

# 8 Multi-Layer-Perzeptron (MLP)

## 8.1 Modellneuron

jedes künstliche neuronale Netz besteht aus verschalteten Modellneuronen

- jedes Neuron hat Eingangsvektor  $\vec{x}$ ,
- der mit Gewichtsvektors  $\vec{w}$  zum Aktivierungszustand  $h$  multipliziert wird:

$$h = \vec{x}^T \vec{w}$$

- die Anwendung der Aktivierungsfunktion  $\sigma$ , die von einem Schwellwert oder bias  $\theta$  abhängt, führt zur Ausgabe  $y$

$$y = \sigma(h - \theta) = \sigma(\vec{x}^T \vec{w} - \theta) = \sigma\left(\sum_{j=1}^N w_j x_j - \theta\right) = \sigma\left(\sum_{j=0}^N w_j x_j\right)$$

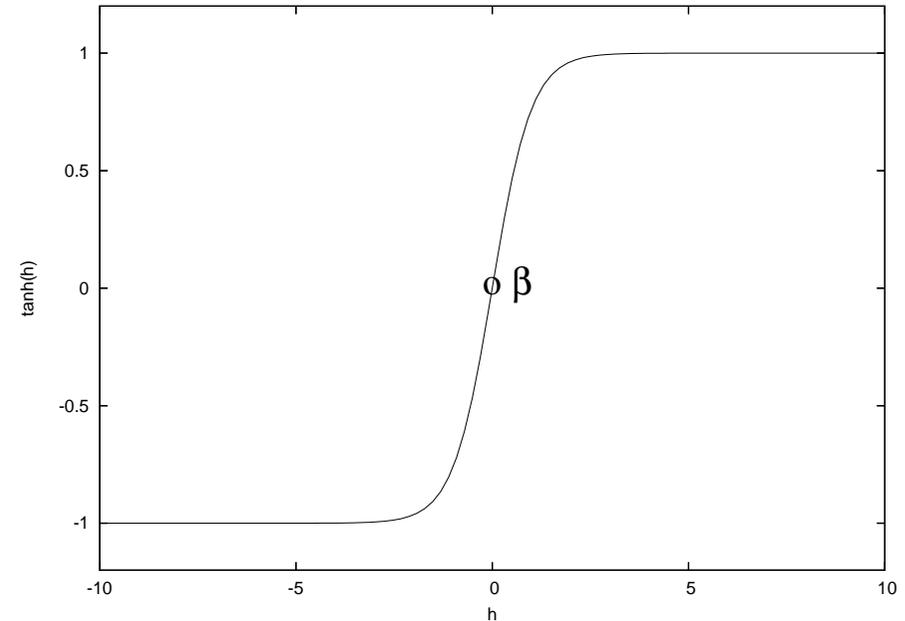
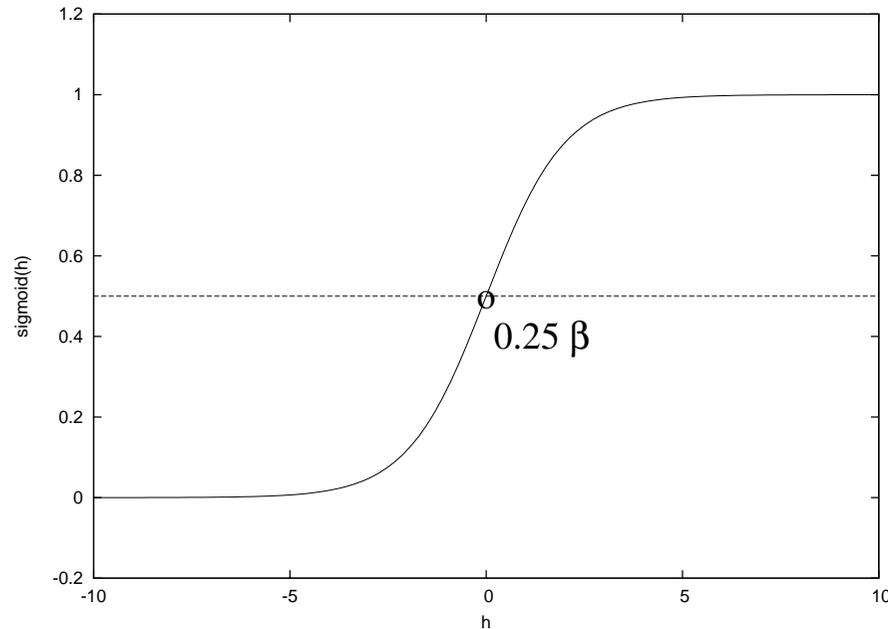
mit  $w_0 = \theta$  und  $x_0 = -1$

## 8.1 Modellneuron

- typische Aktivierungsfunktionen

$$\sigma(h) = \frac{1}{1+e^{-\beta(h-\theta)}} \quad \sigma'(h) = \beta \cdot \sigma(h) \cdot (1 - \sigma(h))$$

$$\sigma(h) = \tanh(\beta(h - \theta)) \quad \sigma'(h) = \beta \cdot (1 - \sigma(h)^2)$$



- $\beta$  bestimmt Steilheit (maximal  $\frac{1}{4}\beta$  bzw.  $\beta$ )
- $\theta$  verschiebt die Kurve

### 8.2 Architektur eines MLP

ein MLP ist ein mehrschichtiges vorwärtsgekoppeltes Netzwerk.

- jede der  $l = 1 \dots L$  Schichten besteht aus  $M^l$  Modellneuronen
- jedes Neuron  $i$  einer Schicht  $l$ 
  - erhält den gleichen Eingangsvektor  $\vec{x}^l$
  - besitzt die Gewichtskoeffizienten  $\vec{w}_i^l$
  - berechnet einen Ausgangswert

$$y_i^l = \sigma \left( \sum_{j=0}^{M^{l-1}} w_{ij}^l \cdot x_j^l \right) = \sigma \left( \vec{w}_i^{lT} \vec{x}^l \right)$$

- Zusammenfassung einer Schicht

$$\vec{y}^l = \vec{\sigma} \left( \underline{W}^{lT} \vec{x}^l \right), \text{ mit } \underline{W}^l := (\vec{w}_1^l, \dots, \vec{w}_{M^l}^l)$$

- der Ausgabevektor der Schicht  $l$  dient als Eingangsvektor der nächsten

$$\vec{x}^{l+1} := \vec{y}^l, \text{ wobei } x_0^{l+1} = -1$$

### 8.3 Einsatz der MLPs zur Musterklassifikation

- erweitere Merkmalsvektor  $\vec{c}$  um eine Komponente als Eingangsvektor der ersten Neuronenschicht benutzt:  $\vec{x}^1 = \begin{pmatrix} -1 \\ \vec{c} \end{pmatrix}$
- MLP als Funktionsapproximator für die Unterscheidungsfunktion:  
 $\vec{d}(\vec{c}) = \vec{y}^L = \vec{y}^L(\vec{c})$
- letzte Schicht besitzt also genau  $K$  (Anzahl der Klassen) Neuronen
- Entscheidungsregel analog zum Polynomklassifikators:  
 $g(\vec{c}) = e(\vec{d}(\vec{c})) = \omega_l, \quad \text{falls } l \text{ maximale Komponente von } \vec{d}(\vec{c})$
- Parameteroptimierung mit gleichen Kriterien wie beim Polynomklassifikator:  
bestimme die  $\vec{w}_i^l$  mit minimalem mittleren quadratischen Fehler:

$$S^2(\underline{W}^1, \dots, \underline{W}^L) = E \left\{ (\vec{y} - \vec{y}^L(\vec{c}))^2 \right\}$$

## 8.3 Einsatz der MLPs zur Musterklassifikation

---

- es existiert keine geschlossene Lösung wie beim Polynomklassifikator  
Optimierung deshalb mit Gradientenabstieg für ein Stichprobenelement  $\vec{c}$  mit Zielvektor  $\vec{y}$ :

$$E(\vec{c}) = (\vec{y} - \vec{a}^L(\vec{c}))^2$$

- Berechnung der partiellen Ableitung  $\frac{dE(\vec{c})}{dw_{ij}^l}$  führt auf **Rückpropagierung (back propagation)** des Fehlers beginnen bei der Ausgabeschicht

$$\delta_i^l := \begin{cases} (y_i - y_i^L) \sigma'(h_i^L) & l = L \\ \left( \sum_{k=1}^{M^{l+1}} \delta_k^{l+1} w_{ki}^{l+1} \right) \sigma'(h_i^l) & \text{sonst} \end{cases}$$

damit Ändern der Gewichte:

$$\Delta w_{ij}^l := \epsilon \delta_i^l y_j^{l-1}$$

mit der Lernschrittweite  $\epsilon$

- iteriertes Trainingsverfahrens über die Elemente der Stichprobe, wobei i.d.R.  $\epsilon$  im Lauf der Iteration monoton abfällt

### Bemerkungen

1. das einschichtige Perzeptron ( $L = 1$ ) ist identisch mit dem Polynomklassifikator vom Grad 1  
(Aktivierungsfunktion  $\sigma$  ändert die Entscheidung nicht)
2. die nicht-lineare Aktivierungsfunktion  $\sigma$  ist entscheidend für die Leistungsfähigkeit des MLP, da es sonst linear wird

$$\vec{d}(\vec{c}) = \underline{W}^{LT} \dots \underline{W}^{1T} \vec{x} = \underline{\tilde{W}} \vec{x}$$

3. auch das MLP ist ein universeller Funktionsapproximator, mit mindestens einer versteckten Schicht ( $L > 1$ ) und genügend Neuronen hierbei werden (im Gegensatz zum Polynomklassifikator) **datenabhängige** Basisfunktionen verwendet, die in den versteckten Schichten berechnet werden
4. bei unserem Optimierungskriterium und Zielvektor gilt:  
MLP schätzt die a posteriori Wahrscheinlichkeit, d.h.

$$y_i^L \sim p(\omega_i \mid \vec{c})$$

(dies folgt letztlich aus dem allgemeinen Resultat des Quadratmittelansatzs)