Neuronale Netze

Prof. Dr. Rudolf Kruse

Computational Intelligence
Institut für Wissens- und Sprachverarbeitung
Fakultät für Informatik
kruse@iws.cs.uni-magdeburg.de

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:

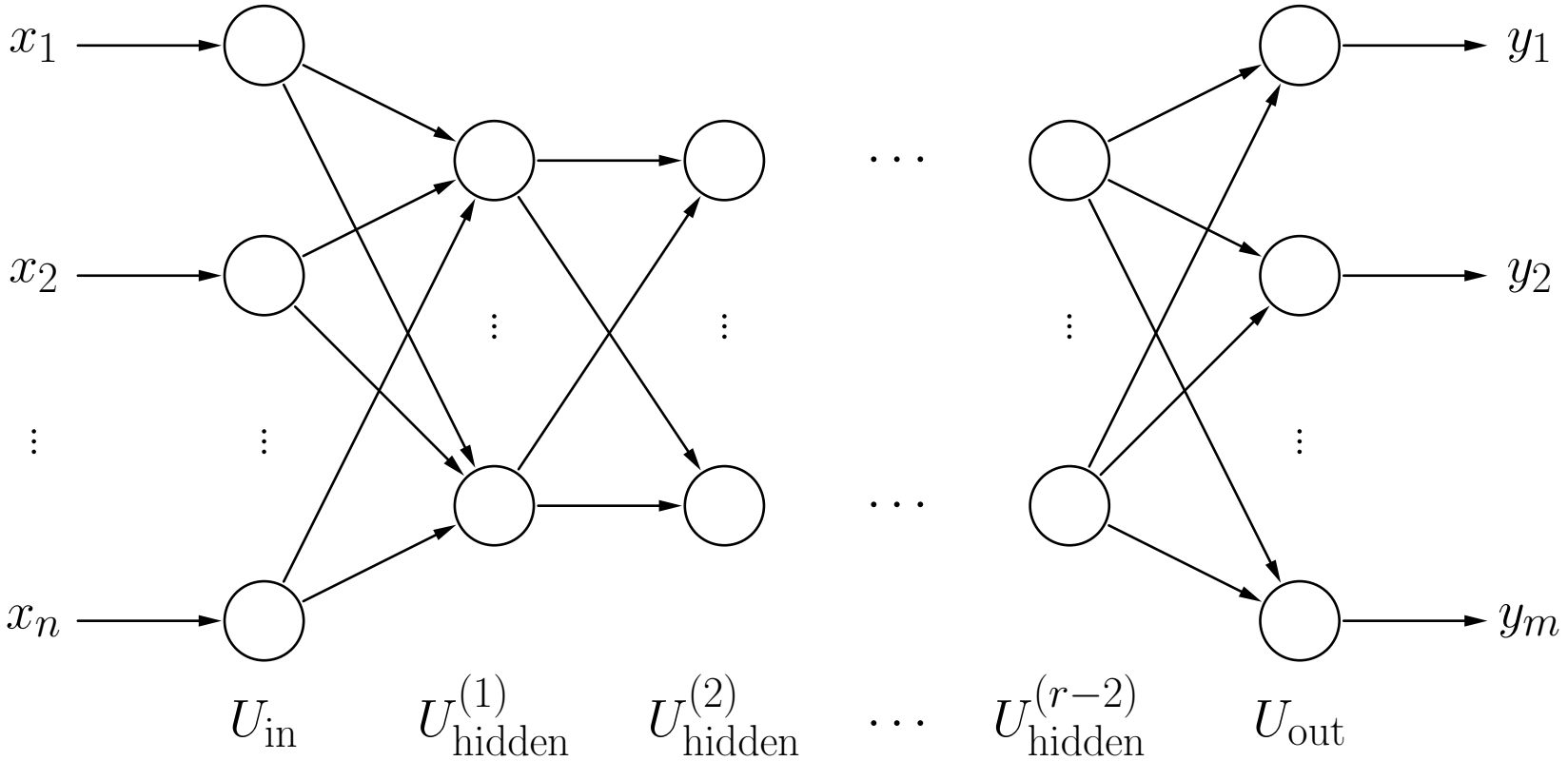
(i) $U_{\text{in}} \cap U_{\text{out}} = \emptyset$,

(ii) $U_{\text{hidden}} = U_{\text{hidden}}^{(1)} \cup \cdots \cup U_{\text{hidden}}^{(r-2)}$,
$\forall 1 \leq i < j \leq r-2: \quad U_{\text{hidden}}^{(i)} \cap U_{\text{hidden}}^{(j)} = \emptyset$,

(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.

Allgemeine Struktur eines mehrschichtigen Perzeptrons



Mehrschichtige Perzeptren

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

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

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

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

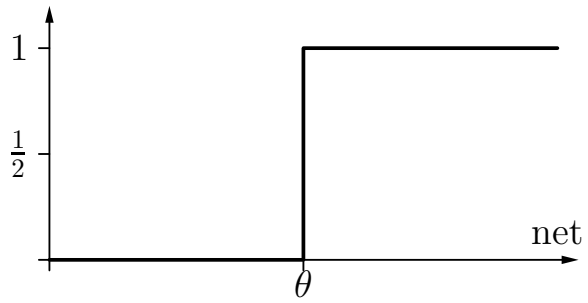
- Die Aktivierungsfunktion jedes Ausgabeneurons ist ebenfalls entweder eine Sigmoide oder eine **lineare Funktion**, d.h.

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

Sigmoide als Aktivierungsfunktionen

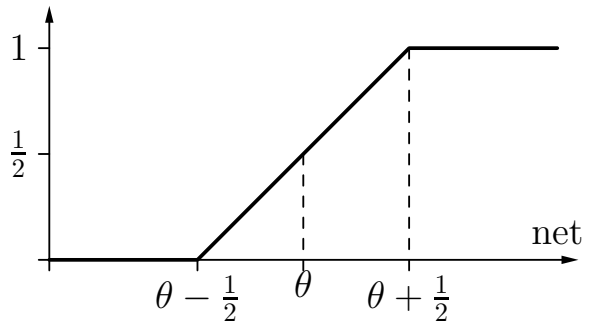
Stufenfunktion:

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



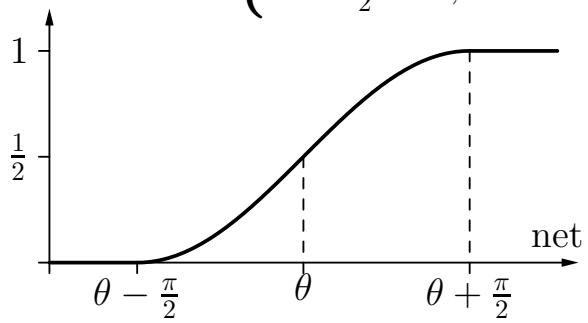
Semilineare Funktion:

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



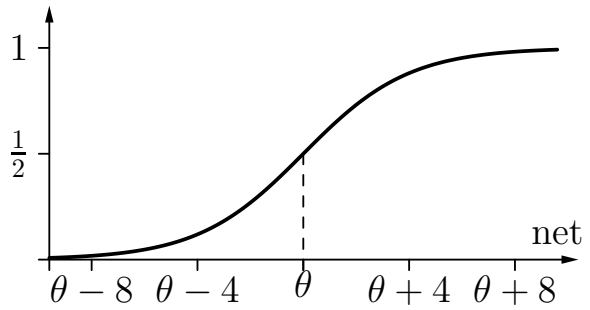
Sinus bis Sättigung:

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



Logistische Funktion:

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

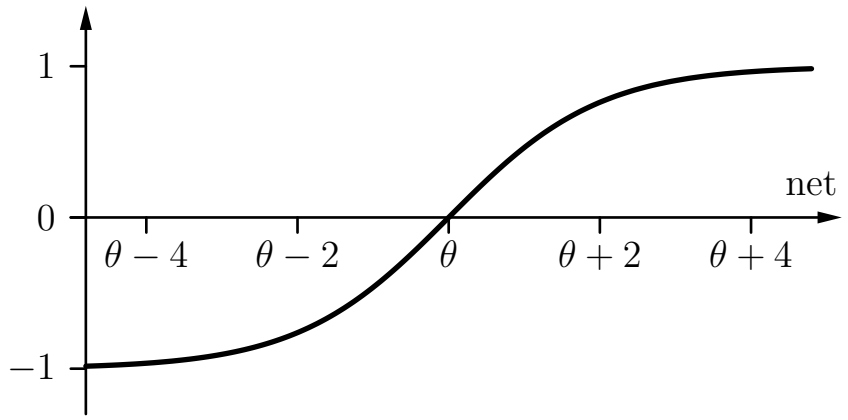


Sigmoide als Aktivierungsfunktionen

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

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

$$\mathbf{W} = \begin{pmatrix} w_{u_1 v_1} & w_{u_1 v_2} & \ldots & w_{u_1 v_m} \\ w_{u_2 v_1} & w_{u_2 v_2} & \ldots & w_{u_2 v_m} \\ \vdots & \vdots & & \vdots \\ w_{u_n v_1} & w_{u_n v_2} & \ldots & w_{u_n v_m} \end{pmatrix},$$

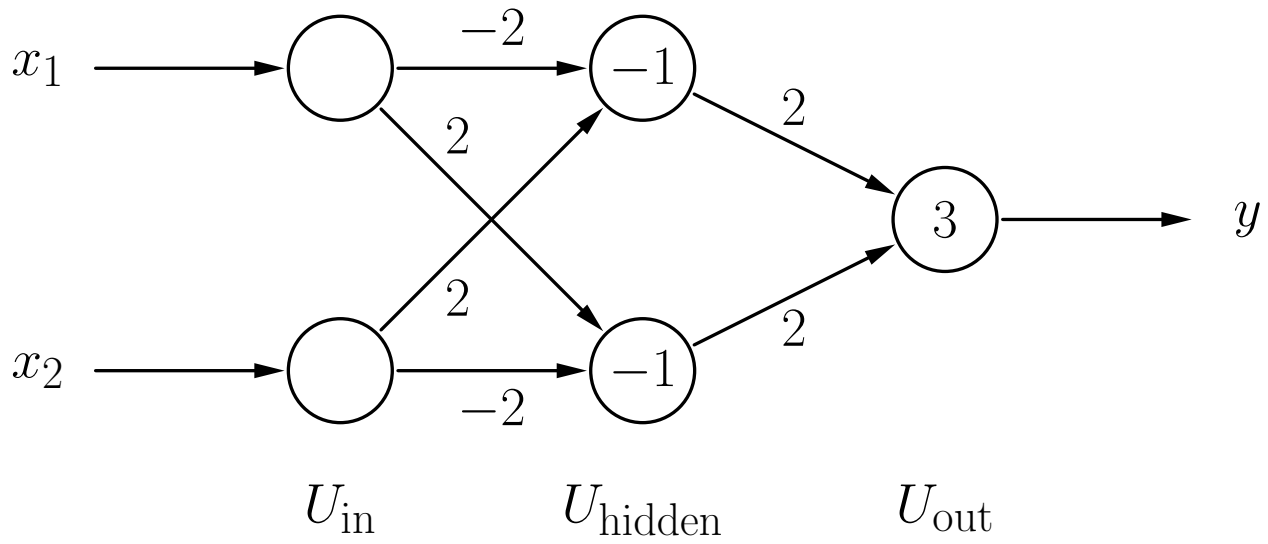
wobei $w_{u_i v_j} = 0$, falls es keine Verbindung von Neuron v_j zu Neuron u_i gibt.

Vorteil: Die Berechnung der Netzwerkeingabe kann geschrieben werden als

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

wobei $\vec{\mathrm{net}}_{U_2} = (\mathrm{net}_{u_1}, \ldots, \mathrm{net}_{u_n})^\top$ und $\vec{\mathrm{in}}_{U_2} = \vec{\mathrm{out}}_{U_1} = (\mathrm{out}_{v_1}, \ldots, \mathrm{out}_{v_m})^\top$.

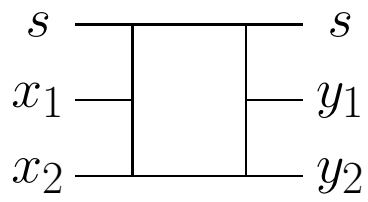
Lösen des Biimplikationsproblems mit einem MLP.

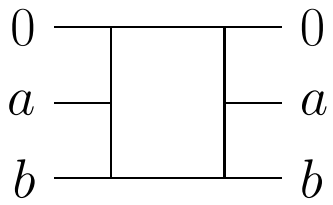


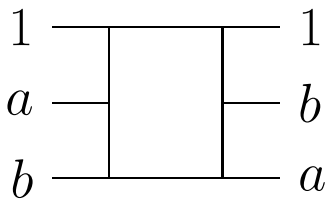
Man beachte die zusätzlichen Eingabeneuronen im Vergleich zur Lösung mit Schwellenwertelementen.

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

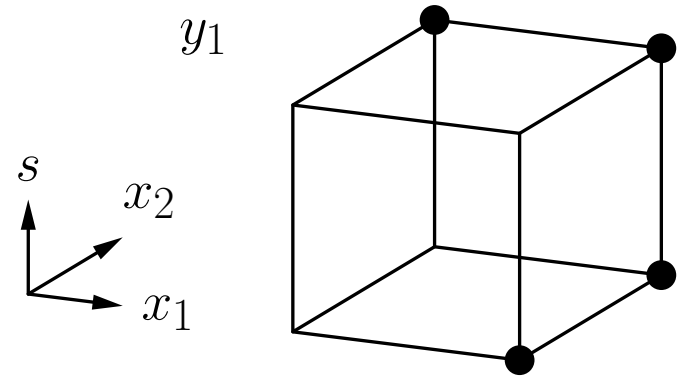
Mehrschichtige Perzeptren: Fredkin-Gatter

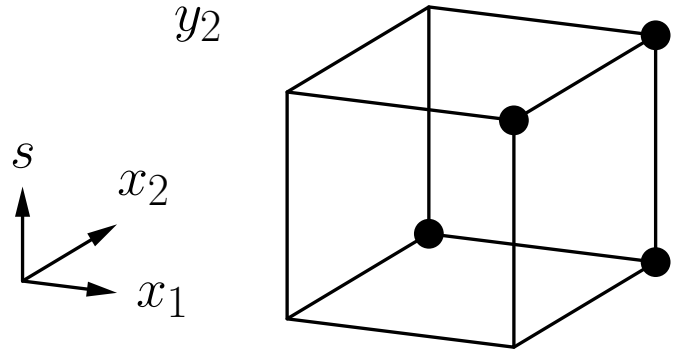




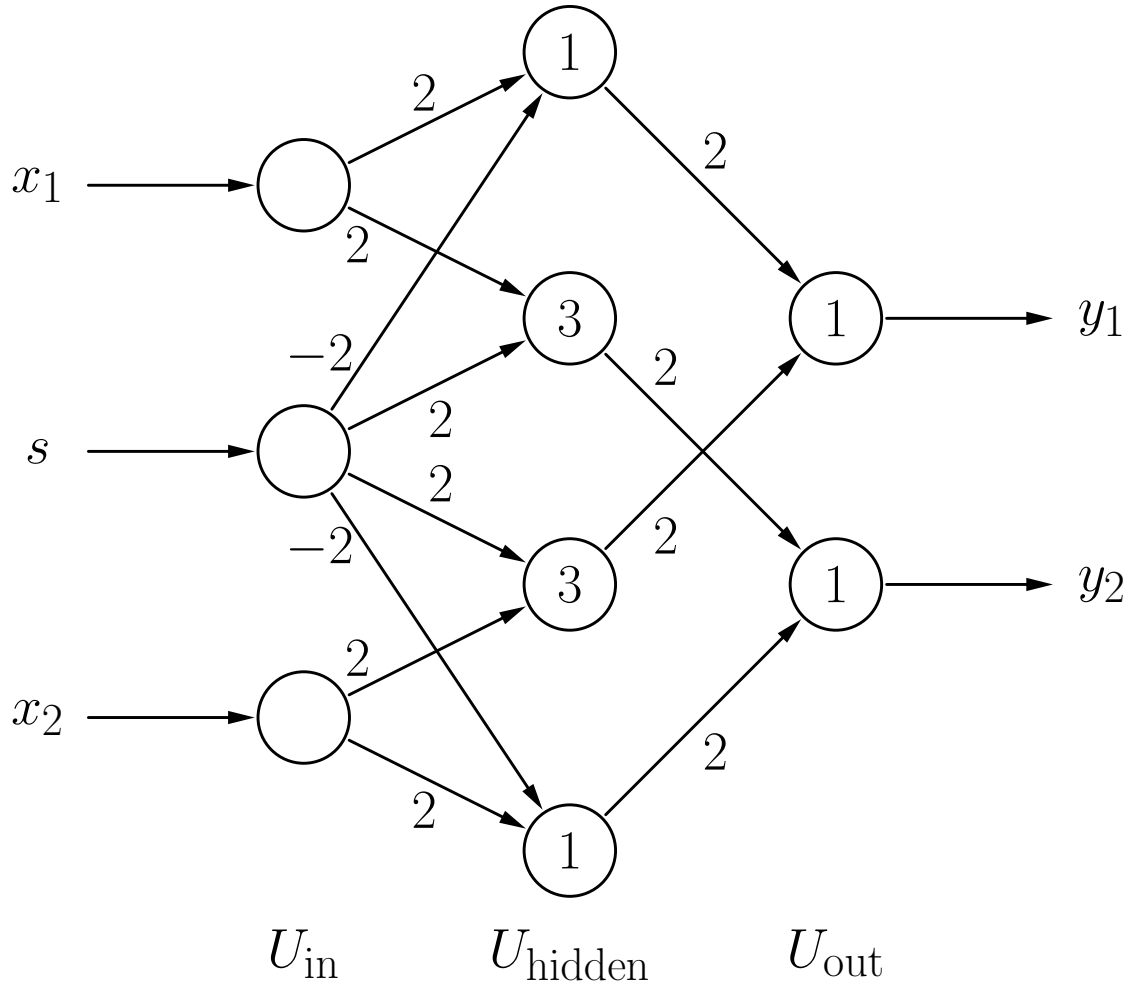


s	0	0	0	0	1	1	1	1
x_1	0	0	1	1	0	0	1	1
x_2	0	1	0	1	0	1	0	1
y_1	0	0	1	1	0	1	0	1
y_2	0	1	0	1	0	0	1	1





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

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

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:

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

wobei

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

Warum nicht-lineare Aktivierungsfunktionen?

Falls die Ausgabefunktion auch linear ist, gilt analog

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

wobei

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

In Kombination ergibt sich

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

und daher

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

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

Warum nicht-lineare Aktivierungsfunktionen?

Daher ergibt sich

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

und

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

für die Berechnungen zwei aufeinanderfolgender Schichten U_2 und U_3.

Diese beiden Berechnungen können kombiniert werden zu

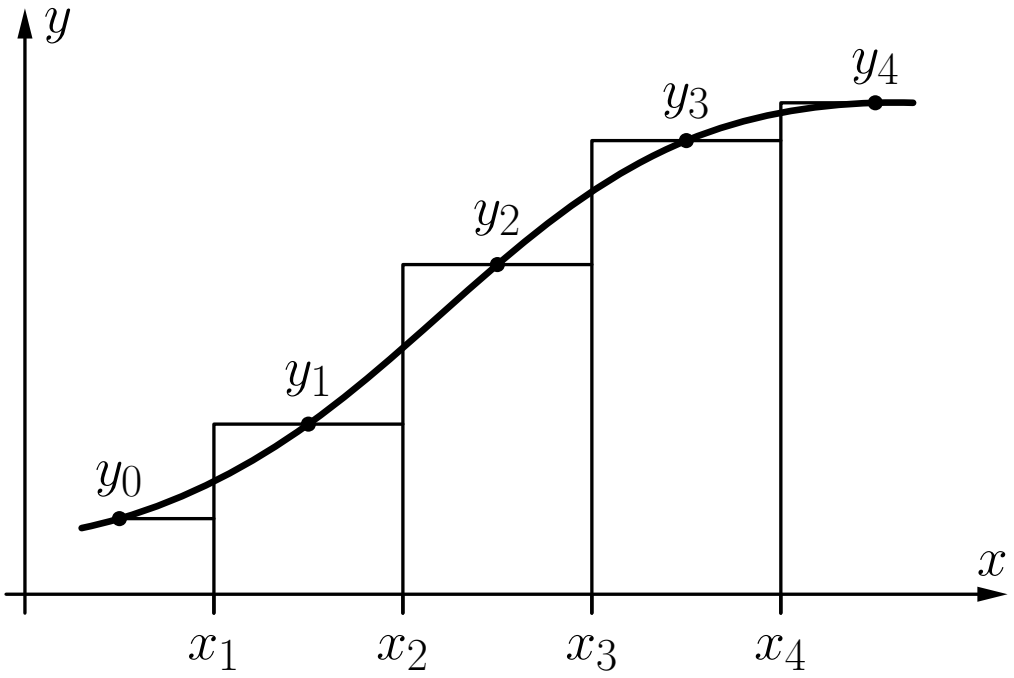
$$\vec{\text{out}}_{U_3} = \mathbf{A}_{13} \cdot \vec{\text{out}}_{U_1} + \vec{b}_{13},$$

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

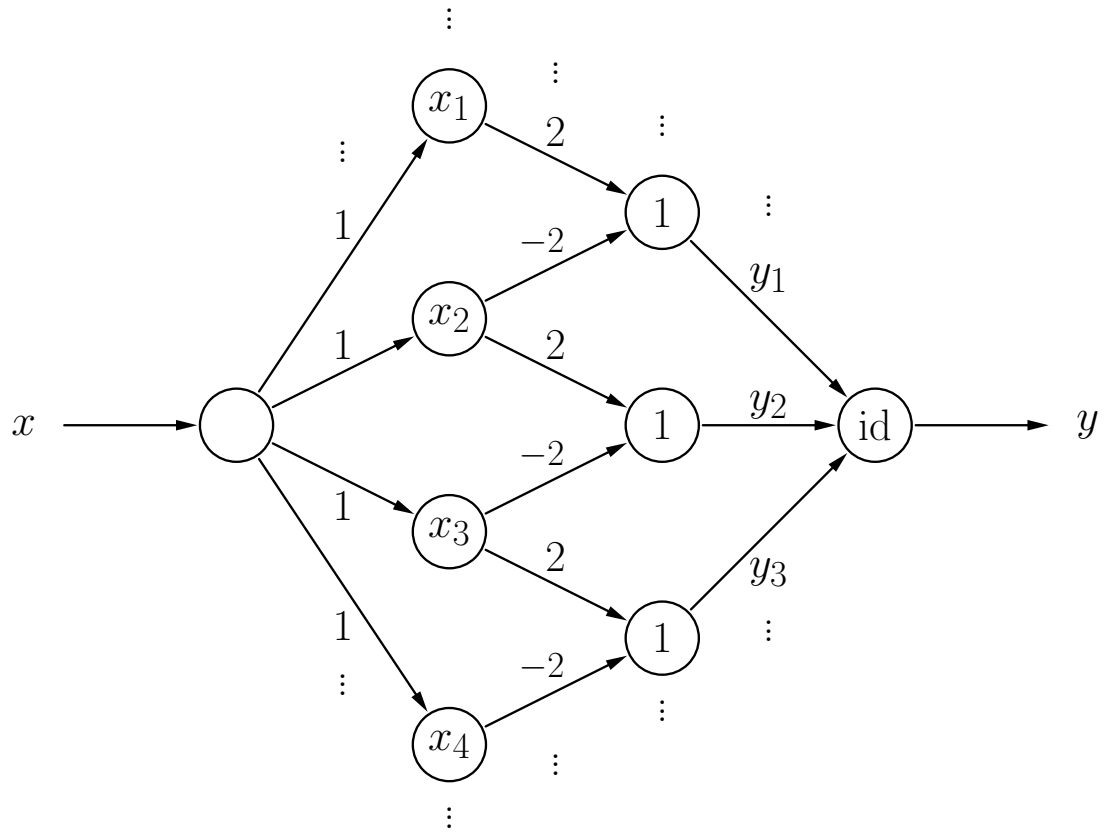
Ergebnis: Mit linearen Aktivierungs- und Ausgabefunktionen können beliebige mehrschichtige Perzeptren auf zweischichtige Perzeptren reduziert werden.

Idee der Funktionsapproximation

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



Mehrschichtige Perzeptren: Funktionsapproximation

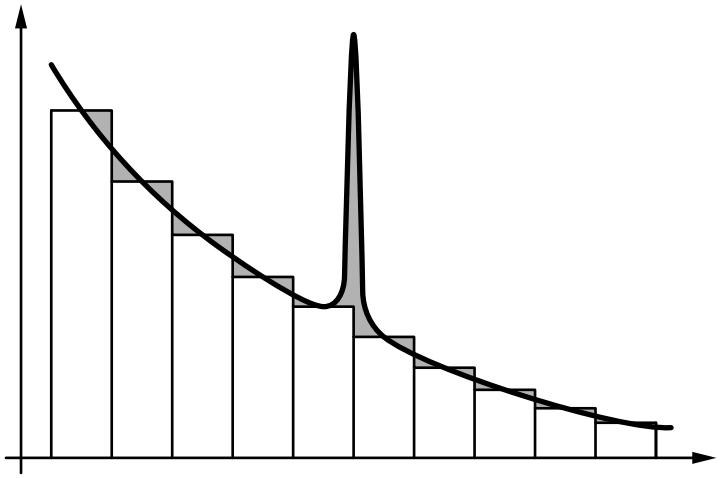


Ein neuronales Netz, das die Treppenfunktion von der vorherigen Folie berechnet. Es ist immer nur eine Stufe passend zum Eingabewert aktiv und die Stufenhöhe wird ausgegeben.

Mehrschichtige Perzeptren: Funktionsapproximation

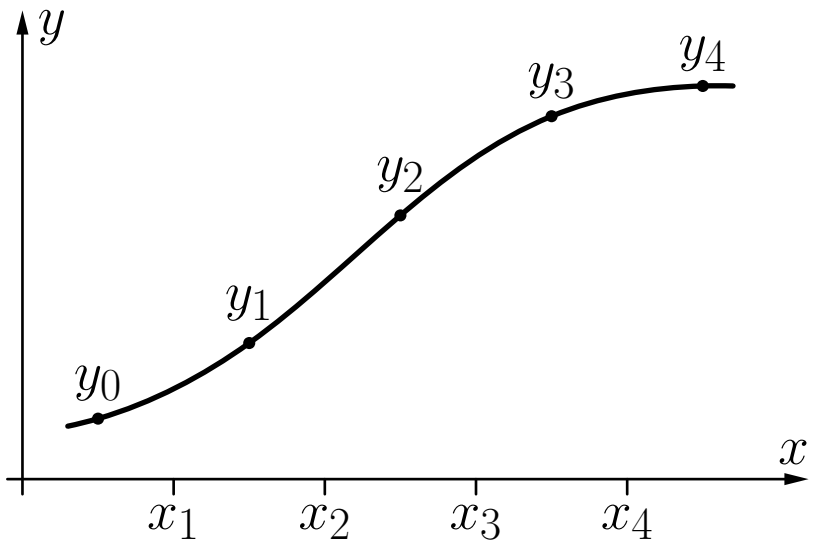
Theorem: Jede Riemann-integrierbare Funktion kann mit beliebiger Genauigkeit durch ein vierschichtiges MLP berechnet werden.

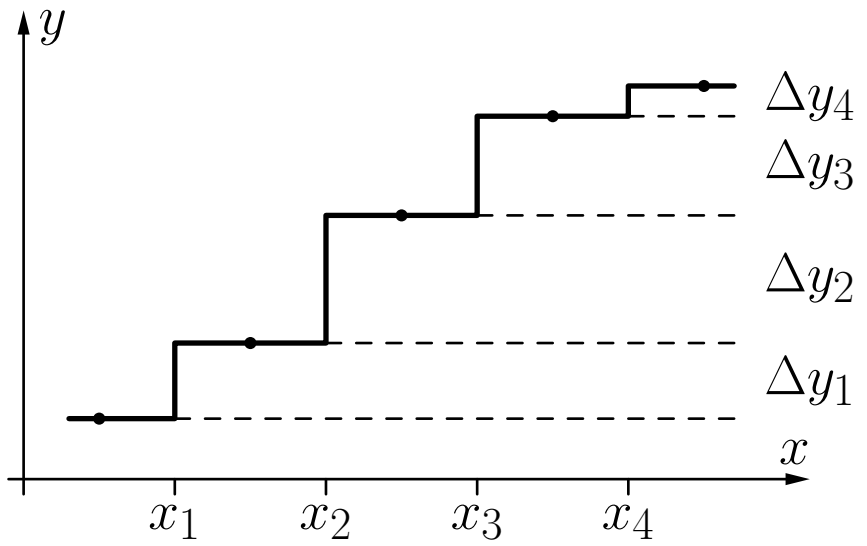
- Aber: Fehler wird bestimmt als die **Fläche** zwischen Funktionen.

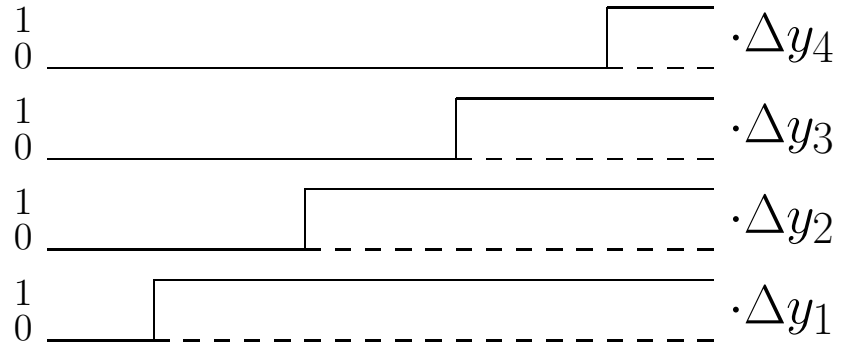


- Weitere mathematische Untersuchungen zeigen, dass sogar gilt: Mit einem dreischichtigen Perzeptron kann jede stetige Funktion mit beliebiger Genauigkeit angenähert werden (Fehlerbestimmung: maximale Differenz der Funktionswerte).

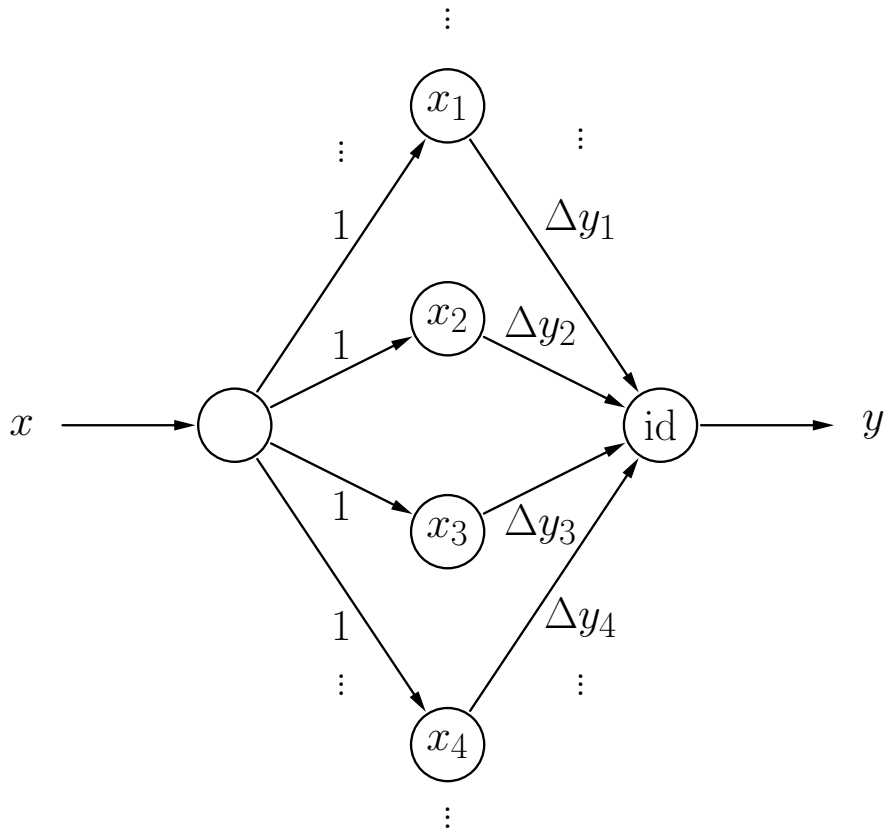
Mehrschichtige Perzeptren: Funktionsapproximation





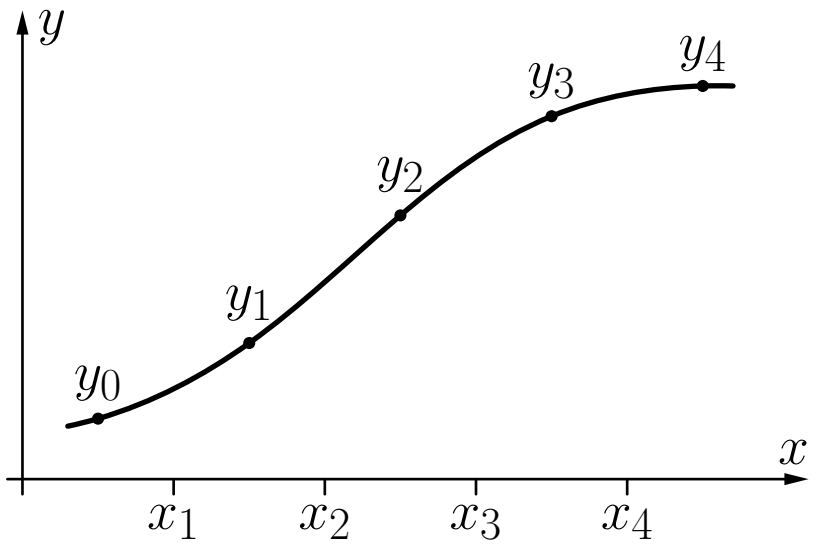


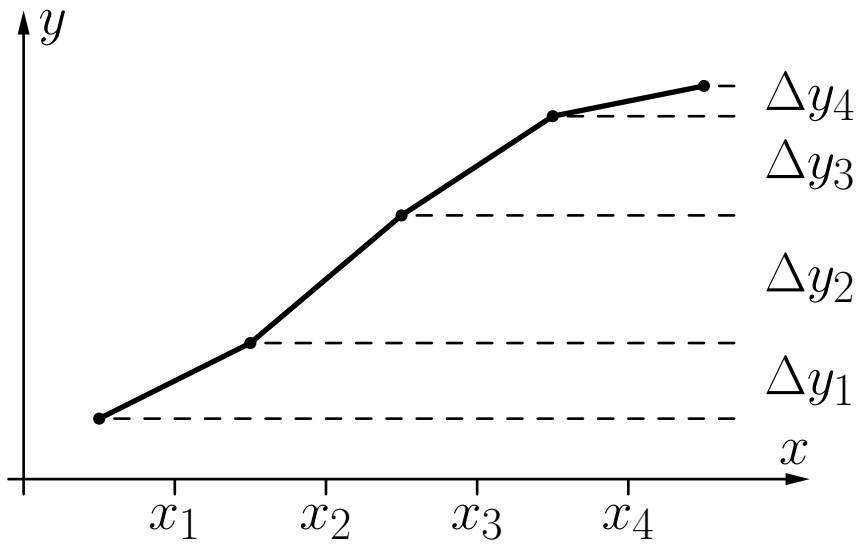
Mehrschichtige Perzeptren: Funktionsapproximation

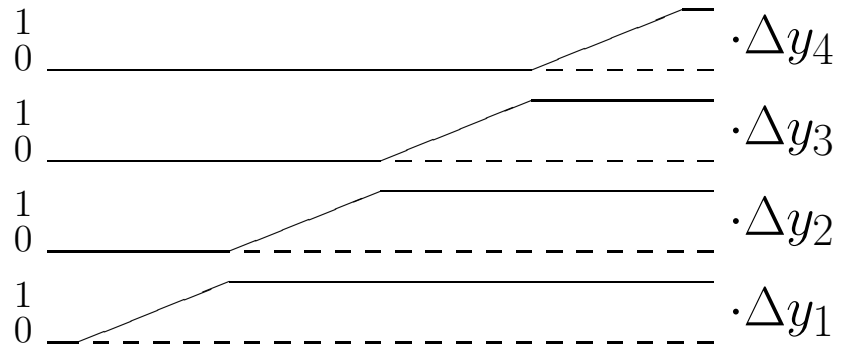


Ein neuronales Netz, das die Treppenfunktion von der vorherigen Folie als gewichtete Summe von Sprungfunktionen berechnet.

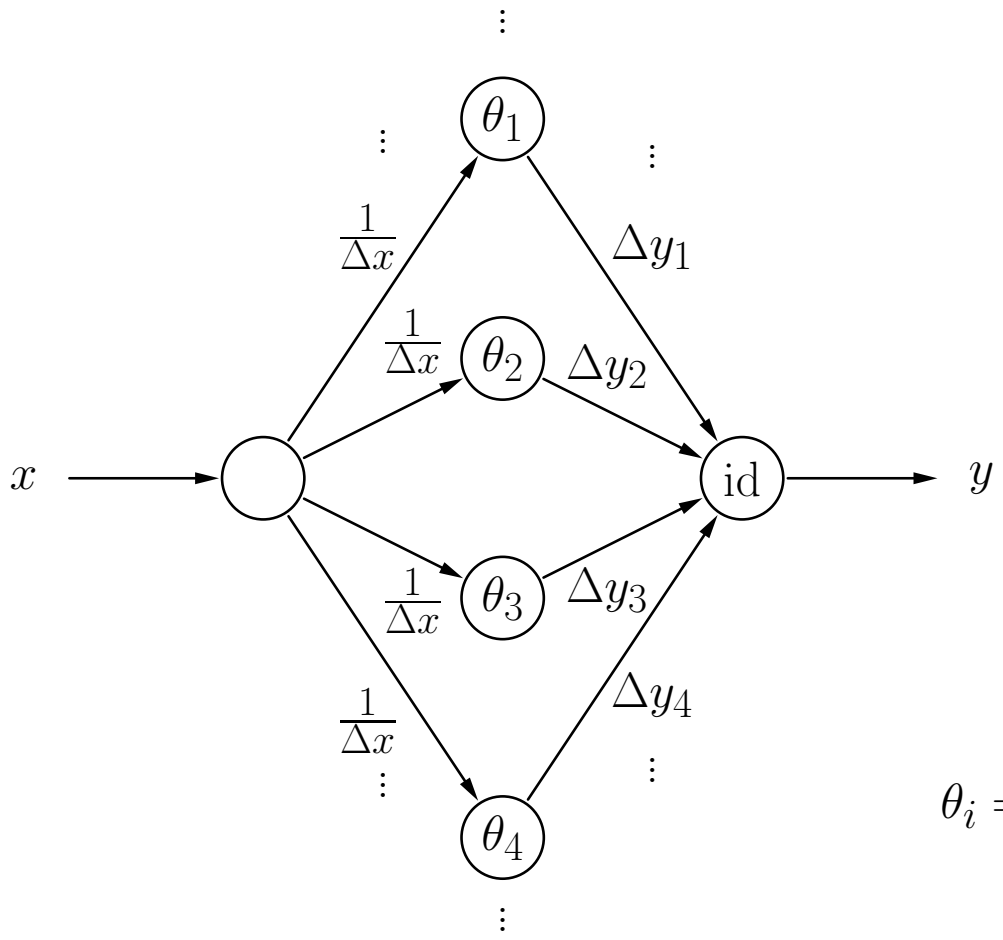
Mehrschichtige Perzeptren: Funktionsapproximation







Mehrschichtige Perzeptren: Funktionsapproximation



$$\theta_i = \frac{x_i}{\Delta x}$$

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

Das Trainieren von NN ist stark verwandt mit Regression

Geg:
- Ein Datensatz $((x_1, y_1), \ldots, (x_n, y_n))$ 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:

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

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

Mathematischer Hintergrund: Lineare Regression

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

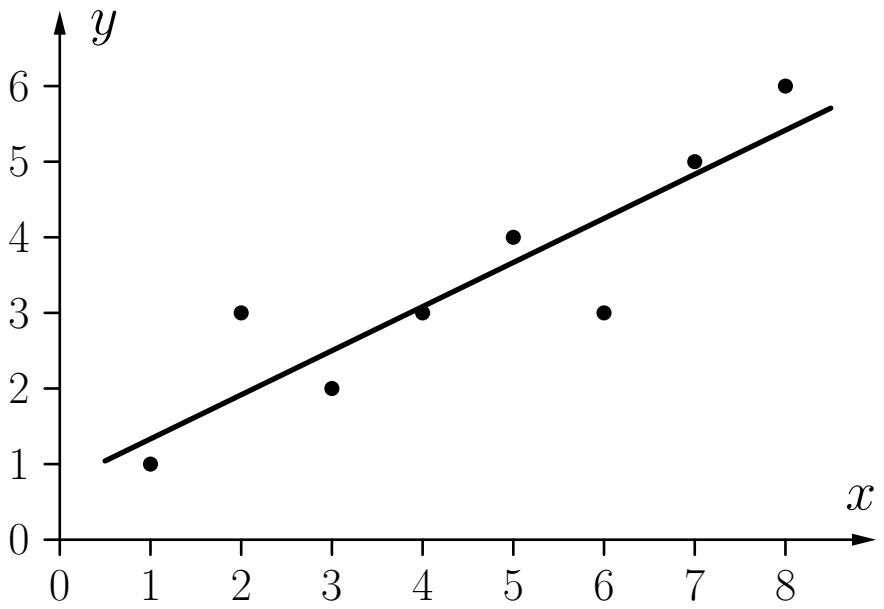
$$\begin{aligned} na + \left(\sum_{i=1}^{n} x_i\right) b &= \sum_{i=1}^{n} y_i, \\ \left(\sum_{i=1}^{n} x_i\right) a + \left(\sum_{i=1}^{n} x_i^2\right) b &= \sum_{i=1}^{n} x_i y_i. \end{aligned}$$

- 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

$$y = p(x) = a_0 + a_1 x + \ldots + a_m x^m$$

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

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

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

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

System von Normalgleichungen für Polynome

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

- $m+1$ lineare Gleichungen für $m+1$ Unbekannte $a_0, \ldots, a_m$.
- 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

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

- 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.

Allgemeiner multilinearer Fall:

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

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

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

wobei

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

Notwendige Bedingungen für ein Minimum:

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

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

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

  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:

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

Multilineare Regression

Einige Regeln für Vektor-/Matrixberechnung und Ableitungen:

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

Ableitung der zu minimierenden Funktion:

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

Multilineare Regression

Notwendige Bedingungen für ein Minimum also:

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

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

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

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

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

$(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top$ 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

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

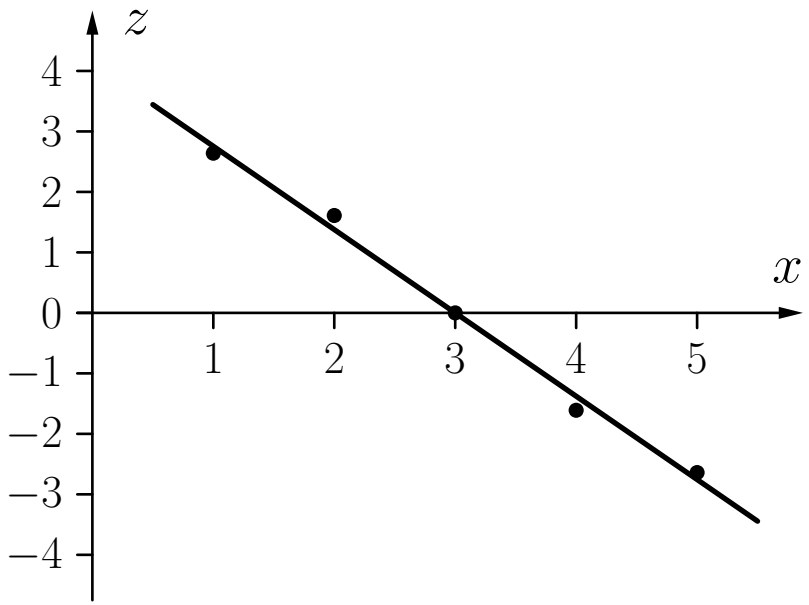
Die transformierten Datenpunkte sind

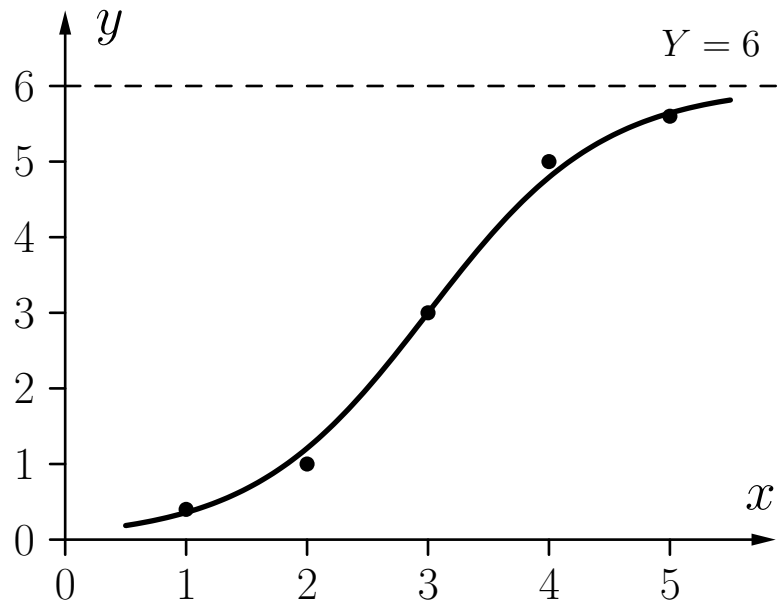
x	1	2	3	4	5
z	2.64	1.61	0.00	-1.61	-2.64

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.