Neuronale Netze

Anna Wallner

15. Mai 2007

Inhalt

Neuronale Netze:

- Motivation
- Grundlagen
- Beispiel: XOR
- Netze mit einer verdeckten Schicht
- Anpassung des Netzes mit Backpropagation
- Probleme
- Beispiel: Klassifikation handgeschriebener Ziffern
- Rekurrente neuronale Netze

Motivation

Gehirn

- arbeitet in hohem Maße parallel
- erkennt Muster
- kann verrauschte/ unvollständige Daten rekonstruieren
- kann Beispiele verallgemeinern
- lernt selbstständig

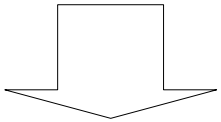
Computer

- numerisch präzise Berechnung
- speichert Daten fehlerlos
- kann zuverlässig auf gespeicherte Daten zugreifen
- vergisst nichts

Motivation

Informationsverarbeitung im Gehirn:

- Interaktion von stark vernetzten Neuronen über elektrische Impulse
- Neuronen können gleichzeitig untereinander Informationen austauschen



Informationsverarbeitung in einem künstlichen neuronalen Netz:

- Künstliche Neuronen aktivieren sich untereinander mit Hilfe von gerichteten Verbindungen

  ~> Aufgaben können anhand von Trainingsbeispielen erlernt werden

  ~> hohe Parallelität bei der Informationsverarbeitung

  ~> hohe Fehlertoleranz

Motivation

Fähigkeiten eines neuronalen Netzes:

- Approximation beliebig komplexer Funktionen
- Erlernen von Aufgaben (z.B. Klassifikation)
- Lösen von Problemen, bei denen eine explizite Modellierung schwierig oder nicht durchführbar ist

Motivation

Anwendungsgebiete neuronaler Netze:

- Frühwarnsysteme
- Optimierung
- Zeitreihenanalysen (z.B. Wetter, Aktien)
- Bildverarbeitung und Mustererkennung
  - Schrifterkennung
  - Spracherkennung
  - Data-Mining
- Informatik: Bei Robotik, virtuellen Agenten und KI-Modulen in Spielen und Simulationen.

Inhalt

Neuronale Netze:

- Motivation
- Grundlagen
- Beispiel: XOR
- Netze mit einer verdeckten Schicht
- Anpassung des Netzes mit Backpropagation
- Probleme
- Beispiel: Klassifikation handgeschriebener Ziffern
- Rekurrente neuronale Netze

Künstliches Neuron:

nicht-lineare parametrisierte beschränkte Funktion $f(X_1,X_2,...,X_p,\omega_1,\omega_2,...,\omega_p)$

$X_i,\ i = 1,...,p$ Eingaben

$\omega_i,\ i = 1,...,p$ Parameter (bzw. Gewichte)

$y = f(X_1,X_2,...,X_p,\omega_1,\omega_2,...,\omega_p)$ Ausgabe

Eingaben:

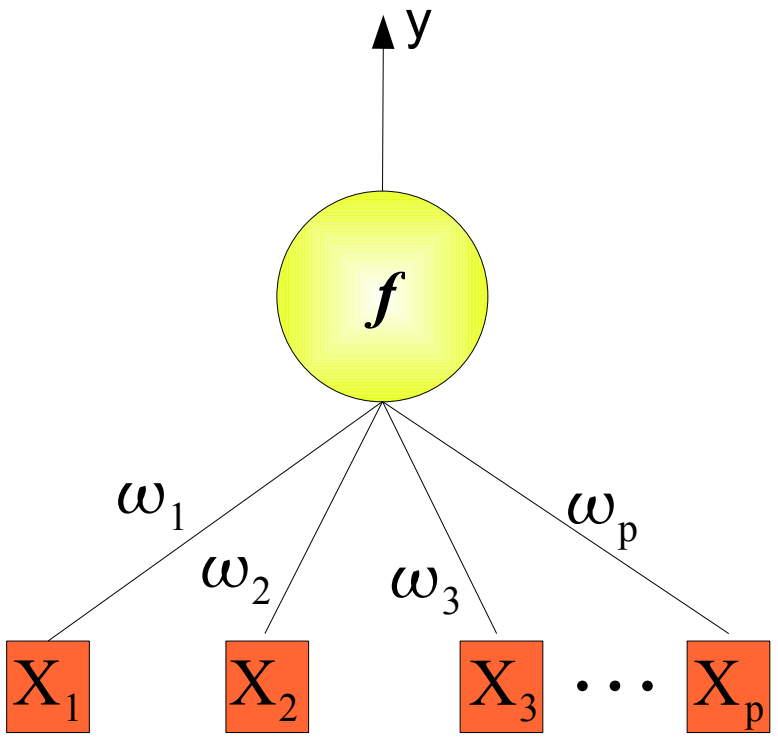
Ausgaben anderer Neuronen oder
beobachtete Werte eines Prozesses

Gewichte:

bestimmen den Grad des Einflusses, den die Eingaben auf die Aktivierungsfunktion f haben
> 0 ~> erregende Wirkung
= 0 ~> keine Wirkung ≙ keine Verbindung
< 0 ~> hemmende Wirkung

Ausgabe:

Ergebnis der Funktion f



2 gängige Möglichkeiten der Parametrisierung von f:

1.) Die Parameter werden den Eingaben zugeordnet:

- Ausgabe entspricht nicht-linearer Kombination der Eingabewerte $\{X_i\}$, gewichtet durch Parameter $\{\omega_i\}$

- häufig verwendetes **Potenzial**:

  gewichtete Summe der Eingabewerte mit zusätzlichem „**Bias**“-Term

$$v = \omega_0 + \sum_{i=1}^{p} \omega_i X_i$$

- die Aktivierungsfunktion f(v) ist meist s-förmig (**sigmoid**)

2 gängige Möglichkeiten der Parametrisierung von f:

2.) Die Parameter gehören zur Definition der Aktivierungsfunktion

Beispiel: **Gauß'sche radiale Basisfunktion**

$$f(X_1, X_2, ..., X_p, \omega_1, \omega_2, ..., \omega_p, \omega_{p+1}) = \exp[-\sum_{i=1}^{p} (X_i - \omega_i)^2 / 2\omega_{p+1}^2]$$

wobei $\omega_i, i = 1,...,p$ $\hat{=}$ Position des Mittelpunktes der Gaußglocke

ω_{p+1} $\hat{=}$ Standardabweichung.

Künstliches neuronales Netz:

Verknüpfung der nicht-linearen Funktionen von Neuronen

„Feedforward“-Netze

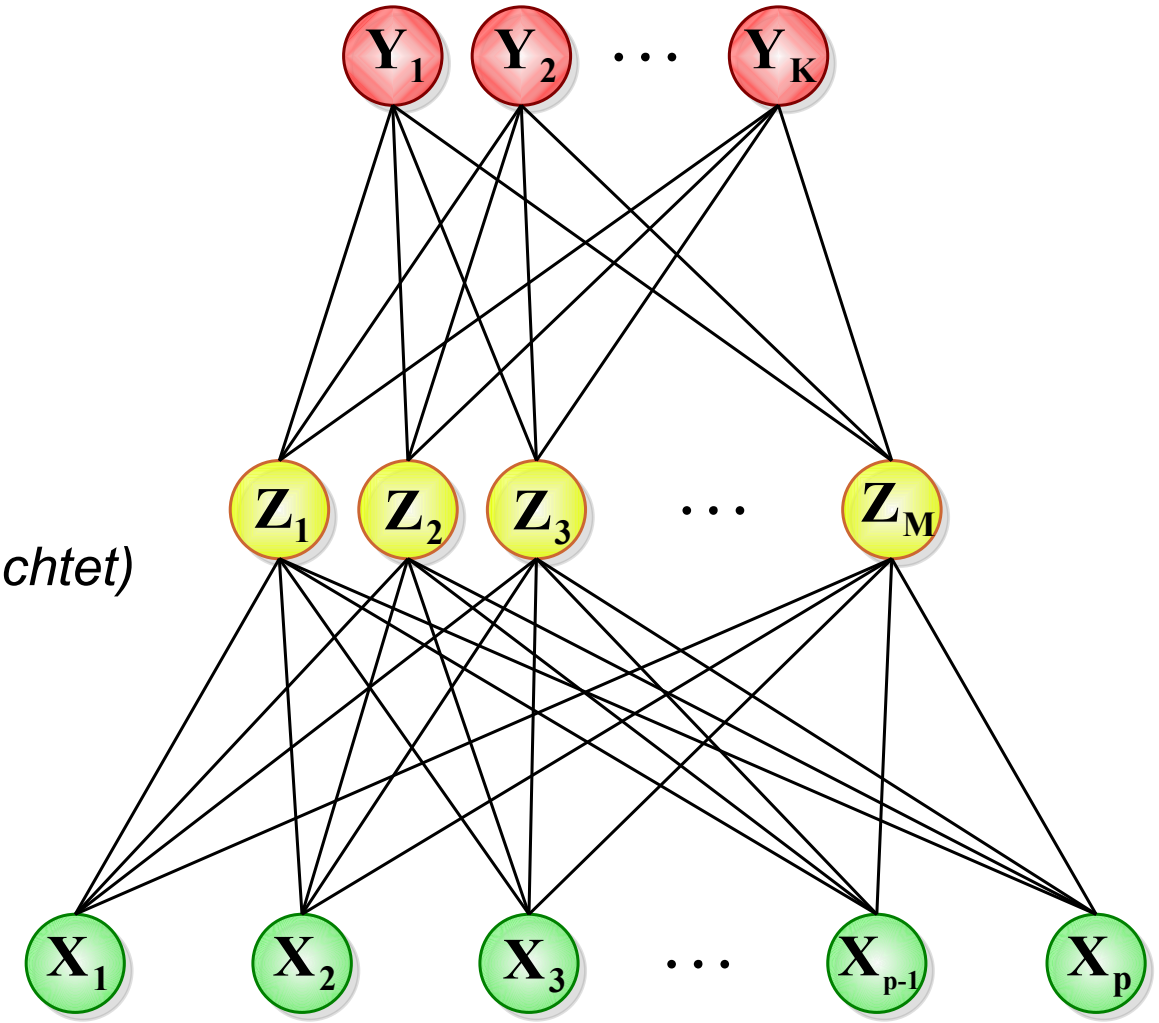
- Informationen fließen nur in eine Richtung
  (von der Eingabe zur Ausgabe)
- stationär (d.h. Eingabewerte konstant
  => Ausgabewerte konstant)

Netzwerkdiagramm eines neuronalen Netzes mit einer verdeckten Schicht

Ausgabeschicht

verdeckte Schicht
(wird nicht direkt beobachtet)

Eingabeschicht



Inhalt

Neuronale Netze:

- Motivation
- Grundlagen
- Beispiel: XOR
- Netze mit einer verdeckten Schicht
- Anpassung des Netzes mit Backpropagation
- Probleme
- Beispiel: Klassifikation handgeschriebener Ziffern
- Rekurrente neuronale Netze

Beispiel: XOR

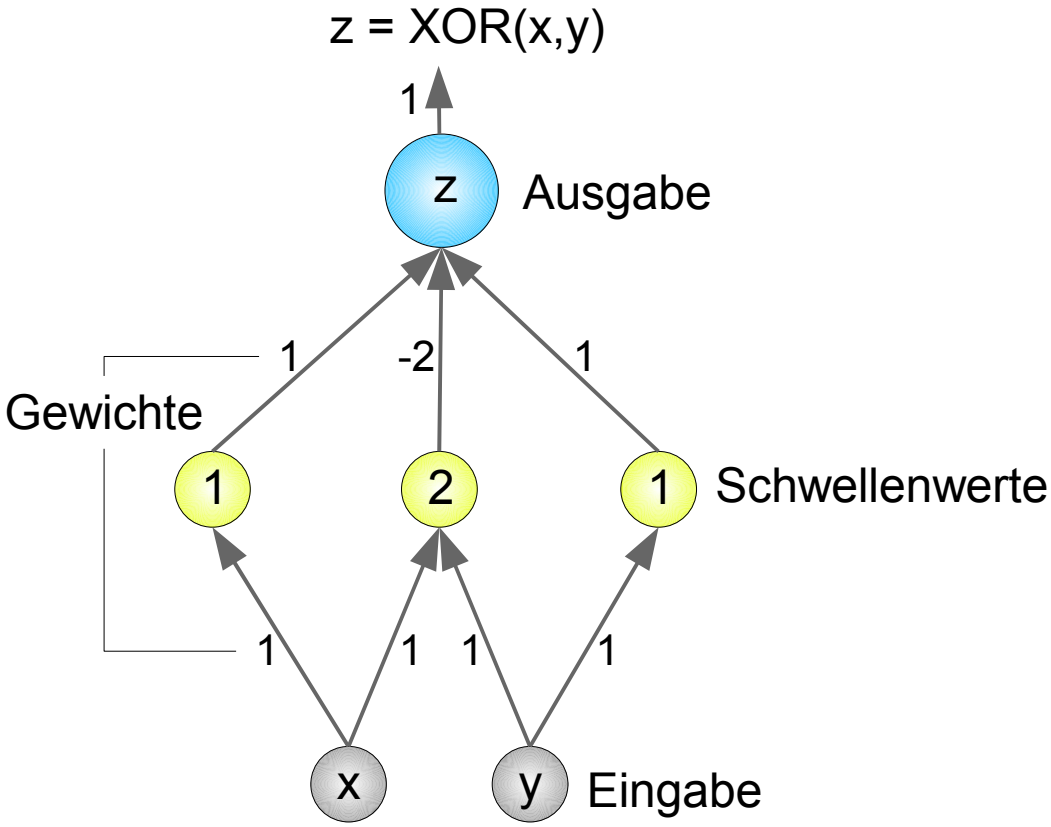
XOR (Exclusive Or)

logische Verknüpfung zweier Operatoren

0 ≙ „false“

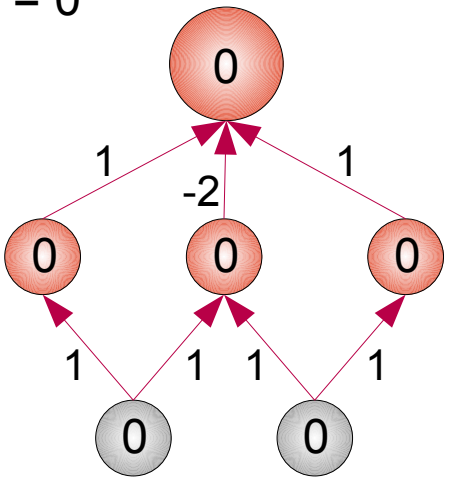
1 ≙ „true“

Ergebnis „true“
<=>
genau ein Operator hat den Wert 1

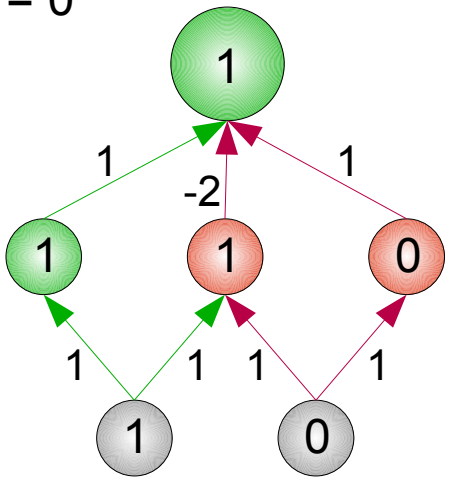


Beispiel: XOR

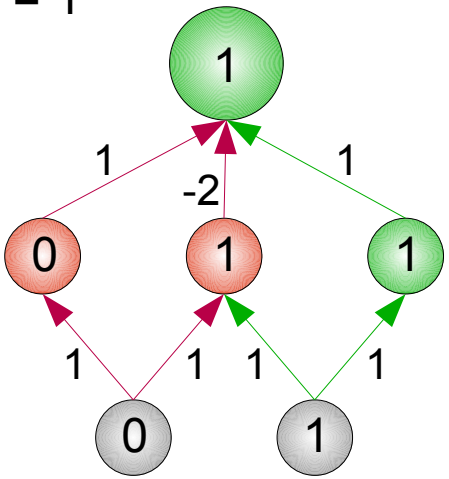
1.) x = 0, y = 0



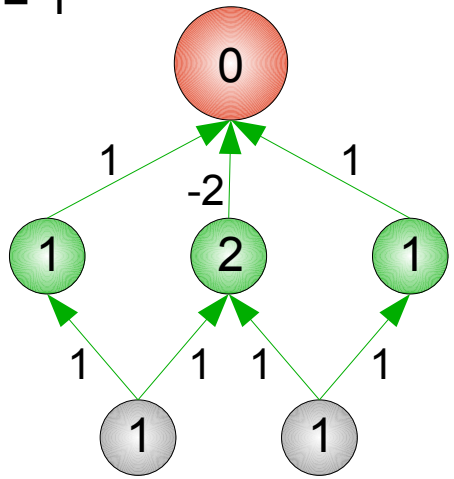
2.) x = 1, y = 0



3.) x = 0, y = 1



4.) x = 1, y = 1



Inhalt

Neuronale Netze:

- Motivation
- Grundlagen
- Beispiel: XOR
- Netze mit einer verdeckten Schicht
- Anpassung des Netzes mit Backpropagation
- Probleme
- Beispiel: Klassifikation handgeschriebener Ziffern
- Rekurrente neuronale Netze

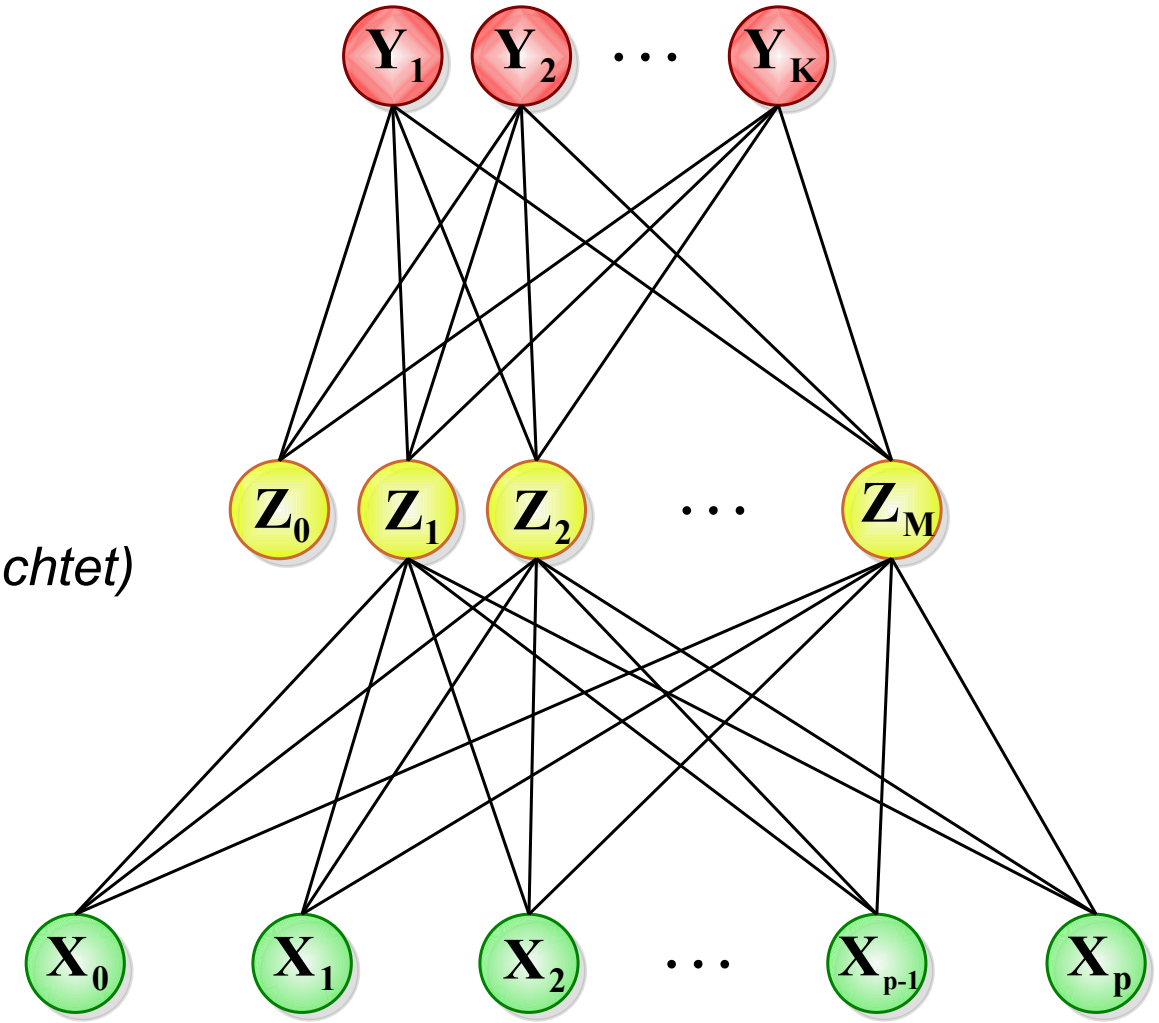
Netze mit einer verdeckten Schicht

Netzwerkdiagramm eines neuronalen Netzes mit einer verdeckten Schicht

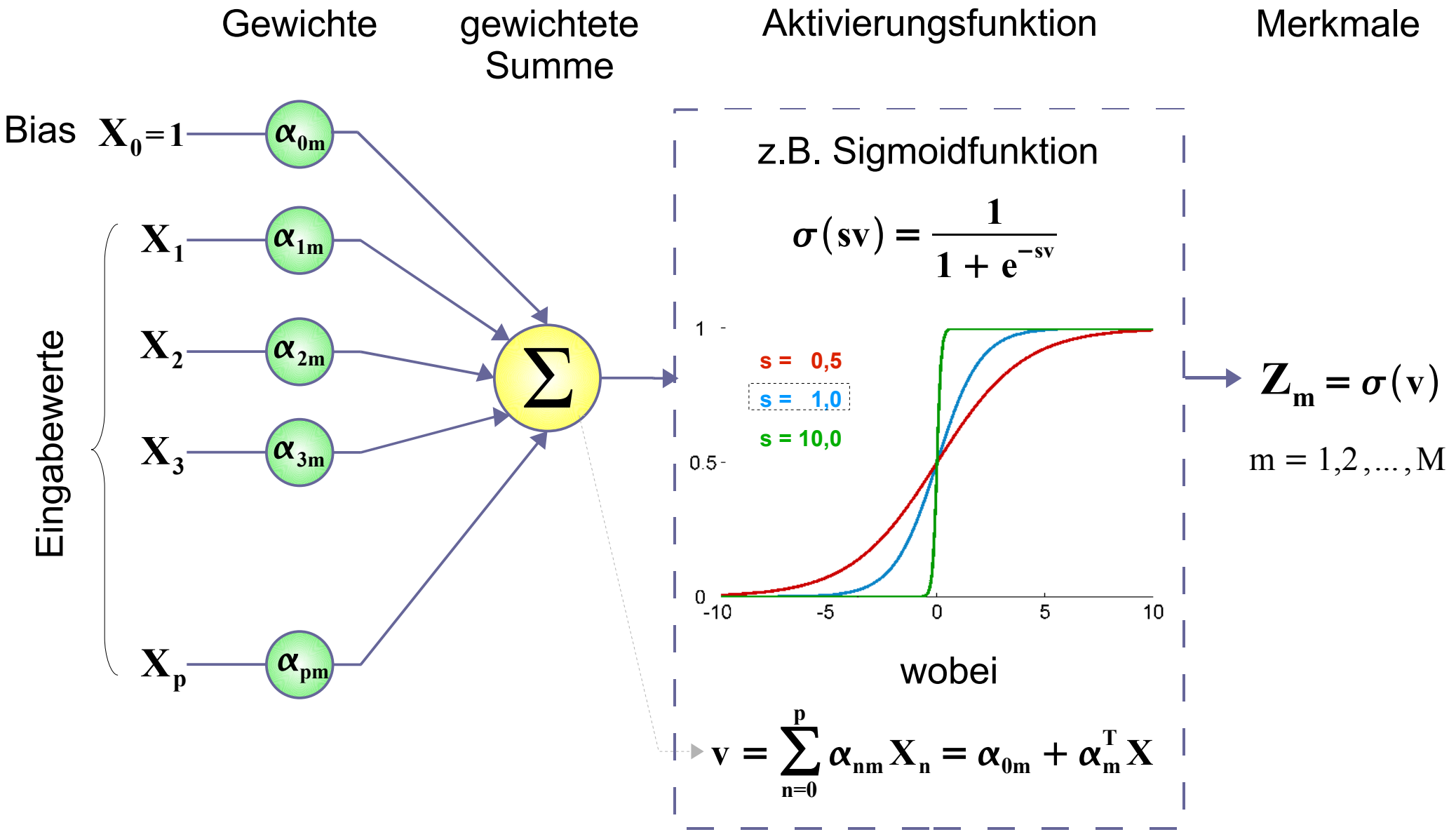
Ausgabeschicht

verdeckte Schicht
(wird nicht direkt beobachtet)

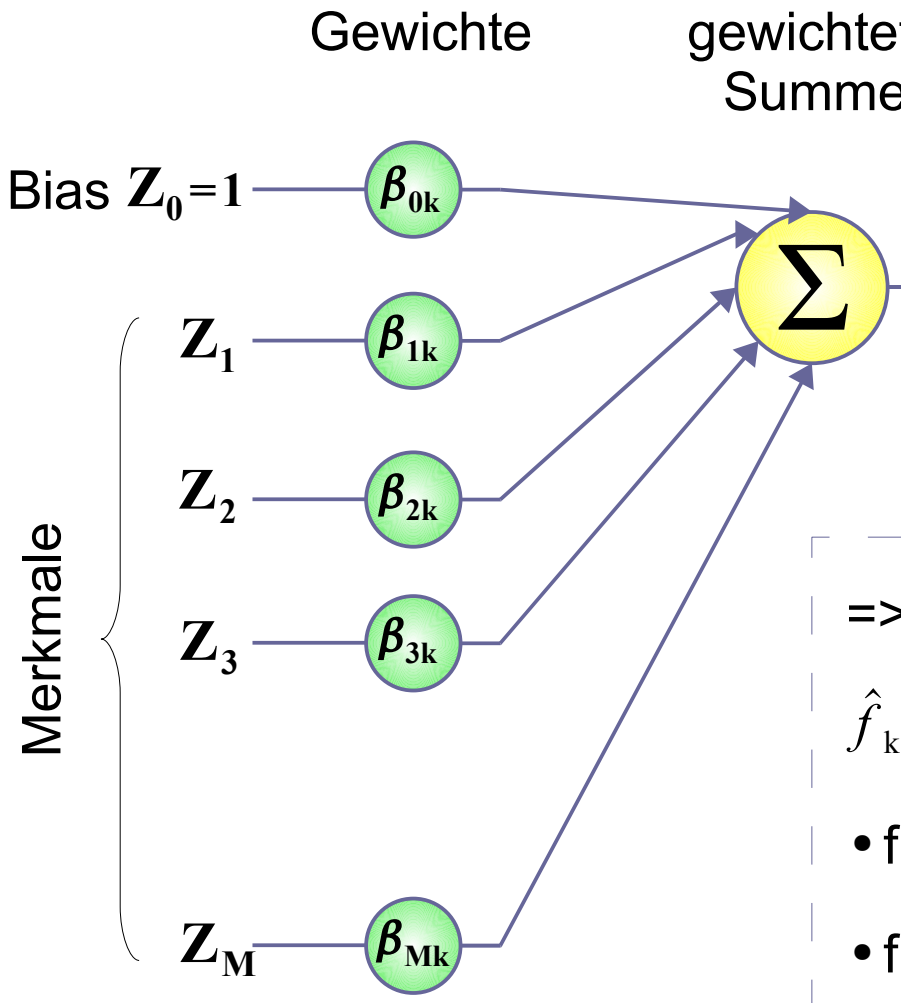
Eingabeschicht



Netze mit einer verdeckten Schicht



Netze mit einer verdeckten Schicht



$$T_k = \sum_{m=0}^{M} \beta_{mk} Z_k = \beta_{0k} + \beta_k^T Z$$

$$k = 1,2,...,K$$

=> Ausgabe des neuronalen Netzes:

$$\hat{f}_k(X) = g_k(T), \; k = 1,2,...,K$$

- für Regressionen meist $g_k(T) = T_k$
- für Klassifikationen meist $g_k(T) = \underbrace{e^{T_k} \cdot \left(\sum_{l=1}^{K} e^{T_l} \right)^{-1}}_{\textit{Softmax-Funktion}}$

Inhalt

Neuronale Netze:

- Motivation
- Grundlagen
- Beispiel: XOR
- Netze mit einer verdeckten Schicht
- Anpassung des Netzes mit Backpropagation
- Probleme
- Beispiel: Klassifikation handgeschriebener Ziffern
- Rekurrente neuronale Netze

Anpassung des Netzes mit Backpropagation

Wie bestimmt man geeignete Gewichte?

Sei θ die Menge aller Gewichte

$$\Rightarrow \theta = \{(\alpha_{0m}, \alpha_m, \beta_{0k}, \beta_k);\ \alpha_{0m},\ \beta_{0k} \in \mathbb{R},\ \alpha_m \in \mathbb{R}^p,\ \beta_k \in \mathbb{R}^M,\ m = 1,2,..,M,\ k = 1,2,..,K\}$$

und $(X,Y),\ X = (X_1,...,X_p),\ Y = (y_1,...,y_K)$ ein Trainingsdatensatz.

~> Fehlerfunktionen als Maß für die Anpassung

~> **Summe der Fehlerquadrate**

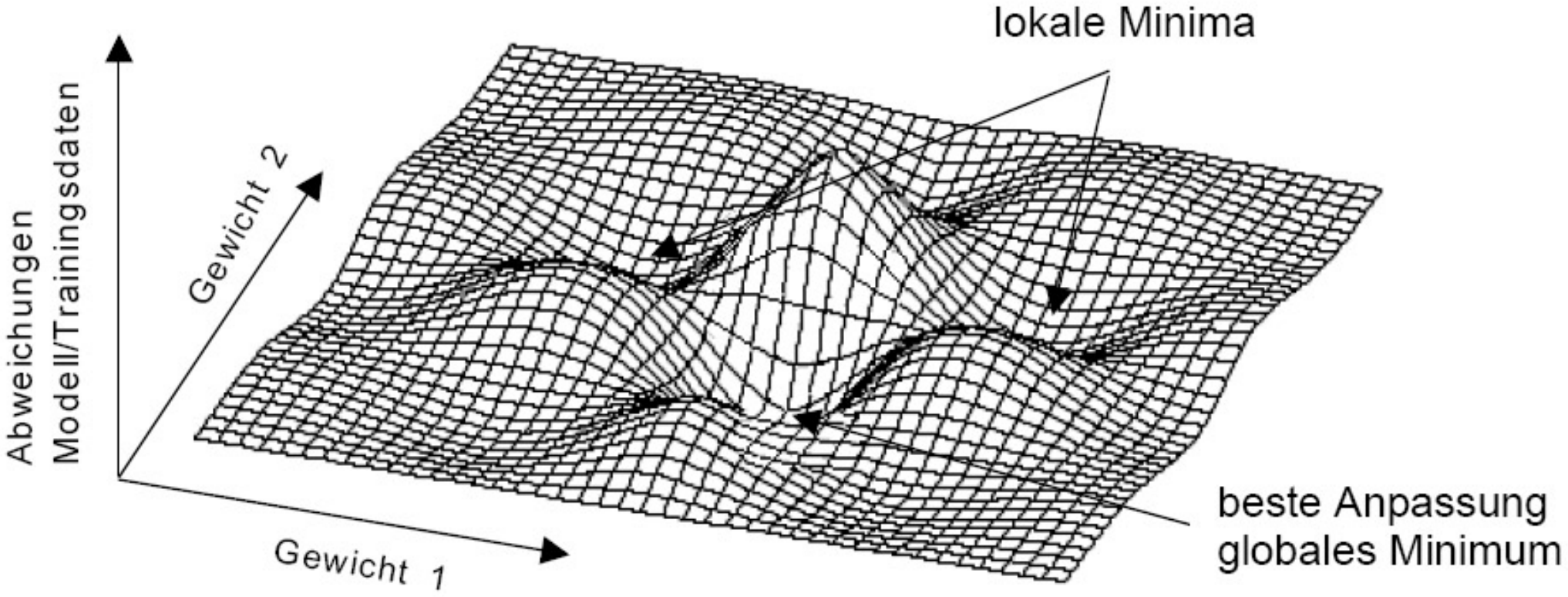
$$R(\theta) = \sum_{k=1}^{K} \underbrace{(y_k - \hat{f}_k(X))^2}$$

Differenz zwischen gewünschter und tatsächlicher Ausgabe

Minimieren von R(θ) mit Hilfe des Gradientenabstiegs

~> **Backpropagation**

Anpassung des Netzes mit Backpropagation



Anpassung des Netzes mit Backpropagation

Backpropagation-Algorithmus:

1. **„forward pass“**:

   - Festlegen der Gewichte

   - Anlegen eines Eingabemusters, das anschließend vorwärts durch das Netz propagiert wird

   - Berechnung der $\hat{f}_k(\mathbf{X}), k = 1,..,K$

2. Vergleich dieser Ausgabe mit den gewünschten Werten

   ~> Differenz = Fehler des Netzes

$$\rightarrow R(\theta) = \sum_{k=1}^{K} (y_k - \hat{f}_k(X))^2, \; X = (X_1,...,X_p)$$

Anpassung des Netzes mit Backpropagation

3. „**backward pass**“

~> der Fehler wird über die Ausgabe- zur Eingabeschicht zurück propagiert, die Gewichtungen der Neuronenverbindungen werden abhängig von ihrem Einfluss auf den Fehler geändert

Es gilt $Z_m = \sigma(\alpha_{0m} + \alpha_m^T X),\ Z = (Z_1,...,Z_M)$

$$\Rightarrow \hat{f}_k(X) = g_k(T_k) = g_k(\beta_{0k} + \beta_k^T Z)$$
$$= g_k(\beta_{0k} + \sum_{m=1}^{M} \beta_{mk}\sigma(\alpha_{0m} + \alpha_m^T X))$$

$$\Rightarrow R(\theta) = \sum_{k=1}^{K}(y_k - \hat{f}_k(X))^2 = \sum_{k=1}^{K}\left[y_k - g_k(\beta_{0k} + \sum_{m=1}^{M}\beta_{mk}\sigma(\alpha_{0m} + \alpha_m^T X))\right]^2$$

Anpassung des Netzes mit Backpropagation

Berechnung der Anpassung der Gewichte:

$$\Delta\beta_{mk} = -\gamma \cdot \frac{\partial R(\theta)}{\partial \beta_{mk}} = -\gamma \cdot \delta_k \cdot Z_m, \quad m = 0,1,...,M, \quad k = 1,...,K$$

$$\Delta\alpha_{nm} = \underbrace{-\gamma}_{\text{Lernrate}} \cdot \frac{\partial R(\theta)}{\partial \alpha_{nm}} = -\gamma \cdot s_m \cdot X_n, \quad n = 0,1,...,p, \quad m = 1,...,M$$

mit $$\delta_k = -2 \cdot (y_k - \hat{f}_k(X)) \cdot \frac{\partial g_k(\beta_{0k} + \beta_k^T Z)}{\partial \beta_{mk}}$$

$$s_m = \sum_{k=1}^{K} \beta_{km} \cdot \delta_k \cdot \frac{\partial \sigma(\alpha_{0m} + \alpha_m^T X)}{\partial \alpha_{nm}}$$

Aktualisierung der Gewichte:

$$\beta_{mk}^{neu} = \beta_{mk}^{alt} + \Delta\beta_{mk}$$

$$\alpha_{nm}^{neu} = \alpha_{nm}^{alt} + \Delta\alpha_{nm}$$

Inhalt

Neuronale Netze:

- Motivation
- Grundlagen
- Beispiel: XOR
- Netze mit einer verdeckten Schicht
- Anpassung des Netzes mit Backpropagation
- Probleme
- Beispiel: Klassifikation handgeschriebener Ziffern
- Rekurrente neuronale Netze

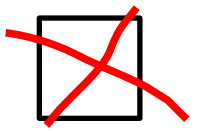
Probleme

- **<u>Wahl der Startwerte für die Gewichte</u>**

  - alle Gewichte identisch 0

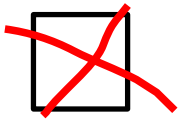
    => Modell ist in allen Komponenten symmetrisch

    => der Algorithmus wiederholt sich ohne tatsächlich etwas zu bewirken



  - zu hohe Werte

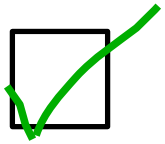
    => eventuell schlechte Ergebnisse



  - Zufallswerte nahe 0

    => Modell ist zu Beginn fast linear,

    wird nicht-linear, wenn Gewichte größer werden



Probleme

- **<u>Anzahl der verdeckten Neuronen und Schichten</u>**

  - zu viele verdeckte Neuronen => zu viele Parameter

    => das neuronale Netz ist zu flexibel

    => Überanpassung

  - zu wenige verdeckte Neuronen => zu wenige Parmeter

    => das neuronale Netz ist nicht komplex genug

    => kann nicht richtig trainiert werden

  - häufig: # verdeckte Neuronen $\in$ {5,...,100}

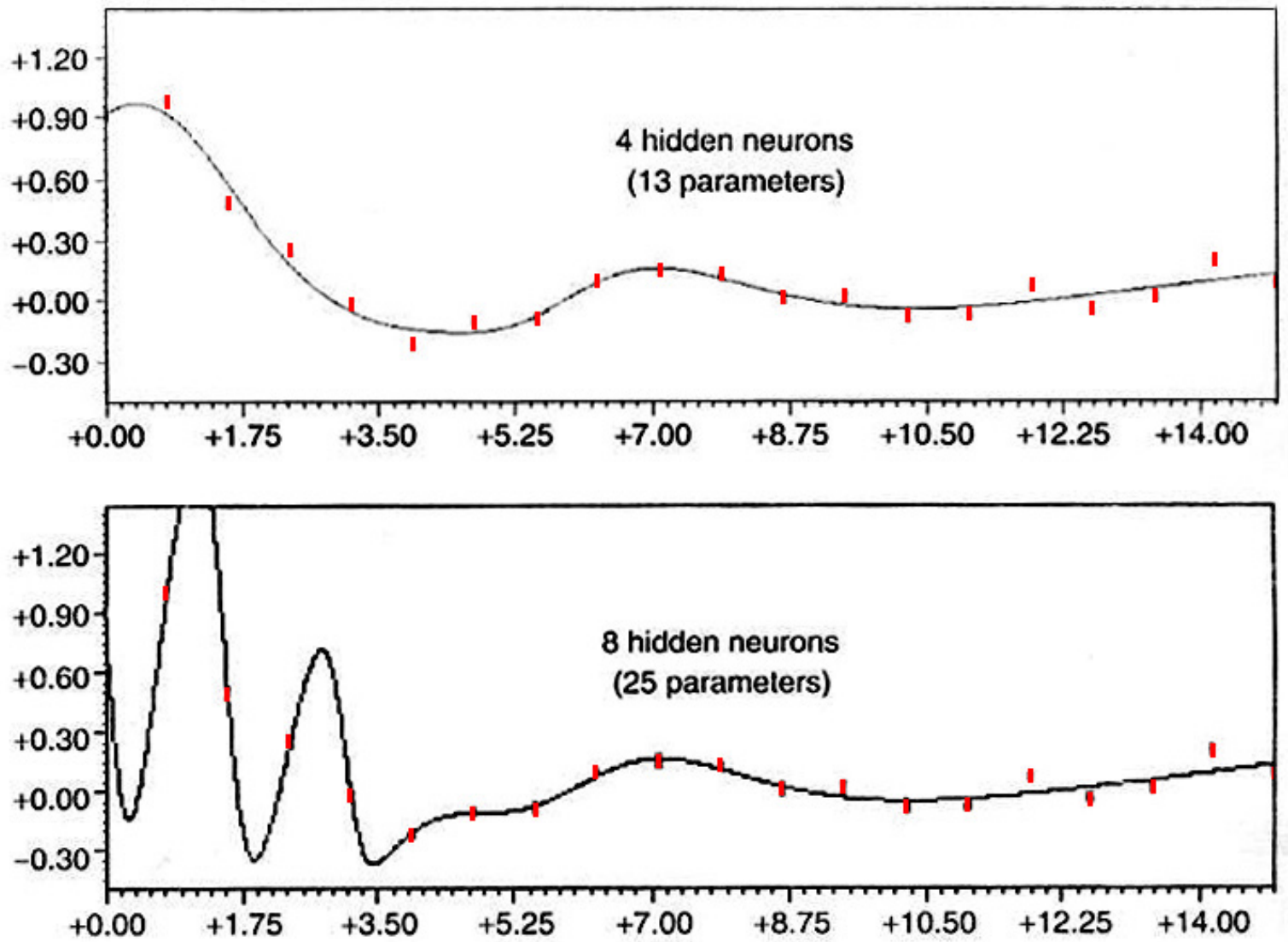
    (steigt mit Anzahl der Eingabewerte und der Trainingseinheiten)

  - # verdeckte Schichten abhängig von

    - Hintergrundwissen
    - Ergebnissen von Experimenten

Probleme

Beispiel:



Probleme

- **<u>Überanpassung</u>**

Modell im globalen Minimum der Fehlerfunktion R oft überangepasst

~> rechtzeitiger Abbruch des Verfahrens

~> **Weight-Decay-Methode**

zur Fehlerfunktion wird ein Strafterm addiert: R(θ) + λJ(θ), wobei

$J(\theta)=\frac{1}{2}\left(\sum_{mk}\beta_{mk}^2+\sum_{nm}\alpha_{nm}^2\right)$ und $\lambda \geqslant 0$ ein Tuningparameter

$$\Rightarrow \Delta\beta_{mk} = -\gamma\cdot\frac{\partial\,(R(\theta)+\lambda J(\theta))}{\partial\beta_{mk}} = -\gamma\cdot\left(\frac{\partial\,R(\theta)}{\partial\beta_{mk}}+\lambda\beta_{mk}\right)$$

$$\Rightarrow \Delta\alpha_{nm} = -\gamma\cdot\frac{\partial\,(R(\theta)+\lambda J(\theta))}{\partial\alpha_{nm}} = -\gamma\cdot\left(\frac{\partial\,R(\theta)}{\partial\alpha_{nm}}+\lambda\alpha_{nm}\right)$$

=> Zu große Werte für λ lassen die Gewichte gegen 0 schrumpfen.

Probleme

- **Skalierung der Eingabewerte**

~> bestimmt die effektive Skalierung der Gewichte in der untersten Schicht

=> kann einen großen Einfluss auf die Qualität des Endergebnisses haben

~> die Eingabewerte werden z.B. so genormt, dass gilt

Mittelwert = 0
Standardabweichung = 1

=> Gleichbehandlung der Eingabewerte im Regulierungsprozess

=> ermöglicht sinnvolle Wahl eines Intervalls für die Startwerte der Gewichte

~> man wählt in diesem Fall zufällige Gewichte aus dem Intervall

[-0,7; +0,7]

Probleme

- **Multiple Minima**

Die Fehlerfunktion R(θ) ist nicht konvex ~> besitzt viele lokale Minima

=> Endergebnis hängt stark von der Wahl der Anfangswerte der Gewichte ab

=> Experimentieren zu Beginn notwendig

=> wähle dasjenige Netz, das den kleinsten Fehler verspricht

Alternative 1: verwende die durchschnittlichen Vorhersagen aus allen Netzen als endgültige Vorhersage

Alternative 2: „bagging“

wie 1, Netze werden jedoch mit zufällig gestörten Trainingsdaten trainiert

Inhalt

Neuronale Netze:

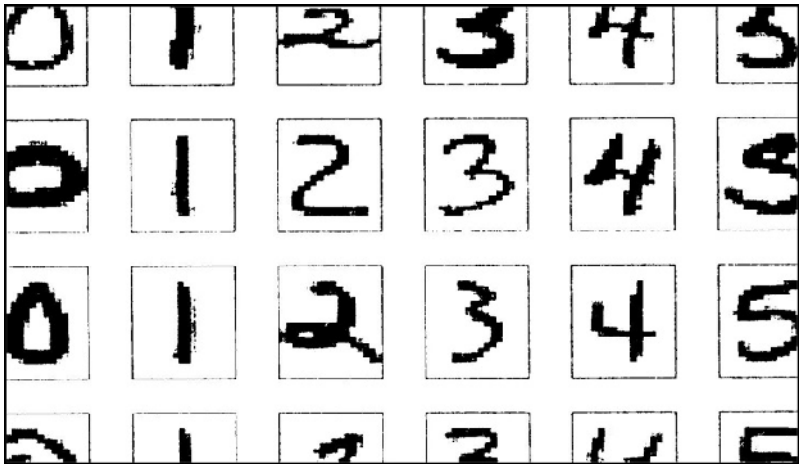
- Motivation
- Grundlagen
- Beispiel: XOR
- Netze mit einer verdeckten Schicht
- Anpassung des Netzes mit Backpropagation
- Probleme
- Beispiel: Klassifikation handgeschriebener Ziffern
- Rekurrente neuronale Netze

Beispiel: Klassifikation handgeschriebener Zahlen

Beispiel: Klassifikation handgeschriebener Ziffern von 0 – 9
(Erkennen von Mustern)

Erzeugung der Eingabemuster:
- Einscannen handgeschriebener Ziffern,
  Auflösung 16x16 Pixel

Umfang:
- Trainingsmenge: 320 Ziffern
- Testmenge: 160 Ziffern



Es werden fünf verschiedene Netze trainiert und anschließend verglichen.

Beispiel: Klassifikation handgeschriebener Zahlen

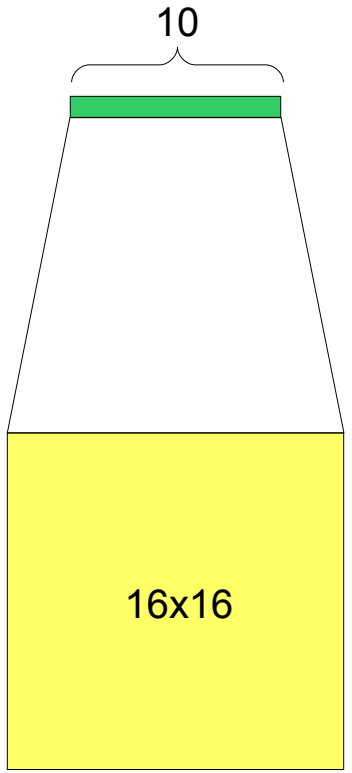
- Eingabeschicht:
  Pro Ziffer 256 Eingabewerte
- Ausgabeschicht:
  10 Neuronen, für die Ziffern 0-9
- $\hat{f}_k(\mathbf{X})$:
  geschätzte Wahrscheinlichkeit, dass Bild X zur Klasse k gehört, $k \in \{0,1,...,9\}$

Net-1: Netz ohne verdeckte Schicht

Gewichte
= # Verknüpfungen

$$10 \cdot 256 + 10 = 2570$$

2570



Erfolgsquote: 80%

Beispiel: Klassifikation handgeschriebener Zahlen

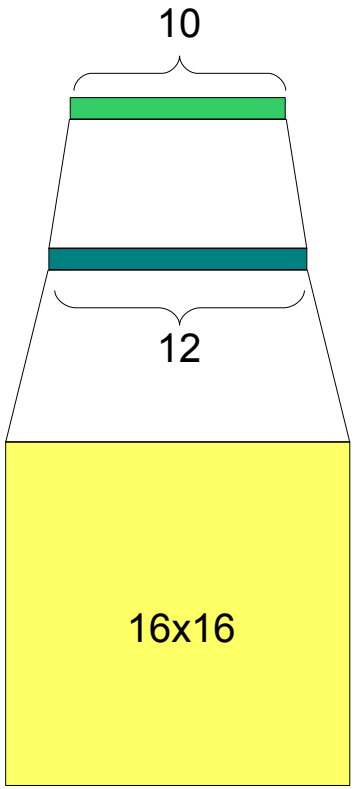
Net-2: Netz mit einer verdeckten Schicht,
deren 12 Neuronen vollständig verknüpft sind

Gewichte = # Verknüpfungen

$$10 \cdot 12 + 10 = \quad 130$$

$$12 \cdot 256 + 12 = 3084$$

3214



Erfolgsquote: 87%

Beispiel: Klassifikation handgeschriebener Zahlen

Net-3: Netz mit zwei verdeckten Schichten, die lokal verknüpft sind

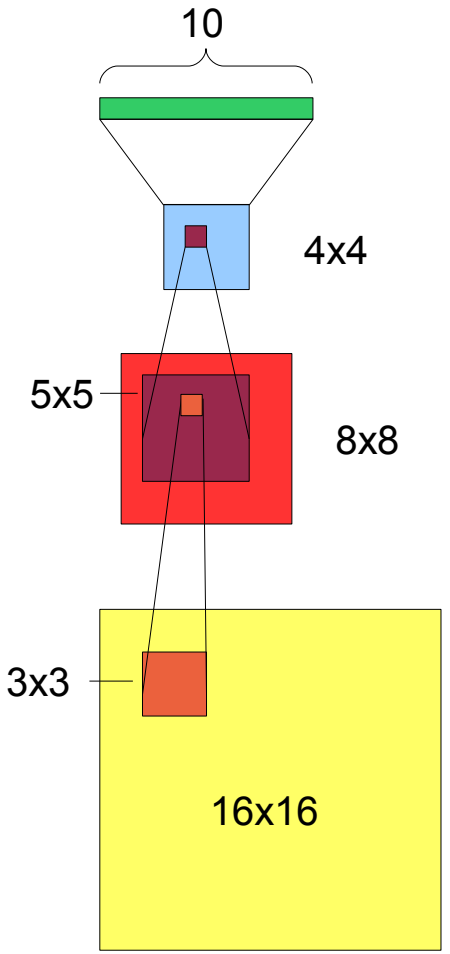
Gewichte = # Verknüpfungen

$$10 \cdot 16 + 10 = 170$$

$$16 \cdot 25 + 16 = 416$$

$$64 \cdot 9 + 64 = 640$$

$$\textbf{1226}$$

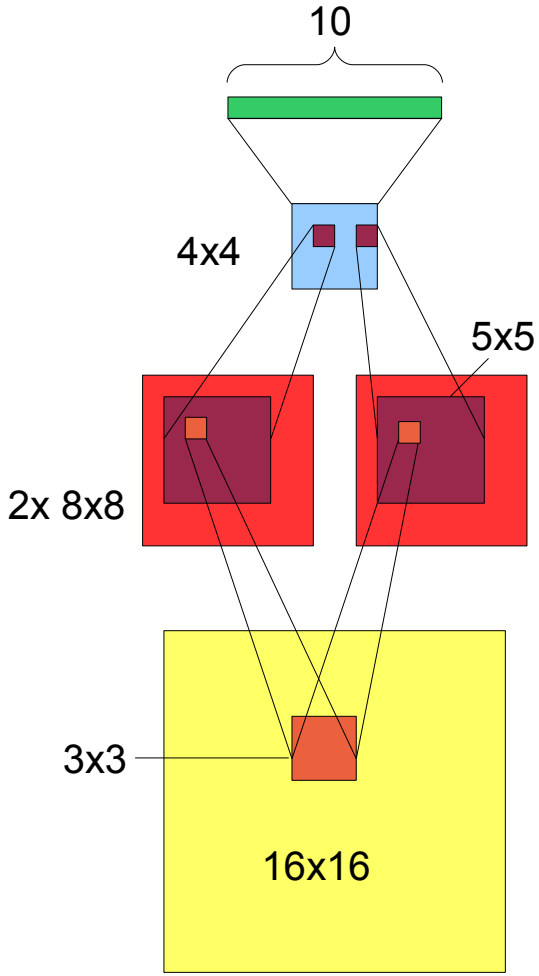


Erfolgsquote: 88,5%

Beispiel: Klassifikation handgeschriebener Zahlen

Net-4: Netz mit zwei verdeckten Schichten, die lokal verknüpft sind
+ weight sharing auf einer Ebene

# Verknüpfungen	# Gewichte
$10 \cdot 16 + 10 = 170$	$10 \cdot 16 + 10 = 170$
$2 \cdot (16 \cdot 25) + 16 = 816$	$2 \cdot (16 \cdot 25) + 16 = 816$
$2 \cdot (64 \cdot 9 + 64) = 1280$	$2 \cdot (9 + 64) = 146$
2266	**1132**

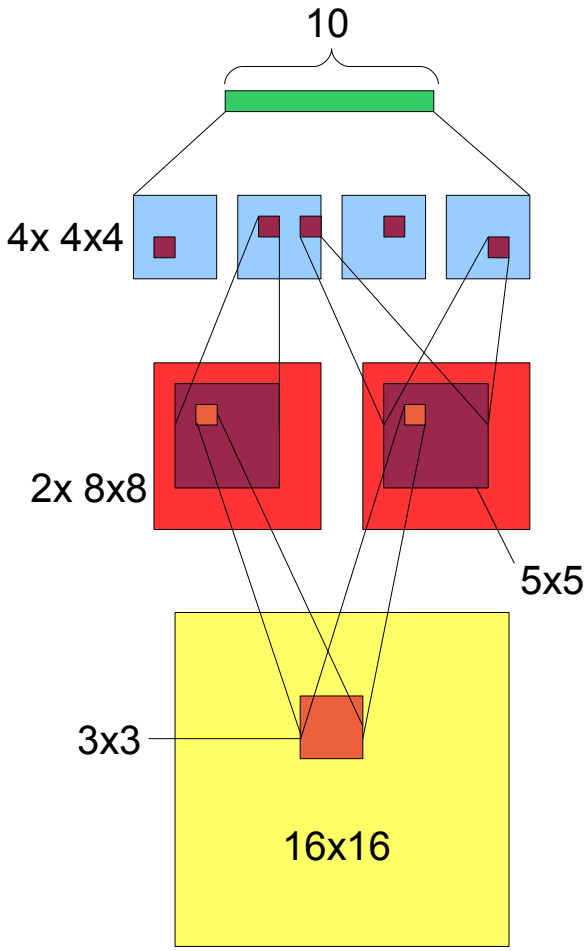


Erfolgsquote: 94%

Beispiel: Klassifikation handgeschriebener Zahlen

Net-4: Netz mit zwei verdeckten Schichten, die lokal verknüpft sind
+ weight sharing auf zwei Ebenen

# Verknüpfungen	# Gewichte
$4\cdot(10\cdot16)+10 = 650$	$4\cdot(10\cdot16)+10 = 650$
$4\cdot(2\cdot(16\cdot25)+16) = 3264$	$4\cdot(2\cdot25+16) = 264$
$2\cdot(64\cdot9+64) = 1280$	$2\cdot(9+64) = 146$
5194	**1060**



Erfolgsquote: 98,4%

Beispiel: Klassifikation handgeschriebener Zahlen

Ergebnis:

	Verknüpfungen	Gewichte	Erfolgsquote
Net-1	2570	2570	80,00%
Net-2	3214	3214	87,00%
Net-3	1226	1226	88,50%
Net-4	2266	1132	94,00%
Net-5	5194	1060	98,40%

=> Net-5 liefert die besten Ergebnisse

Inhalt

Neuronale Netze:

- Motivation
- Grundlagen
- Beispiel: XOR
- Netze mit einer verdeckten Schicht
- Anpassung des Netzes mit Backpropagation
- Probleme
- Beispiel: Klassifikation handgeschriebener Ziffern
- Rekurrente neuronale Netze

Rekurrente neuronale Netze

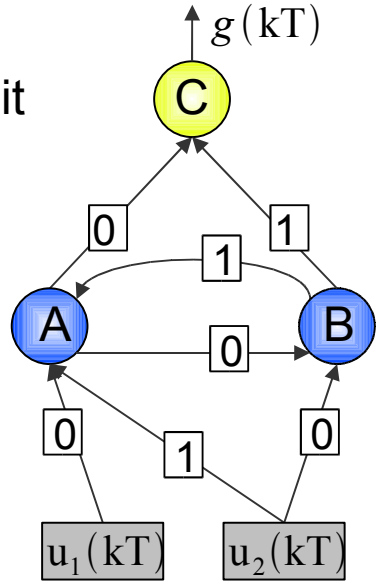
- mindestens eine zyklische Verbindung
- jeder Verknüpfung wird eine bestimmte „Verzögerungszeit“ zugewiesen (Vielfaches einer festgelegten Zeiteinheit)

Beispiel:

☐ Verzögerungszeit

T: Zeiteinheit

k = 1,2,3,...



Inputs und Outputs zum Zeitpunkt kT:

	A	B	C
Inputs	$u_1(kT)$ $u_2[(k-1)T]$ $y_B[(k-1)T]$	$u_2(kT)$ $y_A(kT)$	$u_1(kT)$ $y_A(kT)$
Output	$y_A(kT)$	$y_B(kT)$	$g(kT)$

Quellen

- T. Hastie, R. Tibshirani, J. Friedman. - *The elements of statistical learning*
- G. Dreyfus. - *Neural Networks – Methodology and Applications*
- http://www.wikipedia.org
- http://www.pze.at/linux2/tutor-bu/hege.htm
- http://www.neuronales-netz.de