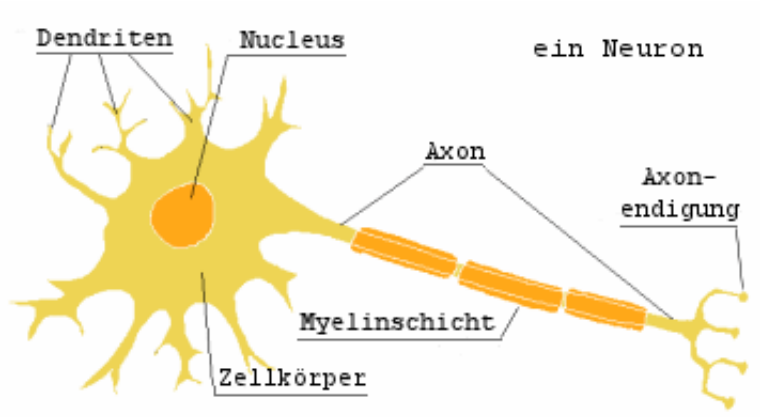
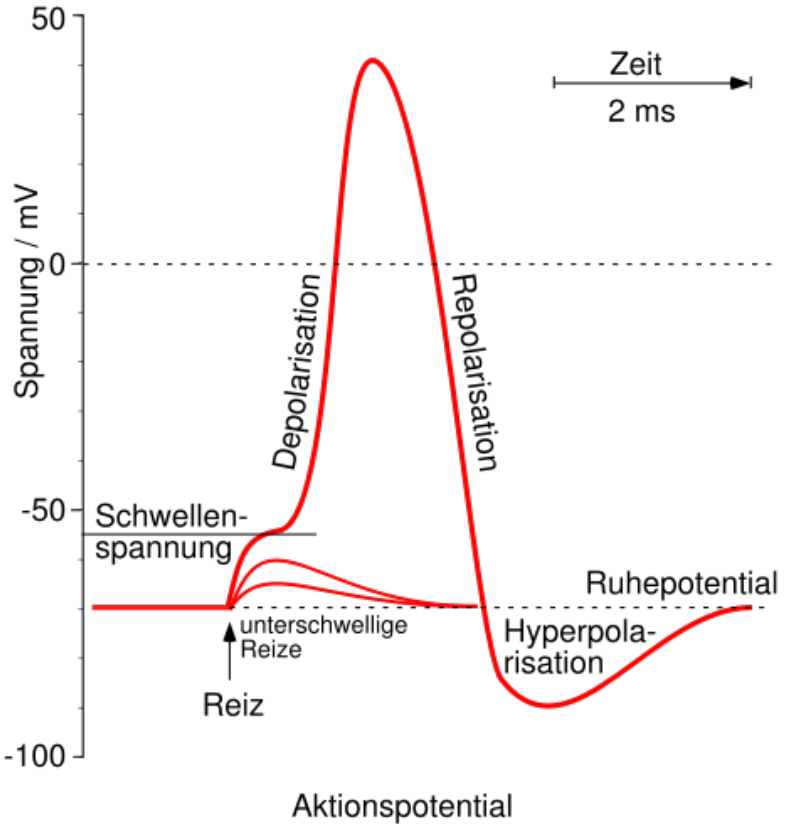
Einführung in künstliche neuronale Netze

Stefan Fürtinger

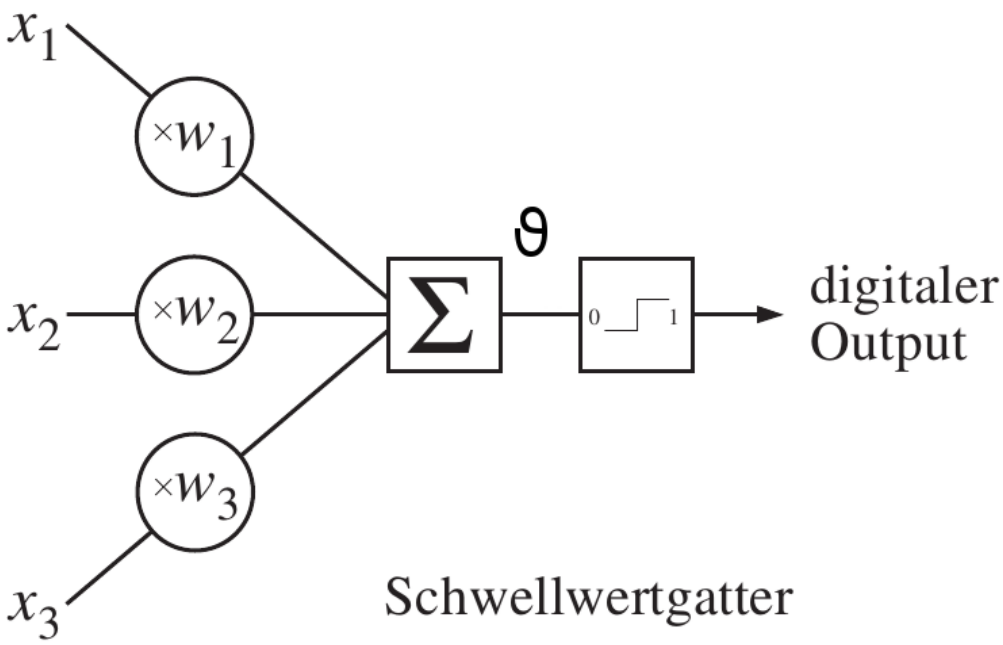
Woche der Modellierung mit Mathematik
5.–11. Februar 2012

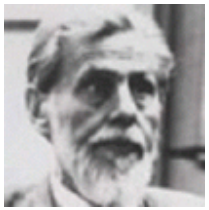
Ein biologisches Neron





Ein künstliches Neuron

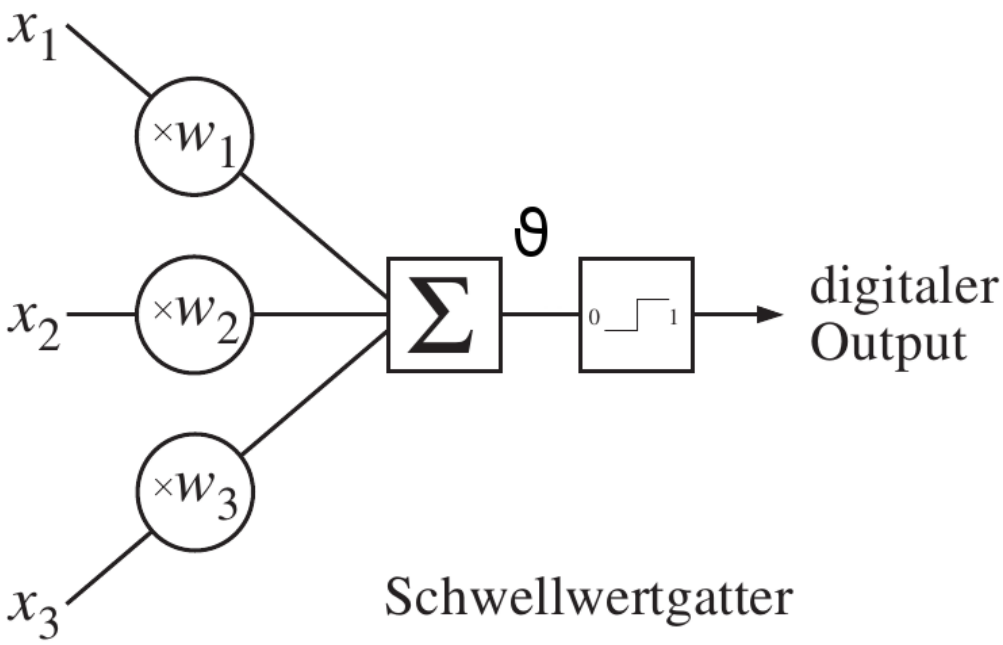


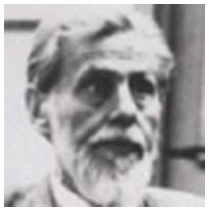


Waren McCulloch

Walter Pitts

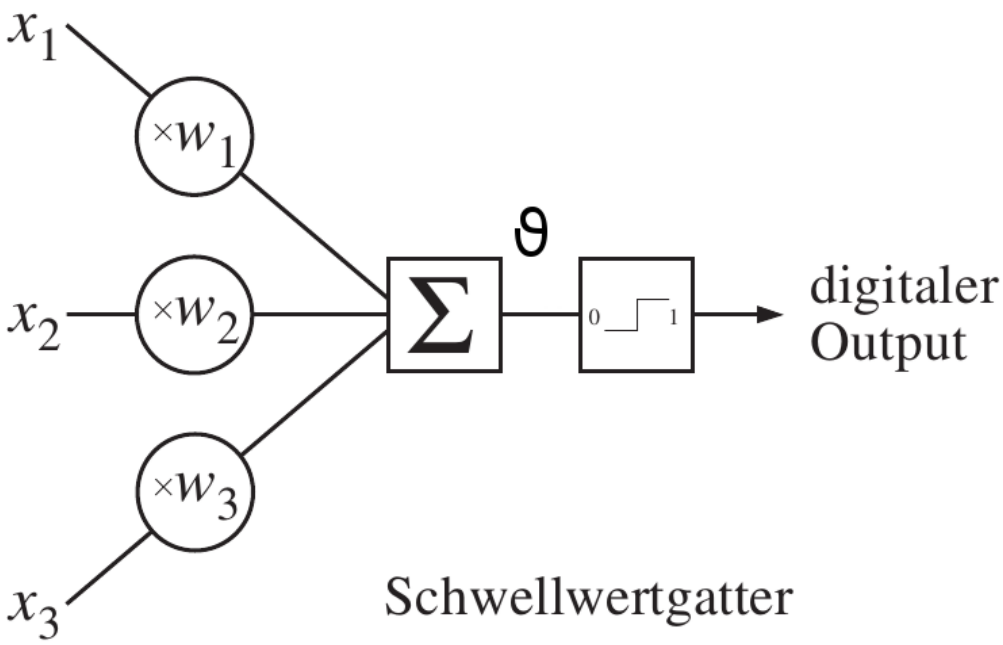
Ein künstliches Neuron

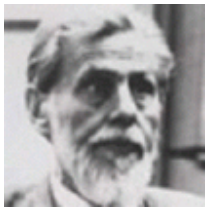




Einführung...

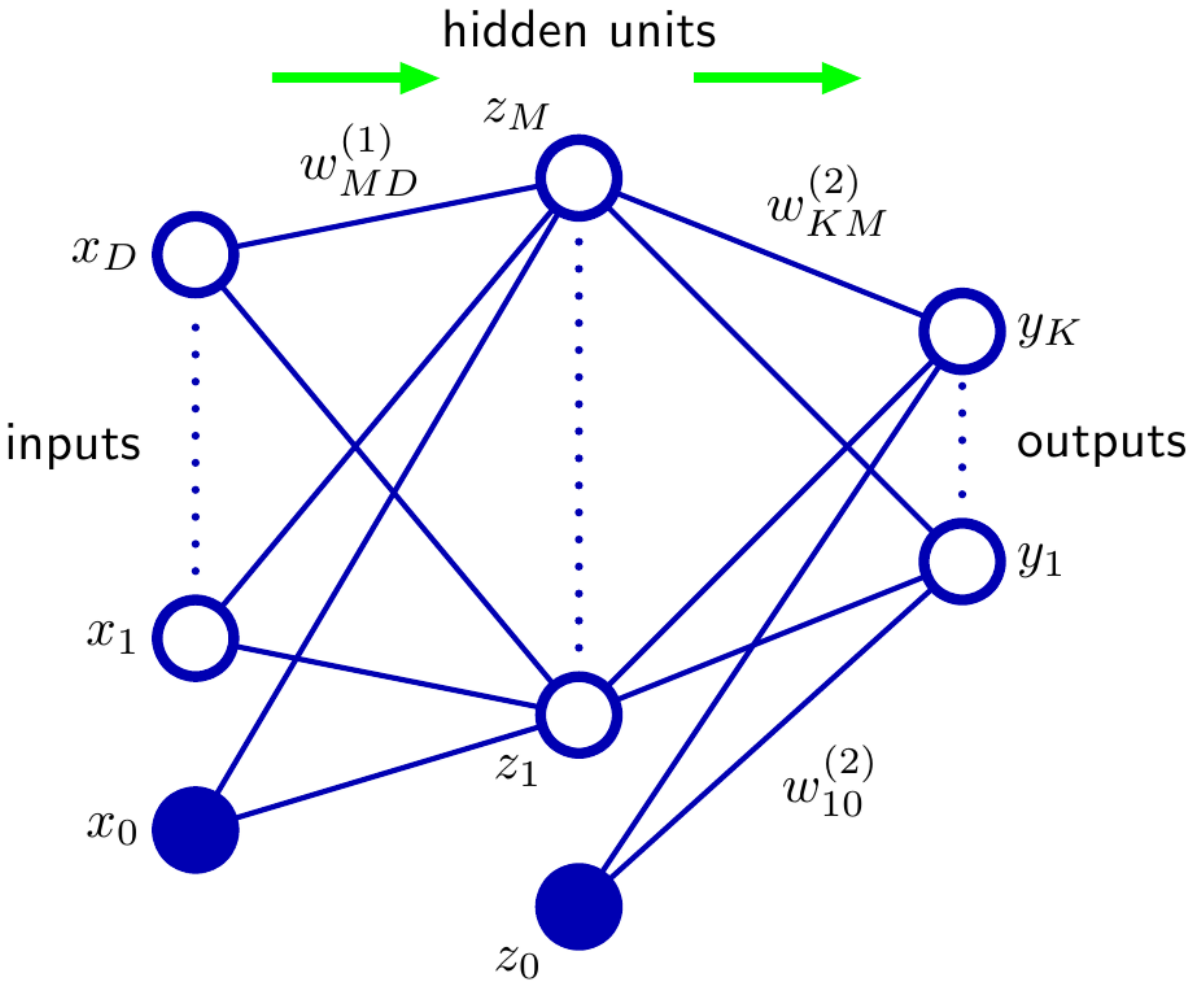
Ein künstliches Neuron





http://www.neuronalesnetz.de/applets/NettoInput.html

Ein neuronales Netz



Terminologie eines Netzes

Was bisher geschah...

1943 McCulloch-Pitts Neuron Das erste mathematische Modell eines Neurons. Heute bekannt als Perceptron

Was bisher geschah...

1943 McCulloch-Pitts Neuron Das erste mathematische Modell eines Neurons. Heute bekannt als Perceptron

1951 Edmond und Minsky bauen den ersten Neurocomputer

Was bisher geschah...

1943 McCulloch-Pitts Neuron Das erste mathematische Modell eines Neurons. Heute bekannt als Perceptron

1951 Edmond und Minsky bauen den ersten Neurocomputer

Problem: Wie lernt der Computer? Vorschläge?

Was bisher geschah...

1943 McCulloch-Pitts Neuron Das erste mathematische Modell eines Neurons. Heute bekannt als Perceptron

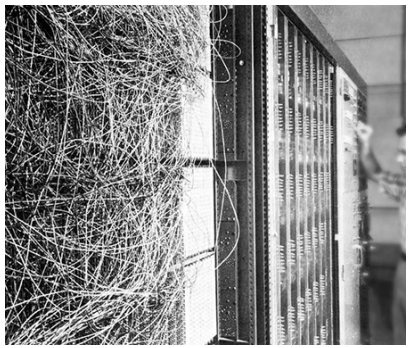
1951 Edmond und Minsky bauen den ersten Neurocomputer

Problem: Wie lernt der Computer? Vorschläge?

$$\Downarrow$$

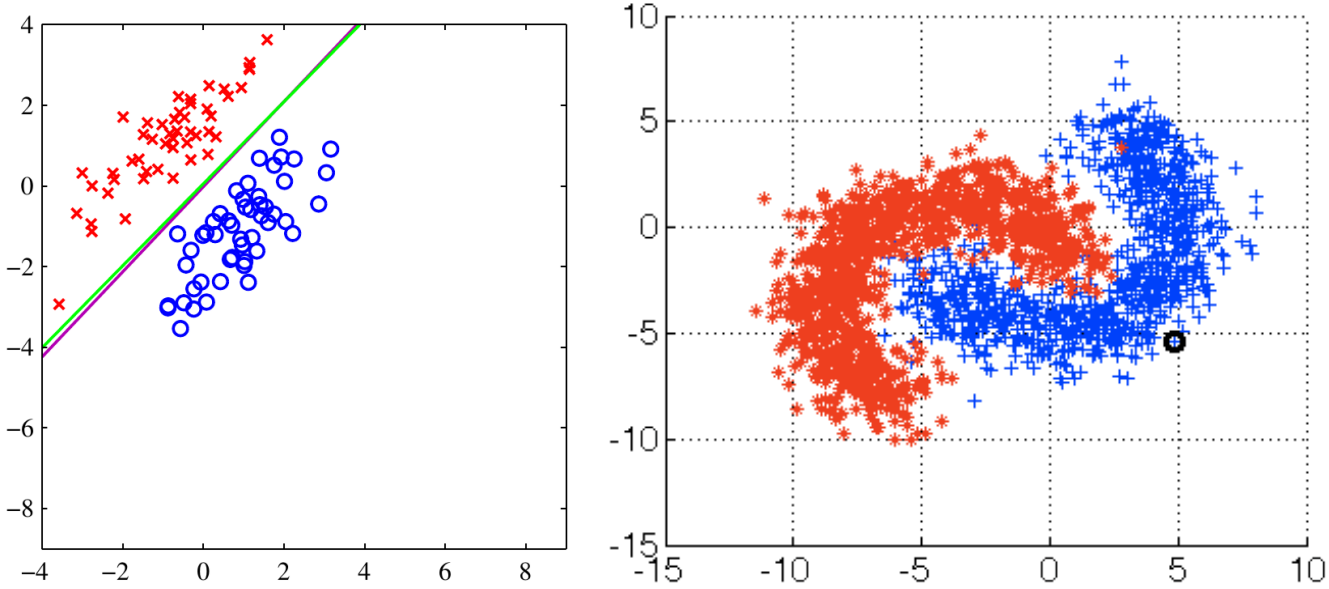
Anpassung der Gewichte

1959 Empory und Rosenblatt das Perceptron kann lernen $\rightarrow$ enthusiastische Forschung



Was sonst noch geschah...

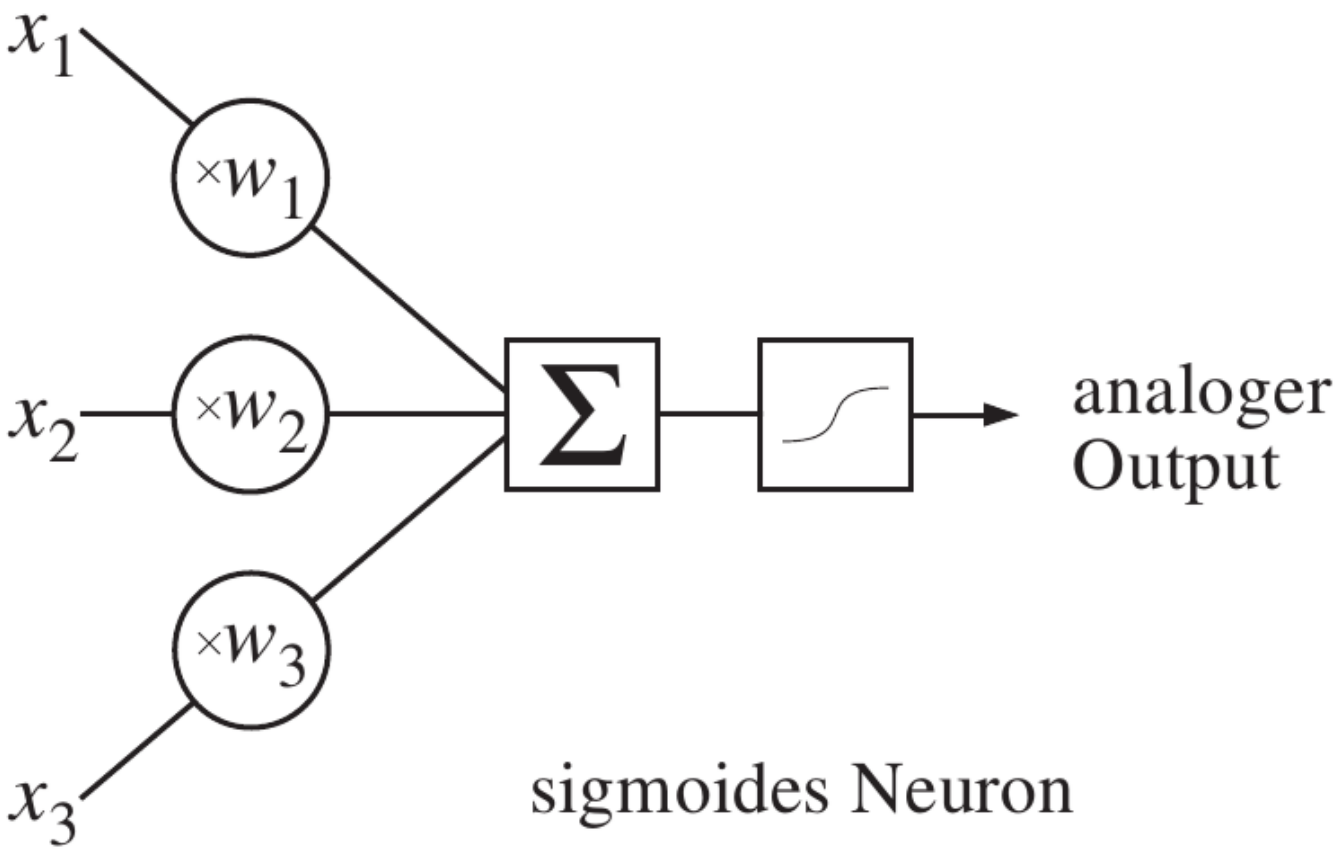
1969 Minsky und Papert Das Buch "Perceptrons" erscheint und dämpft die Euphorie...



Satz

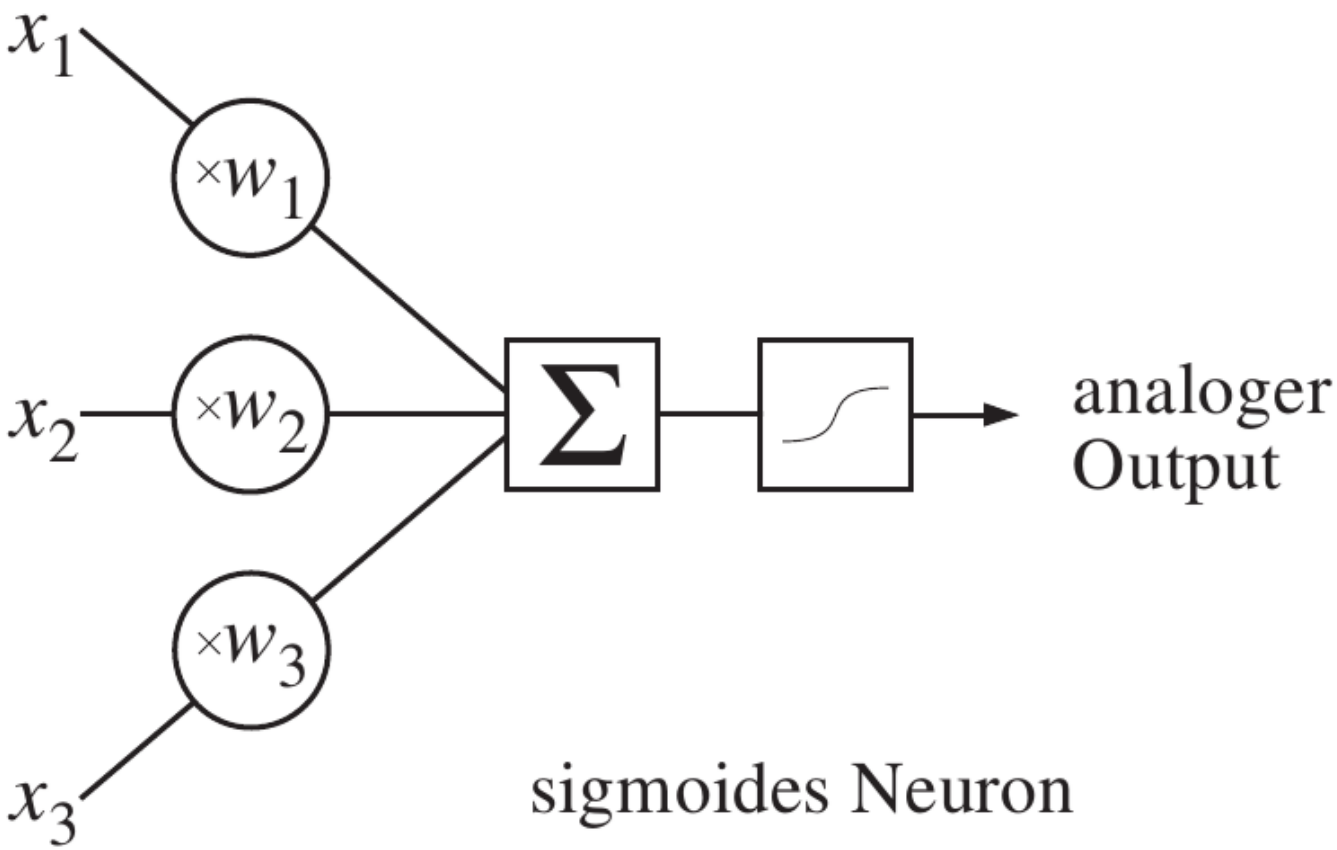
Ein einschichtiges Netzwerk kann nur Lösungen linear separabler Probleme lernen

Ein Schritt zurück...



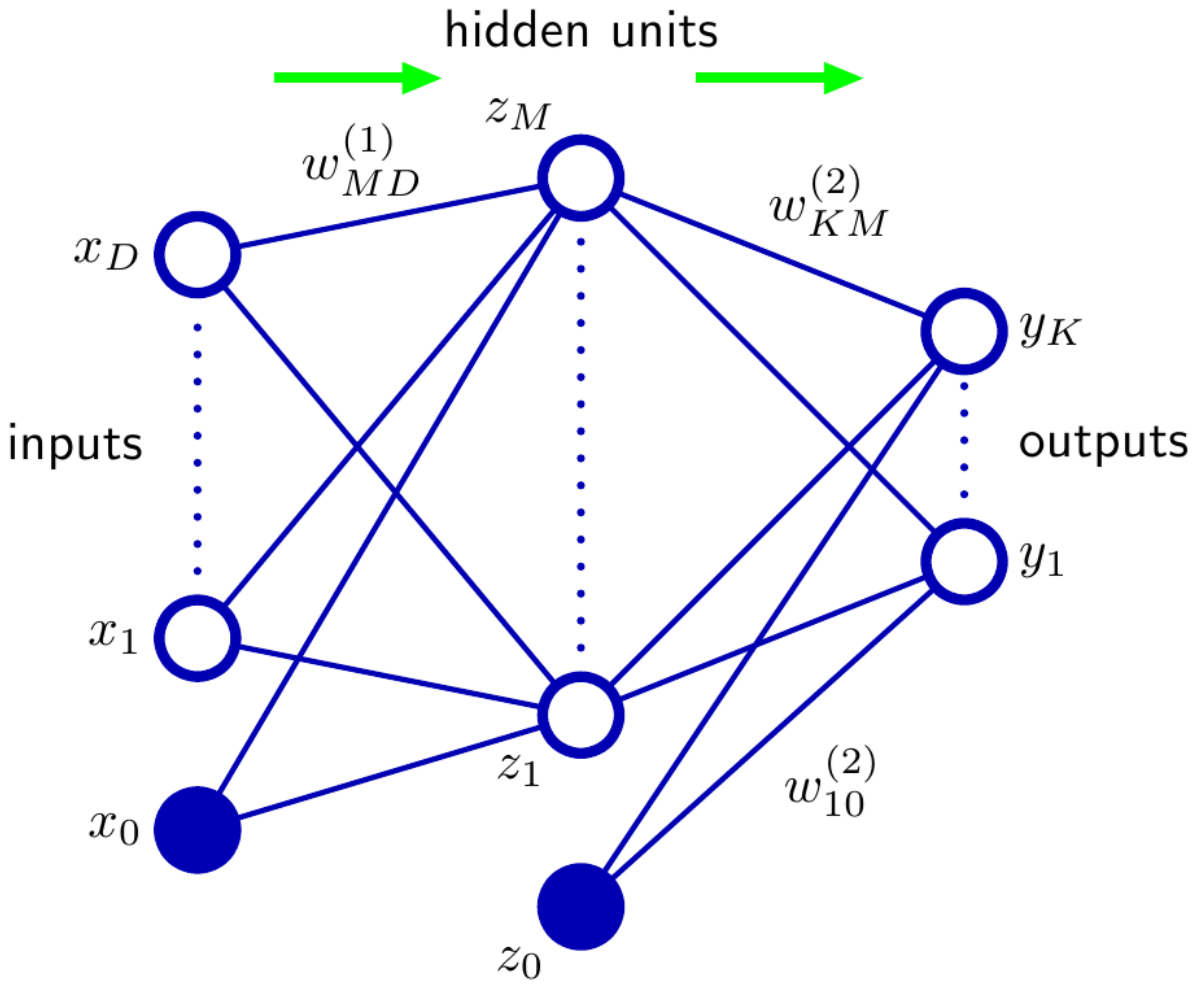
Eine neue Übergangsfunktion

Ein Schritt zurück...



http://www.neuronalesnetz.de/applets/ActivityFunction.html

...ein Schritt vorwärts



Formale Beschreibung eines Netzwerks

Ein Zweischichtiges Netzwerk

Input Variablen

$$x_1, \ldots, x_D \quad \text{mit Bias } x_0$$

Der Netzinput am $j-$ten Neuron des Hidden Layer ist

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}$$

Der Aktivitätslevel des $j-$ten Neurons des Hidden Layer ist

$$z_j = h(a_j), \quad j = 1, \ldots, M$$

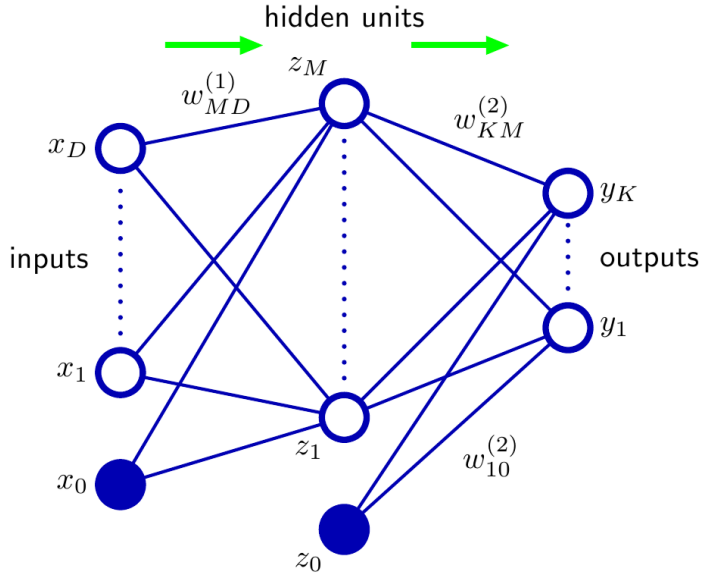
Der Aktivitätslevel am $k-$ten Neuron des Output Layer ist

$$b_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)}$$

Der Output des $k-$ten Neurons des Output Layer ist

$$y_k = g(b_k), \quad k = 1, \ldots, K$$

Formale Beschreibung des Netzwerks



Fasst man Gewichte und Inputs zusammen

$$\boldsymbol{w} = (w^{(1)}_{10}, \ldots, w^{(1)}_{M0}, w^{(1)}_{11}, \ldots w^{(1)}_{M1}, \ldots, w^{(2)}_{10}, \ldots w^{(2)}_{K0}, \ldots)$$
$$\boldsymbol{x} = (x_0, \ldots x_D)$$

so erhält man

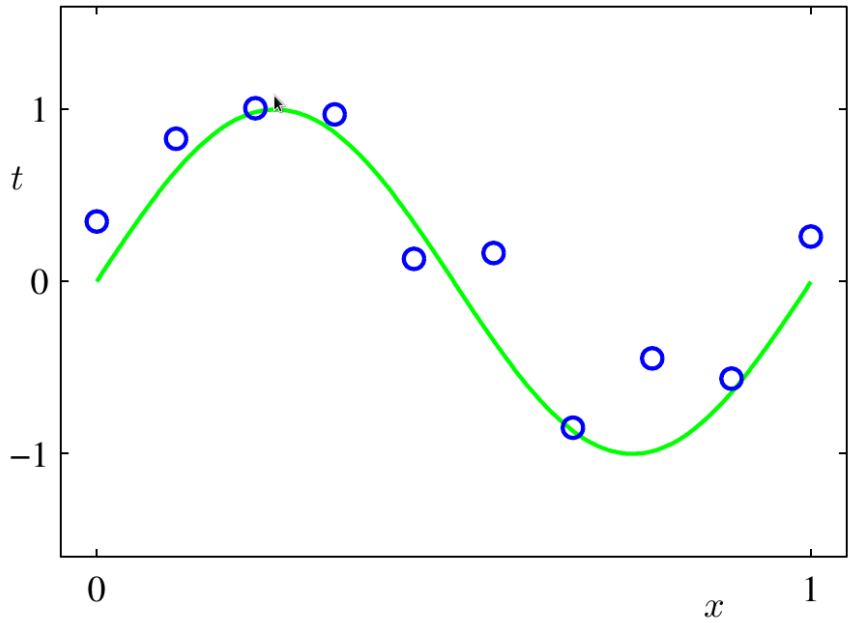
$$y_k(\boldsymbol{x}, \boldsymbol{w}) = g(b_k) = g\left(\sum_{j=0}^{M} w^{(2)}_{kj} h\left(\sum_{i=0}^{D} w^{(1)}_{ji} x_i\right)\right)$$

Was tun mit g und h?

Die Übertragungsfunktionen g und h sind stark problemabhängig

Was tun mit g und h?

Die Übertragungsfunktionen g und h sind stark problemabhängig



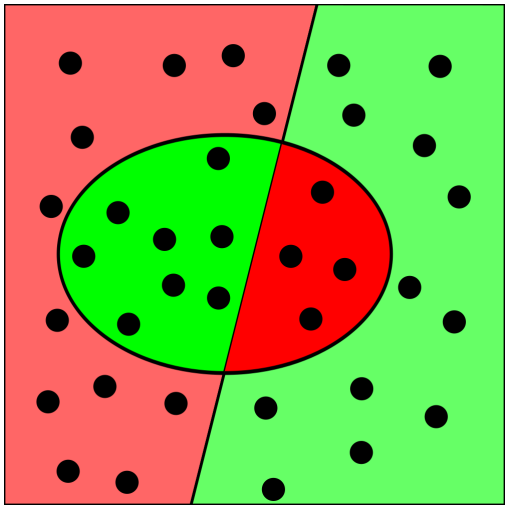
Regressionsproblem

Linear

$$h(a) = ca, \quad c \in \mathbb{R}$$

Was tun mit g und h?

Die Übertragungsfunktionen g und h sind stark problemabhängig



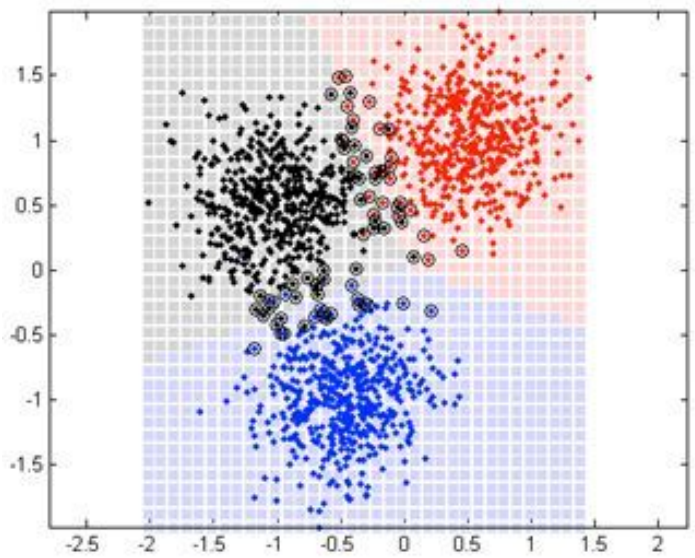
(Multiple) Binäre Klassifizierung

Sigmoid

$$h(a) = \frac{1}{1 + e^{-a}}$$

Was tun mit g und h?

Die Übertragungsfunktionen g und h sind stark problemabhängig



Mehrklassen Klassifizierung

Softmax

$$h(a_1, \ldots, a_K) = \frac{e^{a_k}}{\sum_{j=1}^{K} e^{a_j}}$$

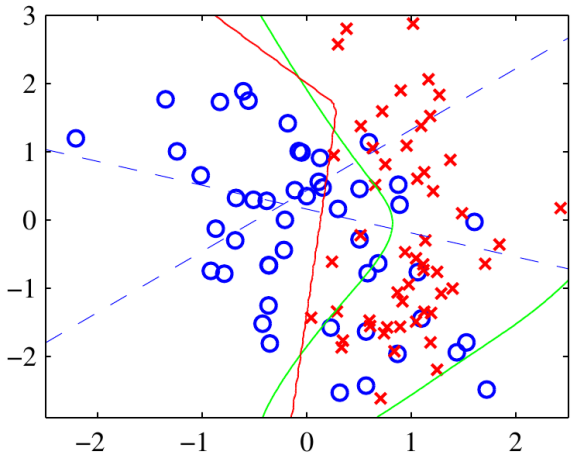
Der nächste Schritt

Beginn 1980er Beweis des folgenden Resultats gelingt

Satz

Ein zweischichtiges neuronales Netzwerk mit linearem Output Neuron und sigmoidalen Hidden Neuronen kann jede beliebige stetige Funktion auf einem kompakten Gebiet mit beliebiger Genauigkeit approximieren, sofern die Anzahl der Neuronen im Hidden Layer nur groß genug ist.

Damit ging's nun Schlag auf Schlag...



Beliebig komplexe Klassifizierungsprobleme und mehr...

Lernen lernen

1986 Rumelhart, Hinton und Williams machen das Konzept des Backpropagation Ansatzes populär $\rightarrow$ Lernen in mehrschichtigen Netzwerken möglich

Wie lernt ein Netzwerk?

Lernen lernen

1986 Rumelhart, Hinton und Williams machen das Konzept des Backpropagation Ansatzes populär $\rightarrow$ Lernen in mehrschichtigen Netzwerken möglich

Wie lernt ein Netzwerk?

Supervised Learning

Lernen lernen

1986 Rumelhart, Hinton und Williams machen das Konzept des Backpropagation Ansatzes populär $\rightarrow$ Lernen in mehrschichtigen Netzwerken möglich

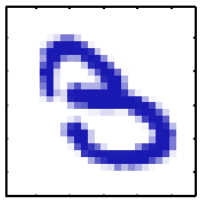
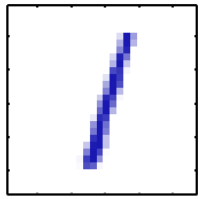
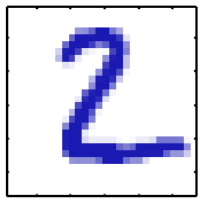
Wie lernt ein Netzwerk?

Supervised Learning

Unsupervised Learning

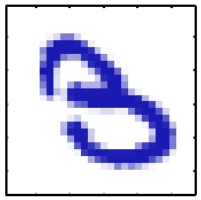
Supervised Learning

Erfordert ein Training Set bestehend aus

Samples	Targets
	3
	1
	2

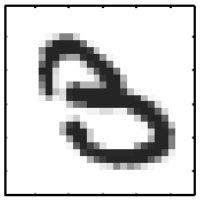
Formale Schreibweise

Datenvorverarbeitung



Formale Schreibweise

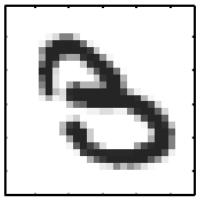
Datenvorverarbeitung



Konvertierung in ein Graustufenbild

Formale Schreibweise

Datenvorverarbeitung

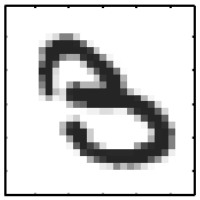


8x8 Pixel 8-Bit Graustufenbild als Vektor

$$\boldsymbol{x} = (0, 1, 6, 15, 12, 1, 0, 0, 0, 7, 16, 6, 6, 10, \dots) \in \mathbb{R}^{64}$$

Formale Schreibweise

Datenvorverarbeitung



8x8 Pixel 8-Bit Graustufenbild als Vektor

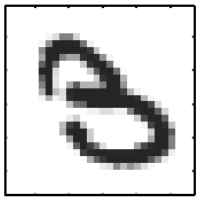
$$x = (0, 1, 6, 15, 12, 1, 0, 0, 0, 7, 16, 6, 6, 10, \dots) \in \mathbb{R}^{64}$$

Alle N Samples

$$\left\{x^1, x^2, \dots, x^N\right\}$$

Formale Schreibweise

Datenvorverarbeitung



8x8 Pixel 8-Bit Graustufenbild als Vektor

$$\boldsymbol{x} = (0, 1, 6, 15, 12, 1, 0, 0, 0, 7, 16, 6, 6, 10, \dots) \in \mathbb{R}^{64}$$

Alle N Samples

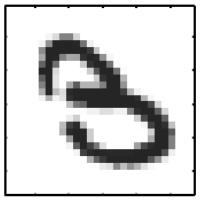
$$\left\{\boldsymbol{x}^1, \boldsymbol{x}^2, \dots, \boldsymbol{x}^N\right\}$$

zusammen mit den Targets $\{1, 2, 3\}$ in einem Targetvektor

$$\boldsymbol{t} = (3, 1, 2, \dots) \in \mathbb{R}^N$$

Formale Schreibweise

Datenvorverarbeitung



8x8 Pixel 8-Bit Graustufenbild als Vektor

$$\boldsymbol{x} = (0, 1, 6, 15, 12, 1, 0, 0, 0, 7, 16, 6, 6, 10, \dots) \in \mathbb{R}^{64}$$

Alle N Samples

$$\left\{\boldsymbol{x}^1, \boldsymbol{x}^2, \dots, \boldsymbol{x}^N\right\}$$

zusammen mit den Targets $\{1, 2, 3\}$ in einem Targetvektor

$$\boldsymbol{t} = (3, 1, 2, \dots) \in \mathbb{R}^N$$

ergibt das Training Set

Was sagt das Netz?

1. Initialisierung des Netzwerks mit Zufallsgewichten w^0

Was sagt das Netz?

1. Initialisierung des Netzwerks mit Zufallsgewichten $\boldsymbol{w}^0$
2. **Fehlerbestimmung**: Auswertung einer Errorfunktion z.B.

$$E(\boldsymbol{w}^0) = \left|y(\boldsymbol{x}^1, \boldsymbol{w}^0) - t_1\right|^2 + \cdots + \left|y(\boldsymbol{x}^N, \boldsymbol{w}^0) - t_N\right|^2 = \sum_{n=1}^{N} \left|y(\boldsymbol{x}^n, \boldsymbol{w}^0) - t_n\right|^2$$

$\rightarrow$Abweichung des Netzwerkoutputs von den gewünschten Targets

Was sagt das Netz?

1. Initialisierung des Netzwerks mit Zufallsgewichten $\boldsymbol{w}^0$
2. **Fehlerbestimmung**: Auswertung einer Errorfunktion z.B.

$$E(\boldsymbol{w}^0) = \left|y(\boldsymbol{x}^1, \boldsymbol{w}^0) - t_1\right|^2 + \cdots + \left|y(\boldsymbol{x}^N, \boldsymbol{w}^0) - t_N\right|^2 = \sum_{n=1}^{N} \left|y(\boldsymbol{x}^n, \boldsymbol{w}^0) - t_n\right|^2$$

$\rightarrow$Abweichung des Netzwerkoutputs von den gewünschten Targets

Ist $E(\boldsymbol{w}^0) = 0 \rightarrow$ Glückstreffer: *perfekte* Netzantwort

Was sagt das Netz?

1. Initialisierung des Netzwerks mit Zufallsgewichten $\boldsymbol{w}^0$
2. **Fehlerbestimmung**: Auswertung einer Errorfunktion z.B.

$$E(\boldsymbol{w}^0) = \left|y(\boldsymbol{x}^1, \boldsymbol{w}^0) - t_1\right|^2 + \cdots + \left|y(\boldsymbol{x}^N, \boldsymbol{w}^0) - t_N\right|^2 = \sum_{n=1}^{N} \left|y(\boldsymbol{x}^n, \boldsymbol{w}^0) - t_n\right|^2$$

$\rightarrow$Abweichung des Netzwerkoutputs von den gewünschten Targets

Ist $E(\boldsymbol{w}^0) = 0 \rightarrow$ Glückstreffer: *perfekte* Netzantwort
Ist $E(\boldsymbol{w}^0) > 0 \rightarrow$ Finde $\boldsymbol{w}^1$ sodass $E(\boldsymbol{w}^1) < E(\boldsymbol{w}^0)$
Hoffnung: nach K Schritten gilt

$$E(\boldsymbol{w}^0) > E(\boldsymbol{w}^1) > \cdots > E(\boldsymbol{w}^K) = \min_{\boldsymbol{w}} E(\boldsymbol{w})$$

Was sagt das Netz?

1. Initialisierung des Netzwerks mit Zufallsgewichten $\boldsymbol{w}^0$
2. **Fehlerbestimmung**: Auswertung einer Errorfunktion z.B.

$$E(\boldsymbol{w}^0) = \left|y(\boldsymbol{x}^1, \boldsymbol{w}^0) - t_1\right|^2 + \cdots + \left|y(\boldsymbol{x}^N, \boldsymbol{w}^0) - t_N\right|^2 = \sum_{n=1}^{N} \left|y(\boldsymbol{x}^n, \boldsymbol{w}^0) - t_n\right|^2$$

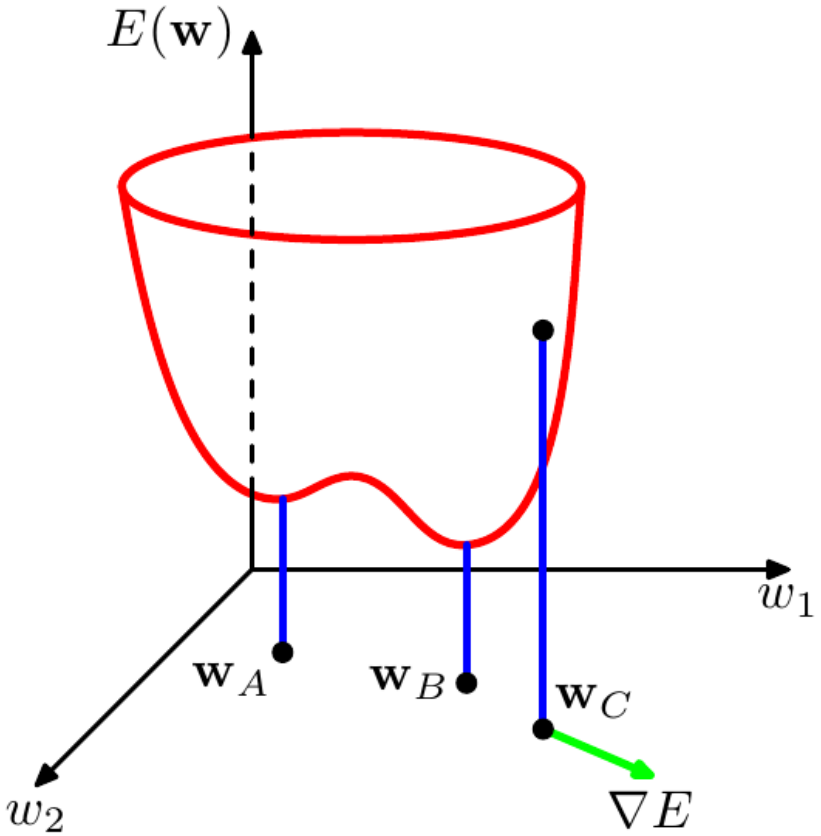
$\rightarrow$Abweichung des Netzwerkoutputs von den gewünschten Targets

Ist $E(\boldsymbol{w}^0) = 0 \rightarrow$ Glückstreffer: *perfekte* Netzantwort
Ist $E(\boldsymbol{w}^0) > 0 \rightarrow$ Finde $\boldsymbol{w}^1$ sodass $E(\boldsymbol{w}^1) < E(\boldsymbol{w}^0)$
Hoffnung: nach K Schritten gilt

$$E(\boldsymbol{w}^0) > E(\boldsymbol{w}^1) > \cdots > E(\boldsymbol{w}^K) = \min_{\boldsymbol{w}} E(\boldsymbol{w})$$

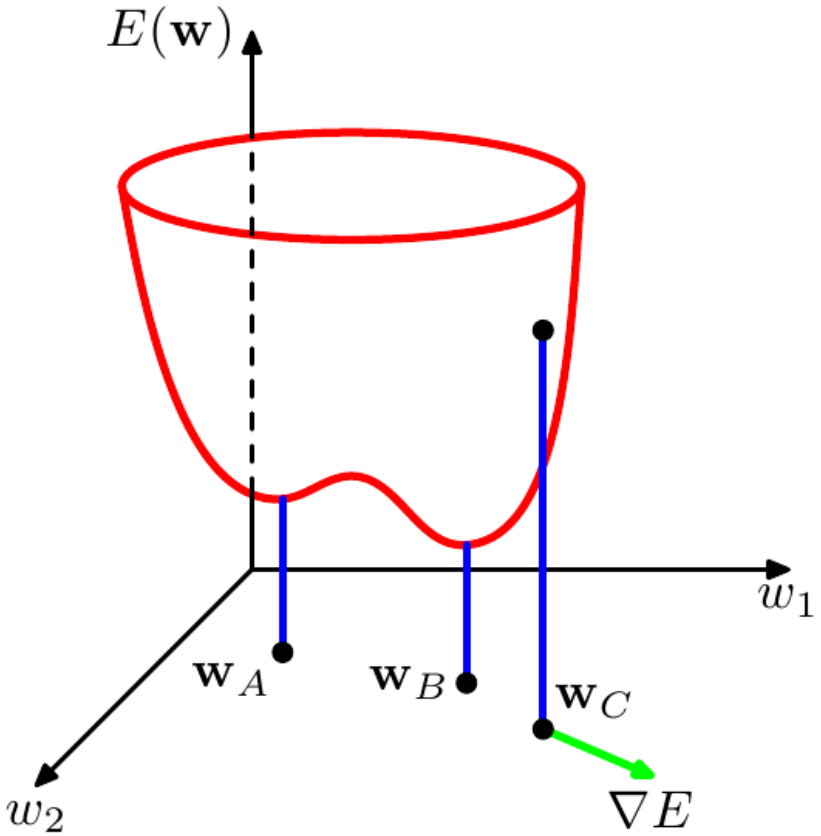
Optimierungsproblem

Ein Minimierungsproblem



Gradientenabstieg

Ein Minimierungsproblem



$$\boldsymbol{w}^{\tau+1} = \boldsymbol{w}^{\tau} - \eta \nabla E(\boldsymbol{w}^{\tau})$$

Backpropagation

Algorithm

Initialisierung: *Zufallsgewichte* $\boldsymbol{w}^0$, *setze* $\tau = 0$, $tol = 1e-3$

1: ***while*** $|\nabla E(\boldsymbol{w}^\tau)| > tol$ ***do***

6: ***end while***

Backpropagation

Algorithm

Initialisierung: *Zufallsgewichte* $\boldsymbol{w}^0$, *setze* $\tau = 0$, $tol = 1e-3$

1: ***while*** $|\nabla E(\boldsymbol{w}^\tau)| > tol$ ***do***

2: ***Forward-Pass***: *Berechnung des Netzoutputs* $y(\boldsymbol{x}^k, \boldsymbol{w}^\tau)$ *für jedes Sample* $\boldsymbol{x}^k$

6: ***end while***

Backpropagation

Algorithm

Initialisierung: *Zufallsgewichte* $\boldsymbol{w}^0$, *setze* $\tau = 0$, $tol = 1e-3$

1: ***while*** $|\nabla E(\boldsymbol{w}^\tau)| > tol$ ***do***

2: ***Forward-Pass***: *Berechnung des Netzoutputs* $y(\boldsymbol{x}^k, \boldsymbol{w}^\tau)$ *für jedes Sample* $\boldsymbol{x}^k$

3: ***Fehlerbestimmung***: *Auswertung von* $E(\boldsymbol{w}^\tau)$

6: ***end while***

Backpropagation

Algorithm

Initialisierung: *Zufallsgewichte* $\boldsymbol{w}^0$, *setze* $\tau = 0$, $tol = 1e-3$

1: ***while*** $|\nabla E(\boldsymbol{w}^\tau)| > tol$ ***do***

2: ***Forward-Pass***: *Berechnung des Netzoutputs* $y(\boldsymbol{x}^k, \boldsymbol{w}^\tau)$ *für jedes Sample* $\boldsymbol{x}^k$

3: ***Fehlerbestimmung***: *Auswertung von* $E(\boldsymbol{w}^\tau)$

4: ***Backward Pass***: *Berechnung der neuen Gewichte*

$$w_{ji}^{\tau+1} = w_{ji}^{\tau} - \eta \frac{\partial E(\boldsymbol{w}^\tau)}{\partial w_{ji}}$$

6: ***end while***

Backpropagation

Algorithm

Initialisierung: *Zufallsgewichte* $\boldsymbol{w}^0$, *setze* $\tau = 0$, $tol = 1e-3$

1: ***while*** $|\nabla E(\boldsymbol{w}^\tau)| > tol$ ***do***

2: ***Forward-Pass***: *Berechnung des Netzoutputs* $y(\boldsymbol{x}^k, \boldsymbol{w}^\tau)$ *für jedes Sample* $\boldsymbol{x}^k$

3: ***Fehlerbestimmung***: *Auswertung von* $E(\boldsymbol{w}^\tau)$

4: ***Backward Pass***: *Berechnung der neuen Gewichte*

$$w_{ji}^{\tau+1} = w_{ji}^{\tau} - \eta \frac{\partial E(\boldsymbol{w}^\tau)}{\partial w_{ji}}$$

5: ***Update***: $\tau \leftarrow \tau + 1$

6: ***end while***

An die Maschinen...