Vorlesung Computational Intelligence:

Teil 3: Künstliche Neuronale Netze

Lernverfahren (Fortsetzung),
Multi-Layer-Perceptron-Netze (MLP-Netze)

Ralf Mikut, Wilfried Jakob, Markus Reischl
Karlsruher Institut für Technologie, Institut für Angewandte Informatik
E-Mail: ralf.mikut@kit.edu, wilfried.jakob@kit.edu

jeden Donnerstag 14:00-15:30 Uhr, Nusselt-Hörsaal

Gliederung

Typen von Lernverfahren

- Überwachtes Lernen ("Lernen **mit** Lehrer"):
  Ausgangsgröße für Lerndatensatz bekannt, z.B.
  - Backpropagation-Algorithmus
  - Levenberg-Marquardt-Algorithmus
- Reinforcement Learning:
  Qualitätsbewertung einer Ausgangsgröße (Lob/Tadel)
- Unüberwachtes oder Selbstorganisiertes Lernen ("Lernen **ohne** Lehrer"):
  Ausgangsgröße für Lerndatensatz unbekannt, z.B.
  - Hebb'sches Lernen
  - Lernverfahren für Kohonen-Karten
- Teilüberwachtes Lernen:
  Ausgangsgröße ist nur für einen Teil der Datentupel im Lerndatensatz bekannt

Selbstorganisiertes Lernen: Hebb'sche Regel

- Das erste Lerngesetz wurde 1949 von HEBB für das biologische Modell formuliert:

  „*Verbindungen zwischen Neuronen werden dann verstärkt werden, wenn die Neuronen gleichzeitig aktiv sind.*“

- mathematische Realisierung für jedes Gewicht w_{ij}, das von Neuron j zu Neuron i verbindet (Doppel-Index i,j jetzt notwendig, um beide Neuronen eindeutig zu adressieren):

  $\Delta w_{i,j}[k+1] = w_{i,j}[k] + \Delta w_{i,j}[k]$ mit

  $\Delta w_{i,j}[k] = f(y_i[k], y_j[k])$, z. B. $\Delta w_{i,j}[k] = \rho \cdot y_i[k] \cdot y_j[k]$

  Lernfaktor $\rho > 0$

- Anlehnung an biologische neuronale Netze
- "Lernen durch Verstärkung"
- keine Ausgangsgröße des Netzes im Lerndatensatz notwendig

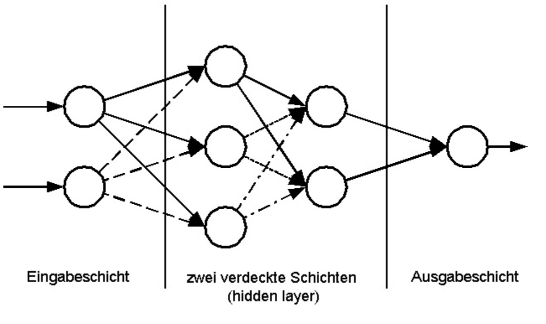
Gliederung

Multi-Layer-Perceptron-Netze (MLP-Netze)

Das MLP-Netz ist dadurch gekennzeichnet, dass

- die Neuronen in mehreren Schichten (Ebenen) angeordnet sind
- Feedforward-Netz
- Bestimmung des Zustands: gewichtete Summe mit Absolutterm
- Aktivierungsfunktion für verdeckte Schicht meist Tansig- oder Sigmoid-Funktion

Veranschaulichung (Beispiel mit V=4 Schichten):



Hierbei handelt es sich um eine Erweiterung des klassischen Perceptrons (besteht nur aus einem einzigen Neuron). Das MLP-Netz ist ein statisches Netz.

Schichten von MLPs (1)

Neuronen der Eingabeschicht (Schicht v = 1):

- verteilen die Eingangswerte auf die Neuronen der ersten verdeckten Schicht
- jedes Neuron hat deshalb nur einen Eingang, dieser Eingang ist auch Eingang des Netzes
- meist lineare Aktivierungsfunktionen

Neuronen der verdeckten Schichten (v = 2...V-1) und der Ausgabeschicht (v=V):

- Eingang eines jeden Neurons ist in der Regel mit allen Neuronen der Vorgängerschicht verbunden
- zusätzlicher Eingang mit Wert Eins (Gewicht w_{i0})
- Zustand z_i des Neurons berechnet sich aus der mit w_{ij} gewichteten Summe der Ausgänge der Neuronen der vorherigen (v-1). Schicht:

$$z_i^{(v)} = w_{i0}^{(v)} + \sum_j w_{ij}^{(v)} y_j^{(v-1)}$$

Schichten von MLPs (2)

- Ausgang des Neurons berechnet sich aus Aktivierungsfunktion für den Zustand z

$$y_i^{(v)} = f_i^{(v)}(z_i^{(v)})$$

- Ausgabeschicht: f meist linear
- Ausgang der Neuronen der Ausgabeschicht = Ausgang des Netzes
  - für Regressionsaufgaben häufig ein Ausgang
  - für Klassifikationsaufgaben meist mehrere Ausgänge (z.B. 3 Klassen o.k., Fehler Typ A, Fehler Typ B) jeweils als ein Neuron kodieren (einfacheres Lernen), Details siehe Vorlesung "Datenanalyse für Ingenieure" im Sommersemester
- Struktur:
  - Anzahl der Schichten V
  - Anzahl der Neuronen in den Schichten
  - Typ der Aktivierungsfunktion
- Parameter:
  - Gewichte der Verbindungen

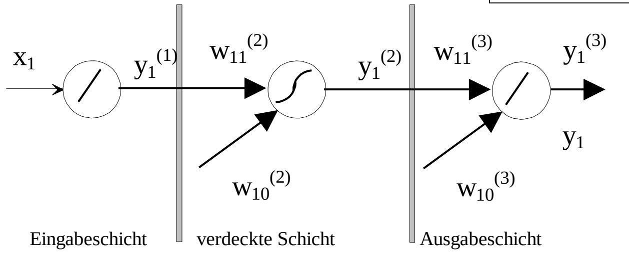
Einfaches MLP-Beispiel

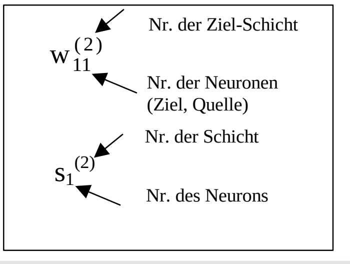
- 1 Eingang x_1, 1 Ausgang y_1, 1 verdeckte Schicht (V=3)
- lineare Aktivierungsfunktionen f für Eingangs- und Ausgangsneuron
- tansig-Aktivierungsfunktion in verdeckter Schicht
- zusätzliche Schwellwerte in verdeckter Schicht und am Ausgangsneuron

$$y_1^{(1)} = f_1^{(1)}(x_1) = x_1$$

$$y_1^{(2)} = f_1^{(2)}(z_1^{(2)}) = \frac{2}{1+e^{-2(w_{11}^{(2)}y_1^{(1)}+w_{10}^{(2)})}} - 1$$

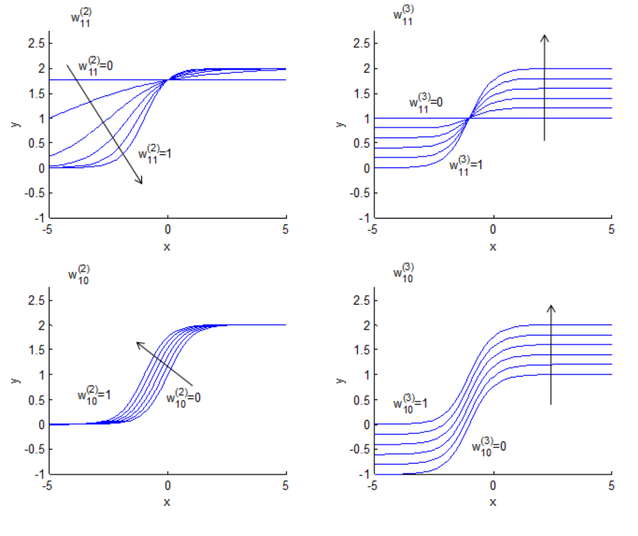
$$\begin{aligned}\hat{y}_1 = y_1^{(3)} &= f_1^{(3)}(z_1^{(3)}) \\ &= w_{11}^{(3)} y_1^{(2)} + w_{10}^{(3)} \\ &= w_{11}^{(3)}\left(\frac{2}{1+e^{-2(w_{11}^{(2)}x_1+w_{10}^{(2)})}} - 1\right) + w_{10}^{(3)}\end{aligned}$$



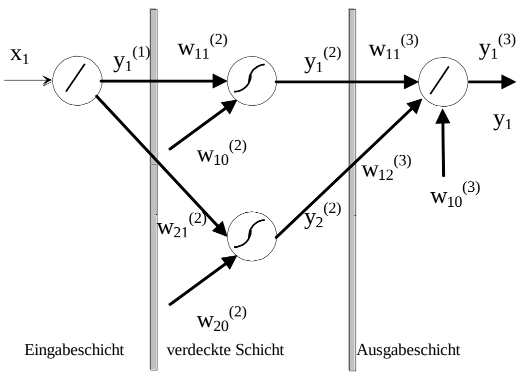


Wirkung der einzelnen Parameter (3 Neuronen)

- $w_{11}^{(2)}$ - Gewicht zwischen Neuron Eingabeschicht und Neuron verdeckte Schicht (oben links)
- $w_{11}^{(3)}$ - Gewicht zwischen Neuron verdeckter Schicht und Neuron Ausgabe-schicht (oben rechts)
- $w_{10}^{(2)}$ - Absolut-Term Neuron verdeckte Schicht (unten links)
- $w_{10}^{(3)}$ - Absolut-Term Neuron Ausgabeschicht (unten rechts)
- Änderung je ein Parameter zwischen 0 und 1, alle anderen Parameter sind 1



Erweiterung um 2. Neuron in verdeckter Schicht

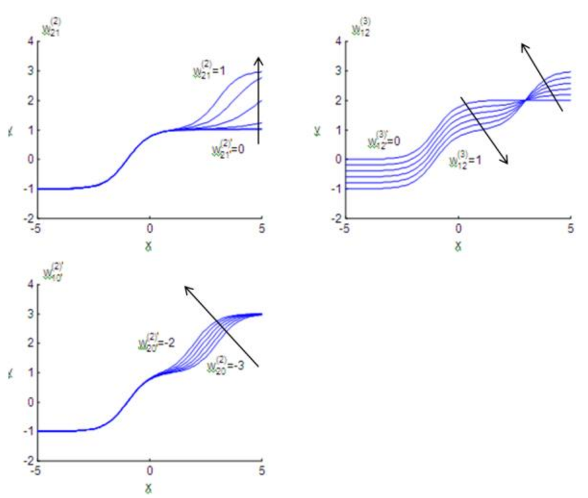


$$y_2^{(2)} = f_2^{(2)}(z_2^{(2)}) = \left(\frac{2}{1+e^{-2(w_{21}^{(2)} y_1^{(1)} + w_{20}^{(2)})}} - 1\right)$$

$$\begin{aligned}
\hat{y}_1 = y_1^{(3)} &= f_1^{(3)}(z_1^{(3)}) \\
&= w_{11}^{(3)} y_1^{(2)} + w_{10}^{(3)} + w_{12}^{(3)} y_2^{(2)} \\
&= w_{11}^{(3)} \left(\frac{2}{1+e^{-2(w_{11}^{(2)} x_1 + w_{10}^{(2)})}} - 1\right) + w_{10}^{(3)} \\
&\quad + w_{12}^{(3)} \left(\frac{2}{1+e^{-2(w_{21}^{(2)} x_1 + w_{20}^{(2)})}} - 1\right)
\end{aligned}$$

Wirkung der einzelnen Parameter (4 Neuronen)

- $w_{21}^{(2)}$ - Gewicht zwischen Neuron Eingabeschicht und 2. Neuron verdeckte Schicht (oben links)
- $w_{12}^{(3)}$ - Gewicht zwischen 2. Neuron verdeckter Schicht und Neuron Ausgabeschicht (oben rechts)
- $w_{20}^{(2)}$ - Absolut-Term 2. Neuron verdeckte Schicht (unten links)
- Standard:
  $w_{21}^{(2)} = 1$,
  $w_{12}^{(3)} = 1$,
  $w_{20}^{(2)} = -3$,
- Änderung:
  $w_{21}^{(2)} = 0...1$,
  $w_{12}^{(3)} = 0...1$,
  $w_{20}^{(2)} = -2...-3$



Jedes Neuron hat <u>globale</u> Wirkung!!!

Hausaufgabe (5 Neuronen)

Erweitern Sie das Netz um ein 3. Neuron in verdeckter Schicht!

Fragen:

1. Welche Parameter kommen dazu?
2. Welche Terme kommen dazu?
3. Ermöglicht die Erweiterung den Aufbau anderer Funktionen?
4. Unter welchen Voraussetzungen beeinflusst das 3. Neuron den Ausgang nicht?

Bestimmung der Parameter (1)

- Alle Gewichte werden in einen Vektor $\mathbf{w}_{MLP}$ geschrieben
- Gütekriterium: Summe der quadratischen Fehler minimieren (entspricht der Euklidischen Distanz d_{Euk} zwischen Ausgangsgrößen im Lerndatensatz und den Ausgangsgrößen des Netzes, überwachtes Lernen)

$$Q = (d_{Euk}(\mathbf{y}, \hat{\mathbf{y}}))^2 = \sum_{n=1}^{N} (y[n] - \hat{y}[n])^2 = (\mathbf{y} - \hat{\mathbf{y}})^T \cdot (\mathbf{y} - \hat{\mathbf{y}})$$

- Ausgangsgrößen des Netzes hängen in komplizierter Form von den Parametern ab, siehe vorherige Beispiele
- deswegen existiert keine geschlossene Lösung
- rekursives Verfahren:

$$\mathbf{w}_{MLP}[k+1] = \mathbf{w}_{MLP}[k] - \rho[k]\mathbf{W}_{rek}[k]\frac{\partial Q}{\partial \mathbf{w}_{MLP}}|_{\mathbf{w}_{MLP}[k]}, \rho \in [0,1]$$

- Startlösung meist zufällig
- $\mathbf{W}_{rek}$ ist Wichtungsmatrix, im einfachsten Fall Einheitsmatrix

Bestimmung der Parameter (2)

- Hauptproblem: Berechnung des Gradienten $\frac{\partial Q}{\partial \mathbf{w}_{MLP}}|_{\mathbf{w}_{MLP}[k]}$
- Zwei Lösungswege:
  - kleine Änderungen an w_{ij} durchführen, neues Q und Gradient ausrechnen (numerisch extrem aufwändig, jedes Mal komplette Netzberechnung)
  - Gradient geschlossen berechnen (eleganter!), dazu muss Q nach jedem (!) Gewicht abgeleitet werden
    - Bei vielen Netzen (z.B. MLP) ist Ausnutzung der Feedforward- und Schichtenstruktur möglich
    - Fehler wird rückwärts von Ausgabeschicht ausgehend nach vorn verteilt
    - daher der Name "Backpropagation-Verfahren"
- Auch noch zu klären: Berechnung der Rekursion und der Gradienten
  - separat für jedes Datentupel oder
  - für alle Datentupel (Batch-Verfahren)

Bestimmung der Parameter (3)

Geschlossene Berechnung des Gradienten (MLP, 4 Neuronen, Auszüge):

$$\hat{y} = w_{11}^{(3)} \left(\frac{2}{1+\exp\left(-2\left(w_{11}^{(2)} y_1^{(1)} + w_{10}^{(2)}\right)\right)} - 1 \right) + w_{10}^{(3)}$$

$$= w_{12}^{(3)} \left(\frac{2}{1+\exp\left(-2\left(w_{21}^{(2)} y_1^{(1)} + w_{20}^{(2)}\right)\right)} - 1 \right)$$

$$Q = \sum_{n=1}^{N} (y[n] - \hat{y}[n])^2$$

$$\frac{\partial Q}{\partial w_{10}^{(3)}} = 2 \sum_{n=1}^{N} (y[n] - \hat{y}[n])$$

$$\frac{\partial Q}{\partial w_{11}^{(3)}} = 2 \sum_{n=1}^{N} (y[n] - \hat{y}[n]) \underbrace{\left(\frac{2}{1+\exp\left(-2\left(w_{11}^{(2)} y_1^{(1)}[n] + w_{10}^{(2)}\right)\right)} - 1 \right)}_{y_1^{(2)}[n]}$$

Bestimmung der Parameter (4)

- Gradient kann für alle Schichten rückwärts und symbolisch berechnet werden
- Formeln in Computerprogrammen hinterlegt
- Zwischenergebnisse müssen abgespeichert werden
- Rolle von $\mathbf{W}_{rek}$ in

$$\mathbf{w}_{MLP}[k+1] = \mathbf{w}_{MLP}[k] - \rho[k]\mathbf{W}_{rek}[k]\frac{\partial Q}{\partial \mathbf{w}_{MLP}}|_{\mathbf{w}_{MLP}[k]}, \rho \in [0,1]$$

  – Gradientenabstieg (manchmal auch als Backpropagation bezeichnet) $\mathbf{W}_{rek}[k] = \mathbf{I}$

  – Levenberg-Marquardt-Verfahren:

  $\mathbf{W}_{rek}[k] = (\hat{\mathbf{H}} + \alpha[k] \cdot \mathbf{I})^{-1},\ \alpha[k]$ – Wichtungsfaktor

  α = 0: Newton-Verfahren

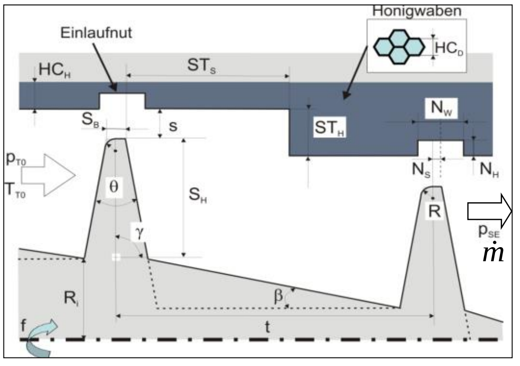
Hesse-Matrix $\mathbf{H}$: Matrix der partiellen zweiten Ableitungen des Bewertungsmaßes nach den Parametern mit Elementen $H_{ij} = \partial^2 Q / (\partial w_{MLP,i} \cdot \partial w_{MLP,j})$

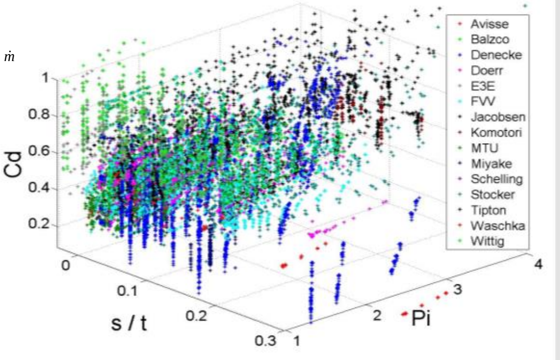
Kommentare

- Verfahren zur Bestimmung der Parameter:
  – garantieren keine optimale Lösung, alle genannten Verfahren können in lokalen Minima der Gütefunktion stagnieren
  – Gradientenabstieg konvergiert oft extrem langsam
  – Levenberg-Marquardt-Verfahren konvergiert viel schneller, erfordert aber die aufwändige Berechnung der Hesse-Matrix (u.U. Laufzeit- oder Speicherprobleme im Computer, insbesondere bei großen Netzen)
  – Backpropagation wird oft auch als Synonym für Gradientenabstieg benutzt
- Struktursuche:
  – oft Heuristiken
  – mit extrem einfachen Netzen beginnen (eine verdeckte Schicht, 2-3 Neuronen) und langsam steigern
  – zu komplizierte Strukturen führen zu Overfitting (siehe Übung)
  – Validierung mit unbekannten Testdaten oder Validierungsverfahren (siehe Vorlesung "Datenanalyse für Ingenieure" im Sommersemester)

Durchblick- und Stufenlabyrinthe [Pychynski09,10]

Datensammlung zum Durchflussverhalten von Durchblick- und Stufenlabyrinthen:





Komplexes System mit 21 Einflussparametern

Datensatz mit 15.297 Datentupeln aus 15 Quellen

[Pychynski09] Pychynski, T.: Anwendung von Data Mining Methoden zur Analyse von Turbomaschinenkomponenten am Beispiel des Durchflussverhaltens von Labyrinthdichtungen. *Karlsruher Institut für Technologie (KIT),* **2009**

[Pychynski10] Pychynski, T.; Blesinger, G.; Mikut, R.; Dullenkopf, K. & Bauer., H.-J.: Modelling the Labyrinth Seal Discharge Coefficient Using Data Mining Methods. *Proc., ASME TURBO EXPO; Glasgow,* **2010**

Hinweise

- Ein Ausgangsneuron: Durchflussbeiwert C_d (dimensionsloser Leckagestrom)
- Strukturentscheidungen:
  - MLP mit einer verdeckten Schicht, Zahl Neuronen in dieser Schicht schrittweise steigern (1...30)
  - Eingangsgrößen u.U. vorher kombinieren (Verhältnisse wie s/t) und als neue Eingangsgrößen verwenden
  - nicht mit allen 21 Merkmalen als Eingangsgrößen des Netzes arbeiten
  - konsequente Merkmalsauswahl durch Durchprobieren von Netzstrukturen, beginnend mit einem Eingang, gute Ergebnisse ab 4 Merkmalen ("Wrapper-Verfahren", siehe "Datenanalyse für Ingenieure" im Sommersemester)
- Gibt es irgendwelche Fallen?
  Daten z.T. auf Papier, schlechte Datenqualität (Abdeckung Merkmalsraum, nicht gemessene Merkmale)

Lösung [Pychynski09,10]

- Regressionsmodelle mit Polynomen und Künstlichen Neuronalen Netzen (MLP mit 30 Neuronen in der verdeckten Schicht)
- Polynome ab Grad 2 o.k., Neuronale Netze besser
- Bewertung mit Korrelationskoeffizienten zwischen Cd und Schätzung von Cd für jedes Modell, je nach Modell zwischen 0.95 - 0.99
- ACHTUNG! Modellgüte nur in der Nähe von existierenden Datentupeln gut, Probleme in schlecht abgedeckten Bereichen