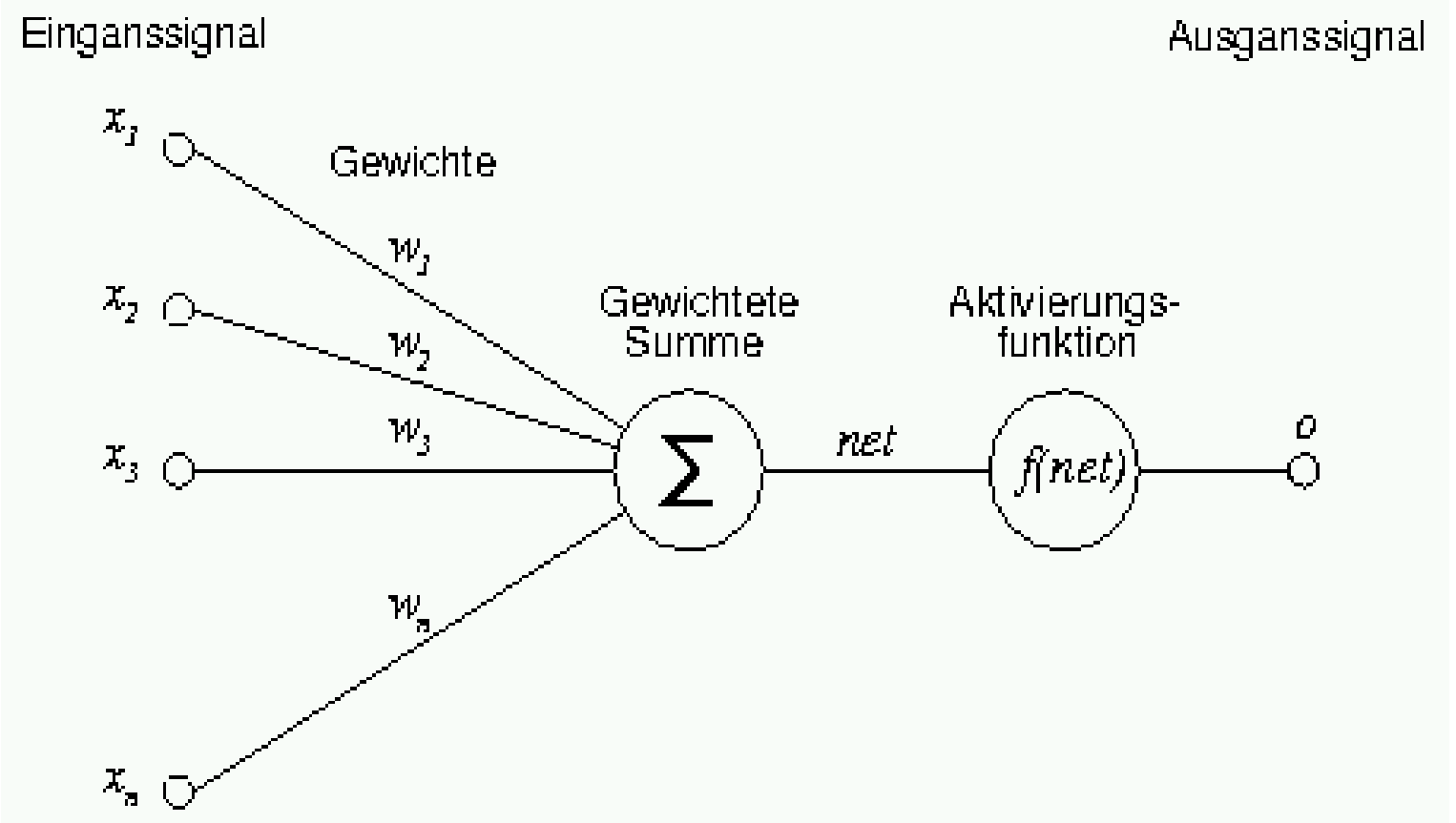
  (nicht viel, dennoch beängstigend, weil Rechner in der Lage menschliches Verhalten hervorzusagen.)

Aufbau des Perzeptrons I

- Eingabeschicht von „Neuronen” S_i
- N “synaptische” Gewichte w_i i=(1,...,N)
- Ausgabeneuron S_0

  (ist mit allen Eingabeneuronen direkt über Synapsen verkoppelt)

Aufbau des Perzeptrons II



Die “Basisfunktion“

- S_0 reagiert auf die Summe der Aktivitäten der Neuronen die über synaptische Verbindungen direkt auf S_0 einwirken.

$$S_0 = sign(\sum_{j=1}^{N} S_j w_j)$$

Die “Basisfunktion“ (biologische Motivation)

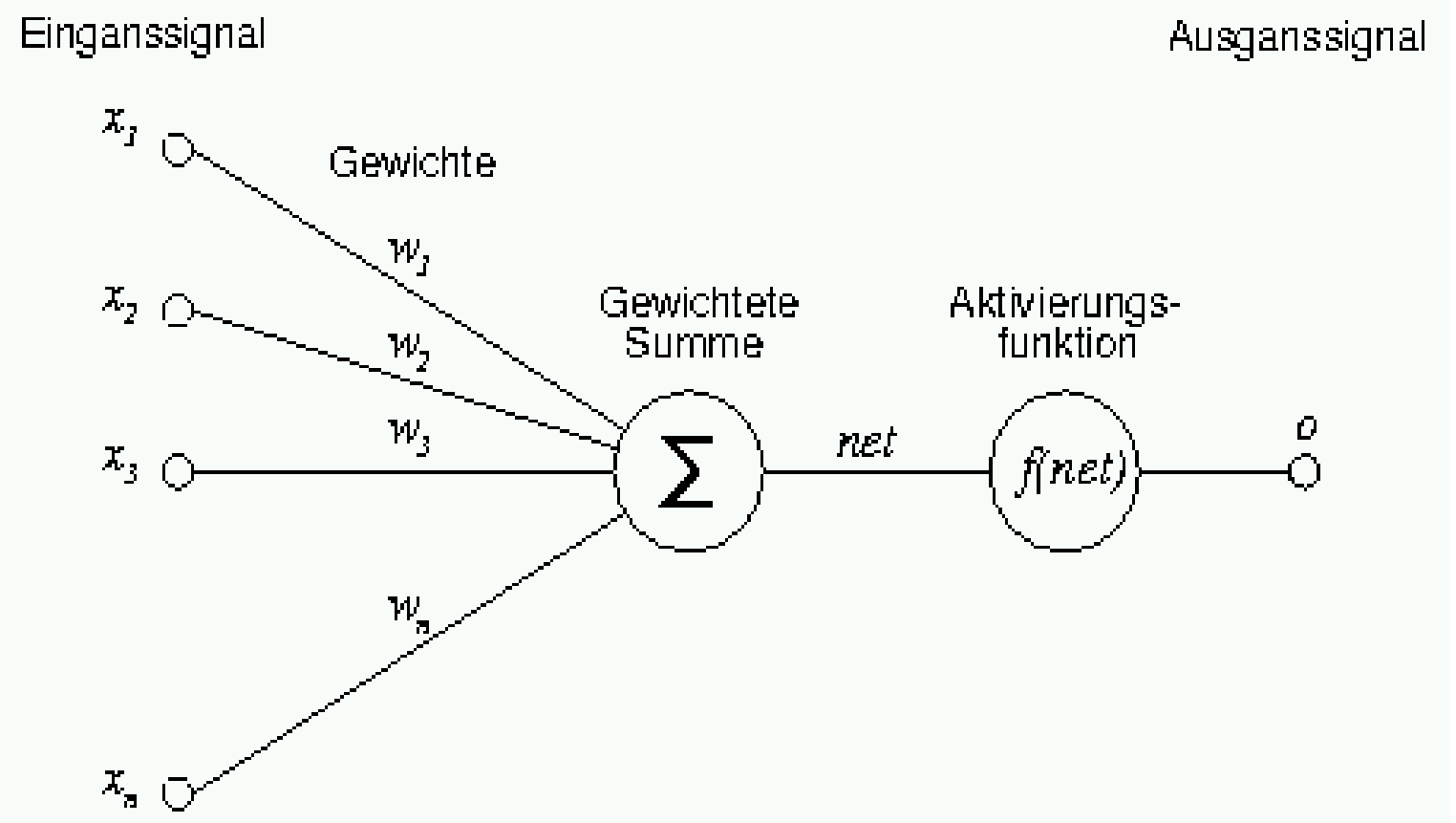
- Neuron S_i: ruht = -1, sendet Impuls=1
- $W_i \in R$ gewichtet Stärke, mit der Signal des Neurons S_i in ein elektrisches Potential im Kern von S_0 umgewandelt wird (Synapse): hemmend(<0), erregend(>0)
- Wenn Summe der Potentiale den Schwellenwert 0 überschreiten, dann „feuert“ es (S_0=+1), sonst ruht (S_0=-1)

Die “Basisfunktion“ (mathematisch)

- Boolsche Funktion, klassifiziert jede Eingabe **S** nach +1 oder -1
- Geometrisch: **S** und **w** sind Vektoren im N-Dimensionalen Raum.
  - Lineare Klassifikationsgrenze der Ausgaben von S0 bei S0=0 => (N-1)-Dimensionale Hyperebene **w**·**S**=0 trennt Eingaben.
  - =>Perzeptron ist linear seperable Boolesche Funktion

$$S_0 = sign(\sum_{j=1}^{N} S_j w_j)$$

Aufbau des Perzeptrons



Die Aktivierungsfunktion

- binäre Klassifikation:

$$y = sign(S_0)$$

Lernen und Verallgemeinern

- Erhält keine Regeln/Programme nur Beispiele
- Beispiele: Menge von Eingabe-/Ausgabepaaren

(x_υ, y_υ), $\upsilon=1,..,M$, $y_\upsilon=\{-1,+1\}$

- Lernen durch Anpassung der Gewichtungen an die Beispiele
- Richtiges klassifizieren durch:

$$y_\upsilon = sign(\vec{w} * \vec{x}_\upsilon) \Rightarrow \quad y = sign(S_0)$$

Lernen

- 1949: Hebb (Psychologe) postulierte: Synapsen passen sich an Aktivität ihrer Ein- und Ausgabeneuronen an.

$$\Delta w = \frac{1}{N} y_v x_v , v = 1,..,M$$

- Bei jedem neuen Beispiel ändert sich die Synapsenstärke w_i ein wenig

  (proportional zum Produkt aus Eingabe- und Ausgabeaktivität)

Lernen

- Angenommen es wird nur ein Beispiel gelernt (**x**,y)=> man erkennt, dass Synapsen die durch das delta w bestimmt werden, die Aktivierungsfkt. erfüllen:

$$\vec{w} * \vec{x} = \sum_{j=1}^{N} \left(\frac{y}{N} * x_j\right) x_j = \frac{y}{N} \sum_{j=1}^{N} x_j x_j = y$$

$$y_v = sign(\vec{w} * \vec{x}_v) \qquad \Delta w = \frac{1}{N} y_v \vec{x}_v, v = 1,..,M$$

Lernen und Verallgemeinern

- Nicht mehr möglich, wenn O(N)=M, wobei N=# der Eingabeneuronen und M=#der Beispiele
- Abänderung der Regel:

  Jedes Beispiel soll nur gelernt werden, wenn es noch nicht richtig durch das momentane **w** abgebildet wird

Lernen und Verallgemeinern

- Perzeptron Lernregel (Rosenblatt, 1960):
  - 1. Lege nacheinander die Eingabesignale an und berechne daraus ein Ausgangssignal.
  - 2. Entspricht das Ausgangssignal dem Sollwert, so bleiben die Gewichte unverändert.
  - 3. Ausgangssignal 0, Sollwert 1 sein, so erhöhe die Gewichte.
  - 4. Ausgangssignal 1, Sollwert 0 sein, so erniedrige die Gewichte.

$$\Delta w = \frac{1}{N} y_v \vec{x}_v \text{ , } wenn(\vec{w} * \vec{x}_v) y_v \leq 0, sonst \Delta \vec{w} = 0$$

Lernen und Verallgemeinern

- Algorithmus stoppt erst, wenn die M Beispiele richtig abgebildet werden konnten.
- Dann existiert ein Gewichtsvektor **w** der für alle Bespiele die Gleichung erfüllt:

$$y_v = sign(\vec{w} * \vec{x}_v)$$

Lernen

- Für diese Lernregel existiert Konvergenzbeweis
  - Zahl t der durchgeführten Schritte ist endlich
  - der Algorithmus stoppt nach höchstens t* Schritten (endet nur, wenn alle Beispiele richtig abgebildet werden)

Lernen

- Für diese Lernregel existiert Konvergenzbeweis
  - Zahl t der durchgeführten Schritte ist endlich
  - der Algorithmus stoppt nach höchstens t* Schritten (endet nur, wenn alle Beispiele richtig abgebildet werden)

Lernen

Wie viele Beispiele kann ein Perzeptron lernen?

- Hängt von Abbildung ab, die Beispiele erzeugen
  - Von anderem Perzeptron, nach Konvergenztheorem jede Anzahl M
  - Zufällig gewählt y_U(Tastatureingabe)
    - für M<N können alle Beispiele gelernt werden
      Für Limes $n \to \infty$ für M<2N
      (Mit Wahrscheinlichkeit 1)

Verallgemeinern

- Bedeutet: Perzeptron kann vorher nicht gelernte Eingabe **S** trotzdem klassifizieren
- Liefert „Lehrer-Funktion“ für Eingabe S das Resultat **S0** => Verallgemeinerungsfähigkeit g ist wie folgt definiert:

$$g = \langle \Theta \left(S_0 \vec{w} \cdot \vec{S} \right) \rangle_S$$

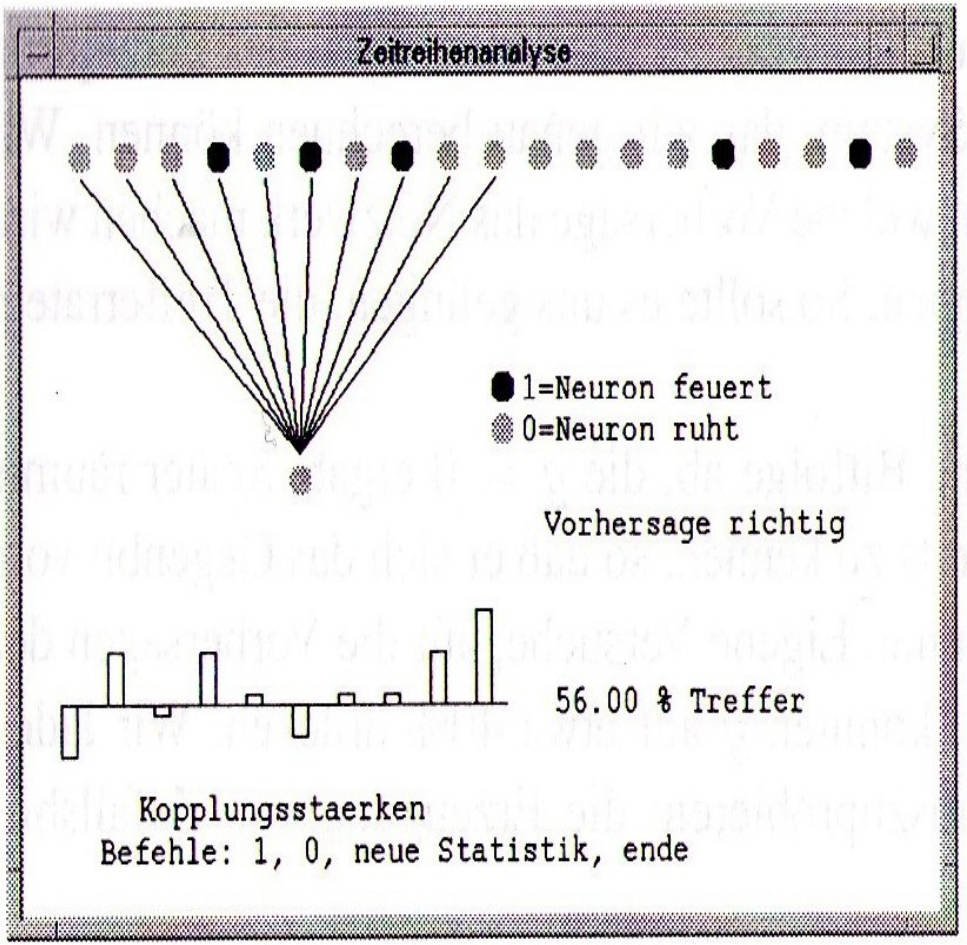
Verfahren des Perzeptrons

- F=(-1,1,1,1,-1,1,-1,-1,1,1,-1,-1,1,-1)
- Perzeptron tastet jeweils ein Fenster von N Bits ab, macht Vorhersage für N+1 über:

$$\widetilde{F}_{\upsilon+N} = sign(\vec{w} * \vec{x}_{\upsilon})$$

- lernt Beispiel (x_{υ},$F_{\upsilon+N}$) mit Rosenblatt-Regel schiebt Fenster um ein Bit nach rechts, macht neue Vorhersage für $x_{\upsilon+1}$
- Iteriert und Trefferrate g bestimmt
- Zufallsfolge: Trefferrate g=50%

Anwendung Perzeptron



- 20x 0 und 1 Eingabe, Perzeptron greift nur auf letzten 10 zurück, macht Vorraussage für folgende Eingabe
- Vorraussage nicht angezeigt, erst nach Eingabe ausgegeben
- Lernvorgang: Änderung d. Synaptischen Gewichtung
- Trefferrate g>50%: Perzeptron hat Struktur in Bitfolge erkannt

Testergebnisse

- Test mit Studenten ergab:

  1000 mal 0 oder 1 zufällig eingetippt
- Auswertung: g=60% mit Werten von 50%<g<80%
- Scheint schwer zu sein Zufallsfolgen per Hand einzutippen

Testergebnisse

- Schon mit 5 Zeilen Computerprogramm kann man menschliche Reaktionen vorhersagen
- Überlistung des Perzeptrons:
  - Wenn man jedoch Spieß umdreht, konnte g auf 40% gedrückt werden

Anwendungsbeispiele

- Frühwarnsystemen
- Zeitreihenanalyse (Wetter, Aktien etc.)
- Bildverarbeitung und Mustererkennung
  - Schrifterkennung (OCR)
  - Spracherkennung

Zusammenfassung

- 3 Schichten: Eingabeschicht S_i, “synaptische” Gewichte w_i, Ausgabeneuron S_0
- Basisfunktion und Aktivierungsfunktion
- Lernen durch Änderung der Synapsengewichtung
- Nach lernen: Verallgemeinern
- g>50%

Literaturverzeichnis

- Physik per Computer (Wolfgang Kinzel / Georg Reents)
- Introduction to the theory of neural computation (John Hertz, Anders Krogh, Richard G. Palmer)
- http://www.iicm.tu-graz.ac.at/greif/node10.ht
- http://www.gc.ssr.upm.es/inves/neural/ann3/concepts/biotype.htm