Fakultät Informatik/Mathematik

Lehrstuhl für Künstliche Intelligenz

HOCHSCHULE FÜR
TECHNIK UND WIRTSCHAFT
DRESDEN
UNIVERSITY OF APPLIED SCIENCES

Vorlesungsmitschrift

NEUROINFORMATIONS-VERARBEITUNG

Mitschrift von

Falk-Jonatan Strube

Vorlesung von

Prof. Dr. Hans-Joachim Böhme

27. März 2018

INHALTSVERZEICHNIS

EINFÜHRUNG

Klausur: Keine Unterlagen oder Hilfsmittel erlaubt

0 BIOLOGISCHER HINTERGRUND

Vorlesung
10.10.2017

NI_intro_bio.pdf
Folie 1

0.1 EINORDNUNG DER NEUROINFORMATIK

Künstliche Intelligenz:

- Logik und Deduktion
- Mustererkennung
- Maschinelles Lernen
- Spracherkennung
- ...
- NEUROINFORMATIONSVERARBEITUNG
  - Biologie
  - Psychologie
  - Computational Neuroscience
  - ...
  - NEURONALEN NETZE

0.2 GESCHICHTE DER NEUROINFORMATIK

NI_intro_bio.pdf
Folie 2

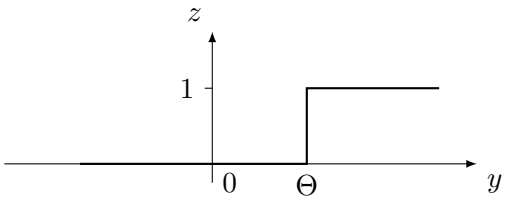
0.2.1 LERNEN

Biologisches System $\longleftrightarrow$ Umwelt

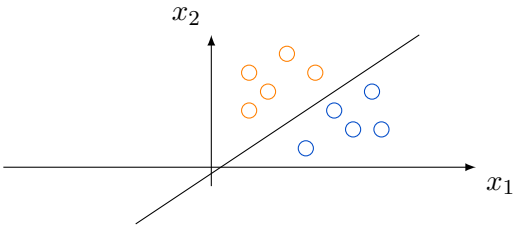
LERNEN: Optimierung der „Fitness“
(Biologisches System soll besser in seiner Umwelt zurecht kommen).
Benötigt dafür: Funktion, die diese Fitness beschreibt.

0.2.2 KLASSIFIKATION

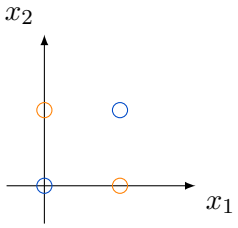
Einfache Klassifikation: Die, die links und die, die rechts von Θ liegen.



Eine Gerade als einfachste Form einer Diskriminanzfunktion (einer Punktwolke). Orthogonal zu dieser Geraden kann wieder die klassifizierende Funktion mit dem Θ an der Diskriminanzfunktion anlegen.



Kontrast: XOR – Welche lineare Funktion kann diese Klassen separieren? Keine!



Lösungsmöglichkeiten: Einteilung in Mengen – die Diagonale und der Rest.

0.3 DAS GEHIRN

0.3.1 ANATOMIE

NI_intro_bio.pdf
Folie 3

0.3.2 VORSTELLUNG DER FUNKTIONALITÄT ANFANG DES 20. JAHRHUNDERTS

NI_intro_bio.pdf
Folie 4

0.3.3 NEURONEN

0.3.3.1 EXPERIMENTELLER NACHWEIS VON EINZELNEURONEN

NI_intro_bio.pdf
Folie 6

0.3.3.2 STRUKTURELLE VIELFALT DER NEURONENTYPEN

NI_intro_bio.pdf
Folie 7

0.3.3.3 STRUKTURELLE UND FUNKTIONELLE DETAILS

NI_intro_bio.pdf
Folie 10

Wichtige Bestandteile:

- Dendrit mit Kontaktstellen (Axon von anderer Zelle)
  Kopplung durch Synapsen beschreibt man mit einem Gewichtsvektor $\underline{w}$, dessen Komponenten adaptierbar sind $\rightarrow$ lernen!
- Axon
  Jeder Eingang bildet EINE Komponente eines Eingabevektors $\underline{x}$.
- Zellkörper
  „verrechnet“ $\underline{x}$ und $\underline{w}$

Die Nervenzelle hat eine besondere Art der Zellmembran: Semipermeable Membran mit Ionen-Kanälen (Teilbild rechts oben).

0.3.4 MEDIZINISCHE VERFAHREN FÜR FUNKTIONELLE UNTERSUCHUNGEN

NI_intro_bio.pdf
Folie 8

0.3.5 HIRNAKTIVITÄT BEI INTELLEKTUELLEN AUFGABEN

NI_intro_bio.pdf
Folie 9

0.4 NEURONALE NETZE IN DER ROBOTIK

0.4.1 LOKALE FAHRZEUGNAVIGATION DURCH EXPERTEN-CLONING

NI_intro_bio.pdf
Folie 12

0.4.2 BIOLOGISCH MOTIVIERTES MODELL EINES 360 GRAD GERÄUSCH-LOKALISATIONSSYSTEMS FÜR ROBOTER

NI_intro_bio.pdf
Folie 13

EXPERIMENTELLE ERGEBNISSE

NI_intro_bio.pdf
Folie 14

1 GRUNDLAGEN

1.1 MATHEMATISCHE NOTATION

2016/NI_WS2016_Kap1_Kap2.pdf
Folie 4

1.1.1 NEURONALE BEZEICHNER

2016/NI_WS2016_Kap1_Kap2.pdf
Folie 5

w_{ij}: Sender i, Empfänger j

2016/NI_WS2016_Kap1_Kap2.pdf
Folie 6

d: Abstandsmaß (bspw. Euklidisch oder Hamming-Distanz)

1.2 NEURONENMODELL

2016/NI_WS2016_Kap1_Kap2.pdf
Folie 7

2016/NI_WS2016_Kap1_Kap2.pdf
Folie 8

1.3 NETZWERK- & LERNPARADIGMEN

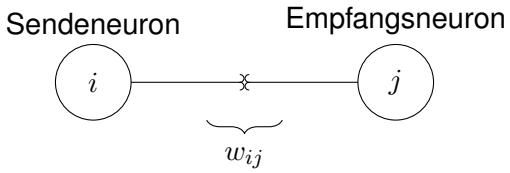
2016/NI_WS2016_Kap1_Kap2.pdf
Folie 9

1.4 LERNPARADIGMEN

1.4.1 HEBB'SCHE LERNREGELN

Korrelationslernen, unüberwacht

2016/NI_WS2016_Kap1_Kap2.pdf
Folie 10



Frage: Wann soll sich der Gewichtswert der Verbindungssynapse w_{ij} ändern?

Idee: Gewicht proportional zur Aktivität prä- und postsynaptische Seite erhöhen.

Als binäre Wertetabelle:

y_i/y_j	0	1
0	0	0
1	0	1

$\Delta w_{ij} = \mu \cdot y_i \cdot y_j$ (μ: Lernrate mit $0 < \mu \leq 1$)
$\rightarrow$ Gewicht der Synapse wird verstärkt, wenn BEIDE Neuronen aktiv sind.
$\Rightarrow$ Korrelations-LR
NACHTEIL: keine Änderung Δw_{ij} in negative Richtung möglich!

1.4.2 SOFM

Kohnen-Algorithmus, unüberwacht

2016/NI_WS2016_Kap1_Kap2.pdf
Folie 11

1.4.3 ERROR-DRIVEN

überwacht

2016/NI_WS2016_Kap1_Kap2.pdf
Folie 12

2 PRINZIPIELLE ANWENDUNGSFÄLLE VON KNNS

2.1 KLASSIFIKATION VON MUSTERN

2016/NI_WS2016_Kap1_Kap2.pdf
Folie 13

2016/NI_WS2016_Kap1_Kap2.pdf
Folie 14

Eingabedaten ($\mathbb{R}^n$) $\rightarrow$ NN $\rightarrow$ (gewünschte) Ausgabedaten ($\mathbb{R}^m$)

NN: Abbildungsfunktion... typischer Weise NICHT als geschlossenen mathematischen Ausdruck gegeben, sondern durch Beispiele $\Rightarrow$ Stützstellen

2016/NI_WS2016_Kap1_Kap2.pdf
Folie 15

2.1.1 STRUKTUR EINES ALLGEMEINEN MUSTERERKENNUNGSPROBLEMS

2016/NI_WS2016_Kap1_Kap2.pdf
Folie 16

2.2 FUNKTIONSAPPROXIMATION

2016/NI_WS2016_Kap1_Kap2.pdf
Folie 17

2016/NI_WS2016_Kap1_Kap2.pdf
Folie 18

2016/NI_WS2016_Kap1_Kap2.pdf
Folie 19

2.3 MUSTERREKONSTRUKTION / -VERVOLLSTÄNDIGUNG

2016/NI_WS2016_Kap1_Kap2.pdf
Folie 20

2.4 ZUSAMMENFASSUNG

2016/NI_WS2016_Kap1_Kap2.pdf
Folie 21

2.4.1 GENERALISIERUNG

Netzwerk soll nicht Daten auswendig lernen (und damit nur in einer festen Umwelt funktioniert), sondern generalisieren.

2016/NI_WS2016_Kap1_Kap2.pdf
Folie 22

$\Rightarrow$ mit dem Training dann aufhören, wenn E_V und E_L gleichzeitig möglichst klein sind.

2016/NI_WS2016_Kap1_Kap2.pdf
Folie 23

Überanpassung, da approximierte Abbildungsfunktion auf allen Punkten der Beispieldaten liegt.

WIEDERHOLUNG KAP. 1+2

Vorlesung
24.10.2017

FORMALES STATISCHES NEURONENMODELL

$$\underline{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \Rightarrow \underline{w}_i = \begin{pmatrix} w_{1i} \\ w_{2i} \\ \vdots \\ w_{ni} \end{pmatrix}$$

$z_i = f(\underline{x}, \underline{w}_i \ldots$ Zustand des Neurons i
$y_i = f(z_i) \ldots$ Ausgabe des Neurons i

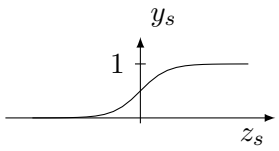
ZUSTANDSFUNKTIONEN

(a) auf Basis des Skalarprodukts
$z_i = w_{1i} \cdot x_1 + w_{2i} \cdot x_2 + \ldots + w_{ni} \cdot x_n - \Theta_i$ (Θ_i: Schwellwert)

(b) auf Basis von Abstandsfunktionen
z.B. $z_i = \sqrt{(x_1 - w_{1i})^2 + \ldots + (x_n - w_{ni})^2} = L_2$-Norm bzw. euklidischer Abstand
Norm: $L_m = \sqrt[n]{(x_1 - w_{1i})^m + \ldots + (x_n - w_{ni})^m}$
(L_1-Norm wäre City-Block-Distanz)

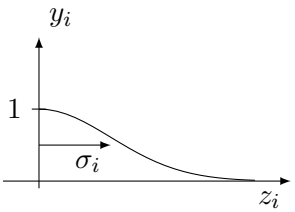
AUSGABE- BZW. TRANSFERFUNKTION

(1) $y_i = f(z_i) = \dfrac{1}{1 + e^{-z_i}} \Rightarrow$ Fermi- bzw. Signoidfunktion



(2) $y_i = f(z_i) = \operatorname{sgn}(z_i) = \begin{cases} 0 & \text{für } z_i \leq 0 \\ 1 & \text{für } z_i > 0 \end{cases}$

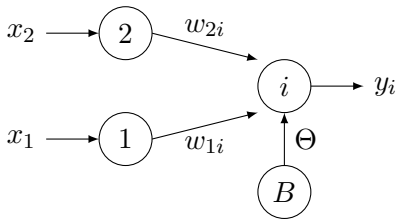
(3) $y_i = f(z_i) = \exp\left(-\dfrac{z_i^2}{2\sigma_i^2}\right)$



Funktionen (1) und (2) setzen Skalarproduktaktivierfunktion voraus, Funktion (3) eine distanzbasierte Zustandsberechunng.

BEISPIEL SKALARPRODUKTAKTIVIERFUNKTION

Welche Funktion kann nur ein Neuron mit Skalarproduktaktivierfunktion realisieren?

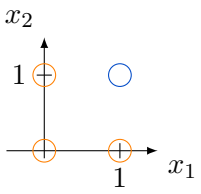


B: Bias-Neuron mit konstanter Ausgabe von 1.

$$z_i = w_{1i} \cdot x_i + w_{2i} x_2 - \Theta_i$$

$$y_i = f(z_i) = z_i$$

Beispieldaten (AND-Abbildung):



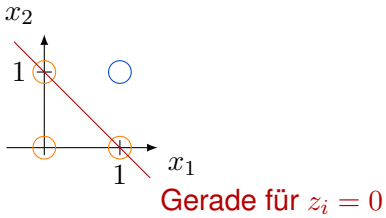
Neuron i soll für $\underline{x} = (1,1)^T$ eine Ausgabe > 0 und für $\underline{x} = (0,0)^T,\ (0,1)^T\ (1,0)^T$ Ausgabe ≥ 0 liefern.

Setze die Parameter $w_{1i},\ w_{2i}$ und Θ_i so, dass die gewünschte Funktion entsteht:
z.B.: $w_{1i} = w_{2i} = 1; \Theta = 1$

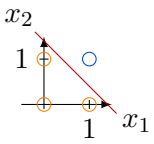
Probe:

$\underline{x}^1 = (0,0)^T$: $z_i = 1 \cdot 0 + 1 \cdot 0 - 1 = -1$ ✓
$\underline{x}^2 = (0,1)^T$: $z_i = 1 \cdot 1 + 1 \cdot 0 - 1 = \ \ 0$ ✓
$\underline{x}^3 = (1,0)^T$: $z_i = 1 \cdot 0 + 1 \cdot 1 - 1 = \ \ 0$ ✓
$\underline{x}^4 = (1,1)^T$: $z_i = 1 \cdot 1 + 1 \cdot 1 - 1 = \ \ 1$ ✓

Neuron bildet eine Lineare Diskriminanzfunktion aus ($\mathbb{R}^2$ ⇒Gerade, $\mathbb{R}^2$ ⇒Ebene, $\mathbb{R}^n$ ⇒ Hyperebene), die den Eingaberaum in zwei Hälften teilt.



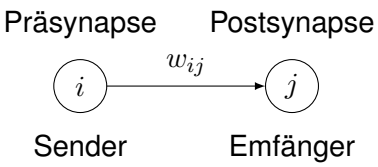
→ Neuron realisiert eine Halbraumtrennung.
mit $y_i = \dfrac{1}{1 + e^{-z_i}}$:



Offset/Schwelle (Θ) gibt Abstand zum Ursprung an.

ERMITTLUNG OPTIMALE PARAMETER

Wie ermittelt lineares Neuron sein optimalen Parameter?
Ausgangspunkt sei Hebb'sche Lernregel:



$$\Delta w_{ij} = \eta \cdot y_i \cdot y_j \quad \eta\text{: Lernrate}$$

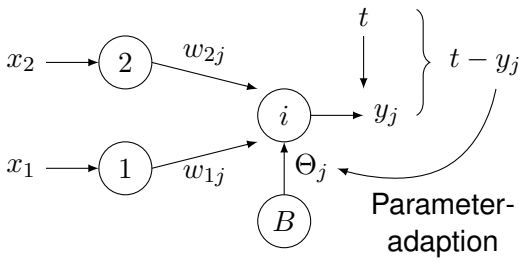
Wir erweitern diese Lernregel und ersetzen den postsynaptischen Term y_j durch einen FEHLERTERM!
Wir kennen zu jedem Eingabevektor die gewünschte Ausgabe (Zielausgabe bzw. Teachvorgabe).
Am Bsp. der AND-Abbildung:

x_1	x_2	t
0	0	0
0	1	0
1	0	0
1	1	1

Fehlerterm:
$(t - y_j) \Rightarrow$ Differenz zwischen tatsächlicher Ausgabe y_j und gewünschter Ausgabe t.
Lernregel lautet jetzt:

$$\boxed{\Delta w_{ij} = \eta \cdot y_i \cdot (t - y_j)}$$

$\rightarrow$ DELTA-LERNREGEL bzw. PERCEPTRON-LERNREGEL
$(t - y_j)$: Fehler auf Ausgabeseite



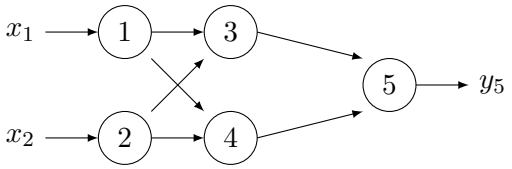
Mit dieser Lernregel durchläuft man den Trainingsdatensatz so lange, bis die Differenz zwischen t und y_i für alle Trainingsbeispiele minimal ist.

Diese Lernregel bildet die Basis für das Perceptron-Netzwerk (60er Jahre). Dies ist ein einschichtiges Netzwerk aus Sklarproduktneuronen: Beliebig viele x_n werden von beliebig vielen j_m verarbeitet.

Wie muss man sowohl das Netzwerk als auch die Lernregel erweitern, damit beliebig geformte Regionen voneinander separiert werden können?
$\Rightarrow$ vom Perceptron zum Multi-Layer-Perceptron (MLP)

BEISPIEL XOR

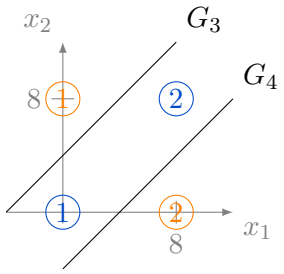
Bei Belegungen mit Anordnungen wie beim XOR:



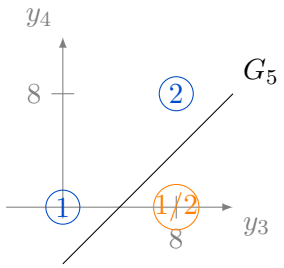
Neuronen 1, . . . , 5 sind Skalarprodukt-neuronen mit $y_i = \frac{1}{1+e^{-z_i}} \rightarrow$ stetig differenzierbare Funktion. Für unsere Darstellung der y_4-y_5-Ebene verwenden wir $y = f(z) = \begin{cases} 1 & \text{für } z > 0 \\ 0 & \text{für } z \leq 0 \end{cases}$

Wertetabelle:

x_1	x_2	t
0	0	0
0	8	1
8	0	1
8	8	0



Graphisches Ablesen/Ausrechnen der Parameter für Neuronen 3 und 4 liefert bspw.:
$w_{13} = 1; w_{23} = -1; \Theta_3 = -3$
$w_{14} = 1; w_{24} = -1; \Theta_4 = 3$
$(0,0)^T$: $z_3 = w_{13}x_1 + w_{23}x_2 - \Theta_3 = 3$; $y_3 = 1$, $z_4 = w_{14} \ldots - \Theta_4 = -3$; $y_4 = 0$
$(0,8)^T$: $z_3 = -5 \rightarrow y_3 = 0$, $z_4 = -11 \rightarrow y_4 = 0$
$(8,0)^T$: $z_4 = 12 \rightarrow y_3 = 1$, $z_4 = 5 \rightarrow y_4 = 1$
$(8,8)^T$: $z_3 = 3 \rightarrow y_3 = 1$, $z_4 = -3 \rightarrow y_4 = 0$



DELTA LERNREGEL

Vorlesung
07.11.2017

$$\Delta w_{ij} = \eta \cdot y_i \cdot (t_j - y_j)$$

$\Rightarrow$ gültig für PERCEPTOREN.
Ziel: Lernregel finden, mit der Parameter eines mehrschichtigen Netzwerks adaptiert werden können.

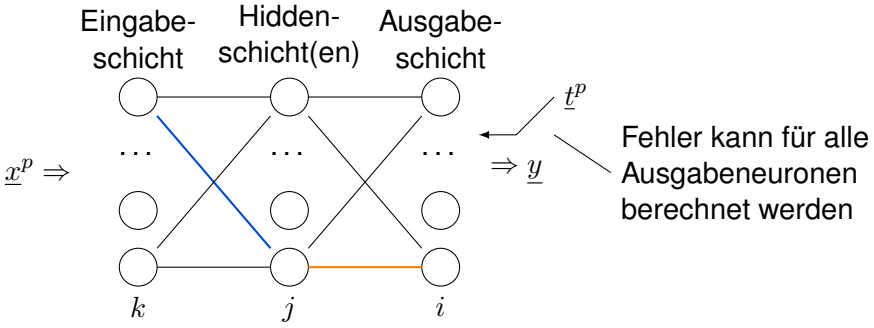
2.4.2 MEHRSCHICHTIGES NETZWERK

$\rightarrow$ MULTILAYER-PERCEPTRON (MLP)

3 MULTILAYER-PERCEPTRON (MLP) UND ERROR-BACKPROPAGATION-ALGORITHMUS

Vorlesung
07.11.2017

Aufbau:



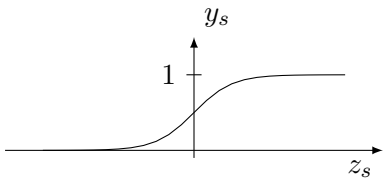
- Feed-forward-network
- vollvernetzt zwischen benachbarten Ebenen

Fehlerfunktion lautet:

$$E^p = \frac{1}{2}\sum_{i=1}^{I}(t_i^p - y_i^p)^2$$

(Quadratischer Ausgabefehler)

Für alle Neuronen, ausgenommen die Eingabeneuronen, gilt:
$z_s = \sum w_{rs} \cdot y_s \Rightarrow$ Skalarprodukt
$y_s = \dfrac{1}{1+e^{-z_s}}$



Für ausgezeichnetes Gewicht w_{ji} würde mit Hilfe der Delta-Lernregel gelten:

$$\Delta w_{ji} = \eta \cdot y_j \cdot (t_i - y_i)$$

Wie adaptiert man das Gewicht w_{kj}?!
Idee: Die Delta-Lernregel mit Hilfe eines Gradientenverfahrens verallgemeinern, so dass sich ausgehend vom Fehler am NW-Ausgang ALLE Parameter adaptieren lassen.

$\Rightarrow$ wir notieren die Fehlerfunktion als Funktion, die vom zu adaptierenden Parameter abhängt (als Argument der Funktion). Zum Beispiel:

$$E^p(\underline{x}^p, \underline{W}) = f(w_{ji})$$

$$E^p(\underline{x}^p, \underline{W}) = f(w_{kj})$$

$$E = \frac{1}{2}\sum(t_i - y_i)^2 = \ldots = f(w_{ji})$$

Für alle Trainingsmuster/-beispiele soll der Fehler (am Ausgang) minimal sein. Dafür muss eben der Wert für die einzelnen Parameter der Übergänge gesucht werden, für die die Ausgabe einen minimalen Fehler hat.

Zum Bestimmen von $f(w_{ji})$ muss folgendes beachtet werden:

$\Delta w \approx -\frac{\partial E}{\partial w}$ (zum Suchen w_{opt} in Richtung des Minimums bewegen)

$\rightarrow \Delta w_{ji} \approx -\frac{\partial E}{\partial w - ji} \qquad \Delta w_{kj} \approx -\frac{\partial E}{\partial w_{kj}}$

Wie hängt die Fehlerfunktion E von Parameter w_{ji} ab?

$\rightarrow E_i = f(y_i) \quad y_i = f(z_i) \quad z_i = f(w_{ji})$

$\Rightarrow E_i(y_i(z_i(w_{ji})))$

$\frac{\partial E_i}{\partial w_{ji}} = \frac{\partial E_i}{\partial y_i} \cdot \frac{\partial y_i}{\partial z_i} \cdot \frac{\partial z_i}{\partial w_{ji}}$ (Kettenregel)

- $\frac{\partial E_i}{\partial y_i} = \left[\frac{1}{2}(t_i - y_i)^2\right]_{y_i} = -(t_i - y_i)$
- $\frac{\partial y_i}{\partial z_i} = \left[\frac{1}{1+e^{-z_i}}\right]_{z_i} = \left[(1+e^{-z_i})^{-1}\right]_{z_i} = -e^{-z_i} \cdot -(1+e^{-z_i})^{-2}$
  $= \frac{e^{-z_i}}{(1+e^{-z_i})^2} = \frac{1}{1+e^{-z_i}} \cdot \frac{e^{-z_i}+1-1}{1+e^{z_i}} = \frac{1}{1+e^{-z_i}} \cdot \left(\frac{1+e^{-z_i}}{1+e^{z_i}} - \frac{1}{1+e^{-z_i}}\right) = y_i(1-y_i)$
- $\frac{\partial z_i}{\partial w_{ji}} = \left[y_1^H \cdot w_{1i} + y_2^H \cdot w_{2i} + \ldots + y_j^H \cdot w_{ji} + \ldots\right]_{w_{ji}} = y_j^H$

Damit gilt:

$\frac{\partial E_i}{\partial w_{ji}} = -(t_i - y_i) \cdot y_i(1-y_i) \cdot y_i$

$\boxed{\Delta w_{ji} = \eta\left(-\frac{\partial E_i}{\partial w_{ji}}\right)}$

$\boxed{\Delta w_{ji} = \eta \cdot (t_i - y_i) \cdot y_i(1-y_i) \cdot y_j}$

Für Δw_{kj} ergibt sich jetzt $E(y_i(z_i(y_j(z_j(w_{kj})))))$:

$\frac{\partial E}{\partial w_{jk}} = \frac{\partial E}{\partial y_i} \cdot \frac{\partial y_i}{\partial z_i} \cdot \frac{\partial z_i}{\partial y_j} \cdot \frac{\partial y_j}{\partial z_j} \cdot \frac{\partial z_j}{\partial w_{kj}}$

$\frac{\partial E}{\partial w_{jk}} = (-1)\sum_i (t_i - y_i) \cdot y_i(1-y_i) \cdot w_{ji} \cdot y_j(1-y_j) \cdot y_k$

Dabei sei $\boxed{\delta_i = (t_i - y_i) \cdot y_i(1-y_i)}$ der Fehler am Knoten i.

Somit ist $\frac{\partial E}{\partial w_{jk}} = (-1)\sum_i \delta_i \cdot w_{ji} \cdot y_j(1-y_j) \cdot y_k$

Dabei sei $\sum_i \delta_i \cdot w_{ji} \cdot y_j(1-y_j) = \delta_j$ der Fehler am Knoten j.

Somit ist $\Delta w_{kj} = \eta \cdot y_j(1-y_j) \cdot y_k \sum_i \delta_i \cdot w_{ji} = \eta \cdot \delta_j \cdot y_k$

$\boxed{\Delta w_{ji} = \eta \cdot \delta_i \cdot y_j}$ $\boxed{\Delta w_{kj} = \eta \cdot \delta_j \cdot y_k}$ usw.

3.1 BEISPIEL

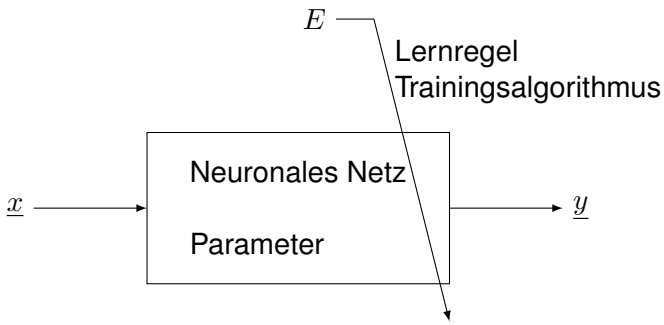
Wichtig: An Neuron wird der z-Wert zusammengerechnet. Für die folgenden Neuronen ist dann der y-Wert relevant, der sich aus dem z-Wert ergibt.

$$\begin{array}{llllll} z_5 = 0,6 \cdot 0 & +0,3 \cdot 1 & +0,1 \cdot 0 & +(-0,3) \cdot 1 & +0 & =0 \\ z_6 = 0,1 \cdot 0 & +0,1 \cdot 1 & +0,1 \cdot 0 & +0,1 \cdot 1 & -0,2 & =0 \\ z_7 = 0,4 \cdot 0 & +(-0,1) \cdot 1 & +0,1 \cdot 0 & +(-0,2) \cdot 1 & +0,3 & =0 \end{array}$$

Vorlesung
14.11.2017

Wichtig: Eingangsvariable $\underline{x}^p$ und Vorgabe $\underline{t}^p$.
Ergebnis im Vergleich zur Vorgabe:

$$\underline{y}^p = \begin{pmatrix} y_8 \\ y_9 \end{pmatrix} = \begin{pmatrix} 0,5 \\ 0,5 \end{pmatrix} \Leftrightarrow \underline{t}^p = \begin{pmatrix} t_8 \\ t_9 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$



$$\Delta w(t) = \eta \cdot \left(-\frac{\partial E}{\partial w} \right)$$

$$w(t+1) = w(t) + \Delta w(t)$$

Beispielweg von einem Eingang zum Ausgang:
$\Delta w_{58} = \eta \cdot \delta_8 \cdot y_5$ mit $\delta_8 = \underbrace{(t_8 - y_8)}_{F'(y_8)\cdot(-1)} \cdot \underbrace{y_8 \cdot (1 - y_8)}_{F'(z_8)}$
$\Delta w_{15} = \eta \cdot \delta_5 \cdot y_1$ mit $\delta_5 = (w_{58} \cdot \delta_8 + w_{59} \cdot \delta_9) \cdot y_5 \cdot (1 - y_5)$
Damit lässt sich berechnen:

3.1.1 ERSTER TRAININGSSCHRITT

Gesucht: $\Delta\underline{w}_8$, $\Delta\underline{w}_9$, $\Delta\underline{w}_5$, $\Delta\underline{w}_6$, $\Delta\underline{w}_7$ (Training)

- $\Delta\underline{w}_8 = (\Delta w_{B8},\ \Delta w_{58},\ \Delta w_{68},\ \Delta w_{78})^T$
  - $\Delta w_{B8} = 0,1 \cdot (0 - 0,5) \cdot 0,5 \cdot (1 - 0,5) \cdot 1 = -0,0125$
  - $\Delta w_{58} = 0,1 \cdot \underbrace{(0 - 0,5) \cdot 0,5 \cdot (1 - 0,5)}_{\delta_8 = -0,125} \cdot 0,5 = -0,00625$
  - $\Delta w_{68} = \Delta w_{58}$
  - $\Delta w_{78} = \Delta w_{58}$
- $\Delta\underline{w}_9 = (\Delta w_{B9},\ \Delta w_{59},\ \Delta w_{69},\ \Delta w_{79})^T$
  - $\Delta w_{B9} = 0,0125$
  - $\Delta w_{59} = 0,1 \cdot \underbrace{(1 - 0,5) \cdot 0,5 \cdot (1 - 0,5)}_{\delta_9 = 0,125} \cdot 0,5 = 0,00625$
  - $\Delta w_{69} = \Delta w_{59}$

  - $\Delta w_{79} = \Delta w_{59}$

- $\Delta \underline{w}_5 = (w_{B5},\ w_{15},\ w_{25},\ w_{35},\ w_{45})^T$
  - $\Delta w_{B5} = 0,1 \cdot \underbrace{[(-0,6) \cdot (-0,125) + (-0,6) \cdot 0,125]}_{=0} \cdot 0,5 \cdot (1-0,5) \cdot 0 = 0$
  - $\Delta w_{15} = 0$
  - $\Delta w_{25} = 0$
  - $\Delta w_{35} = 0$
  - $\Delta w_{45} = 0$
- $\Delta \underline{w}_6 = (w_{B6},\ w_{16},\ w_{26},\ w_{36},\ w_{46})^T$
  - $\Delta w_{B6} = 0,1 \cdot [0,2 \cdot (-0,125) + (-0,4) \cdot 0,125] \cdot 0,5 \cdot (1-0,5) \cdot 1 = -0,001875$
  - $\Delta w_{16} = 0$
  - $\Delta w_{26} = \Delta w_{B6}$
  - $\Delta w_{36} = 0$
  - $\Delta w_{46} = \Delta w_{B6}$
- $\Delta \underline{w}_7 = (w_{B7},\ w_{17},\ w_{27},\ w_{37},\ w_{47})^T$
  - $\Delta w_{B7} = 0,1 \cdot [(-0,6) \cdot (-0,125) + (-0,4) \cdot 0,125] \cdot 0,5 \cdot (1-0,5) \cdot 1 = -0,000625$
  - $\Delta w_{17} = 0$
  - $\Delta w_{27} = \Delta w_{B7}$
  - $\Delta w_{37} = 0$
  - $\Delta w_{47} = \Delta w_{B7}$

Optionen beim Korrigieren: Entweder Änderungen direkt drauf rechnen (direct learning), oder als zusätzlichen „Delta-Bias“ in einer Variable speichern und dann drauf rechnen (batch learning).

Warum initialisieren wir die Parameter mit ZUFÄLLIGEN Werten?

- Damit die Änderungen möglichst nicht klein werden.

Was passiert, wenn alle Parameter mit Null initialisiert werden?

- $z_5 = z_6 = z_7 = 0$
- $y_5 = y_6 = y_7 = 0.5$
- $z_8 = z_9 = 0$
- $y_8 = y_9 = 0.5$

$\rightarrow$ die y-Werte bleiben gleich.
Beim Training:
1.Schritt:

- $\Delta \underline{w}_8$ und $\Delta \underline{w}_9$ bleiben wie bei der zufälligen Verteilung.
- $\Delta \underline{w}_5 = \Delta \underline{w}_6 = \underline{w}_7$ ist dann $\underline{0}$.

Damit:

- $\underline{w}_8(t+1) = \underline{w}_8(t) + \Delta w_8(t) = (-0,0125, -0,00625, \ldots)^T$
- $\underline{w}_9(t+1) = \underline{w}_9(t) + \Delta w_9(t) = (0,0125, 0,00625, \ldots)^T$

- $\underline{w}_4(t+1) = \underline{w}_4(t)$, gleiches für $\underline{w}_5$ und $\underline{w}_6$

Und somit:

- $z_5 = z_6 = z_7 = 0$
- $y_5 = y_6 = y_7 = 0.5$
- $z_8 = -0.0125 \cdot 1 + 3 \cdot ((-0.00625) \cdot 0.5) = -0.021875$
- $z_9 = 0.0125 \cdot 1 + 3 \cdot (0.00625 \cdot 0.5) = 0.021875$
- $y_8 = 0.5055$
- $y_9 = 0.4945$

2. Schritt:

- $\Delta\underline{w}_8 = (-0.0124, -0.0062, -0.0062, \ldots)^T$
- $\Delta\underline{w}_9 = (0.0124, 0.0062, 0.0062, \ldots)^T$
- $\Delta\underline{w}_5 = \Delta\underline{w}_6 = \Delta\underline{w}_7 = (0.00004, 0, 0.00004, 0, 0.00004)^T$

Damit:

- $\underline{w}_5(t+2) = \underline{w}_6(t+2) = \underline{w}_7(t+2) = (0.00004, 0, 0.00004, 0, 0.00004)$

$\rightarrow$ alle Hidden-Knoten sind gleich $\rightarrow$ man könnte die Hidden-Knoten auch auf einen reduzieren. Damit kann aber bspw. nur eine Gerade abgebildet werden (Vergleich XOR – braucht 2 Geraden) (vgl. Symmetriebruch).
$\Rightarrow$ man braucht unterschiedliche/zufällige Initialwerte, damit UNTERSCHIEDLICHE Geraden angelernt werden können!

3.2 MULTI LAYERED PERCEPTRON

NI_WS2017_Kap3_MLP.pdf
Folie 1

3.3 GRUNDPRINZIP TRAINING

NI_WS2017_Kap3_MLP.pdf
Folie 2

3.4 ERROR-BACKTRACKING-ALGORITHMUS

NI_WS2017_Kap3_MLP.pdf
Folie 3

3.4.1 GRADIENTANSTIEG

NI_WS2017_Kap3_MLP.pdf
Folie 4

3.4.2 RÜCKPROPAGIERUNG DES FEHLERS AM NETZWERK-AUSGANG

NI_WS2017_Kap3_MLP.pdf
Folie 5

NI_WS2017_Kap3_MLP.pdf
Folie 6

3.4.3 ADAPTIONSVORSCHRIFTEN

NI_WS2017_Kap3_MLP.pdf
Folie 7

3.4.4 FAZIT

NI_WS2017_Kap3_MLP.pdf
Folie 8

4 ERWEITERUNG ERROR-BACKPROPAGATION-ALGORITHMUS

Vorlesung
21.11.2017

4.1 PROBLEME DES EBP-ALGORITHMUS

4.1.1 RESULTIEREND AUS GRADIENTENABSTIEG

NI_WS2016_Kap4_BP_ext.pdf
Folie 4

Kommt aus lokalem Minimum nicht mehr raus, weil dort der Gradient 0 ist $\Rightarrow$ es kann keine Änderung mehr statt finden. Einzige Lösung: neu Initialisieren bzw. willkürlicher Schritt.

NI_WS2016_Kap4_BP_ext.pdf
Folie 5

NI_WS2016_Kap4_BP_ext.pdf
Folie 6

NI_WS2016_Kap4_BP_ext.pdf
Folie 7

4.1.2 RESULTIEREND AUS NETZWERK-TOPOLOGIE

NI_WS2016_Kap4_BP_ext.pdf
Folie 8

Sehr weit außerhalb des Arbeitsbereichs bekommt man einen kleinen Gradienten ($\widehat{=}$ Adaptionsgeschwindigkeit gering)

NI_WS2016_Kap4_BP_ext.pdf
Folie 9

4.2 TRAINING MIT MOMENTUM-TERM

NI_WS2016_Kap4_BP_ext.pdf
Folie 10

NI_WS2016_Kap4_BP_ext.pdf
Folie 11

„Langsamer ins Tal gleiten, Schwung von starkem Abstieg ins Plateau mitnehmen."

4.3 MODIFIKATION DER ABLEITUNG DER TRANSFERFUNKTION

NI_WS2016_Kap4_BP_ext.pdf
Folie 12

Schrittgröße hat ein festes Minimum, Gradient gibt dann nur die Richtung an.

4.4 TRAINING MIT WEIGHT-DECAY

NI_WS2016_Kap4_BP_ext.pdf
Folie 13

NI_WS2016_Kap4_BP_ext.pdf
Folie 14

NI_WS2016_Kap4_BP_ext.pdf
Folie 15

NI_WS2016_Kap4_BP_ext.pdf
Folie 16

4.5 MANHATTEN-TRAINING

NI_WS2016_Kap4_BP_ext.pdf
Folie 17

NI_WS2016_Kap4_BP_ext.pdf
Folie 18

$\Rightarrow$ wie Modifikation der Ableitung der Transferfunktion, nur für beide Richtungen.

4.6 EIGENE SCHRITTWEITE FÜR JEDEN PARAMETER

NI_WS2016_Kap4_BP_ext.pdf
Folie 19

NI_WS2016_Kap4_BP_ext.pdf
Folie 20

4.7 RESILIENT BACKPROPAGATION (RPROP)

NI_WS2016_Kap4_BP_ext.pdf
Folie 24

NI_WS2016_Kap4_BP_ext.pdf
Folie 25

NI_WS2016_Kap4_BP_ext.pdf
Folie 26

NI_WS2016_Kap4_BP_ext.pdf
Folie 27

NI_WS2016_Kap4_BP_ext.pdf
Folie 28

4.8 QUICKPROPAGATION (QUICKPROP)

NI_WS2016_Kap4_BP_ext.pdf
Folie 21

NI_WS2016_Kap4_BP_ext.pdf
Folie 22

NI_WS2016_Kap4_BP_ext.pdf
Folie 23

4.9 FAZIT

NI_WS2016_Kap4_BP_ext.pdf
Folie 29

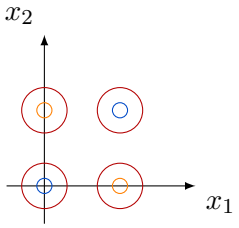
NI_WS2016_Kap4_BP_ext.pdf
Folie 30

5 RADIAL BASIS FUNCTION-NETZWERK (RBF-NETZWERK)

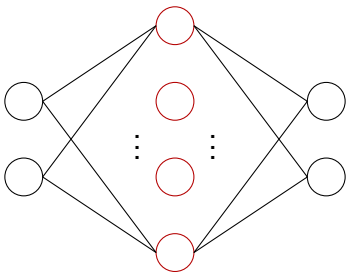
5.1 MOTIVATION

NI_WS2016_Kap5_RBF.pdf
Folie 1

Datenpunkte als Prototypen der Abbildung → diese Umkreisen.



Kombiniere die Ausgaben des Netzwerks aus den Ausgaben der Prototyp- bzw. Referenz-Neuronen.
Jedes Neuron der Hidden-Schicht bildet einen (Radius von einem) Datenpunkt dar:



Skalarprodukt:

$$z = \underline{w}^T \cdot \underline{w}$$

Distanzbasierte Aktionsgerade:

$$z = f(\underline{x}, \underline{w})$$

→ Vektornormen
→ L_2-Norm
→ $\boxed{z = \sqrt{(x_1 - w_1)^2 + \ldots + (x_n - w_n)^2}}$ (Euklidischer Abstand) ... z als Abstand zwischen $\underline{x}$ und $\underline{w}$.

$$y = f(z) \approx e^{-z}$$

$\boxed{f = \exp\left(-\frac{z^2}{2\sigma^2}\right)}$ Gaußfunktion

NI_WS2016_Kap5_RBF.pdf
Folie 2

5.2 TOPOLOGIE

NI_WS2016_Kap5_RBF.pdf
Folie 3

5.3 BERECHNUNGSVORSCHRIFTEN

NI_WS2016_Kap5_RBF.pdf
Folie 4

NI_WS2016_Kap5_RBF.pdf
Folie 5

5.4 TRAININGSREGIME

NI_WS2016_Kap5_RBF.pdf
Folie 6

5.4.1 BESTIMMUNG DER GEWICHTE VON DER EINGANGS- ZUR RBF-SCHICHT

NI_WS2016_Kap5_RBF.pdf
Folie 7

NI_WS2016_Kap5_RBF.pdf
Folie 8

5.4.2 BESTIMMUNG DER VARIANZEN

NI_WS2016_Kap5_RBF.pdf
Folie 9

NI_WS2016_Kap5_RBF.pdf
Folie 10

5.4.3 BESTIMMUNG DER GEWICHTE VON DER RBF- ZUR AUSGABESCHICHT

NI_WS2016_Kap5_RBF.pdf
Folie 11

$\underline{A} = f(\underline{X}, \underline{W}, \underline{\sigma})$

NI_WS2016_Kap5_RBF.pdf
Folie 12

Achtung bei Inverser: Elemente dürfen nicht linear abhängig sein!

5.4.4 ITERATIVES NACHTRAINING

NI_WS2016_Kap5_RBF.pdf
Folie 13

5.4.5 WEITERE MATHEMATISCHE BETRACHTUNGEN

NI_WS2016_Kap5_RBF.pdf
Folie 14

5.5 NOTIZEN

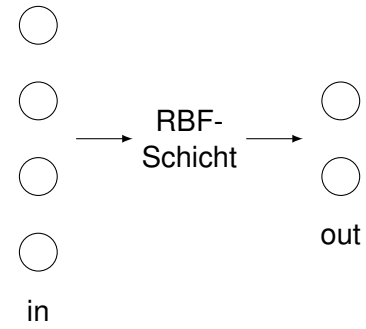
Vorlesung
05.12.2017

- Genauigkeit des Netzwerks ist geringer, je weiter die Trainingspunkte auseinander liegen (je kleiner damit die Einzugsbereiche werden).
- Gelerntes Netzwerk bildet nur Daten in der Nähe der gelernten Punkte ab. Das kann bspw. gut sein, wenn die Frage ist, wann der Arbeitsbereich verlassen wird.
- Bei RBF gibt es immer nur eine Hidden-Schicht!

5.6 BERECHNUNGSBEISPIEL

5.6.1 GEGEBEN

Abbildung von 4 Merkmalen/Eingängen auf 2 Klassen.



$$\underline{x}^1 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} \quad \underline{x}^2 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \quad \underline{x}^3 = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} \quad \underline{t}^1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad \underline{t}^2 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \underline{t}^3 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

5.6.2 BESTIMMUNG DER RBF-ZENTREN

Einfacher Ansatz: für jeden Trainingspunkt ein Zentrum.
Besser: Ansatz mit möglichst geringem Quantisierungsfehler.

5.6.2.1 QUANTISIERUNGSFEHLER

Quantisierung: Einteilung
$\Rightarrow$ Quantisierung eines Vektorraumes durch m Referenz- bzw. Codebook-Vektoren.
$\Rightarrow$ Bestimme $\forall \underline{x} \in X$ die Differenz zwischen dem am besten passenden Referenzvektor und $\underline{x}$.
Referenzvektoren mit Varianz $\sigma^2 \forall$ Referenzvektoren identische Varianz angenommen:
Jeder Punkt hat Gaußsche Verteilung um sich. Damit ist das am besten passende (Best-Matching-Neuron) das, welches am nahsten am Punkt dran liegt. Die Differenz ist somit die Distanz (i.d.R. euklidischer Abstand) zwischen dem am besten passenden Neuron und dem betrachteten Punkt.
Mit $BM = (w_1, w_2)^T$ und betrachteter Punkt $\underline{x} = (x_1, x_2)^T$ ist die Differenz:

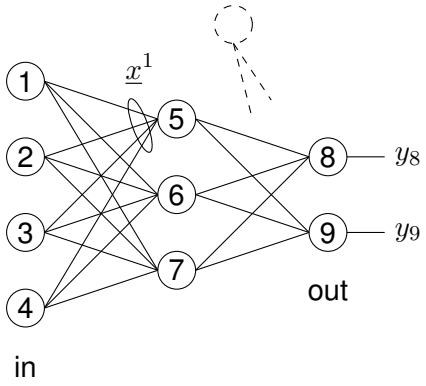
$$d(\underline{x}, BM) = \sqrt{(x_1 - w_1)^2 + (x_2 - w_2)^2}$$

Der Einzugsbereich eines Neuron ist durch den halben Abstand zu seinen Nachbarneuronen begrenzt (Delanney-Triangulation). Den daraus erhaltenen Einzugsbereich nennt man auch Voronoi-Region.

5.6.2.2 BESTIMMUNG

- Definiere einen maximalen Quantisierungsfehler $d^2_{max} = 1$, der nicht überschritten werden darf.
  $\rightarrow d^2(\underline{x}, \underline{w}^{I,RBF}) \leq 1 \; \forall x$
- Ordne $\underline{w}_5^{I,RBF}$ Vektor $\underline{x}^1$ zu:
  $$\underline{w}_5^{I,RBF} = (0, 1, 0, 1)^T$$
- Prüfe, ob $\underline{x}^2$ und $\underline{x}^3$ durch Referenzvektor $\underline{w}_5^{I,RBF}$ abgedeckt sind:
  $d(\underline{x}^2, \underline{w}_5^{I,RBF}) = 3 \quad > d_{max}$ ✓
  $d(\underline{x}^3, \underline{w}_5^{I,RBF}) = 2 \quad > d_{max}$ ✓
- Ordne $\underline{w}_6^{I,RBF}$ Vektor $\underline{x}^2$ zu:
  $$\underline{w}_6^{I,RBF} = (0, 0, 1, 0)^T$$
- Prüfe, ob $\underline{x}^3$ durch $\underline{w}_6^{I,RBF}$ abgedeckt ist:
  $d(\underline{x}^3, \underline{w}_6^{I,RBF}) = 3 \quad > d_{max}$ ✓
- Ordne $\underline{w}_7^{I,RBF}$ Vektor $\underline{x}^3$ zu:
  $$\underline{w}_7^{I,RBF} = (1, 1, 0, 0)^T$$

Damit:



5.6.3 AUSGABEVEKTOR

Weiterhin gegeben:

$$\underline{w}_8^{RBF,o} = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \quad \underline{w}_9^{RBF,o} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \quad \sigma^2 = 0.5$$

Gesucht: $\underline{y}^o = \begin{pmatrix} y_8 \\ y_9 \end{pmatrix}$ für Eingabevektor $\underline{x}^1$

Rechnung: (siehe Abschnitt 5.3)

- $z_5 = \sqrt{(x_1^1 - w_{15})^2 + \ldots + (x_4^1 - w_{45})^2} = 0$
- $y_5 = \exp\left(-\frac{z_5^2}{2\sigma^2}\right) = 1$
- $z_6 = \sqrt{(x_1^1 - w_{16})^2 + \ldots + (x_4^1 - w_{46})^2} = \sqrt{3}$
- $y_6 = e^{-3}$
- $z_7 = \sqrt{(x_1^1 - w_{17})^2 + \ldots + (x_4^1 - w_{47})^2} = \sqrt{2}$
- $y_7 = e^{-2}$
- $y_{RBF} = \left(1, e^{-3}, e^{-2}\right)^T$

Damit:

- $y_8 = z_8 = \underline{w}_8^T \cdot \underline{y}_{RBF} = e^{-3} + e^{-2}$
- $y_9 = z_9 = \underline{w}_9^T \cdot \underline{y}_{RBF} = e^{-3}$

5.6.4 LERNSCHRITT

Mittels Delta-Lernregel ($\eta = 0,3$) einen Lernschritt für das Gewicht w_{58} ausführen. Gesucht: Δw_{58}

6 LEARNING VECTOR QUANTIZATION (LVQ)

6.1 MOTIVATION

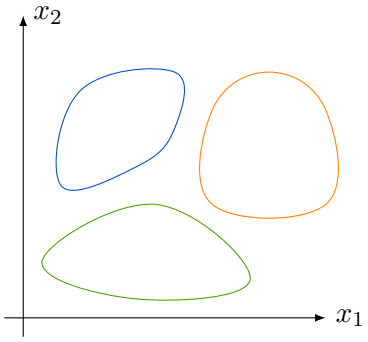
Vorlesung
09.01.2018

Ziel Repräsentation der Eingabedaten erzeugen, die

- sowohl deren Statistik
- als auch deren Klassenzugehörigkeit

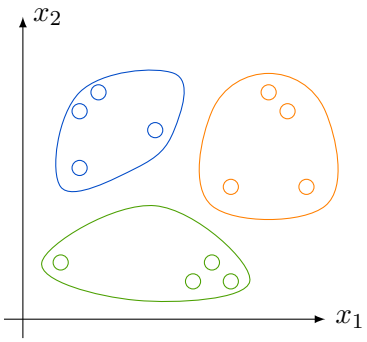
berücksichtigt.

Vorgehen Anzahl M Referenzvektoren sollen eine Anzahl P Datenpunkte repräsentieren/abbilden:



(3 Klassen)

- Definiere eine feste Anzahl von Referenzvektoren pro Klasse.
- Initialisiere diese mit zufällig ausgewählten Beispielen der jeweiligen Klassen. $\rightarrow$ bspw. 4 pro Klasse:



Lernverfahren

- Versuche, die Lage der Referenzvektoren zu optimieren.
- Nach Optimierung ist die Klassifikation von unbekannten Eingangsdaten möglich:
  Das Klassenattribut des BEST-MATCHING-NEURONS (-Referenzvektors) wird dem Eingangsdatum $\underline{x}$ zugeordnet (Zuweisung zum Referenzvektor, der am nächsten zum Eingangsvektor liegt).

6.2 OPTIMIERVERFAHREN

Für LVQ existieren VIELE unterschiedliche Varianten.

6.2.1 LVQ 1

In jedem Adaptionsschritt wird NUR das Best-Matching-Neuron (im Bezug zu der Menge der Beispiel- bzw.Trainingsdaten) angepasst.
Format der Trainingsdaten: $\underline{x}^n$ Label k ($k \in$ mögliche Klassen [in Beispiel $1, 2, 3$])

LERNALGORITHMUS

1. Wähle ein $\underline{x}$ (aus Trainingsdaten).
2. Bestimme Best-Matching-Neuron b nach Lernvorschrift:

   $$\underline{w}_b = \underset{i=1...P}{\operatorname{argmin}} \|\underline{x} - \underline{w}_i\|$$

   Sowohl $\underline{x}$ als auch $\underline{w}_b$ verfügen über ihr jeweiliges Klassenlabel/-attribut.
3. Zwei Fälle abhängig davon, ob die Klassenattribute gleich sind oder nicht:
   - Falls $\underline{x}$ und $\underline{w}_b$ gleiches Klassenattribut aufweisen:

     $$\underline{w}_b(t+1) = \underline{w}_b(t) + \eta(t)\,[\underline{x}(t) - \underline{w}_b(t)]$$

     $\Rightarrow$ Verschiebung von $\underline{w}_b$ in Richtung $\underline{x}$.
   - Falls $\underline{x}$ und $\underline{w}_b$ verschiedene Klassenattribute aufweisen:

     $$\underline{w}_b(t+1) = \underline{w}_b(t) - \eta(t)\,[\underline{x}(t) - \underline{w}_b(t)]$$

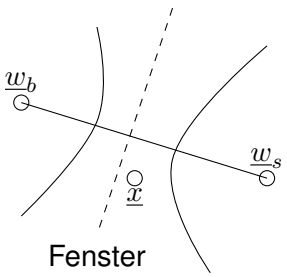
     $\Rightarrow$ Verschiebung von $\underline{w}_b$ von $\underline{x}$ weg.

   Zeitabhängige Lernrate $\eta(t)$ ist mit $\eta(t+1) = \eta(t) \cdot \alpha$ und $\eta(t=0) \approx 0.5$ definiert. Dabei liegt $0 < \alpha \leq 1$, typische Größen sind $0.9,\ 0.99$ usw.
   Durch die zeitabhängige Lernrate „friert“ die Verteilung irgendwann ein (und stellt somit den optimalen Zustand dar).
   Wichtig: t der Lernrate meint die gesamte Trainingsepoche (ein Mal mit allen Trainingsdaten trainiert).
   Anmerkung: $\alpha = 1$ wäre nur sinnvoll, wenn die Lernrate $\eta(t=0)$ sehr klein ist.

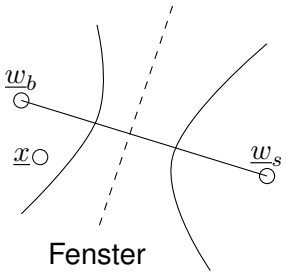
6.2.2 LVQ 2.1

Im Unterschied zu LVQ 1:

- Nutzt sowohl Best-Matching-Neuron b a-s auch Second-Best-Matching-Neuron s.
- $\underline{x}$ muss in einem „Fenster“ bzw. zwischen $\underline{w}_b$ und $\underline{w}_s$ liegen:



Falls $\underline{x}$ nicht in Fenster liegt, erfolgt KEINE Adaption:



$\underline{w}_b$ und $\underline{w}_s$ müssen verschiedene Klassen repräsentieren!

Fenster: Wird aus Relationen zwischen den Abständen $d_b = \|\underline{x} - \underline{w}_b\|$ und $d_s = \|\underline{x} - \underline{w}_s\|$ berechnet. Mit $0.2 \leq v \leq 0.3$ muss $\min\left(\frac{d_b}{d_s}, \frac{d_s}{d_b}\right) > \frac{1-v}{1+v}$ sein, damit $\underline{x}$ im Fenster liegt.

LERNALGORITHMUS

1. Wähle ein $\underline{x}$.
2. Bestimme Best- und Second-Best-Matching-Neuronen.
3. Prüfe, ob $\underline{x}$ im Fenster liegt. Falls $\underline{x}$ im Fenster, dann:
   (a) $\mathrm{Klasse}(\underline{w}_b) = \mathrm{Klasse}(\underline{x}(t))$

$$\underline{w}_b(t+1) = \underline{w}_b(t) + \eta(t) \cdot [\underline{x}(t) - \underline{w}_b(t)]$$

$$\underline{w}_s(t+1) = \underline{w}_s(t) - \eta(t) \cdot [\underline{x}(t) - \underline{w}_s(t)]$$

   (b) $\mathrm{Klasse}(\underline{w}_s) = \mathrm{Klasse}(\underline{x}(t))$

$$\underline{w}_b(t+1) = \underline{w}_b(t) - \eta(t) \cdot [\underline{x}(t) - \underline{w}_b(t)]$$

$$\underline{w}_s(t+1) = \underline{w}_s(t) + \eta(t) \cdot [\underline{x}(t) - \underline{w}_s(t)]$$

LVQ 2.1 adaptiert ausschließlich die Referenzvektoren, die entlang der Klassengrenzen liegen. $\Rightarrow$ Die Referenzvektoren, die „innerhalb“ der Klassengebiete liegen, bleiben unverändert. $\Rightarrow$ Gegebenenfalls schlechte Generalisierung innerhalb der Klassengebiete.

6.2.3 LVQ 3

Arbeitet analog zu LVG 2.1 und lässt zusätzlich den Fall zu, dass $\mathrm{Klasse}(\underline{w}_b) = \mathrm{Klasse}(\underline{w}_s) = \mathrm{Klasse}(\underline{x}(t))$ gilt. Für diesen Fall werden $\underline{w}_b$ und $\underline{w}_s$ adaptiert mit:

$$\underline{w}_{b/s}(t+1) = \underline{w}_{b/s}(t) + \eta(t) \cdot \varepsilon \cdot \left[\underline{x}(t) - \underline{w}_{b/s}(t)\right]$$

Dabei ist $\varepsilon \approx 0.5$.
Damit kann auch innerhalb der Klassen adaptiert werden. Somit ist die Repräsentation der Statistik der Klassengebiete möglich.

6.2.4 OPTIMIZED LVQ (OLVQ)

Ziel ist, dass jedes Datum der Trainingsdaten den gleichen Einfluss auf die Lage des Referenzvektors hat. Dazu führt man für jeden Referenzvektor eine eigene Schrittweite, die dies sicherstellt.

6.2.5 HANDHABUNG DER LVQ-VERFAHREN

Die Performance/Güte eines LVQ-Netzwerks hängt von folgenden Faktoren ab:

(a) Zahl der Referenzvektoren pro Klasse

(b) Initialisierung der Referenzvektoren

(c) Verwendeten LVQ-Verfahren

(d) Zeitlichem Verlauf der Lernrate

(e) Einem geeigneten Abbruchkriterium

Als „best praxis“ Vorgehen zeit sich oft:

1. Mit OLVQ-Training beginnen.
2. Mit LVQ 1 bzw. LVQ 3 nachtrainieren.

7 UNÜBERWACHTE LERNVERFAHREN

7.1 ABGRENZUNG ÜBERWACHTE LERNVERFAHREN

- MLP/RBF $\rightarrow$ supervised learning
  $\underline{y}$ und $\underline{t}$ (Teacher) steuern das Neuronale Netzwerk.
- LVQ $\rightarrow$ semi-supervised learning
  Nutzen zwar Klassenzugehörigkeit der Beispieldaten aber geben keinen Teacher vor.
- Unüberwachte Lernverfahren: unsupervised learning
  - Ausschließlich auf Basis der Eingabedaten (absolut kein Teacher mehr).
  - KOMPETITIVES LERNVERFAHREN: Neuronen konkurrieren um die Teilnahme am Adaptionsprozess.
  - Vektorquantisierung.

7.2 NEURAL-GAS-NETZWERK (NG)

Gas $\rightarrow$ Bezüge aus Thermodynamik und Molekülbewegungen.
Ziel: Anhand von P Eingabedaten eine Menge von M Referenzvektoren so verteilen, dass ein optimaler Vektorquantisierer entsteht (typischer Weise $P \gg M$).

7.2.1 OPTIMALE VEKTORQUANTISIERER

Zum Bestimmen eines optimalen Vektorquantisierers wird ein Fehlermaß benötigt (Funktion). Der optimale Vektorquantisierer hat dann ein minimales Fehlermaß.

Quantisierungsfehler $E = \sum_{p=1}^{P} \|\underline{x}^p - \underline{w}_b(\underline{x}^p)\|$ (Abstand der Punkte vom Best-Matching-Neuron)

Man kann nachweisen, dass der NG-Algorithmus Optimalität sicherstellt.

7.2.2 ALGORITHMUS

Vorlesung
23.01.2018

1. Wähle ein $\underline{x}$ aus der Menge der (gleichverteilten) Eingangsdaten.
2. Sortiere alle Referenzvektoren ihres euklidischen Abstandes zu $\underline{x}$.
   $\Rightarrow$ Ranking aller Referenzvektoren

Ranking-Faktor: k_i, Lernreichweite: $h_\lambda(k_i)$.
Reichweite wird mit zunehmender Trainingsdauer kleiner. Das Best-Matching-Neuron hat die größte Adaption, weitere Neuronen eine geringere. Die Reichweite beschreibt dann, welche Neuronen beim Training noch (wie stark) einbezogen werden.

$$\Delta\underline{w}_i = \eta(t) \cdot h_\lambda(k_i)(\underline{x} - \underline{w}_i)$$

$$\lambda(t) = \lambda_A \cdot \left(\frac{\lambda_E}{\lambda_A}\right)^{\frac{t}{t_{max}}}$$

$$h_\lambda(k_i) = e^{\frac{-k_i}{\lambda(t)}}$$

$$\eta(t) = \eta_A \cdot \left(\frac{\eta_E}{\eta_A}\right)^{\frac{t}{t_{max}}}$$

Mit $\bullet_A$: Initialwert (Anfang), $\bullet_E$: Endwert, t: Trainingsepoche
Anmerkung: auf den Folien $\eta = \varepsilon$.

7.3 KOHONEN-NETZWERKE

- Vektorquantisierer
- benannt noch „Erfinder“
- nutzt ncahbarschaft innerhalb der Netzwerk-Topologie
- trainiert wird jeweils das Best-Matching-Neuron $\underline{r}'$ (Koordinate im Netzwerk) mit

  $$\underline{r}' = \underset{\underline{r}}{\operatorname{argmin}}(\underline{x} - \underline{w}_{\underline{r}})$$

  und dessen Nachbarn in der TOPOLOGIE $\Rightarrow$ „Dimension“ des Netzwerks

Beispiel „Dimension“/Topologie:

- 1D $\rightarrow$ Kette (von Neuronen)
- 1,5D $\rightarrow$ Ring
- 2D $\rightarrow$ Karte/Grid/Gitter

Definition Nachbarschaftsfunktionen:

$$h_{\underline{r}\underline{r}'}(t) = \exp(-\|\underline{r}' - \underline{r}\|)$$

Reichweite nimmt mit zunehmender Trainingsdauer ab.
$r(t) = r(t-1) \cdot \alpha$ mit $0 < \alpha < 1$, z.B. $\alpha = 0,99$
$r(t=0) = 10$ (als Richtwert)
$\eta(t)$: Analog zu Neural-Gas-Network oder einfacher $\eta(t) = \eta(0-1) \cdot \alpha$
$\eta(t=0) = 0,8$ als Richtwert

$$\Delta\underline{w}_{\underline{r}} = \eta(t) \cdot h_{\underline{r}\underline{r}'}(t) \cdot (\underline{x} - \underline{w}_r)$$

Definition der Nachbarschaft (Dimension der Topologie) führt zur Eigenschaft der topologieerhaltenden Abbildung (bzw.: Nachbarschaften im Eingaberaum werden auf Nachbarschaften in der neuronalen Repräsentation abgebildet – benachbarte Punkte bleiben benachbart).
Dies setzt voraus, dass die (Unter-)Mannigfaltigkeit (Dimensionen) der Eingabedaten mit der Dimension der Netzwerk-Topologie übereinstimmt. Sonst topologische Defekte zu erwarten (topologischer Defekt: Nachbarschaft in der neuronalen Repräsentation repräsentiert keine Nachbarschaft im Eingaberaum, bzw. umgekehrt). Beispiel Defekt: Kette (1D) in einer 2-D Ebene. Dort ist aber Beginn und Ende der Kette ggf. Nebeneinander, was nicht dem tatsächlichen Abstand entspricht.

Das Kohonen-Modell hat den stärksten biologischen Background.

7.4 GROWING-NEURAL-GAS

$\Rightarrow$ Netzwerk bestimmt selbst:

- Lage der Referenzvektoren
- Anzahl der Referenzvektoren