

# Mehrschichtige Perzeptren

Multilayer Perceptrons (MLPs)

# Mehrschichtige Perzeptren

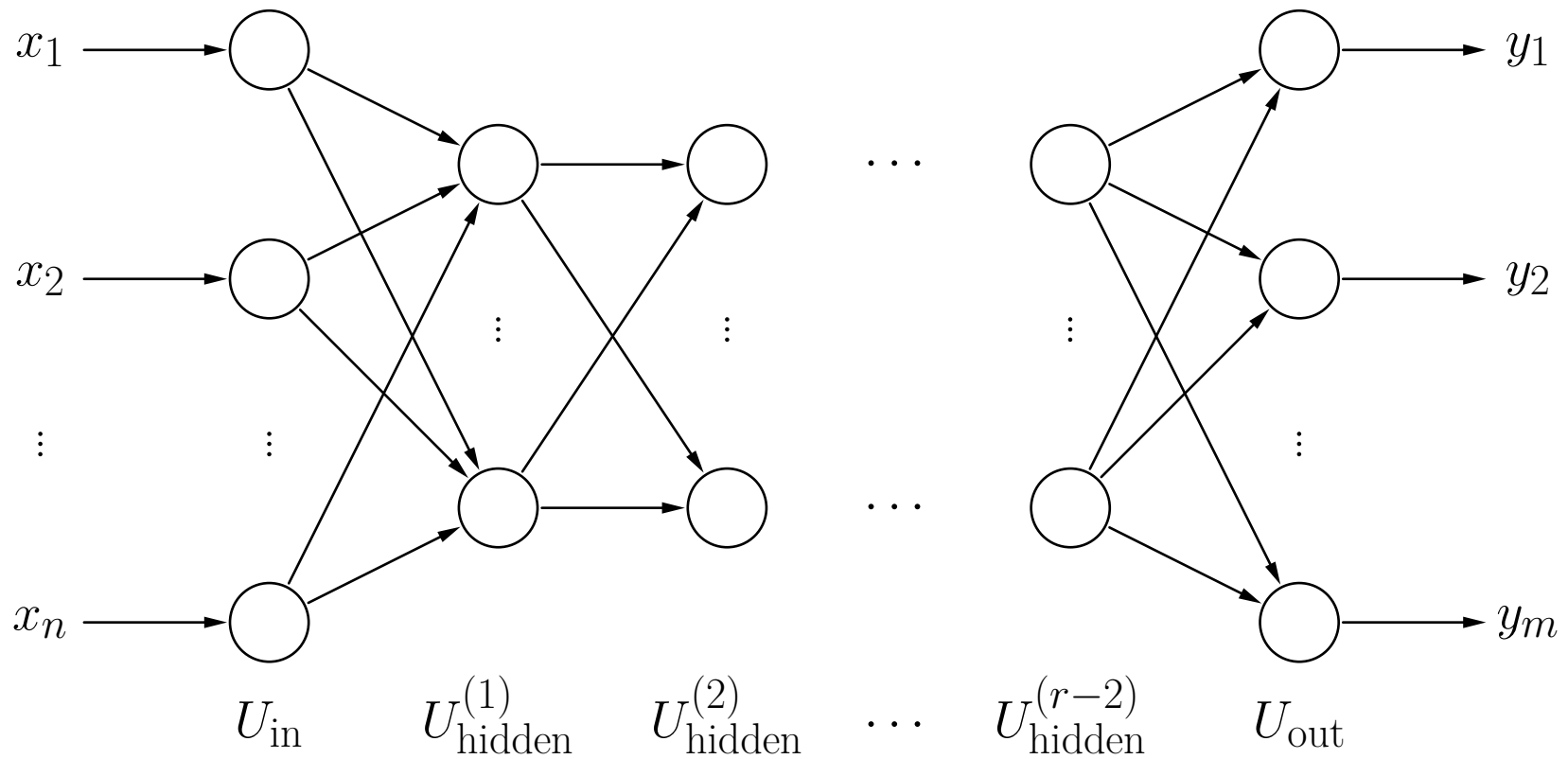
Ein **r-schichtiges Perzeptron** ist ein neuronales Netz mit einem Graph  $G = (U, C)$  das die folgenden Bedingungen erfüllt:

- (i)  $U_{\text{in}} \cap U_{\text{out}} = \emptyset$ ,
- (ii)  $U_{\text{hidden}} = U_{\text{hidden}}^{(1)} \cup \dots \cup U_{\text{hidden}}^{(r-2)}$ ,  
 $\forall 1 \leq i < j \leq r - 2 : U_{\text{hidden}}^{(i)} \cap U_{\text{hidden}}^{(j)} = \emptyset$ ,
- (iii)  $C \subseteq \left( U_{\text{in}} \times U_{\text{hidden}}^{(1)} \right) \cup \left( \bigcup_{i=1}^{r-3} U_{\text{hidden}}^{(i)} \times U_{\text{hidden}}^{(i+1)} \right) \cup \left( U_{\text{hidden}}^{(r-2)} \times U_{\text{out}} \right)$   
oder, falls keine versteckten Neuronen vorhanden ( $r = 2, U_{\text{hidden}} = \emptyset$ ),  
 $C \subseteq U_{\text{in}} \times U_{\text{out}}$ .

- Vorwärtsgerichtetes Netz mit streng geschichteter Struktur.

# Mehrschichtige Perzeptren

## Allgemeine Struktur eines mehrschichtigen Perzeptrons



# Mehrschichtige Perzeptren

- Die Netzwerkeingabe jedes versteckten Neurons und jedes Ausgabeneurons ist die **gewichtete Summe** seiner Eingaben, d.h.

$$\forall u \in U_{\text{hidden}} \cup U_{\text{out}} : \quad f_{\text{net}}^{(u)}(\mathbf{w}_u, \mathbf{in}_u) = \mathbf{w}_u \mathbf{in}_u = \sum_{v \in \text{pred}(u)} w_{uv} \text{out}_v .$$

- Die Aktivierungsfunktion jedes versteckten Neurons ist eine sogenannte **Sigmoide**, d.h. eine monoton steigende Funktion

$$f : \mathbb{R} \rightarrow [0, 1] \quad \text{with} \quad \lim_{x \rightarrow -\infty} f(x) = 0 \quad \text{and} \quad \lim_{x \rightarrow \infty} f(x) = 1 .$$

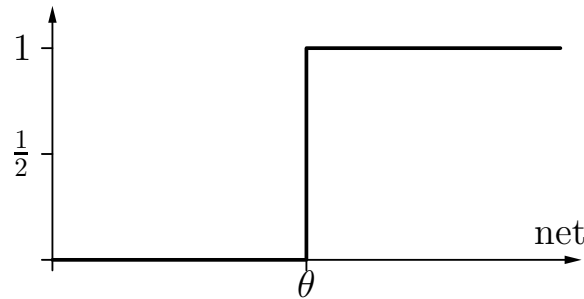
- Die Aktivierungsfunktion jedes Ausgabeneurons ist ebenfalls entweder eine Sigmoide oder eine **lineare Funktion**, d.h.

$$f_{\text{act}}(\text{net}, \theta) = \alpha \text{net} - \theta .$$

# Sigmoide als Aktivierungsfunktionen

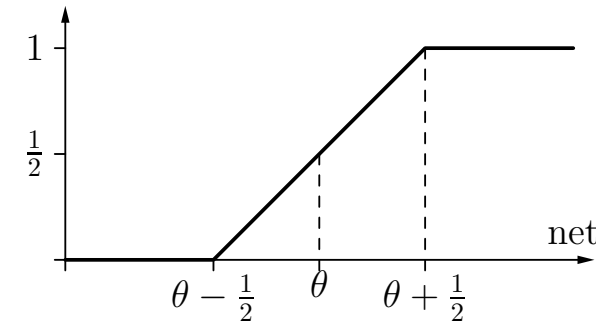
Stufenfunktion:

$$f_{\text{act}}(\text{net}, \theta) = \begin{cases} 1, & \text{falls } \text{net} \geq \theta, \\ 0, & \text{sonst.} \end{cases}$$



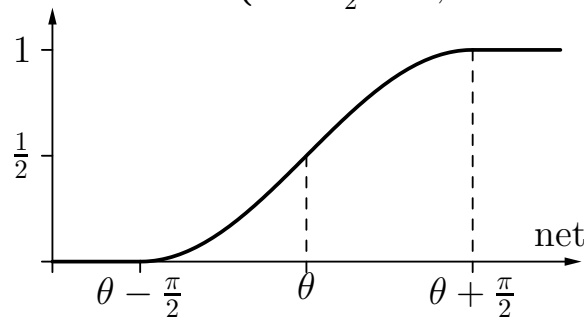
Semilineare Funktion:

$$f_{\text{act}}(\text{net}, \theta) = \begin{cases} 1, & \text{falls } \text{net} > \theta + \frac{1}{2}, \\ 0, & \text{falls } \text{net} < \theta - \frac{1}{2}, \\ (\text{net} - \theta) + \frac{1}{2}, & \text{sonst.} \end{cases}$$



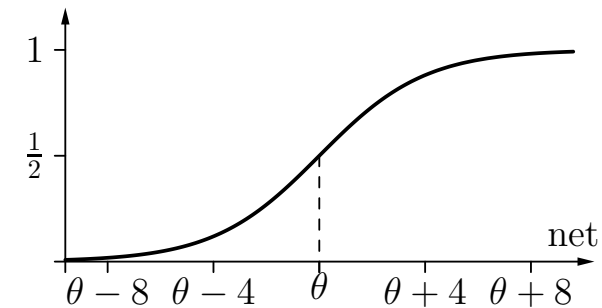
Sinus bis Sättigung:

$$f_{\text{act}}(\text{net}, \theta) = \begin{cases} 1, & \text{falls } \text{net} > \theta + \frac{\pi}{2}, \\ 0, & \text{falls } \text{net} < \theta - \frac{\pi}{2}, \\ \frac{\sin(\text{net} - \theta) + 1}{2}, & \text{sonst.} \end{cases}$$



Logistische Funktion:

$$f_{\text{act}}(\text{net}, \theta) = \frac{1}{1 + e^{-(\text{net} - \theta)}}$$

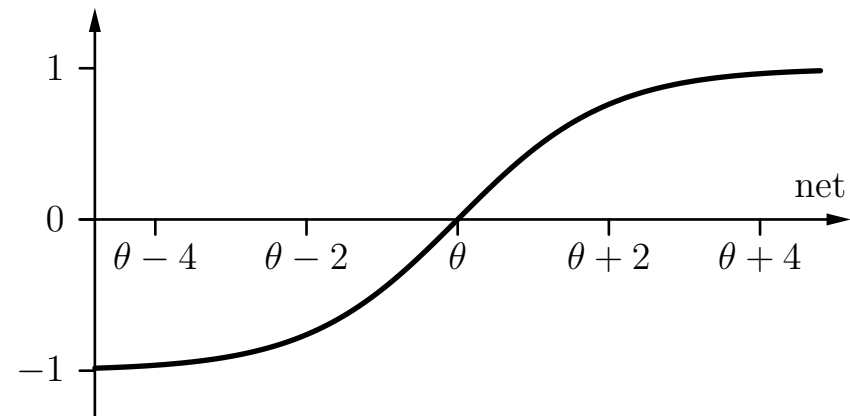


# Sigmoide als Aktivierungsfunktionen

- Alle Sigmoiden auf der vorherigen Folie sind **unipolar**, d.h. sie reichen von 0 bis 1.
- Manchmal werden **bipolare** sigmoidale Funktionen verwendet, wie beispielsweise der *tangens hyperbolicus*.

tangens hyperbolicus:

$$f_{\text{act}}(\text{net}, \theta) = \tanh(\text{net} - \theta)$$
$$= \frac{2}{1 + e^{-2(\text{net} - \theta)}} - 1$$



# Mehrschichtige Perzeptren: Gewichtsmatrizen

Seien  $U_1 = \{v_1, \dots, v_m\}$  and  $U_2 = \{u_1, \dots, u_n\}$  die Neuronen zwei aufeinanderfolgender Schichten eines MLP.

Ihre Verbindungsgewichte werden dargestellt als eine  $n \times m$ -Matrix

$$\mathbf{W} = \begin{pmatrix} w_{u_1v_1} & w_{u_1v_2} & \dots & w_{u_1v_m} \\ w_{u_2v_1} & w_{u_2v_2} & \dots & w_{u_2v_m} \\ \vdots & \vdots & & \vdots \\ w_{u_nv_1} & w_{u_nv_2} & \dots & w_{u_nv_m} \end{pmatrix},$$

wobei  $w_{u_iv_j} = 0$ , falls es keine Verbindung von Neuron  $v_j$  zu Neuron  $u_i$  gibt.

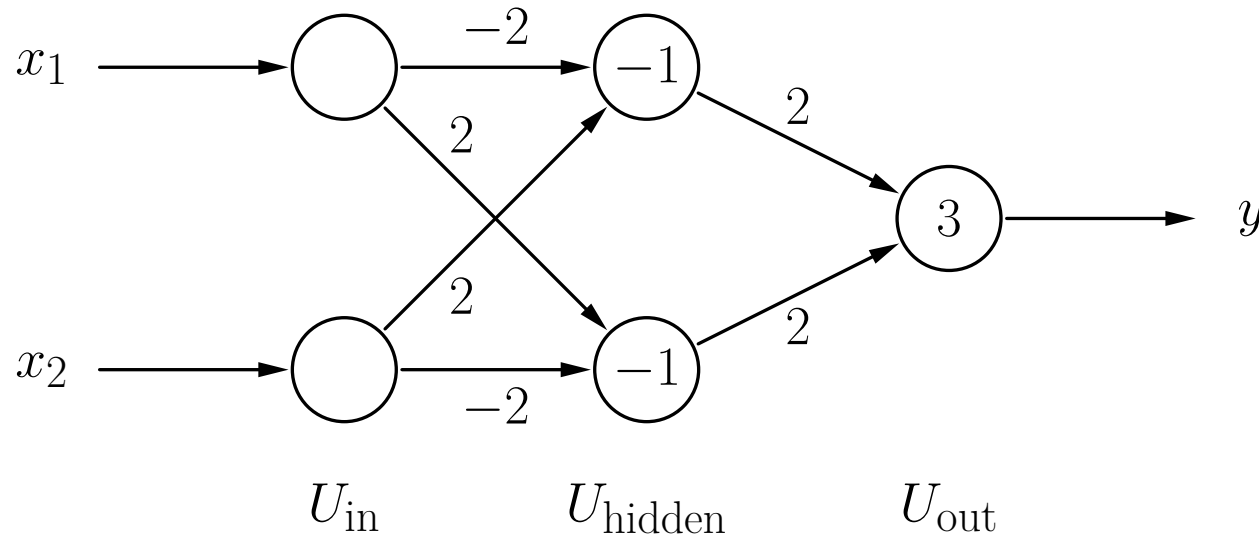
Vorteil: Die Berechnung der Netzwerkeingabe kann geschrieben werden als

$$\mathbf{net}_{U_2} = \mathbf{W} \cdot \mathbf{in}_{U_2} = \mathbf{W} \cdot \mathbf{out}_{U_1}$$

wobei  $\mathbf{net}_{U_2} = (\text{net}_{u_1}, \dots, \text{net}_{u_n})^\top$  und  $\mathbf{in}_{U_2} = \mathbf{out}_{U_1} = (\text{out}_{v_1}, \dots, \text{out}_{v_m})^\top$ .

# Mehrschichtige Perzeptren: Biimplication

Lösen des Biimplikationsproblems mit einem MLP.

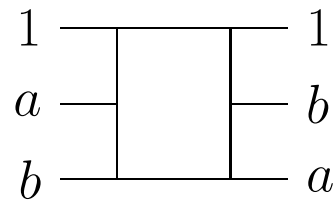
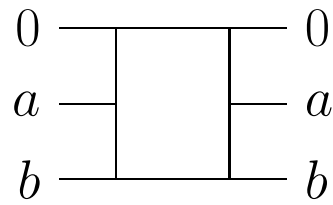
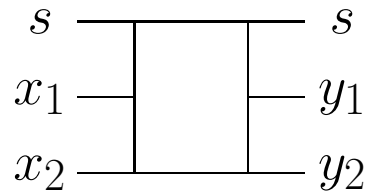


Man beachte die zusätzlichen Eingabeneuronen im Vergleich zur Lösung mit Schwellenwertelementen.

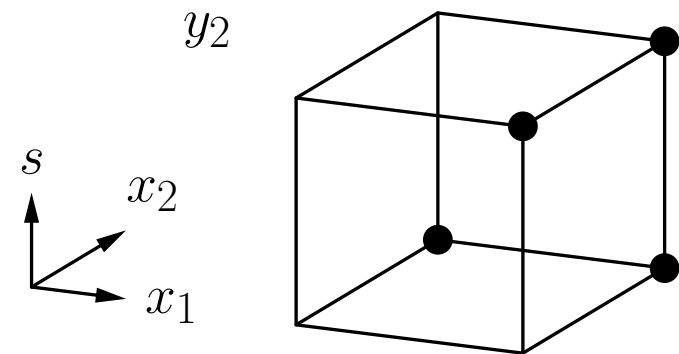
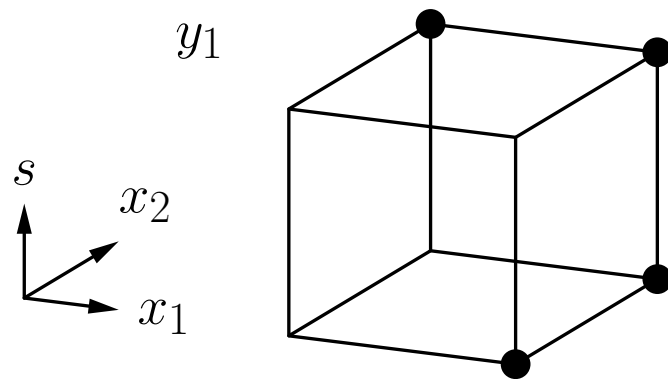
$$\mathbf{W}_1 = \begin{pmatrix} -2 & 2 \\ 2 & -2 \end{pmatrix} \quad \text{und} \quad \mathbf{W}_2 = \begin{pmatrix} 2 & 2 \end{pmatrix}$$



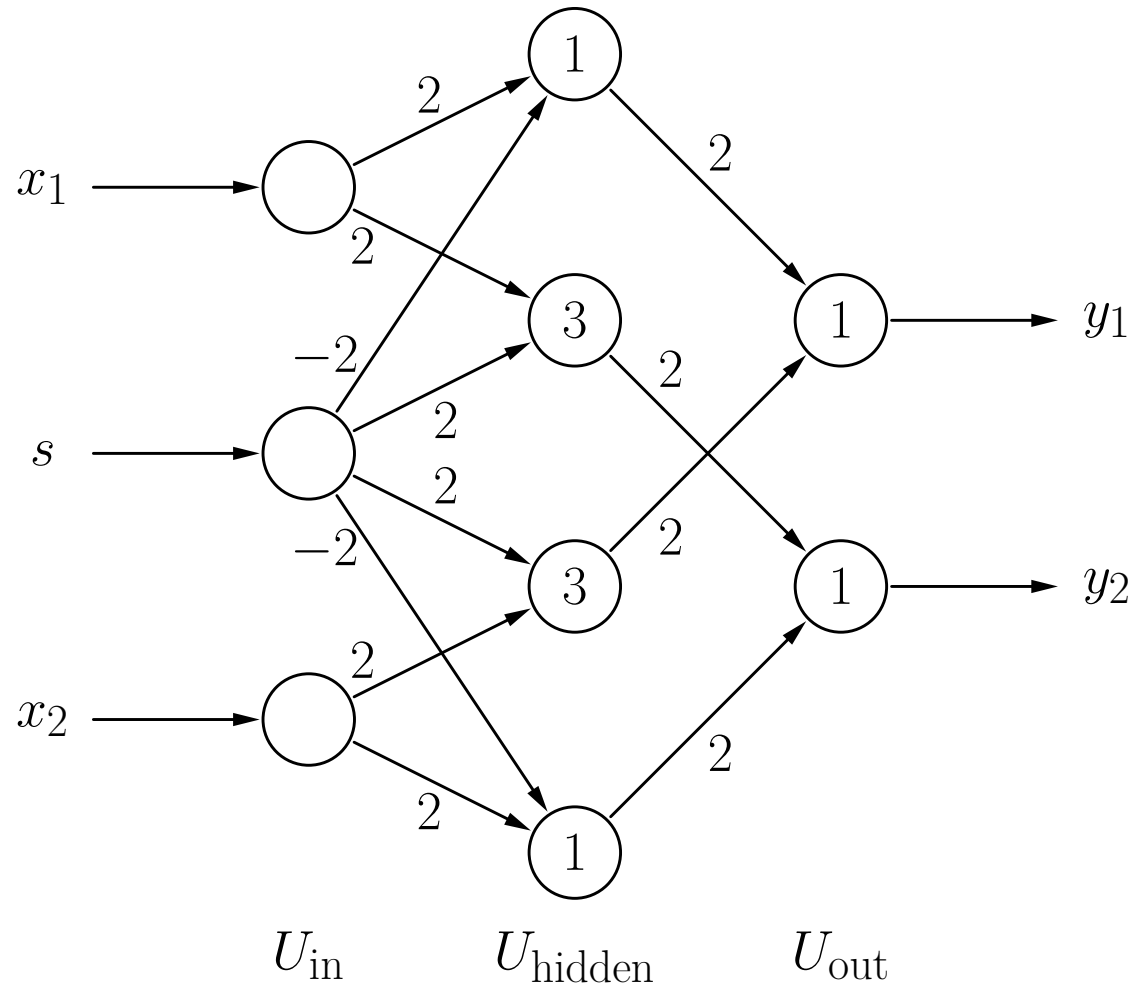
# Mehrschichtige Perzeptren: Fredkin-Gatter



$s$	0	0	0	0	1	1	1	1
$x_1$	0	0	1	1	0	0	1	1
$x_2$	0	1	0	1	0	1	0	1
$y_1$	0	0	1	1	0	1	0	1
$y_2$	0	1	0	1	0	0	1	1



# Mehrschichtige Perzeptren: Fredkin-Gatter



$$\mathbf{W}_1 = \begin{pmatrix} 2 & -2 & 0 \\ 2 & 2 & 0 \\ 0 & 2 & 2 \\ 0 & -2 & 2 \end{pmatrix}$$

$$\mathbf{W}_2 = \begin{pmatrix} 2 & 0 & 2 & 0 \\ 0 & 2 & 0 & 2 \end{pmatrix}$$

# Warum nicht-lineare Aktivierungsfunktionen?

In Matrixschreibweise ergibt sich für zwei aufeinanderfolgende Schichten  $U_1$  und  $U_2$

$$\mathbf{net}_{U_2} = \mathbf{W} \cdot \mathbf{in}_{U_2} = \mathbf{W} \cdot \mathbf{out}_{U_1}.$$

Wenn die Aktivierungsfunktionen linear sind, d.h.

$$f_{\text{act}}(\text{net}, \theta) = \alpha \text{net} - \theta.$$

können die Aktivierungen der Neuronen in der Schicht  $U_2$  wie folgt berechnet werden:

$$\mathbf{act}_{U_2} = \mathbf{D}_{\text{act}} \cdot \mathbf{net}_{U_2} - \boldsymbol{\theta},$$

wobei

- $\mathbf{act}_{U_2} = (\text{act}_{u_1}, \dots, \text{act}_{u_n})^\top$  der Aktivierungsvektor ist,
- $\mathbf{D}_{\text{act}}$  eine  $n \times n$  Diagonalmatrix aus den Faktoren  $\alpha_{u_i}$ ,  $i = 1, \dots, n$ , ist und
- $\boldsymbol{\theta} = (\theta_{u_1}, \dots, \theta_{u_n})^\top$  der Bias-Vektor ist.

# Warum nicht-lineare Aktivierungsfunktionen?

Falls die Ausgabefunktion auch linear ist, gilt analog

$$\mathbf{out}_{U_2} = \mathbf{D}_{\text{out}} \cdot \mathbf{act}_{U_2} - \boldsymbol{\xi},$$

wobei

- $\mathbf{out}_{U_2} = (\text{out}_{u_1}, \dots, \text{out}_{u_n})^\top$  der Ausgabevektor ist,
- $\mathbf{D}_{\text{out}}$  wiederum eine  $n \times n$  Diagonalmatrix aus Faktoren ist und
- $\boldsymbol{\xi} = (\xi_{u_1}, \dots, \xi_{u_n})^\top$  ein Biasvektor ist.

In Kombination ergibt sich

$$\mathbf{out}_{U_2} = \mathbf{D}_{\text{out}} \cdot (\mathbf{D}_{\text{act}} \cdot (\mathbf{W} \cdot \mathbf{out}_{U_1}) - \boldsymbol{\theta}) - \boldsymbol{\xi}$$

und daher

$$\mathbf{out}_{U_2} = \mathbf{A}_{12} \cdot \mathbf{out}_{U_1} + \mathbf{b}_{12}$$

mit einer  $n \times m$ -Matrix  $\mathbf{A}_{12}$  und einem  $n$ -dimensionalen Vektor  $\mathbf{b}_{12}$ .

# Warum nicht-lineare Aktivierungsfunktionen?

Daher ergibt sich

$$\mathbf{out}_{U_2} = \mathbf{A}_{12} \cdot \mathbf{out}_{U_1} + \mathbf{b}_{12}$$

und

$$\mathbf{out}_{U_3} = \mathbf{A}_{23} \cdot \mathbf{out}_{U_2} + \mathbf{b}_{23}$$

für die Berechnungen zwei aufeinanderfolgender Schichten  $U_2$  und  $U_3$ .

Diese beiden Berechnungen können kombiniert werden zu

$$\mathbf{out}_{U_3} = \mathbf{A}_{13} \cdot \mathbf{out}_{U_1} + \mathbf{b}_{13},$$

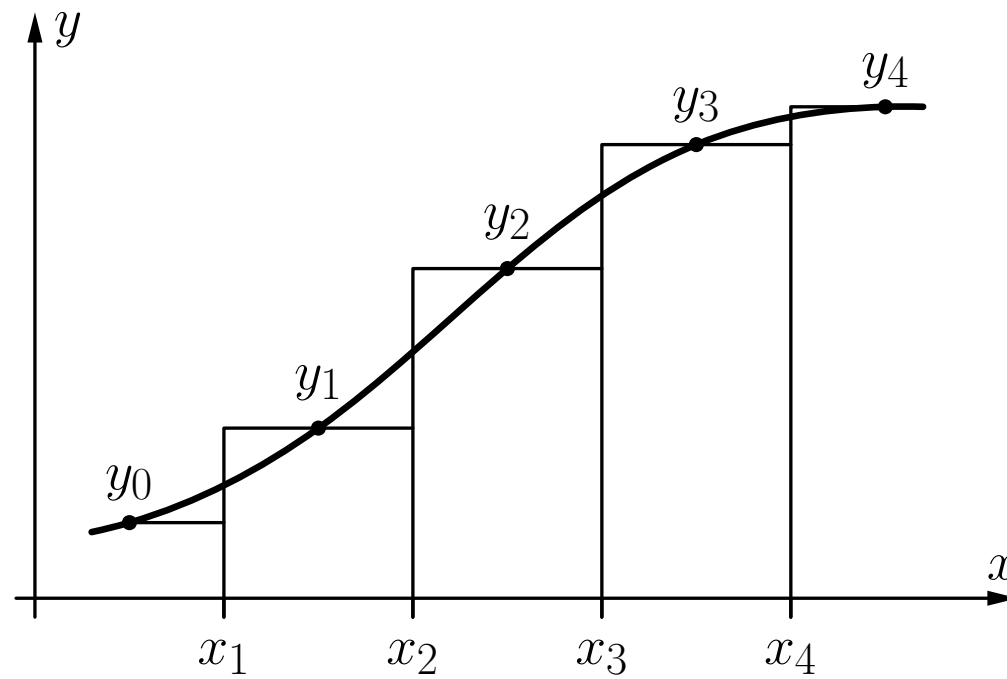
wobei  $\mathbf{A}_{13} = \mathbf{A}_{23} \cdot \mathbf{A}_{12}$  und  $\mathbf{b}_{13} = \mathbf{A}_{23} \cdot \mathbf{b}_{12} + \mathbf{b}_{23}$ .

**Ergebnis:** Mit linearen Aktivierungs- und Ausgabefunktionen können beliebige mehrschichtige Perzeptren auf zweischichtige Perzeptren reduziert werden.

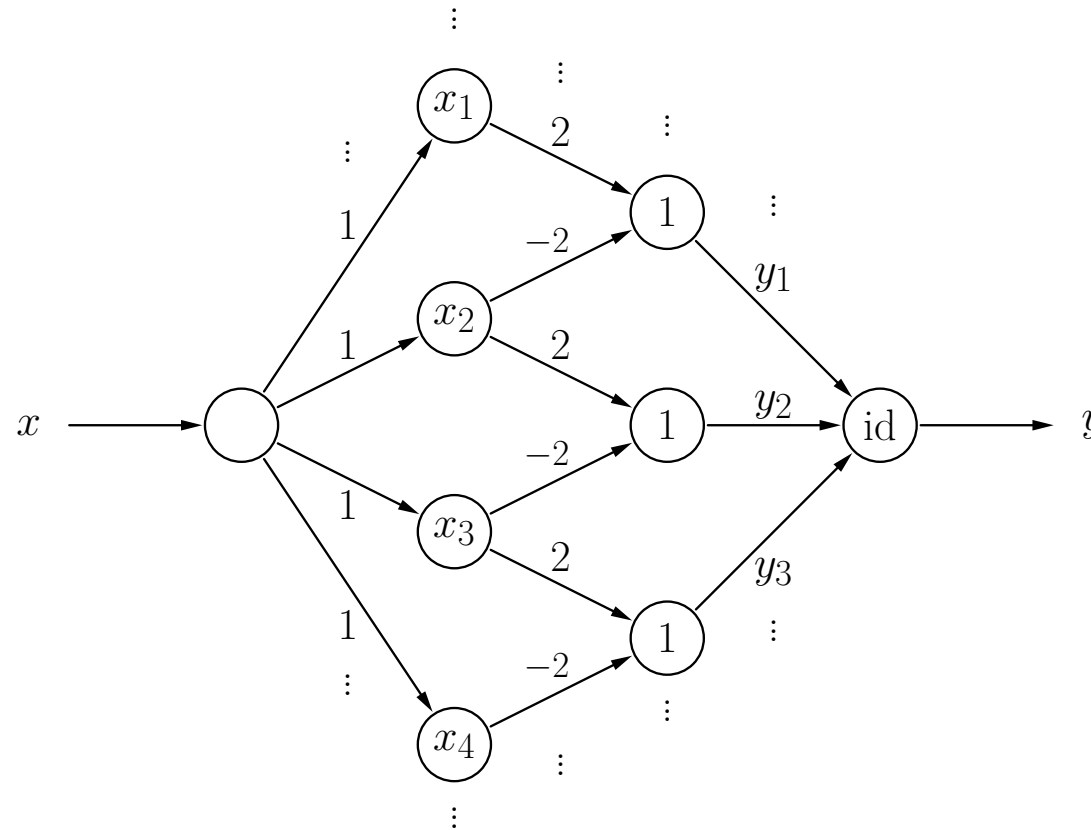
# Mehrschichtige Perzeptren: Funktionsapproximation

## Idee der Funktionsapproximation

- Nähere eine gegebene Funktion durch eine Stufenfunktion an.
- Konstruiere ein neuronales Netz, das die Stufenfunktion berechnet.



# Mehrschichtige Perzeptren: Funktionsapproximation

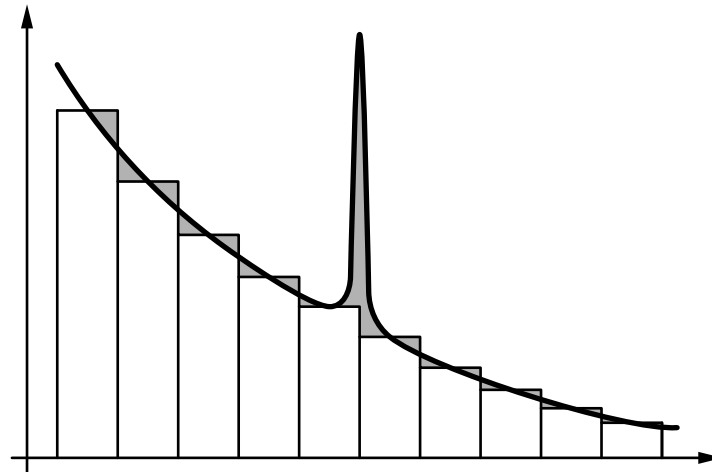


Ein neuronales Netz, das die Treppenfunktion von der vorherigen Folie berechnet. Es ist immer nur eine Stufe passend zum Eingabewert aktiv und die Stufenhöhe wird ausgegeben.

# Mehrschichtige Perzeptren: Funktionsapproximation

**Theorem:** Jede Riemann-integrierbare Funktion kann mit beliebiger Genauigkeit durch ein vierschichtiges MLP berechnet werden.

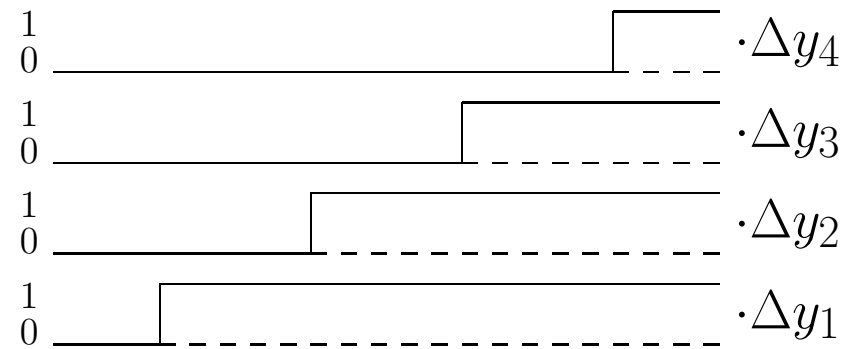
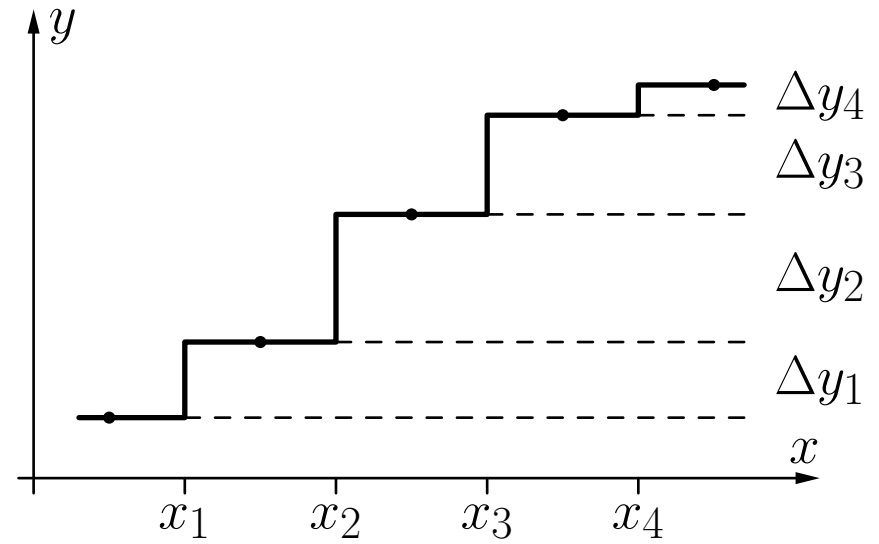
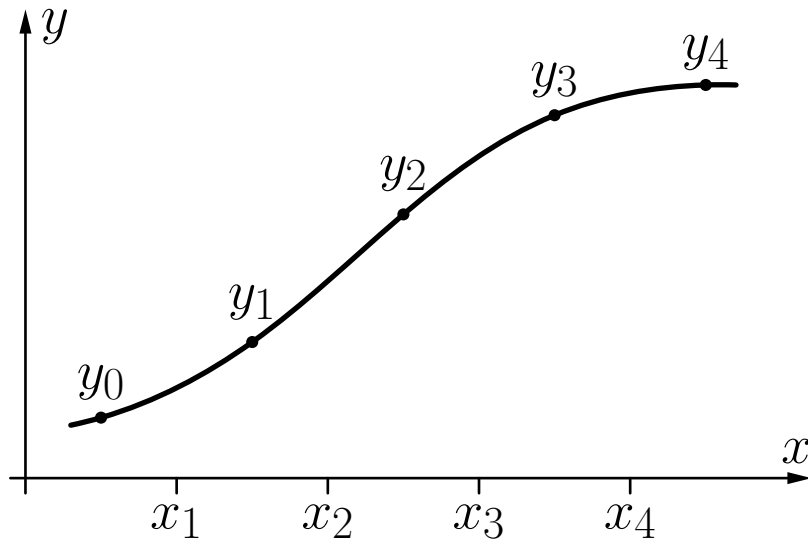
- Aber: Fehler wird bestimmt als die **Fläche** zwischen Funktionen.



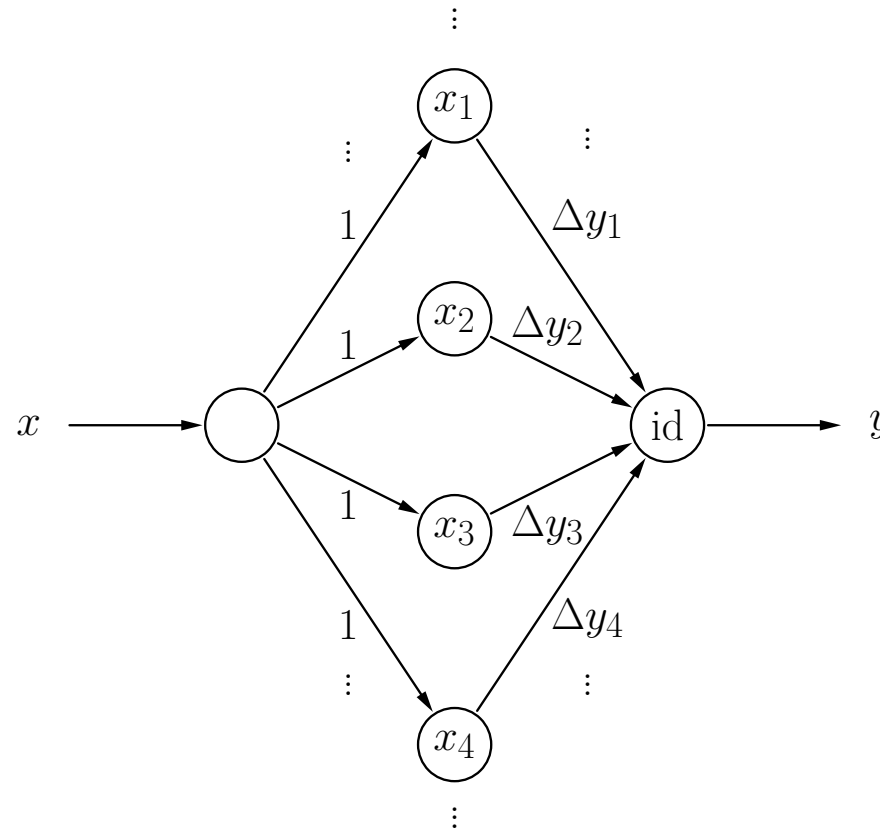
- Weitere mathematische Untersuchungen zeigen, dass sogar gilt: Mit einem dreischichtigen Perzeptron kann jede stetige Funktion mit beliebiger Genauigkeit angenähert werden (Fehlerbestimmung: maximale Differenz der Funktionswerte).



# Mehrschichtige Perzeptren: Funktionsapproximation

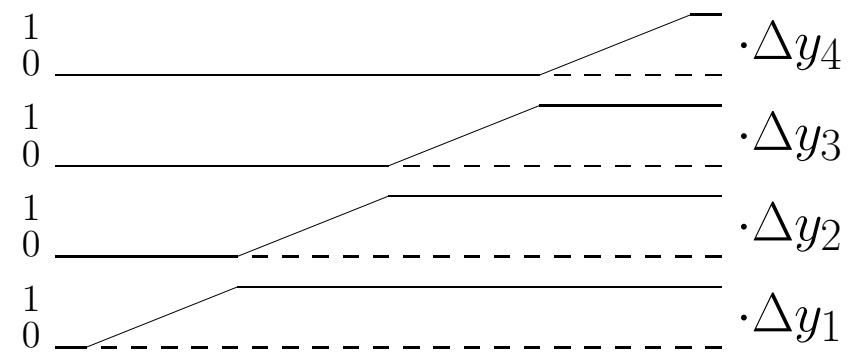
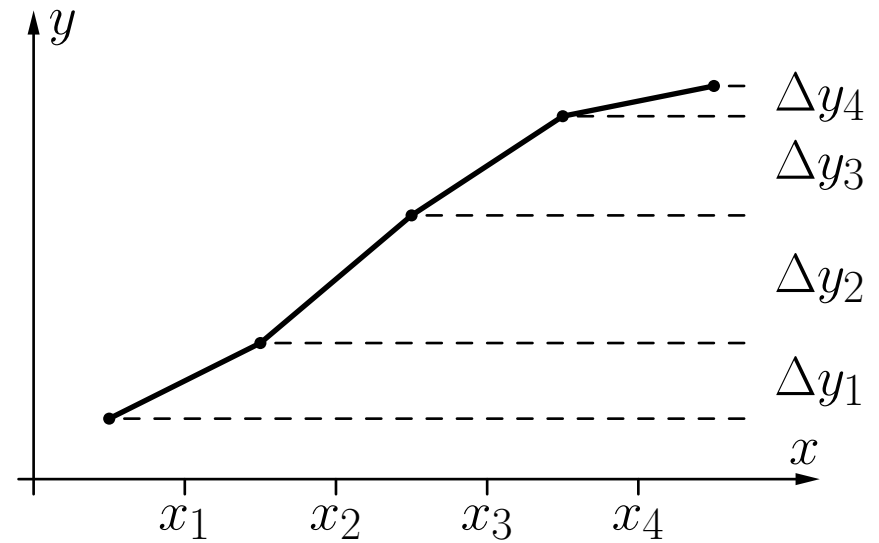
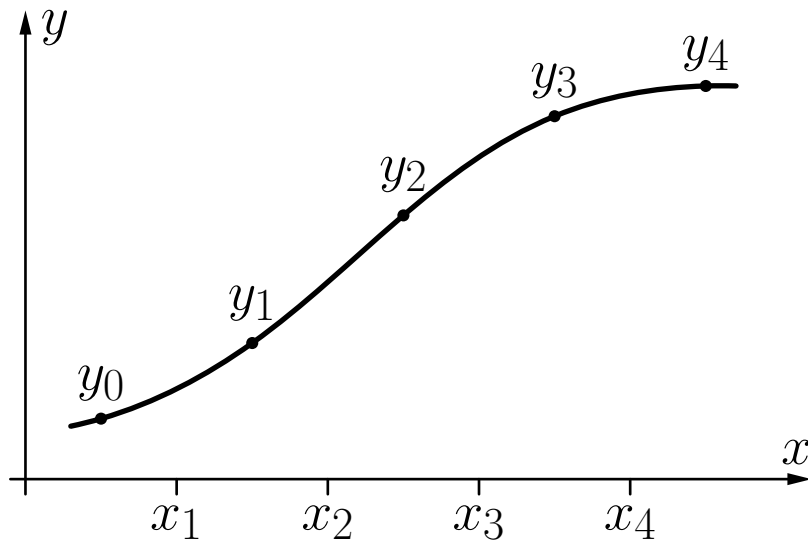


# Mehrschichtige Perzeptren: Funktionsapproximation

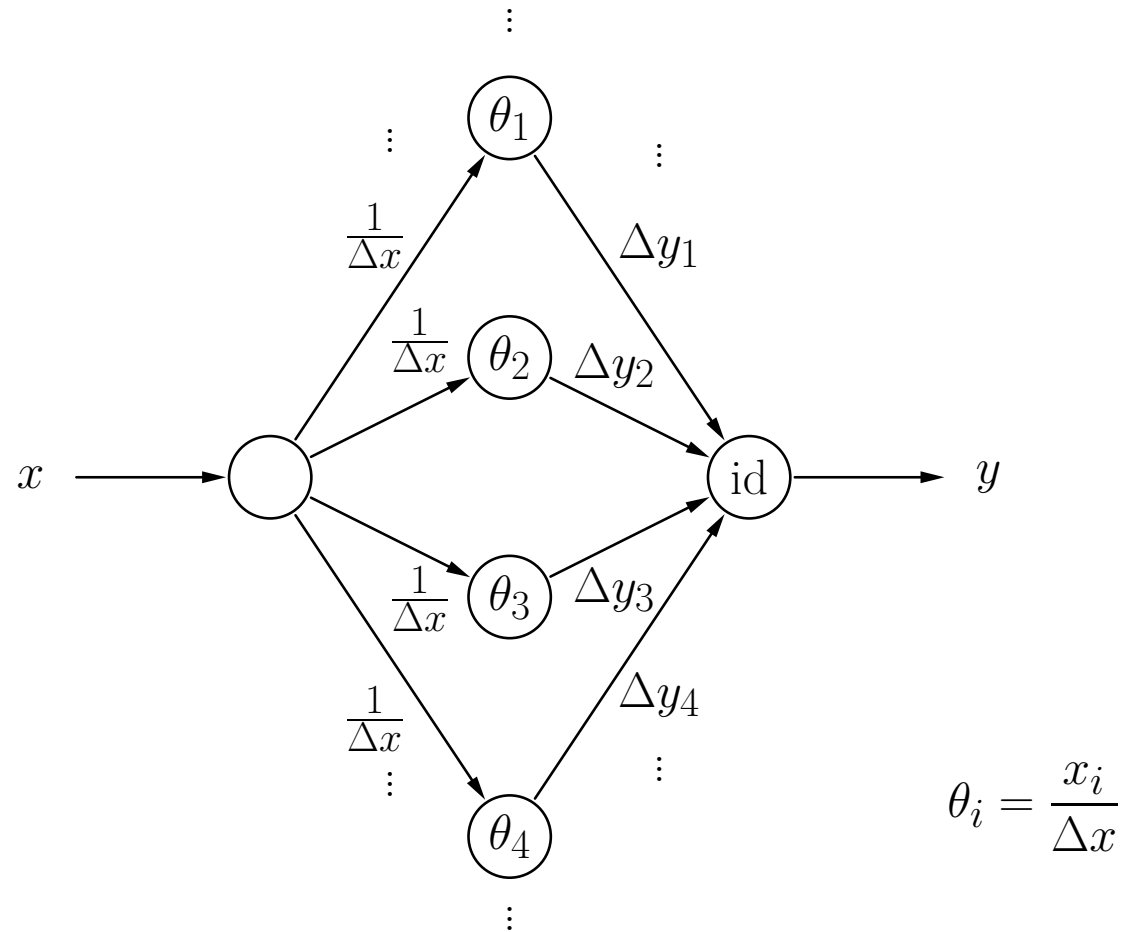


Ein neuronales Netz, das die Treppenfunktion von der vorherigen Folie als gewichtete Summe von Sprungfunktionen berechnet.

# Mehrschichtige Perzeptren: Funktionsapproximation



# Mehrschichtige Perzeptren: Funktionsapproximation



Ein neuronales Netz, das die stückweise lineare Funktion von der vorherigen Folie durch eine gewichtete Summe von semi-linearen Funktionen berechnet, wobei  $\Delta x = x_{i+1} - x_i$ .

# Mathematischer Hintergrund: Regression

# Mathematischer Hintergrund: Lineare Regression

## Das Trainieren von NN ist stark verwandt mit Regression

- Geg:
- Ein Datensatz  $((x_1, y_1), \dots, (x_n, y_n))$  aus  $n$  Daten-Tupeln und
  - Hypothese über den funktionellen Zusammenhang, also z.B.  $y = g(x) = a + bx$ .

Idee: Minimiere die Summe der quadrierten Fehler, d.h.

$$F(a, b) = \sum_{i=1}^n (g(x_i) - y_i)^2 = \sum_{i=1}^n (a + bx_i - y_i)^2.$$

Notwendige Bedingungen für ein Minimum:

$$\frac{\partial F}{\partial a} = \sum_{i=1}^n 2(a + bx_i - y_i) = 0 \quad \text{und}$$

$$\frac{\partial F}{\partial b} = \sum_{i=1}^n 2(a + bx_i - y_i)x_i = 0$$

# Mathematischer Hintergrund: Lineare Regression

Resultat der notwendigen Bedingungen: System sogenannter **Normalgleichungen**, d.h.

$$na + \left( \sum_{i=1}^n x_i \right) b = \sum_{i=1}^n y_i,$$

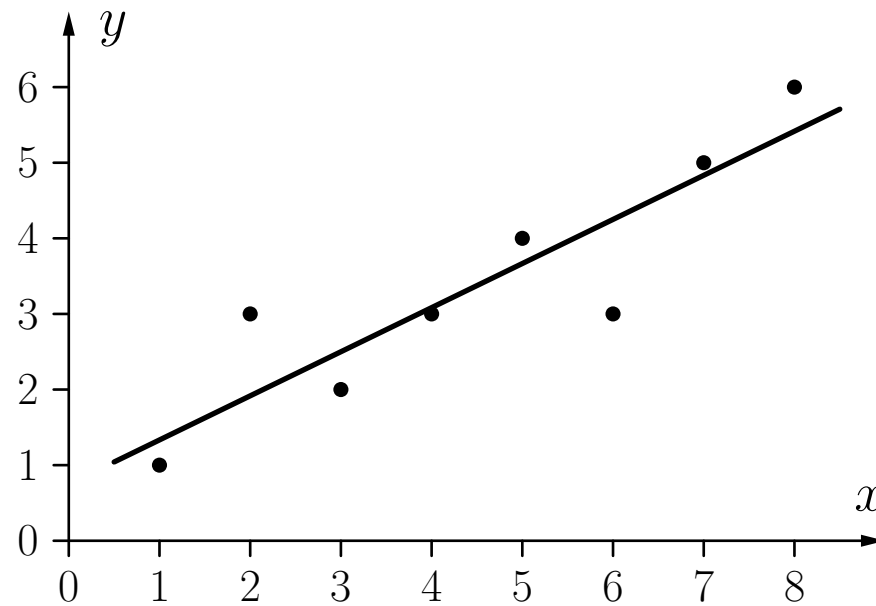
$$\left( \sum_{i=1}^n x_i \right) a + \left( \sum_{i=1}^n x_i^2 \right) b = \sum_{i=1}^n x_i y_i.$$

- Zwei lineare Gleichungen für zwei Unbekannte  $a$  und  $b$ .
- System kann mit Standardmethoden der linearen Algebra gelöst werden.
- Die Lösung ist eindeutig, falls nicht alle  $x$ -Werte identisch sind.
- Die errechnete Gerade nennt man **Regressionsgerade**.

# Lineare Regression: Beispiel

$x$	1	2	3	4	5	6	7	8
$y$	1	3	2	3	4	3	5	6

$$y = \frac{3}{4} + \frac{7}{12}x.$$





## Generalisierung auf Polynome

$$y = p(x) = a_0 + a_1x + \dots + a_mx^m$$

Idee: Minimiere die Summe der quadrierten Fehler, d.h.

$$F(a_0, a_1, \dots, a_m) = \sum_{i=1}^n (p(x_i) - y_i)^2 = \sum_{i=1}^n (a_0 + a_1x_i + \dots + a_mx_i^m - y_i)^2$$

Notwendige Bedingungen für ein Minimum: Alle partiellen Ableitungen verschwinden, d.h.

$$\frac{\partial F}{\partial a_0} = 0, \quad \frac{\partial F}{\partial a_1} = 0, \quad \dots, \quad \frac{\partial F}{\partial a_m} = 0.$$

## System von Normalgleichungen für Polynome

$$\begin{aligned} na_0 + \left(\sum_{i=1}^n x_i\right) a_1 + \dots + \left(\sum_{i=1}^n x_i^m\right) a_m &= \sum_{i=1}^n y_i \\ \left(\sum_{i=1}^n x_i\right) a_0 + \left(\sum_{i=1}^n x_i^2\right) a_1 + \dots + \left(\sum_{i=1}^n x_i^{m+1}\right) a_m &= \sum_{i=1}^n x_i y_i \\ \vdots & \\ \left(\sum_{i=1}^n x_i^m\right) a_0 + \left(\sum_{i=1}^n x_i^{m+1}\right) a_1 + \dots + \left(\sum_{i=1}^n x_i^{2m}\right) a_m &= \sum_{i=1}^n x_i^m y_i, \end{aligned}$$

- $m + 1$  lineare Gleichungen für  $m + 1$  Unbekannte  $a_0, \dots, a_m$ .
- System kann mit Standardmethoden der linearen Algebra gelöst werden.
- Die Lösung ist eindeutig, falls nicht alle  $x$ -Werte identisch sind.

## Generalisierung auf mehr als ein Argument

$$z = f(x, y) = a + bx + cy$$

Idee: Minimiere die Summe der quadrierten Fehler, d.h.

$$F(a, b, c) = \sum_{i=1}^n (f(x_i, y_i) - z_i)^2 = \sum_{i=1}^n (a + bx_i + cy_i - z_i)^2$$

Notwendige Bedingungen für ein Minimum: Alle partiellen Ableitungen verschwinden, d.h.

$$\begin{aligned}\frac{\partial F}{\partial a} &= \sum_{i=1}^n 2(a + bx_i + cy_i - z_i) = 0, \\ \frac{\partial F}{\partial b} &= \sum_{i=1}^n 2(a + bx_i + cy_i - z_i)x_i = 0, \\ \frac{\partial F}{\partial c} &= \sum_{i=1}^n 2(a + bx_i + cy_i - z_i)y_i = 0.\end{aligned}$$

## System von Normalgleichungen für mehrere Argumente

$$na + \left( \sum_{i=1}^n x_i \right) b + \left( \sum_{i=1}^n y_i \right) c = \sum_{i=1}^n z_i$$

$$\left( \sum_{i=1}^n x_i \right) a + \left( \sum_{i=1}^n x_i^2 \right) b + \left( \sum_{i=1}^n x_i y_i \right) c = \sum_{i=1}^n z_i x_i$$

$$\left( \sum_{i=1}^n y_i \right) a + \left( \sum_{i=1}^n x_i y_i \right) b + \left( \sum_{i=1}^n y_i^2 \right) c = \sum_{i=1}^n z_i y_i$$

- 3 lineare Gleichungen für 3 Unbekannte  $a$ ,  $b$  und  $c$ .
- System kann mit Standardmethoden der linearen Algebra gelöst werden.
- Die Lösung ist eindeutig, falls nicht alle  $x$ -Werte oder alle  $y$ -Werte identisch sind.

# Multilineare Regression

## Allgemeiner multilinearer Fall:

$$y = f(x_1, \dots, x_m) = a_0 + \sum_{k=1}^m a_k x_k$$

Idee: Minimiere die Summe der quadrierten Fehler, d.h.

$$F(\mathbf{a}) = (\mathbf{X}\mathbf{a} - \mathbf{y})^\top (\mathbf{X}\mathbf{a} - \mathbf{y}),$$

wobei

$$\mathbf{X} = \begin{pmatrix} 1 & x_{11} & \dots & x_{m1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{1n} & \dots & x_{mn} \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}, \quad \text{und} \quad \mathbf{a} = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{pmatrix}$$

Notwendige Bedingungen für ein Minimum:

$$\nabla_{\mathbf{a}} F(\mathbf{a}) = \nabla_{\mathbf{a}} (\mathbf{X}\mathbf{a} - \mathbf{y})^\top (\mathbf{X}\mathbf{a} - \mathbf{y}) = \mathbf{0}$$

# Multilineare Regression

- $\nabla_{\mathbf{a}} F(\mathbf{a})$  kann einfach berechnet werden mit der Überlegung, dass der Nabla-Operator

$$\nabla_{\mathbf{a}} = \left( \frac{\partial}{\partial a_0}, \dots, \frac{\partial}{\partial a_m} \right)$$

sich formell wie ein Vektor verhält, der mit der Summe der quadrierten Fehler “multipliziert” wird.

- Alternativ kann man die Differentiation komponentenweise beschreiben.

Mit der vorherigen Methode bekommen wir für die Ableitung:

$$\begin{aligned} & \nabla_{\mathbf{a}} (\mathbf{X}\mathbf{a} - \mathbf{y})^\top (\mathbf{X}\mathbf{a} - \mathbf{y}) \\ &= (\nabla_{\mathbf{a}} (\mathbf{X}\mathbf{a} - \mathbf{y}))^\top (\mathbf{X}\mathbf{a} - \mathbf{y}) + ((\mathbf{X}\mathbf{a} - \mathbf{y})^\top (\nabla_{\mathbf{a}} (\mathbf{X}\mathbf{a} - \mathbf{y})))^\top \\ &= (\nabla_{\mathbf{a}} (\mathbf{X}\mathbf{a} - \mathbf{y}))^\top (\mathbf{X}\mathbf{a} - \mathbf{y}) + (\nabla_{\mathbf{a}} (\mathbf{X}\mathbf{a} - \mathbf{y}))^\top (\mathbf{X}\mathbf{a} - \mathbf{y}) \\ &= 2\mathbf{X}^\top (\mathbf{X}\mathbf{a} - \mathbf{y}) \\ &= 2\mathbf{X}^\top \mathbf{X}\mathbf{a} - 2\mathbf{X}^\top \mathbf{y} = \mathbf{0} \end{aligned}$$

# Multilineare Regression

Einige Regeln für Vektor-/Matrixberechnung und Ableitungen:

$$\begin{aligned}(\mathbf{A} + \mathbf{B})^\top &= \mathbf{A}^\top + \mathbf{B}^\top & \nabla_{\mathbf{z}} f(\mathbf{z})\mathbf{A} &= (\nabla_{\mathbf{z}} f(\mathbf{z}))\mathbf{A} \\ (\mathbf{AB})^\top &= \mathbf{B}^\top \mathbf{A}^\top & \nabla_{\mathbf{z}} (f(\mathbf{z}))^\top &= (\nabla_{\mathbf{z}} f(\mathbf{z}))^\top \\ \nabla_{\mathbf{z}} \mathbf{A}\mathbf{z} &= \mathbf{A} & \nabla_{\mathbf{z}} f(\mathbf{z})g(\mathbf{z}) &= (\nabla_{\mathbf{z}} f(\mathbf{z}))g(\mathbf{z}) + f(\mathbf{z})(\nabla_{\mathbf{z}} g(\mathbf{z}))^\top\end{aligned}$$

Ableitung der zu minimierenden Funktion:

$$\begin{aligned}\nabla_{\mathbf{a}} F(\mathbf{a}) &= \nabla_{\mathbf{a}} (\mathbf{X}\mathbf{a} - \mathbf{y})^\top (\mathbf{X}\mathbf{a} - \mathbf{y}) \\ &= \nabla_{\mathbf{a}} ((\mathbf{X}\mathbf{a})^\top - \mathbf{y}^\top) (\mathbf{X}\mathbf{a} - \mathbf{y}) \\ &= \nabla_{\mathbf{a}} ((\mathbf{X}\mathbf{a})^\top \mathbf{X}\mathbf{a} - (\mathbf{X}\mathbf{a})^\top \mathbf{y} - \mathbf{y}^\top \mathbf{X}\mathbf{a} + \mathbf{y}^\top \mathbf{y}) \\ &= \nabla_{\mathbf{a}} (\mathbf{X}\mathbf{a})^\top \mathbf{X}\mathbf{a} - \nabla_{\mathbf{a}} (\mathbf{X}\mathbf{a})^\top \mathbf{y} - \nabla_{\mathbf{a}} \mathbf{y}^\top \mathbf{X}\mathbf{a} + \nabla_{\mathbf{a}} \mathbf{y}^\top \mathbf{y} \\ &= (\nabla_{\mathbf{a}} (\mathbf{X}\mathbf{a})^\top) \mathbf{X}\mathbf{a} + ((\mathbf{X}\mathbf{a})^\top (\nabla_{\mathbf{a}} \mathbf{X}\mathbf{a}))^\top - 2\nabla_{\mathbf{a}} (\mathbf{X}\mathbf{a})^\top \mathbf{y} \\ &= ((\nabla_{\mathbf{a}} \mathbf{X}\mathbf{a})^\top) \mathbf{X}\mathbf{a} + (\nabla_{\mathbf{a}} \mathbf{X}\mathbf{a})^\top \mathbf{X}\mathbf{a} - 2(\nabla_{\mathbf{a}} (\mathbf{X}\mathbf{a})^\top) \mathbf{y} \\ &= 2(\nabla_{\mathbf{a}} \mathbf{X}\mathbf{a})^\top \mathbf{X}\mathbf{a} - 2(\nabla_{\mathbf{a}} \mathbf{X}\mathbf{a})^\top \mathbf{y} \\ &= 2\mathbf{X}^\top \mathbf{X}\mathbf{a} - 2\mathbf{X}^\top \mathbf{y}\end{aligned}$$

# Multilineare Regression

Notwendige Bedingungen für ein Minimum also:

$$\begin{aligned}\nabla_{\mathbf{a}}F(\mathbf{a}) &= \nabla_{\mathbf{a}}(\mathbf{X}\mathbf{a} - \mathbf{y})^{\top}(\mathbf{X}\mathbf{a} - \mathbf{y}) \\ &= 2\mathbf{X}^{\top}\mathbf{X}\mathbf{a} - 2\mathbf{X}^{\top}\mathbf{y} \stackrel{!}{=} \mathbf{0}\end{aligned}$$

Als Ergebnis bekommen wir das System von **Normalgleichungen**:

$$\mathbf{X}^{\top}\mathbf{X}\mathbf{a} = \mathbf{X}^{\top}\mathbf{y}$$

Dieses System hat eine Lösung, falls  $\mathbf{X}^{\top}\mathbf{X}$  nicht singulär ist. Dann ergibt sich

$$\mathbf{a} = (\mathbf{X}^{\top}\mathbf{X})^{-1}\mathbf{X}^{\top}\mathbf{y}.$$

$(\mathbf{X}^{\top}\mathbf{X})^{-1}\mathbf{X}^{\top}$  heißt die (Moore-Penrose-) **Pseudoinverse** der Matrix  $\mathbf{X}$ .

Mit der Matrix-Vektor-Repräsentation des Regressionsproblems ist die Erweiterung auf **Multipolynomiale Regression** naheliegend:

Addiere die gewünschten Produkte zur Matrix  $\mathbf{X}$ .



## Generalisierung auf nicht-polynomiale Funktionen

Einfaches Beispiel:  $y = ax^b$

Idee: Finde Transformation zum linearen/polynomiellen Fall.

Transformation z.B.:  $\ln y = \ln a + b \cdot \ln x$ .

## Spezialfall: **logistische Funktion**

$$y = \frac{Y}{1 + e^{a+bx}} \quad \Leftrightarrow \quad \frac{1}{y} = \frac{1 + e^{a+bx}}{Y} \quad \Leftrightarrow \quad \frac{Y - y}{y} = e^{a+bx}.$$

Ergebnis: Wende sogenannte **Logit-Transformation** an:

$$\ln \left( \frac{Y - y}{y} \right) = a + bx.$$

# Logistische Regression: Beispiel

$x$	1	2	3	4	5
$y$	0.4	1.0	3.0	5.0	5.6

Transformiere die Daten mit

$$z = \ln\left(\frac{Y - y}{y}\right), \quad Y = 6.$$

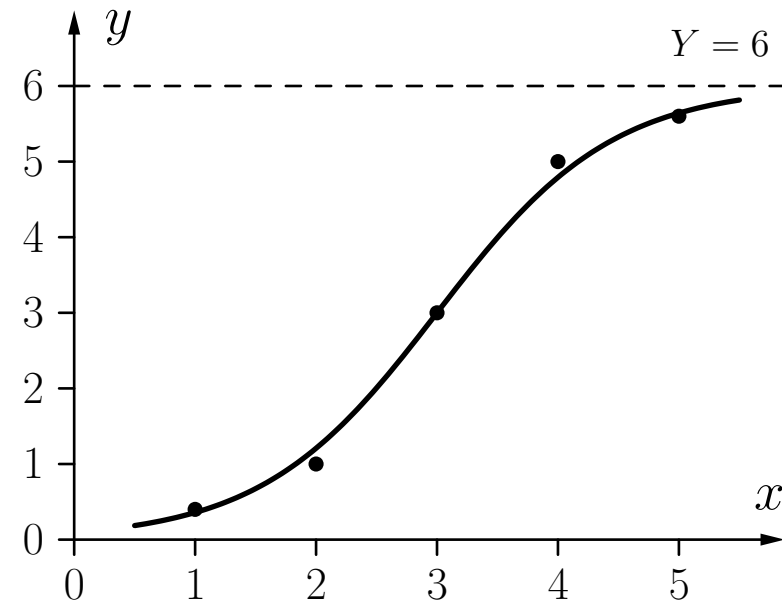
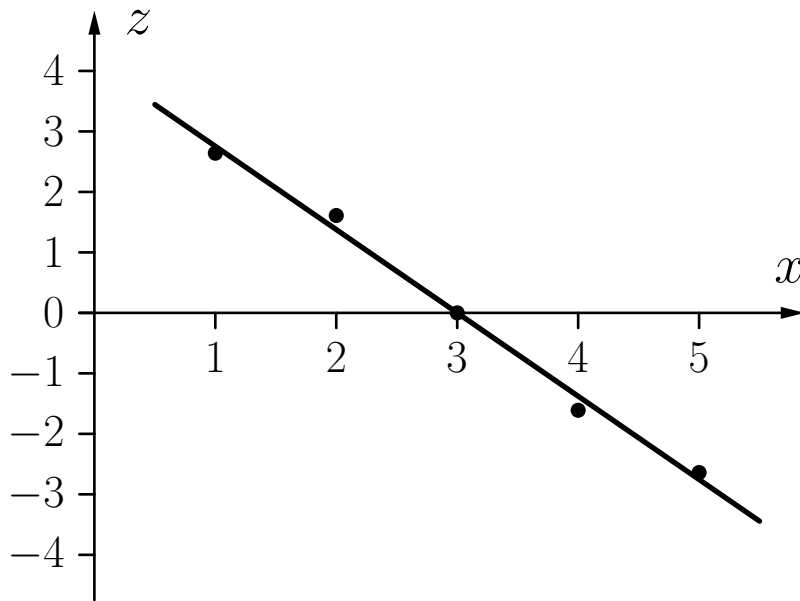
Die transformierten Datenpunkte sind

$x$	1	2	3	4	5
$z$	2.64	1.61	0.00	-1.61	-2.64

Die sich ergebende Regressionsgerade ist

$$z \approx -1.3775x + 4.133.$$

# Logistische Regression: Beispiel



Die logistische Regressionsfunktion kann von einem einzelnen Neuron mit

- Netzeingabefunktion  $f_{\text{net}}(x) \equiv wx$  mit  $w \approx -1.3775$ ,
  - Aktivierungsfunktion  $f_{\text{act}}(\text{net}, \theta) \equiv (1 + e^{-(\text{net} - \theta)})^{-1}$  mit  $\theta \approx 4.133$  und
  - Ausgabefunktion  $f_{\text{out}}(\text{act}) \equiv 6 \text{ act}$
- berechnet werden.