JOHANN WOLFGANG GOETHE
UNIVERSITÄT
FRANKFURT AM MAIN

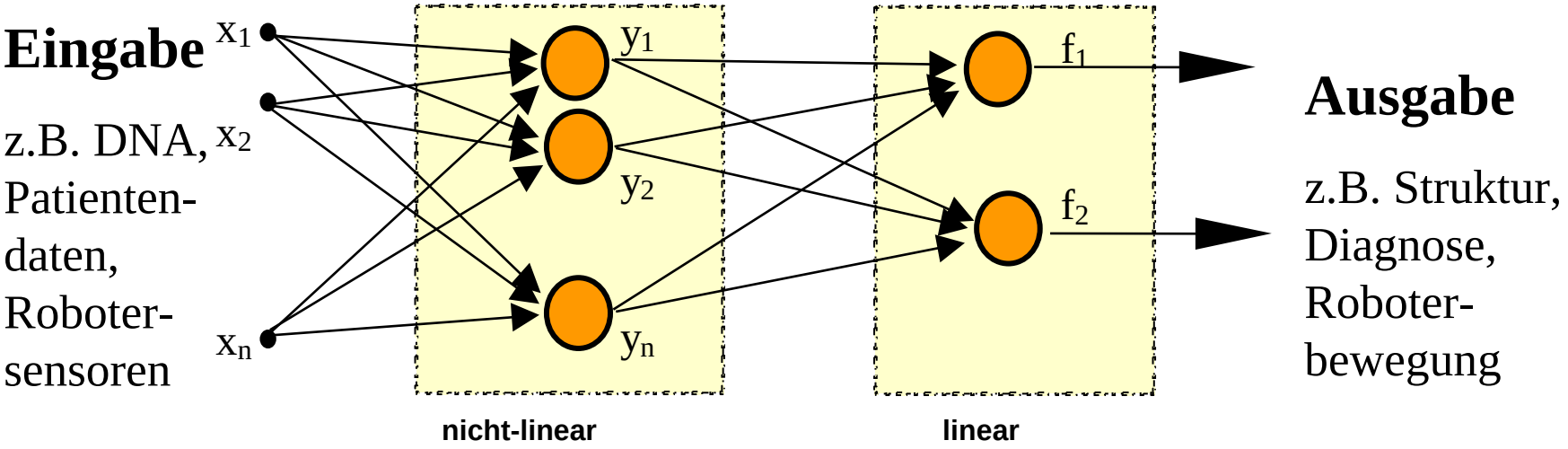
www.uni-frankfurt.de

Mehrschichten-Netze: Error-Backpropagation

Praktikum Adaptive Systeme

Mehrschichten-Netze

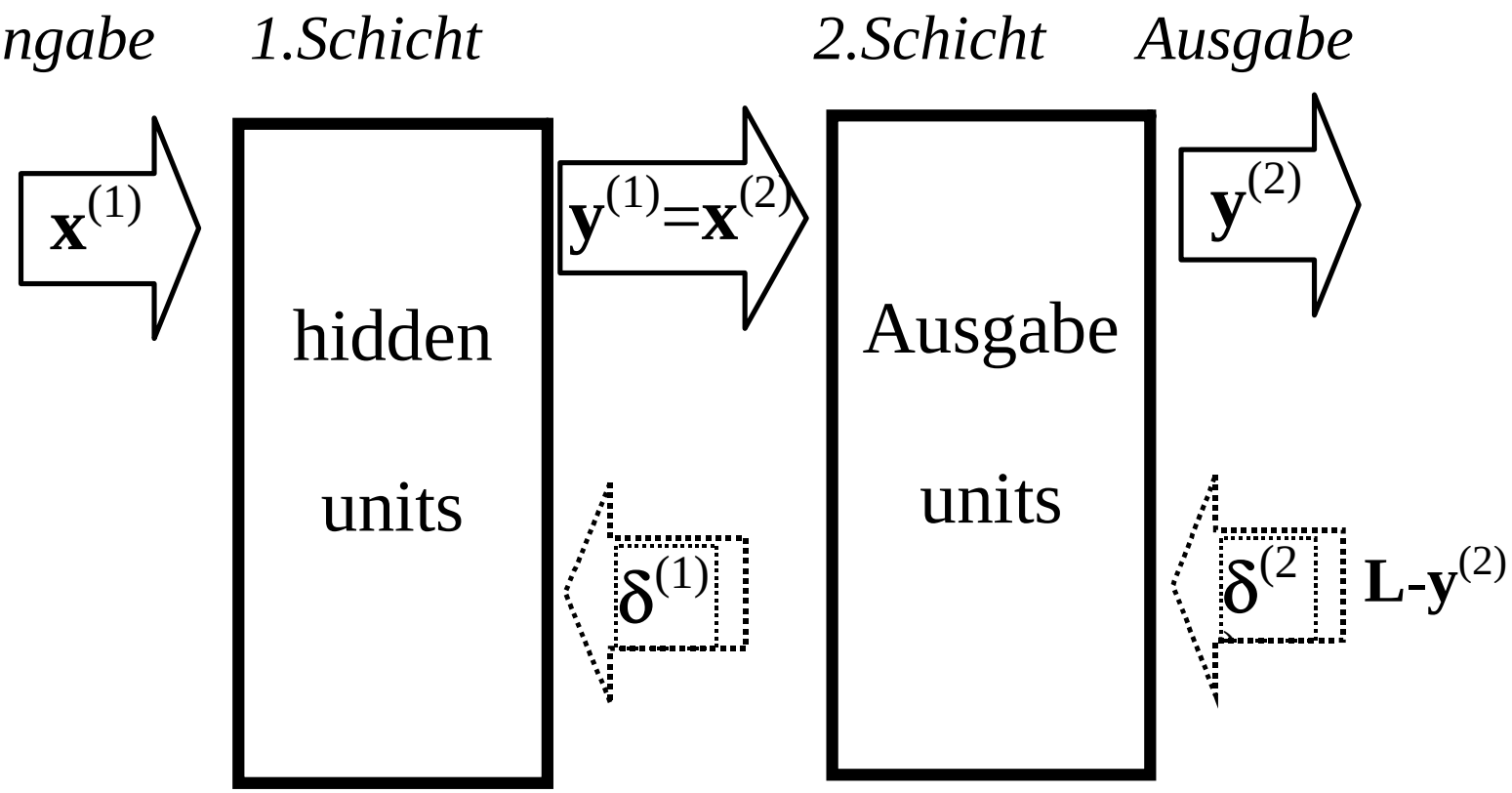
Fähigkeiten von Mehrschicht-Netzen:



- Ein 2-Schichtennetzwerk mit nicht-linearer Ausgabefunktion S(z) kann JEDE beliebige Funktion so genau wie gewünscht approximieren, wenn genügend Neuronen ex.

Neuronenzahl gegeben. Lernalgorithmus?

Backpropagation-Grundidee



Schichtweise Verbesserung durch Rückführung des Fehlers

Backpropagation-Lernregel letzte Schicht

JOHANN WOLFGANG GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Lernziel: $R(w^*) = \min E(y(w) - L(x))^2$ *min.mittl. quadr. Fehler*

$$w_i(t+1) = w_i(t) - \gamma \frac{\partial R}{\partial w_i}$$ Gradienten-Lernregel

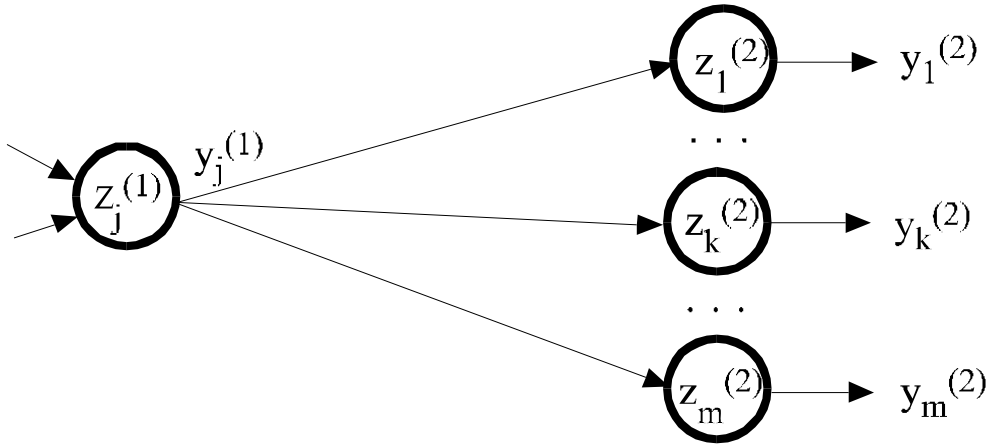
$$w_i(t+1) = w_i(t) - \gamma\,(y(w_i)-L(x))\frac{\partial S(z)}{\partial w_i}$$ *stoch. Approximation*

mit $$\frac{\partial S(z)}{\partial w_i} = \frac{S'(z)\partial z}{\partial w_i} = \frac{S'(z)\partial}{\partial w_i}\sum_j w_j x_j = S'(z)x_i$$

$\delta_i := -(y(w_i)-L(x))\,S'(z)x_i$

$\Delta w_{ij}(x) = \gamma\,\delta_i\,x_j$ *Delta-Regel*

Fehler-Backpropagation



Beeinflussung voriger Schichten $z_i^{(1)} \rightarrow R$

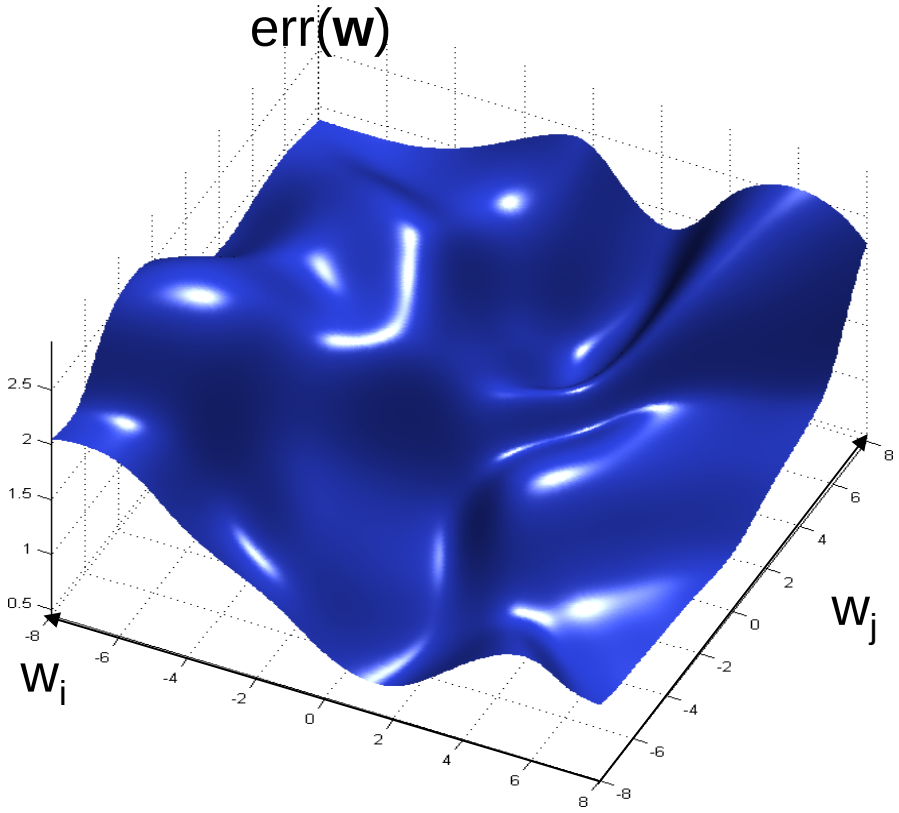
Delta-Regel für Schicht 1

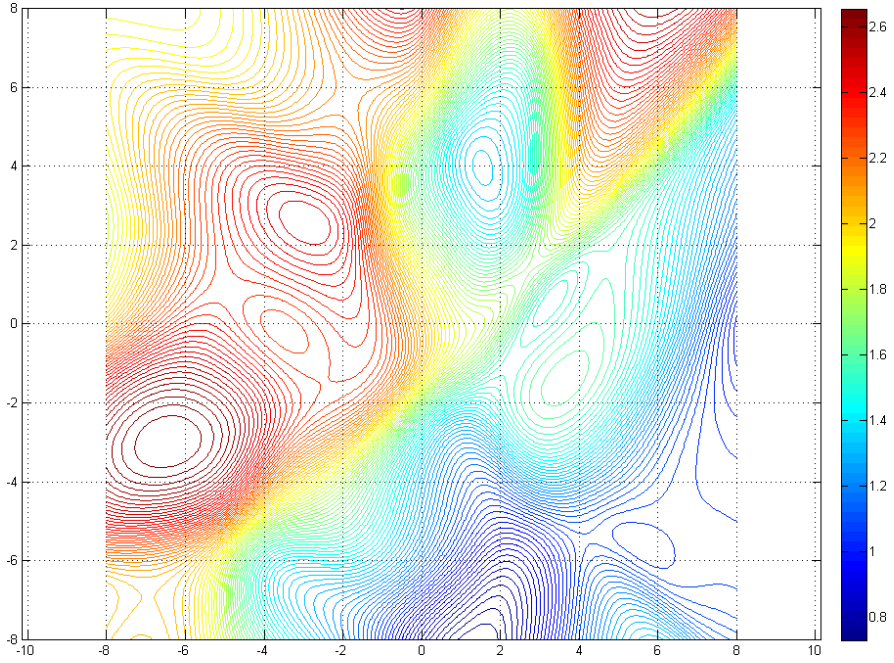
$$\delta_i^{(1)} = \frac{-\partial R_x}{\partial y_i^{(1)}} \frac{\partial y_i^{(1)}}{\partial z_i^{(1)}} = \left(\sum_{k=1}^{m} \delta_k^{(2)} w_{ki}^{(2)} \right) S'\left(z_i^{(1)}\right)$$

Nicht-lineare Ausgabe S(z)

- **Approximation durch Gradientenabstieg** in der Fehlerfunktion

Nicht-lineare Ausgabe y = S(z) wirkt sich aus: **zusätzliche lokale Minima**!

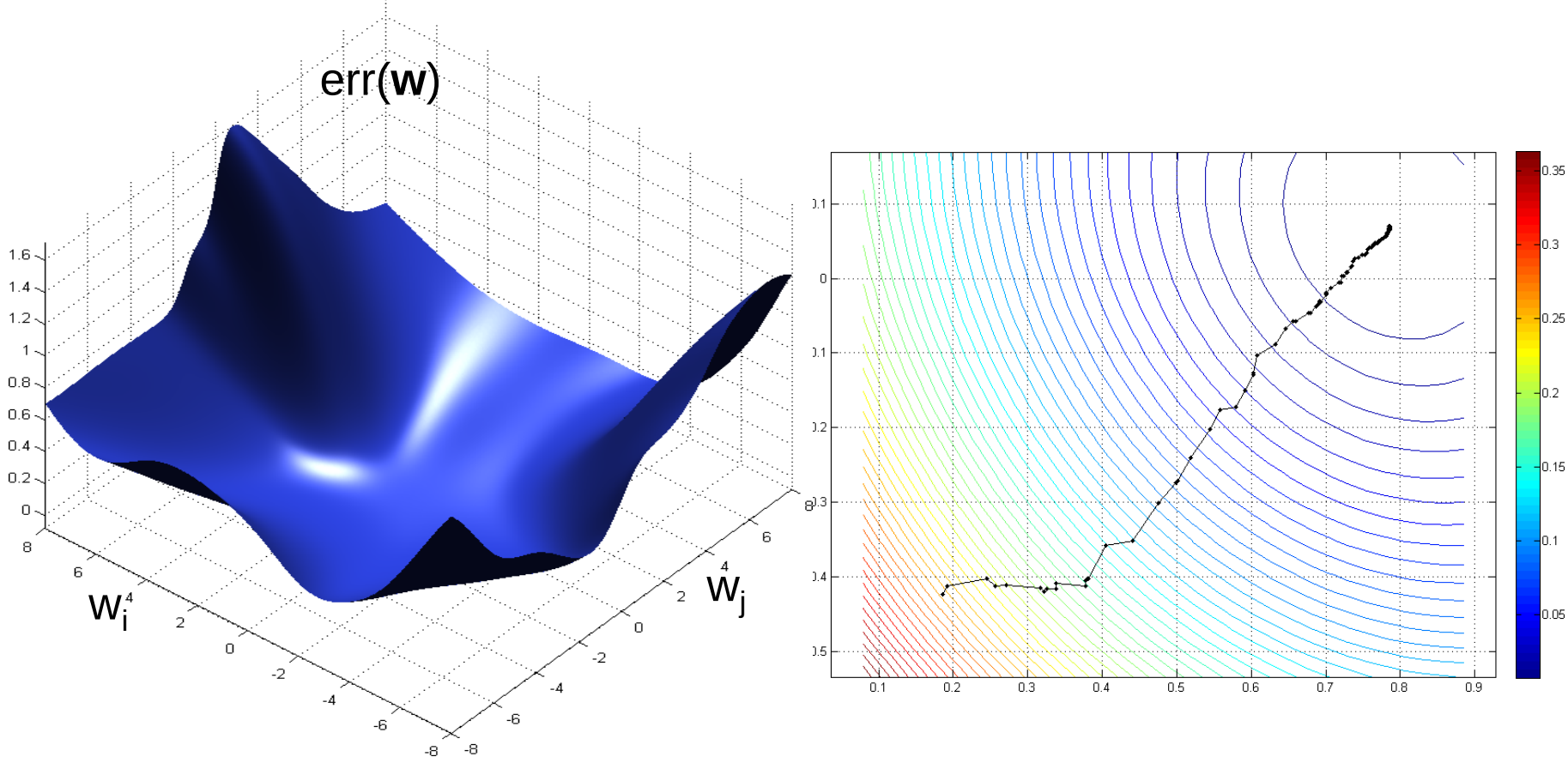




Nicht-lineare Ausgabe S(z)

- **Approximation durch Gradientenabstieg** in der Fehlerfunktion

  Anpassung der Gewichte = **Abstieg** innerhalb der Fehlerfunktion



Online vs Offline-Lernen

JOHANN WOLFGANG GOETHE
UNIVERSITÄT
FRANKFURT AM MAIN

- **ONLINE-Learning:**

```
WHILE NOT Abbruchbedingung erfüllt:
    Delta := 0
    FORALL Trainingsmuster x
        berechne Delta(W(x))
        W(t) := W(t-1) + Delta  // Lernen mit jedem Muster
    END FOR
END WHILE
```

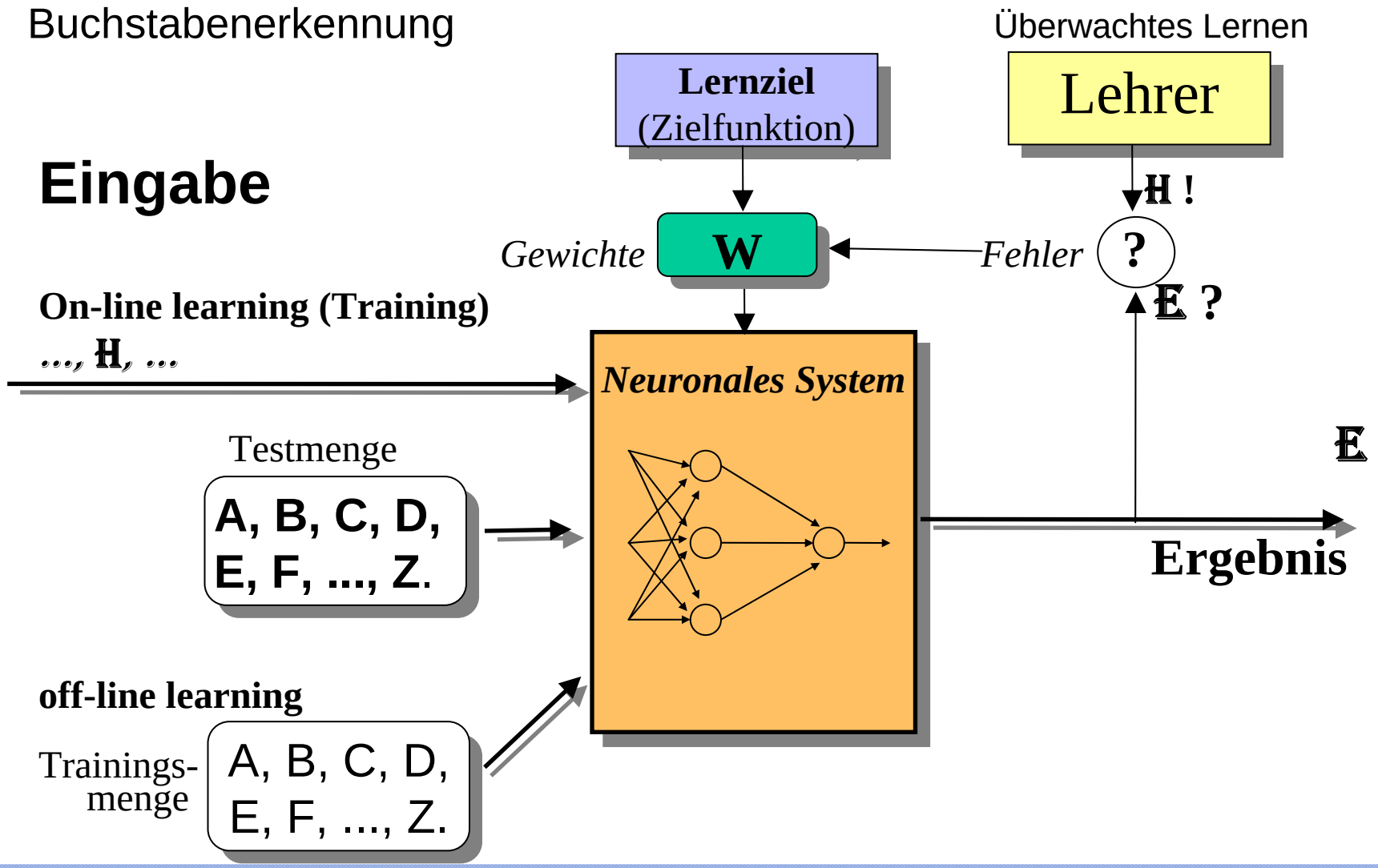
Online vs Offline-Lernen

JOHANN WOLFGANG GOETHE
UNIVERSITÄT
FRANKFURT AM MAIN

- **OFFLINE-Learning:**

```
WHILE NOT Abbruchbedingung erfüllt:
    GesamtDelta := 0
    FORALL Trainingsmuster x
        berechne Delta(W(x))
        GesamtDelta := GesamtDelta + Delta(W(x))
    END FOR
    W(t) := W(t-1) + GesamtDelta  // Lernen am Schluss!
END WHILE
```

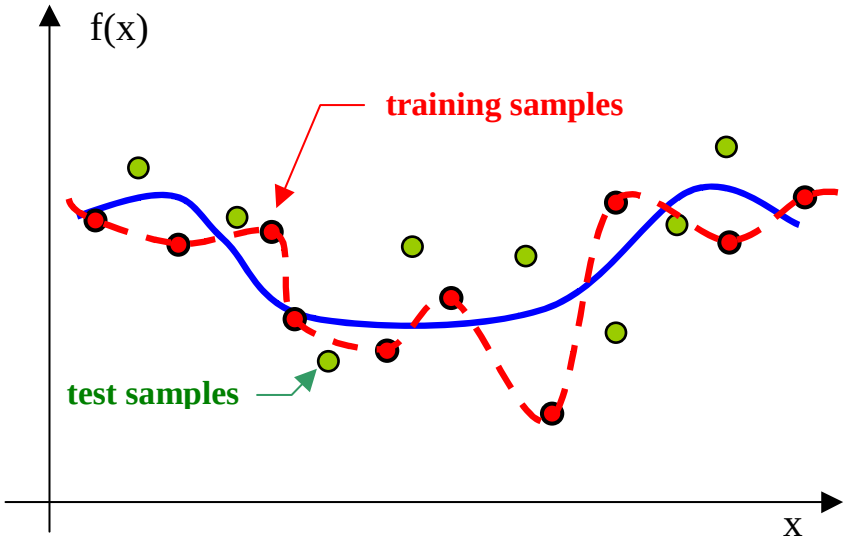
Arten des Lernens: Beispiel



Overfitting and Stopped Training

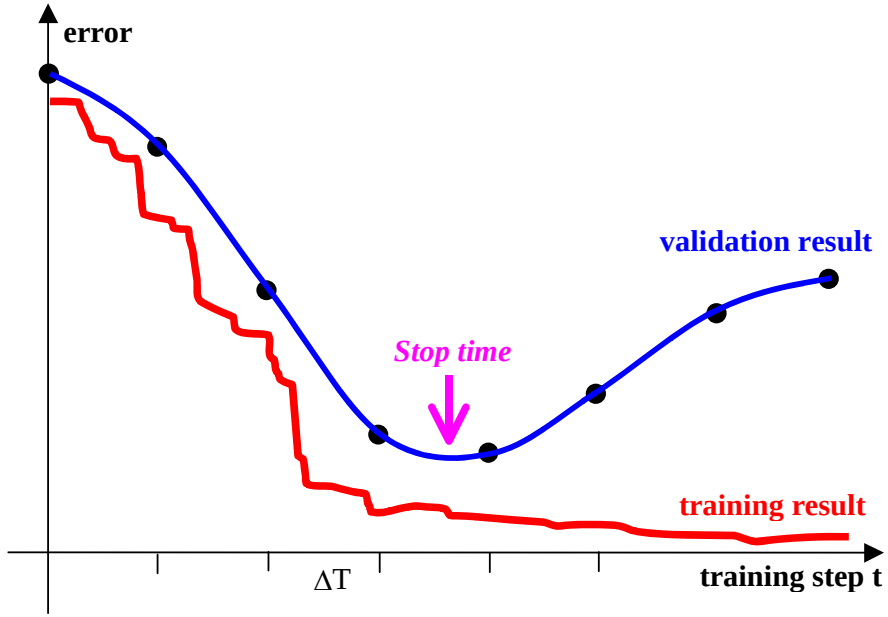
- **Problem**

  Überanpassung vs. Generalisierung



- **Lösung**

  stopped training



- **Wenige Daten** → *p-fold cross validation*:
  - *p-1* Trainingsmengen, 1 Testmenge
  - Bilde Mittelwert aus *p* Wiederholungen

Problem Ausgabefunktion

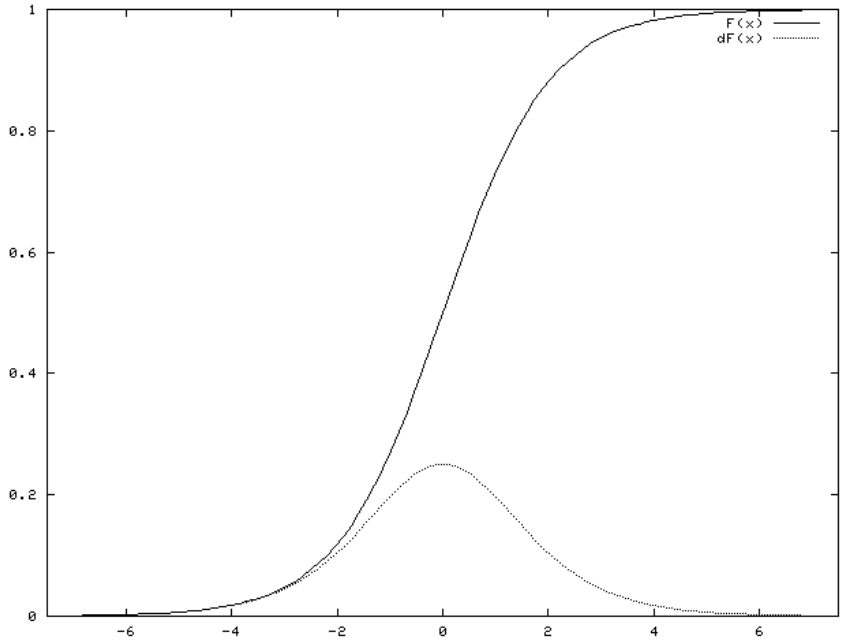
Bei Wahl von S = Fermifunktion

ist die Ableitung eine Glocken-Funktion mit S‘(-∞) = 0 = S‘(∞)

und damit bei sehr grossem oder kleinem x

$$\delta(x) = 0$$

Kein Lernen mehr möglich!



$$S'(z) = \frac{\partial}{\partial z}\frac{1}{1+e^{-z}} = \frac{+1-1+e^{-z}}{\left(1+e^{-z}\right)^2} = \left(1-S(z)\right)S(z)$$

Problem Ausgabefunktion

JOHANN WOLFGANG GOETHE
UNIVERSITÄT
FRANKFURT AM MAIN

Abhilfen:

- Andere Ausgabefunktion wählen (z.B. Sinus)
- Andere Zielfunktion wählen (Information statt quadr.Fehler)
- Andere Lernfunktion wählen, z.B. Ergänzung durch Backpropagation *Trägheitsmoment*

$$\delta(t+1) = \alpha\delta(t) + (1-\alpha)\delta(t-1) \qquad \text{z.B. } \alpha = 0.9$$

Fragen ?