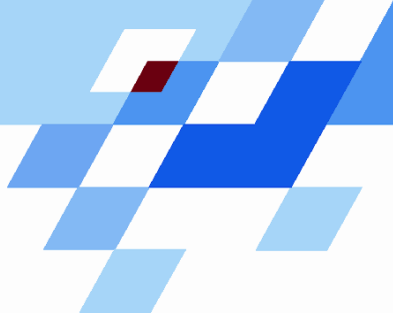


Wintersemester 2006/07

Fundamente der Computational Intelligence

(Vorlesung)

Prof. Dr. Günter Rudolph

Fachbereich Informatik

Lehrstuhl für Algorithm Engineering

Inhalt

- Lernverfahren für Multi-Layer-Perceptrons

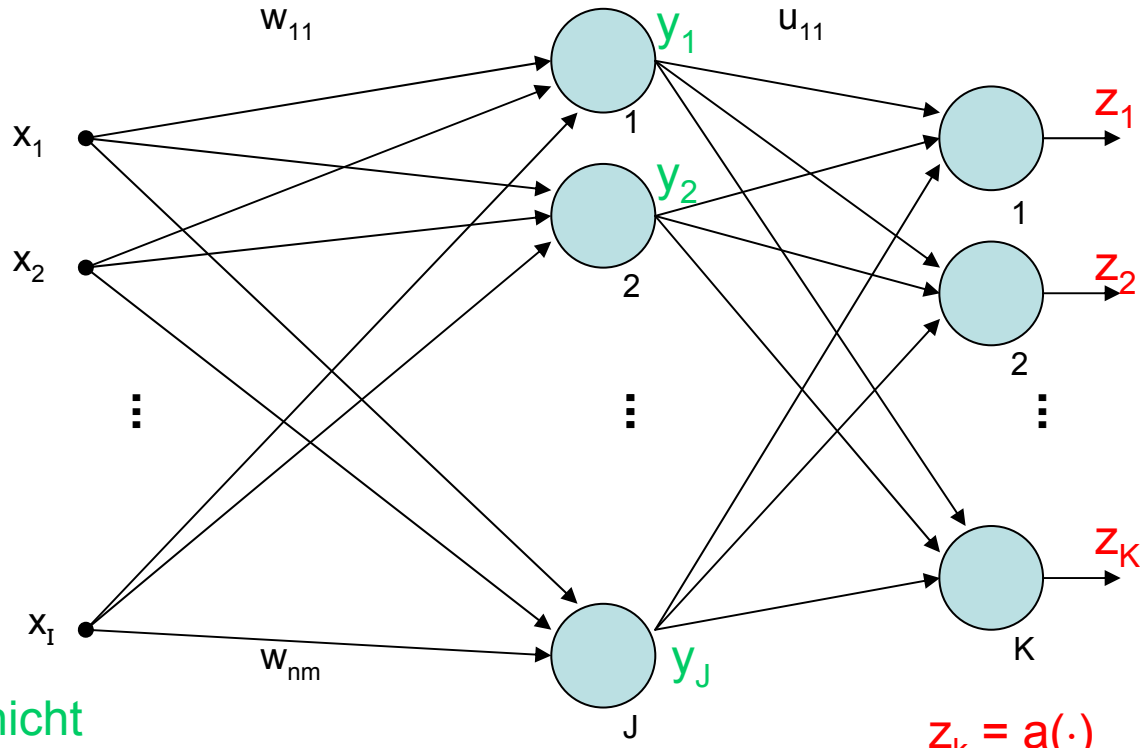
Lernalgorithmus für Multi-Layer-Perceptrons

Gradientenverfahren

$f(w_t, u_t) = TSSE$

$u_{t+1} = u_t - \gamma \nabla_u f(w_t, u_t)$

$w_{t+1} = w_t - \gamma \nabla_w f(w_t, u_t)$



x_i : Eingabe an Eingabeschicht

y_j : Ausgabe der verdeckten Schicht

z_k: Ausgabe der Ausgabeschicht

$z_k = a(\cdot)$

$y_j = h(\cdot)$

$$y_j = h\left(\sum_{i=1}^{I} w_{ij} \cdot x_i\right) = h(w_j' x)$$

Ausgabe von Neuron j in der verdeckten Schicht

$$z_k = a\left(\sum_{j=1}^{J} u_{jk} \cdot y_j\right) = a(u_k' y)$$

Ausgabe von Neuron k in der Ausgabeschicht

$$= a\left(\sum_{j=1}^{J} u_{jk} \cdot h\left(\sum_{i=1}^{I} w_{ij} \cdot x_i\right)\right)$$

Fehler bei Eingabe x:

$$f(w,u;x) = \sum_{k=1}^{K} (z_k(x) - z_k^*(x))^2 = \sum_{k=1}^{K} (z_k - z_k^*)^2$$

Netzausgabe

Sollausgabe bei Eingabe x

Fehler bei Eingabe x und Sollausgabe z*:

$$f(w,u;x,z^*) = \sum_{k=1}^{K} \Bigg[\underbrace{a\Bigg(\sum_{j=1}^{J} u_{jk} \cdot \underbrace{h\Bigg(\overbrace{\sum_{i=1}^{I} w_{ij} \cdot x_i}^{w_j' x} \Bigg)}_{y_j} \Bigg)}_{z_k} - z_k^*(x) \Bigg]^2$$

Gesamtfehler für alle Beispiele (x, z*) ∈ B:

$$f(w,u) = \sum_{(x,z^*) \in B} f(w,u;x,z^*) \qquad \text{(TSSE)}$$

Gradient des Gesamtfehlers:

$$\nabla f(w,u) = \sum_{(x,z^*)\in B} \nabla f(w,u;x,z^*)$$

Vektor der partiellen Ableitungen nach den Gewichten u_{jk} und w_{ij}

also:

$$\frac{\partial f(w,u)}{\partial u_{jk}} = \sum_{(x,z^*)\in B} \frac{\partial f(w,u;x,z^*)}{\partial u_{jk}}$$

bzw.

$$\frac{\partial f(w,u)}{\partial w_{ij}} = \sum_{(x,z^*)\in B} \frac{\partial f(w,u;x,z^*)}{\partial w_{ij}}$$

Annahme: $a(x) = \dfrac{1}{1+e^{-x}} \quad \Rightarrow \quad \dfrac{d\,a(x)}{dx} = a'(x) = a(x) \cdot (1 - a(x))$

und: $h(x) = a(x)$

Kettenregel der Differentialrechnung:

$$[\,p(q(x))\,]' = \underbrace{p'(q(x))}_{\text{äußere Ableitung}} \cdot \underbrace{q'(x)}_{\text{innere Ableitung}}$$

$$f(w,u;x,z^*) \;=\; \sum_{k=1}^{K} [\, a(u_k' y) - z_k^* \,]^2$$

partielle Ableitung nach u_{jk}:

$$\frac{\partial f(w,u;x,z^*)}{\partial u_{jk}} \;=\; 2\,[\, a(u_k' y) - z_k^* \,] \cdot a'(u_k' y) \cdot y_j$$

$$\;=\; 2\,[\, a(u_k' y) - z_k^* \,] \cdot a(u_k' y) \cdot (1 - a(u_k' y)) \cdot y_j$$

$$\;=\; \underbrace{2\,[\, z_k - z_k^* \,] \cdot z_k \cdot (1 - z_k)}_{\text{„Fehlersignal“ } \delta_k} \cdot y_j$$

partielle Ableitung nach w_{ij}:

$$\frac{\partial f(w,u;x,z^*)}{\partial w_{ij}} = 2\sum_{k=1}^{K}[\underbrace{a(u_k'y)}_{z_k} - z_k^*]\cdot\underbrace{a'(u_k'y)}_{z_k(1-z_k)}\cdot u_{jk}\cdot\underbrace{h'(w_j'x)}_{y_j(1-y_j)}\cdot x_i$$

$$= 2\cdot\sum_{k=1}^{K}[z_k - z_k^*]\cdot z_k\cdot(1-z_k)\cdot u_{jk}\cdot y_j(1-y_j)\cdot x_i$$

Faktoren umordnen

$$= x_i\cdot y_j\cdot(1-y_j)\cdot\sum_{k=1}^{K}\underbrace{2\cdot[z_k - z_k^*]\cdot z_k\cdot(1-z_k)}_{\text{Fehlersignal } \delta_k \text{ aus vorheriger Schicht}}\cdot u_{jk}$$

Fehlersignal δ_j aus „aktueller“ Schicht

Verallgemeinerung

Das neuronale Netz habe L Schichten (*layer*) S_1, S_2, ... S_L.

Seien Neuronen aller Schichten durchnummeriert von 1 bis N.

Alle Gewichte w_{ij} sind in Gewichtsmatrix W zusammengefasst.

Sei o_j Ausgabe (*output*) von Neuron j.

$j \in S_m \rightarrow$ Neuron j ist in m-ter Schicht

Fehlersignal:

$$\delta_j = \begin{cases} o_j \cdot (1 - o_j) \cdot (o_j - z_j^*) & \text{falls } j \in S_L \text{ (Ausgabeneuron)} \\ o_j \cdot (1 - o_j) \cdot \sum_{k \in S_{m+1}} \delta_k \cdot w_{jk} & \text{falls } j \in S_m \text{ und } m < L \end{cases}$$

Korrektur:

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} - \gamma \cdot o_i \cdot \delta_j$$

beim Online-Lernen:
Korrektur nach **jedem** präsentierten Beispiel

Fehlersignal eines Neurons einer inneren Schicht bestimmt durch

- Fehlersignale aller Neuronen der nachfolgenden Schicht und
- zugehörige Verbindungsgewichte.

$\Downarrow$

- Erst Fehlersignale der Ausgabeneuronen bestimmen,
- daraus Fehlersignale der Neuronen der vorhergehenden Schicht berechnen,
- daraus Fehlersignale der Neuronen der vorhergehenden Schicht berechnen,
- usw. bis zur ersten inneren Schicht.

$\Downarrow$

Fehler wird also von Ausgabeschicht zur ersten inneren Schicht zurückgeleitet.

$\Rightarrow$ **Backpropagation** (of error)

$\Rightarrow$ andere Optimierverfahren einsetzbar!

neben **Backpropagation** (Gradienenabstieg) auch:

- **Backpropagation mit Momentum**
  zusätzlich vorherige Gewichtsänderung mit einbeziehen:

$$\Delta w_{ij}^{(t)} = -\gamma_1 \cdot o_i \cdot \delta_j - \gamma_2 \cdot \Delta w_{ij}^{(t-1)}$$

- **QuickProp**
  Annahme: Fehlerfunktion durch quadratische Funktion lokal approximierbar!
  Formel verwendet die beiden letzten Gewichtsterme zur Zeit t – 1 und t – 2.

- **Resilient Propagation (RPROP)**
  Nutzt Vorzeichen der partiellen Ableitungen:
  2 mal negativ bzw. positiv $\Rightarrow$ Schrittweite vergrößern!
  Vorzeichenwechsel $\Rightarrow$ Rücknahme letzter Schritt und Schrittweite verkleinern!
  Typische Werte: Verkleinerungsfaktor 0,5 / Vergrößerungsfaktor 1,2

- **Evolutionäre Algorithmen**
  Individuum = Gewichtsmatrix