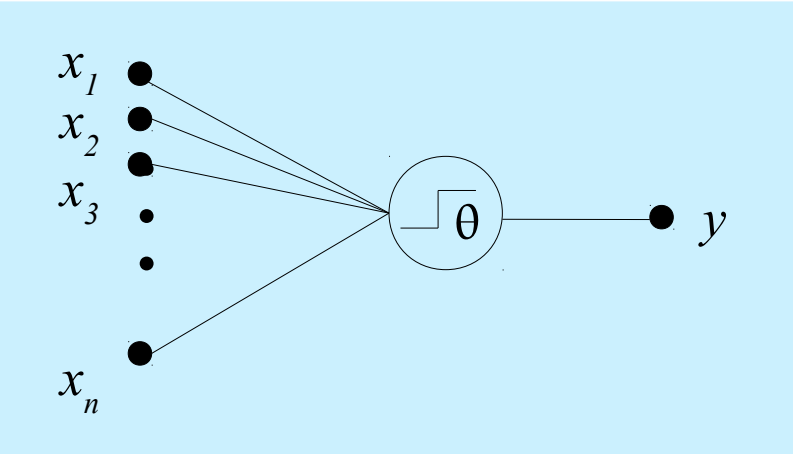
Fachhochschule
Münster University of
Applied Sciences

Adaptive Systeme

Mehrere Neuronen, Assoziative Speicher und Mustererkennung

Prof. Dr. rer. nat. Nikolaus Wulff

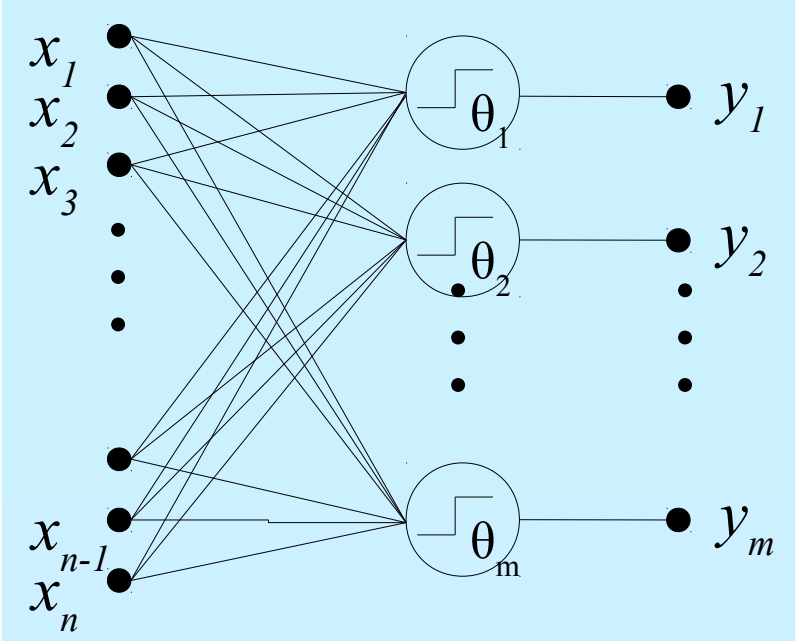
Modell eines Neuron



$$y = \sigma\left(\sum_{k=1}^{n} w_k x_k - \theta\right)$$

- Die n binären Eingangssignale $x_k \in \{0,1\}$ werden vom Neuron aufsummiert.
- Sobald die (gewichtete) Summe der Eingangssignale x_k eine gewisse Schwelle θ übersteigt, „feuert“ das Axon und das Ausgangssignal y geht von 0 auf 1, realisiert durch die Stufenfunktion σ.

Neuronen Schicht



$$y_j = \sigma\left(\sum_{k=1}^{n} w_{jk} x_k - \theta_j\right)$$

$$y_j = \sigma\left(\sum_{k=0}^{n} w_{jk} x_k\right)$$

$$\vec{y} = \sigma\left(W \vec{x}\right)$$

- Werden *m* Neuronen zu einer Schicht mit *n* gemeinsamen Eingängen zusammengefasst, so werden den *n* Eingangssignalen *m* Ausgangssignale zugeordnet.
- Es entsteht eine Abbildung $f: \{0,1\}^n \rightarrow \{0,1\}^m$.

Mehrere Neuronen als Schicht

- Wenn k Neuronen unabhängig voneinander denselben Eingangsvektor $\vec{x}$ verarbeiten, so liefert jedes Neuron ein Teilergebnis des Ausgabevektors $\vec{y}$.
- Bei m Neuronen entstehen aus dem n dimensionalen Eingangssignal m unabhängige Ausgangssignale y_m.
- Eine solche Neuronenschicht kann genauso trainiert werden wie ein einzelnes Neuron:
  - Jedes Neuron k wird solange mit den p Eingangsvektoren $\vec{x}^{(\nu)}$, $\nu = 1, \ldots, p$ trainiert, bis alle erwünschten Ausgabewerte $y_k^{(\nu)} \equiv (\vec{y}^{(\nu)})_k$ erlernt wurden.
  - Für die Konvergenz gelten die selben Aussagen, wie bei einem einzelnen Perzeptron.

Mustererkennung

- Mit Hilfe einer Schicht von Neuronen lassen sich recht einfach und schnell Muster erkennen und klassifizieren.
- Nach der Trainingsphase hat das Neuronale Netz ein „Gedächtnis“ der erlernten Muster und assoziiert die zugehörigen Ausgabe- mit den Eingangsdaten.
- Mit *m* Neuronen gibt es maximal 2^m Zustände.
- Es lässt sich allerdings nur eine beschränkte Anzahl von Mustern erlernen und diese dürfen nicht „zu ähnlich“ sein, ansonsten kommt es zu Fehlklassifikationen.

Assoziative Mustererkennung

- Ein assoziativer Speicher soll eine Menge $\{\boldsymbol{x}^{(1)},\ldots,\boldsymbol{x}^{(p)}\}$ von Mustern mit dem Satz $\{\boldsymbol{d}^{(1)},\ldots,\boldsymbol{d}^{(p)}\}$ von Ausgaben korrekt klassifizieren.
- Alle Muster sollen $k \ll n$ Einsen enthalten.
- Initialisiert wird die Gewichtsmatrix mit $\boldsymbol{W}^{(0)}=0$.
- Das Training erfolgt mit den p Eingabemustern:

$$\Delta w_{jk}^{(\nu)} = d_j^{(\nu)} \cdot x_k^{(\nu)} \in \{0,1\}$$

$$W^{(\nu)} = W^{(\nu-1)} + \Delta W^{(\nu)} \quad \nu = 1,\ldots,p$$

- D. h. in der Matrix $\boldsymbol{W}$ werden alle Elemente auf 1 gesetzt, wenn sowohl x_j als auch d_k gleich 1 sind.

Das assoziative Gedächtnis

- Als Bias wird $w_{j0}=k-½$ gewählt, nach dem Training hat die Matrix dann die Belegung:

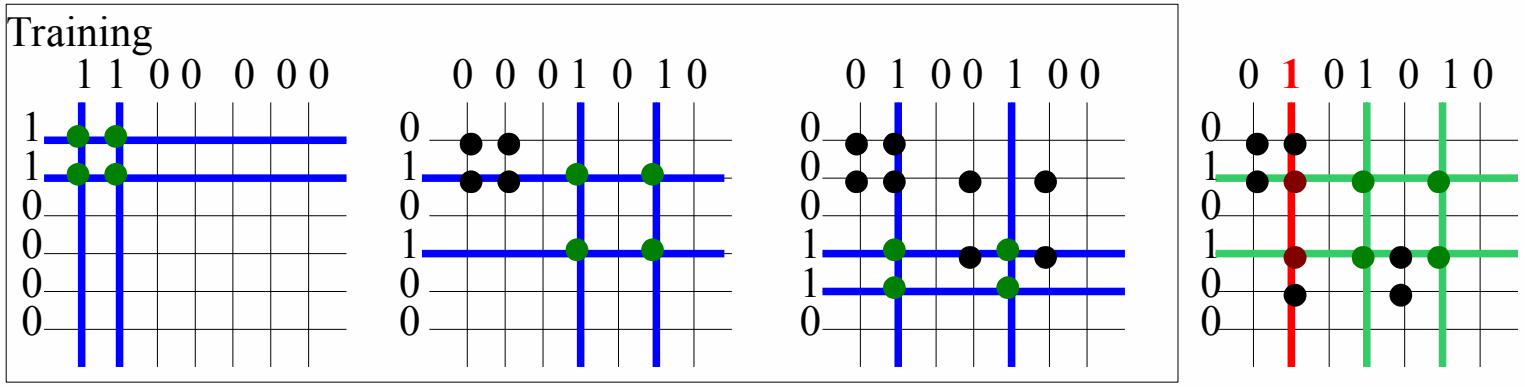
$$(W)_{jk} = \max_{\nu=1\ldots p} \left(d_j^{(\nu)} x_k^{(\nu)}\right)$$

- und für alle Vektoren gilt daher:

$$(W\,\vec{x}^{(\nu)})_j = \sum_k w_{jk} x_k^{(\nu)} = \sum_k d_j^{(\nu)} x_k^{(\nu)} = \sum_k x_k^{(\nu)} = k > w_{j0}$$

- Falsche Einsen im Ausgabekanal y_j können nur dann entstehen, wenn ein Vektor $\vec{x}^{(\mu)}$ bei allen k Einträgen auf einen passenden Gewichtsvektor $\vec{w}_k$ trifft.
- Die Wahl von w_{j0} ermöglicht keine Fehlertoleranz.

Falsche Assoziationen



- Das Beispiel illustriert das Zustandekommen einer falschen Ausgabe am Beispiel $k=2$.
- Erlernt wurden drei Tupel $(x^{(\nu)},d^{(\nu)})$ die entsprechenden Matrixeinträge sind mit • gekennzeichnet.
- Eine spätere Abfrage mit dem zweiten Vektor ergibt die in rot gekennzeichnete falsche Ausgabe.

$$d_{Hamming}(\vec{x}^{(\nu)},\vec{x}^{(\mu)})\neq 0 \wedge d_{Hamming}(\vec{d}^{(\nu)},\vec{d}^{(\mu)})\neq 0$$

Adaline

- Adaline ist ein Kunstwort aus ADAptive LInear NEuron und ist formal analog zum Perzeptron gebaut.
- Adaline verwendet einen gegenüber dem Perzeptron abgewandelten Trainingsalgorithmus:

$$\Delta w_{jk} = \mu \left(d_j - \sum_\nu w_{j\nu} x_\nu \right) \cdot x_k \qquad \mu > 0$$

- D.h. die Hebbsche Lernregel wird ohne Auswertung der Transferfunktion angewendet.
- Im Trainingsmodus besteht Adaline nur aus einer reinen Matrizenmultiplikation, d.h. ist eine lineare Abbildung, für die das Superpositionsprinzip gilt.

Linearer assoziativer Speicher

- Ohne Anwendung der Transferfunktion entsteht ein linearer assoziativer Speicher dessen optimale Lösung sich analytisch berechnen lässt:

$$E[W] = \frac{1}{2}\sum_{\nu=1}^{p} \left\| y^{(\nu)} - d^{(\nu)} \right\|^2$$

$$E[W] = \frac{1}{2}\sum_{\nu=1}^{p} \left(\sum_{j=0}^{m} \left(\sum_{k=0}^{n} w_{jk} x_k^{(\nu)} - d_j^{(\nu)} \right)^2 \right)$$

- Das Minimum erfüllt das Gleichungssystem

$$(\nabla_W E)_{jk} = \frac{\partial E}{\partial w_{jk}} = \sum_{\nu=1}^{p} \left(\sum_i w_{ji} x_i^{(\nu)} - d_j^{(\nu)} \right) x_k^{(\nu)} = 0$$

Matrixgleichung des LAM

- Die zugehörige Matrizengleichung lautet:

$$W \cdot XX^T = D \cdot X^T$$

- welche formal die Lösung $W = D \cdot X^T \cdot (XX^T)^{-1}$ besitzt, sofern die quadratische Matrix (XX^T) invertierbar ist.
- Da die Mustervektoren $x^{(v)}$ jedoch i.A. nicht linear unabhängig sind, ist dies meistens nicht der Fall und es existiert eine ganze Schar von Lösungsmatrizen W.
- Als zusätzliche Forderung wird daher gefordert, dass die Quadratsumme $\sum_{jk} w_{jk}^2$ der Gewichte gleichzeitig mit $E[W]$ minimiert wird. Die neue Forderung lautet:

$$E[W] + \lambda \sum_{jk} w_{jk}^2 = E[W] + \lambda \left\| W \right\|^2 = min$$

Pseudoinverse

- Dies führt zur Matrixgleichung

$$W \cdot (XX^T + \lambda \mathbf{1}) = D \cdot X^T$$

- Die Matrix $(XX^T + \lambda \mathbf{1})$ ist für alle Werte $\lambda > 0$ positiv definit und invertierbar.

$$W = \lim_{\lambda \to 0} DX^T \cdot (XX^T + \lambda \mathbf{1})^{-1} =: D \cdot \tilde{X}$$

- Mit der Pseudoinversen

$$\tilde{X} = \lim_{\lambda \to 0} X^T (XX^T + \lambda \mathbf{1})^{-1}$$

Gradientenverfahren

- Ebenso ist es möglich mittels eines Gradientenverfahrens eine Nährung zu finden, in dem als Lernregel

$$\Delta W = \mu \sum_{\nu=1}^{p} \left(\vec{d}^{(\nu)} - W \cdot \vec{x}^{(\nu)} \right) \left(\vec{x}^{(\nu)} \right)^T$$

- angewandt wird. Diese erfüllt i.A. nicht mehr die Forderung, dass ein Minimum der Gewichte gefunden wird. Sind die Vektoren statistisch mit gleichem Gewicht verteilt, kann die vereinfachte Lernregel

$$\Delta W = \tilde{\mu} \left(\vec{d}^{(\nu)} - W \cdot \vec{x}^{(\nu)} \right) \left(\vec{x}^{(\nu)} \right)^T, \quad \tilde{\mu} = \mu / p$$

- angewandt werden, d.h. zusätzliche Vektoren erfordern keine komplette Neuberechnung von W.

Einbeziehung der Transferfunktion

- Mit einer stetig differenzierbaren Transferfunktion gilt für ein beliebiges Musterpaar (x,d):

$$\frac{\partial E}{\partial w_{jk}} = \left(\sigma\left(\sum_i w_{ji} x_i\right) - d_j\right) \sigma'\left(\sum_i w_{ji} x_i\right) x_k$$

- und die Anwendung des Gradientenverfahrens liefert die Lernregel:

$$\Delta w_{jk} = \mu \left(d_j - y_j\right) \sigma'(\tilde{y}_j) x_k = \mu e_j \sigma'(\tilde{y}_j) x_k$$

- Die Bedingung $y_j \in \{0,1\}$ ist jetzt nicht mehr streng gegeben, sondern die Bildmenge ist das Intervall $[0,1]$

Übung/Praktikum

- Formulieren Sie die Hebbschen Lernregeln für die stetigen Transferfunktionen.
  - Berechnen Sie die Ableitung $\sigma_c'(x)$ der Fermi- und Tanh-Funktion und drücken Sie diese durch $\sigma_c(x)$ aus.
  - Implementieren Sie entsprechende Lernalgorithmen.
  - Testen Sie mit den Beispielen des And und Or Problems.
- Sie wollen Oberflächenvektoren $x \in \mathbb{R}^3;\ \|x\| = 1$ der Einheitskugel klassifizieren nach den drei Kriterien oben/unten, links/rechts und vorne/hinten, d.h auf Zielvektoren $d \in \{0,1\}^3$ abbilden.
  - Formulieren Sie einen entsprechenden Lernalgorithmus.
  - Wogegen konvergiert die Matrix W?