

Neuronale Netze

Prof. Dr. Rudolf Kruse
Christoph Doell, M.Sc.

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



Zu meiner Person: Rudolf Kruse

1979 Dipl. Mathematik (Nebenfach Informatik) von TU Braunschweig

dort 1980 promoviert, 1984 habilitiert

2 Jahre hauptamtlicher Mitarbeiter bei Fraunhofer

1986 Ruf als Professor für Informatik der TU Braunschweig

seit 1996 Professor an der Universität Magdeburg

Forschung: Data Mining, Explorative Datenanalyse, Fuzzy-Systeme, Neuronale Netze, Evolutionäre Algorithmen, Bayes'sche Netze

`mailto:kruse@iws.cs.uni-magdeburg.de`

Büro: G29-008, Telefon: 0391 67-58706

Sprechstunde: Mi., 11:00–12:00 Uhr

Zur Arbeitsgruppe: Computational Intelligence

Lehre:

Intelligente Systeme	Bachelor (2 V + 2 Ü, 5 CP)
Evolutionäre Algorithmen	Bachelor (2 V + 2 Ü, 5 CP)
Neuronale Netze	Bachelor (2 V + 2 Ü, 5 CP)
Fuzzy-Systeme	Master (2 V + 2 Ü, 6 CP)
Bayes-Netze	Master (2 V + 2 Ü, 6 CP)
Intelligente Datenanalyse	Master (2 V + 2 Ü, 6 CP)
Seminare: Clustering/Classification Algorithms, Hot Topics in Neural Networks	

Forschungsbeispiele:

Dynamische Analyse von sozialen Netzen	(Pascal Held)
Interaktive Datenanalyse	(Christian Braune)
Maschinelles Lernen mit Hilfe von Kognitiven Architekturen	(Christoph Doell)

Zur Vorlesung

Vorlesungstermine: Mo., 13:15–14:45 Uhr, G29-307

Vorlesungsausfall: 16.05.2016 (Pfingstmontag)

Vorlesungsende: 04.07.2016

Informationen zur Vorlesung:

<http://fuzzy.cs.ovgu.de/wiki/pmwiki.php?n=Lehre.NN2016>

- wöchentliche Vorlesungsfolien als PDF
- Übungsblätter ebenfalls
- wichtige Ankündigungen und Termine!

Inhalte und Lernziele der Vorlesung

Einführung in die Grundlagen der (Künstlichen) Neuronalen Netze

Behandlung von Lernparadigmen und -algorithmen

Behandlung von verschiedenen Netzmodellen

Anwendungsbeispiele

Zur Übung

Zielsetzung:

Anwendung von Methoden der Datenanalyse mit Neuronalen Netzen zur Lösung von Klassifikations-, Regressions- und weiteren statistischen Problemen

Bewertung und Anwendung neuronaler Lernverfahren zur Analyse komplexer Systeme

Befähigung zur Entwicklung von Neuronalen Netzen

Ihre Aufgabe:

Nacharbeiten des Vorlesungsstoffs

Bearbeitung der Übungsaufgaben

aktive Teilnahme an den Übungen

Durchführung der Übungen

Sie werden aktiv und erklären Ihre Lösungen!

Tutor macht auf Fehler aufmerksam und beantwortet Fragen

das „Vorrechnen“ der Aufgaben ist nicht Sinn der Übung

ganz bewusst: keine ausgearbeiteten Musterlösungen

Tutor: Christoph Doell <mailto:doell@iws.cs.uni-magdeburg.de>

Raum G29-013, Sprechstunde: immer, wenn Bürotür offen steht

Tutor: Marco Dankel <mailto:marco.dankel@st.ovgu.de>

PNK-Tutorin: Michelle Bieber <mailto:ibieber@iws.cs.ovgu.de>

Übung: 2 Termine zur Auswahl

Do., 11:15–12:45 Uhr in Raum G29-037

Fr., 09:15–10:45 Uhr in Raum G29-037

Anmeldung: <https://iws.cs.uni-magdeburg.de:8080/frs/subscribe/NN>

Übungsbeginn der regulären Übung : nächste Woche (Übungsblatt 1)

Extraübung für PNK-Studierende

Di., 17:15–18:45 Uhr in Raum G22A-362

Beginn der PNK-Übung: morgen

Zur Prüfung

schriftliche Klausur: 120 Minuten

voraussichtlich Mitte Juli

Termine, Räume etc. werden in Vorlesung u. WWW angekündigt

Durchführung ohne Hilfsmittel (nur Taschenrechner)

nur Schreibmaterial (Stifte/Füller, die blau oder schwarz schreiben)

Bekanntgabe der Ergebnisse: HISQIS

Einsichtnahme in die Klausur ist möglich (Termin im WWW)

Schein- und Prüfungskriterien

Studierende, die den Kurs mit Prüfung oder (un)benotetem Schein beenden wollen, müssen

regelmäßig und gut in Übungen mitarbeiten,

mind. 50% der Aufgaben votieren,

mind. 2x Lösung zu schriftlicher Aufgabe präsentieren,

Klausur nach dem Kurs bestehen

Studierende der **Philosophie-Neurowissenschaften-Kognition**

müssen außerdem aktiv an einer zusätzlichen Übung teilnehmen

Inhalt der Vorlesung

Einleitung

Motivation, biologischer Hintergrund

Schwellenwertelemente

Definition, geometrische Deutung, Grenzen, Netze von SWE, Training

Allgemeine Neuronale Netze

Struktur, Arbeitsweise, Training

Mehrschichtige Perzeptren

Definition, Funktionsapproximation, Gradientenabstieg, Backpropagation, Varianten, Sensitivitätsanalyse

Radiale-Basis-Funktions-Netze

Definition, Funktionsapproximation Initialisierung, Training, allgemeine Version

Selbstorganisierende Karten

Definition, Lernende Vektorquantisierung, Nachbarschaft von Ausgabeneuronen

Hopfield-Netze

Definition, Konvergenz, Assoziativspeicher, Lösen von Optimierungsproblemen

Rekurrente Neuronale Netze

Differentialgleichungen, Vektornetze, Backpropagation über die Zeit

Support-Vektor-Maschinen

Empirische Risikominimierung, statistische Lerntheorie, Kernmethoden

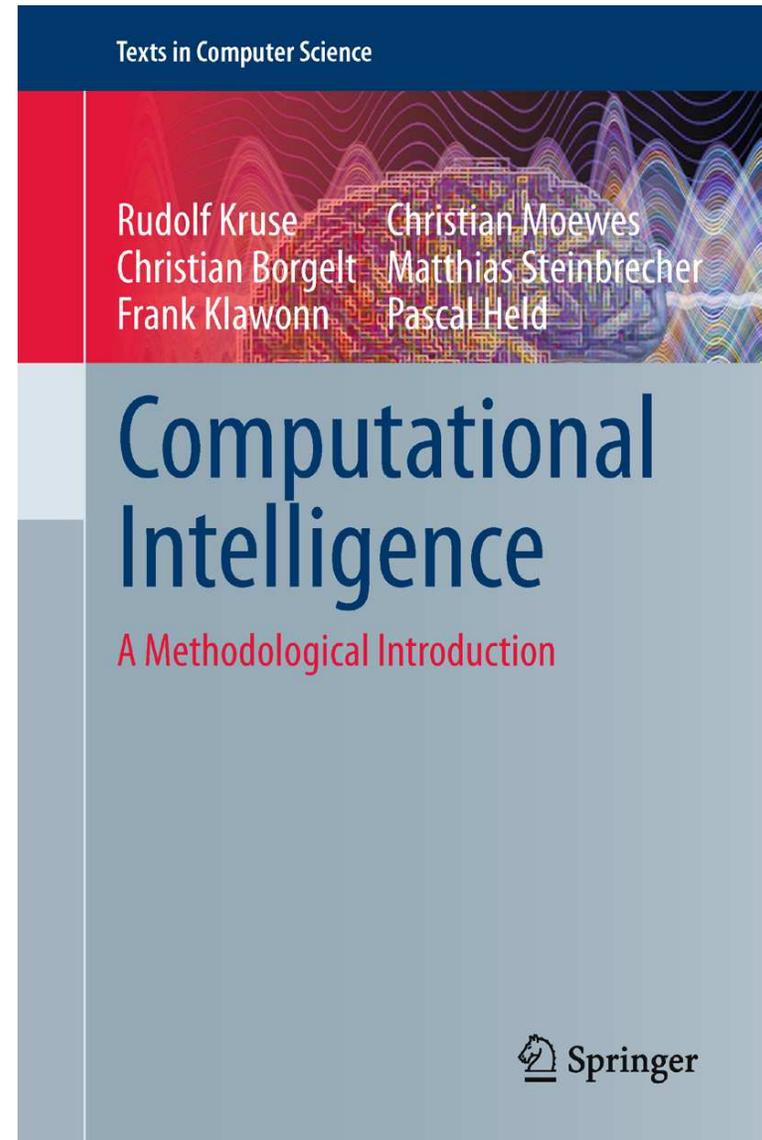
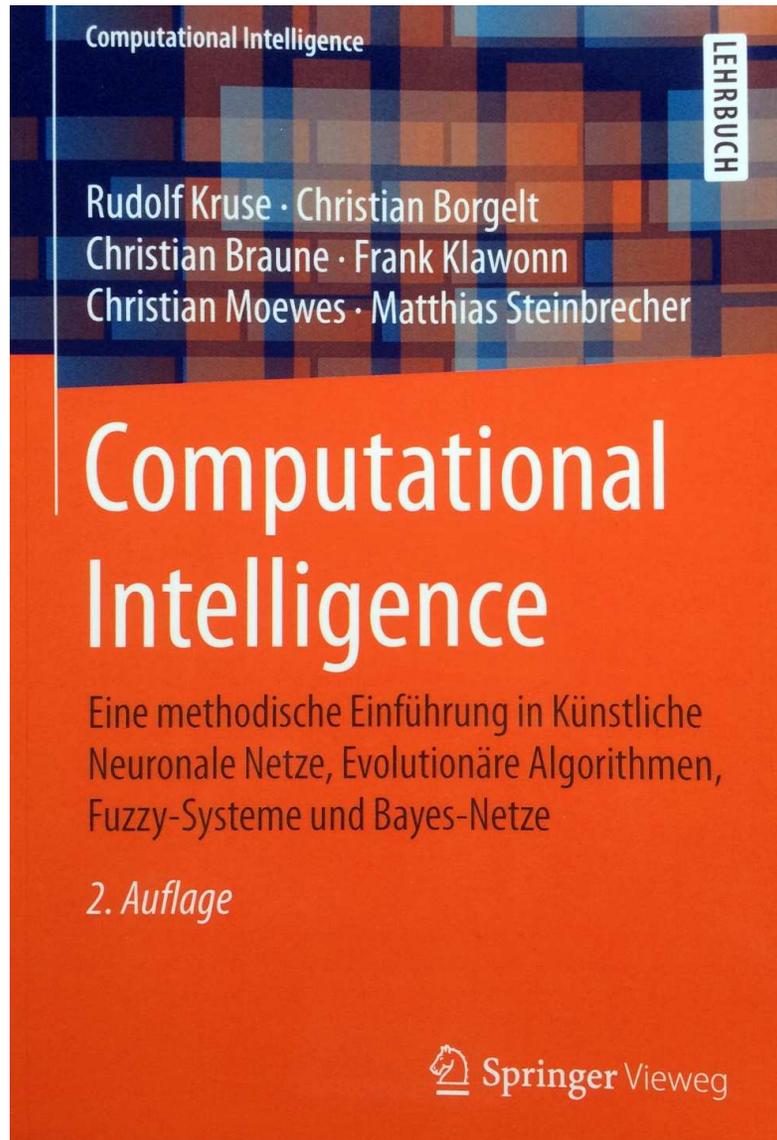
Gepulste Neuronale Netze

Realistischere Simulation des Gehirns, Anwendung in Computerchips

Lernen vielschichtiger Neuronaler Netze (Deep Learning)

Probleme bei vielschichtigen Netzen, Lösungsansätze, Praktische Anwendung

Bücher zur Vorlesung



Literatur zur Lehrveranstaltung

Kruse, Borgelt, Klawonn, Moewes, Steinbrecher und Held. *Computational Intelligence: A Methodological Introduction*. Springer, London, 2013.

Kruse, Borgelt, Braune, Klawonn, Moewes und Steinbrecher. *Computational Intelligence: Eine methodische Einführung in Künstliche Neuronale Netze, Evolutionäre Algorithmen, Fuzzy-Systeme und Bayes-Netze*. Springer Vieweg, Wiesbaden, 2015.

Kriesel. *Ein kleiner Überblick über neuronale Netze*. Manuskript, erhältlich auf <http://www.dkriesel.com>, 2007.

Aktuelle Motivation: AlphaGo

2016: AlphaGo besiegt Lee Sedol 4:1

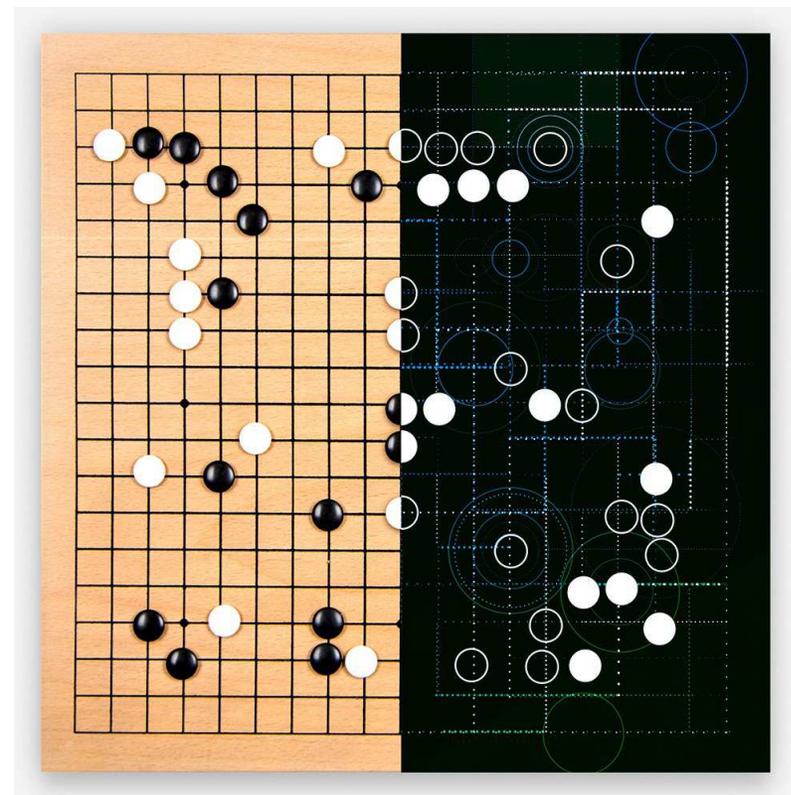
Es nutzt dabei tiefe Neuronale Netze:

Policy-Networks zur Vorauswahl möglicher Züge

Value-Networks zur Bewertung von Spielpositionen

Monte-Carlo Baumsuche zum Finden der besten Züge

Reinforcement Learning zur Verbesserung der Neuronalen Netze



Motivation: Warum (künstliche) neuronale Netze?

(Neuro-)Biologie / (Neuro-)Physiologie / Psychologie:

- Ausnutzung der Ähnlichkeit zu echten (biologischen) neuronalen Netzen
- Modellierung zum Verständnis Arbeitsweise von Nerven und Gehirn durch Simulation

Informatik / Ingenieurwissenschaften / Wirtschaft

- Nachahmen der menschlichen Wahrnehmung und Verarbeitung
- Lösen von Lern-/Anpassungsproblemen sowie Vorhersage- und Optimierungsproblemen

Physik / Chemie

- Nutzung neuronaler Netze, um physikalische Phänomene zu beschreiben
- Spezialfall: Spin-Glas (Legierungen von magnetischen und nicht-magnetischen Metallen)

Konventionelle Rechner vs. Gehirn

	Computer	Gehirn
Verarbeitungseinheiten	1 CPU, 10^{10} Transistoren	10^{11} Neuronen
Speicherkapazität	10^{11} Bytes RAM, 10^{13} Bytes Festspeicher	10^{11} Neuronen, 10^{14} Synapsen
Verarbeitungsgeschwindigkeit	10^{-8} Sekunden	10^{-3} Sekunden
Bandbreite	$10^{12} \frac{\text{bits}}{\text{s}}$	$10^{14} \frac{\text{bits}}{\text{s}}$
Neuronale Updates pro Sekunde	10^6	10^{14}

Konventionelle Rechner vs. Gehirn

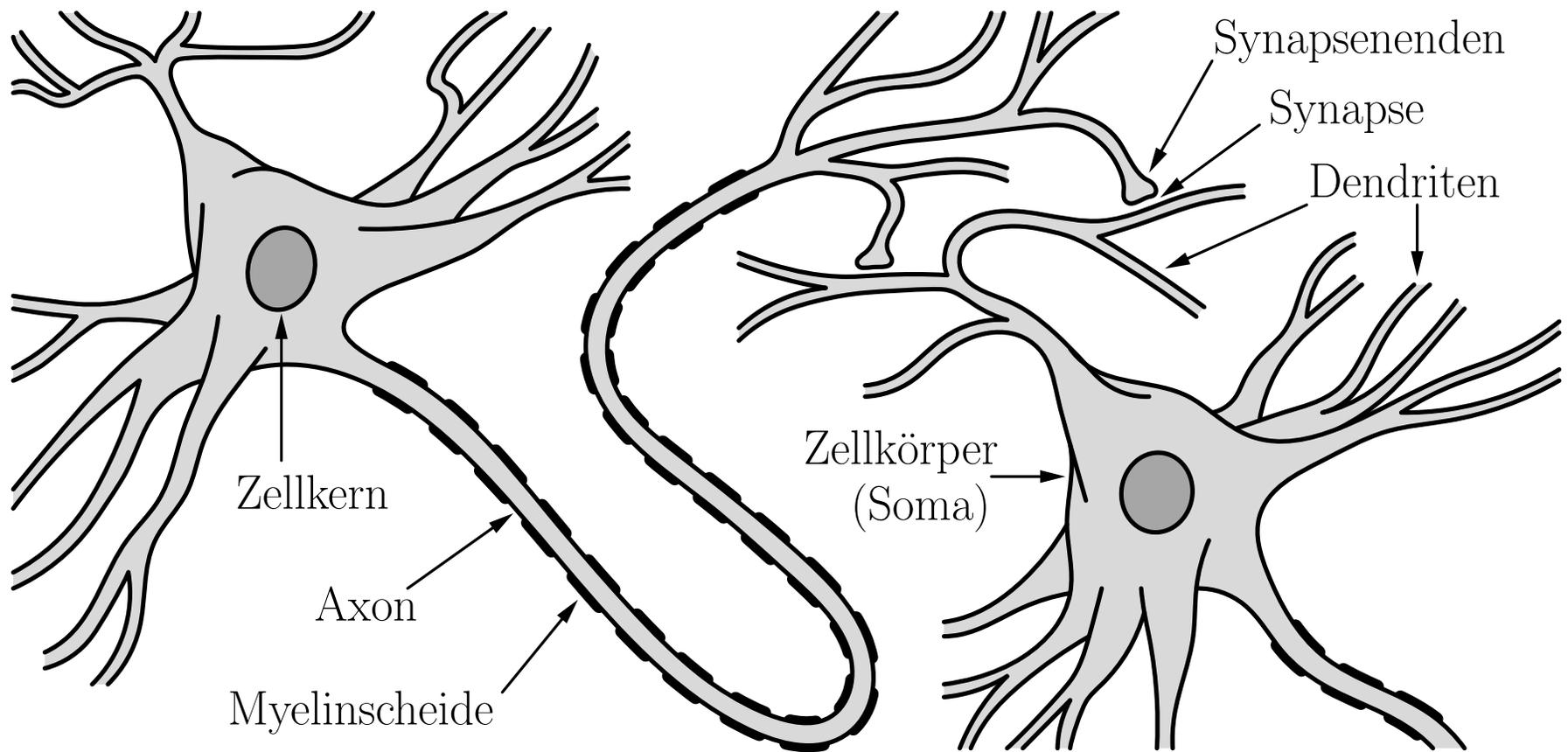
Beachte: die Hirnschaltzeit ist mit 10^{-3} s recht langsam, aber Updates erfolgen parallel. Dagegen braucht die serielle Simulation auf einem Rechner mehrere hundert Zyklen für ein Update.

Vorteile neuronaler Netze:

- Hohe Verarbeitungsgeschwindigkeit durch massive Parallelität
- Funktionstüchtigkeit selbst bei Ausfall von Teilen des Netzes (Fehlertoleranz)
- Langsamer Funktionsausfall bei fortschreitenden Ausfällen von Neuronen (*graceful degradation*)
- Gut geeignet für induktives Lernen

Es erscheint daher sinnvoll, diese Vorteile natürlicher neuronaler Netze künstlich nachzuahmen.

Struktur eines prototypischen biologischen Neurons



(Stark) vereinfachte Beschreibung neuronaler Informationsverarbeitung

Das Axonende gibt Chemikalien ab, **Neurotransmitter** genannt.

Diese bewirken an der Membran des Empfängerendriten die Veränderung der Polarisierung.

(Das Innere ist typischerweise 70mV negativer als die Außenseite.)

Abnahme in der Potentialdifferenz: **anregende** Synapse

Zunahme in der Potentialdifferenz: **hemmende** Synapse

Wenn genügend anregende Information vorhanden ist, wird das Axon depolarisiert.

Das resultierende **Aktionspotential** pflanzt sich entlang des Axons fort.

(Die Geschwindigkeit hängt von der Bedeckung mit Myelin ab.)

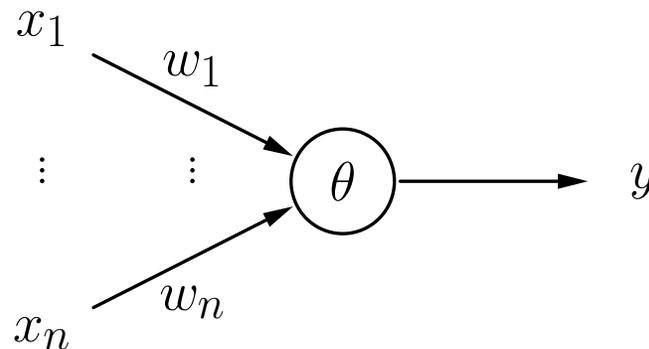
Wenn das Aktionspotential die Synapsenenden erreicht, löst es die Abgabe von Neurotransmittern aus.

Schwellenwertelemente

Schwellenwertelemente

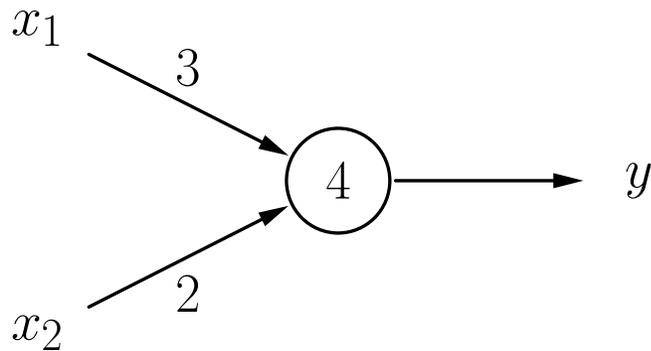
Ein **Schwellenwertelement** (Threshold Logic Unit, TLU) ist eine Verarbeitungseinheit für Zahlen mit n Eingängen x_1, \dots, x_n und einem Ausgang y . Das Element hat einen **Schwellenwert** θ und jeder Eingang x_i ist mit einem **Gewicht** w_i versehen. Ein Schwellenwertelement berechnet die Funktion

$$y = \begin{cases} 1, & \text{falls } \vec{x}\vec{w} = \sum_{i=1}^n w_i x_i \geq \theta, \\ 0, & \text{sonst.} \end{cases}$$



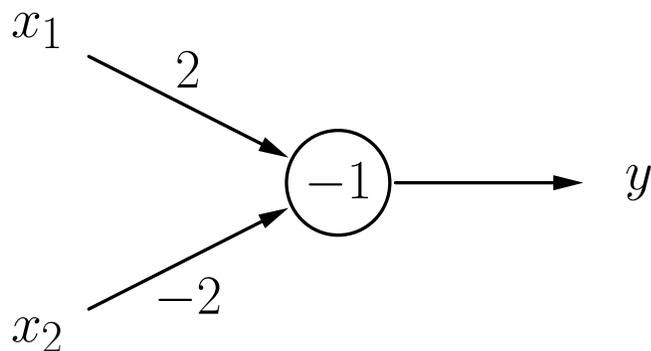
Schwellenwertelemente: Beispiele

Schwellenwertelement für die Konjunktion $x_1 \wedge x_2$.



x_1	x_2	$3x_1 + 2x_2$	y
0	0	0	0
1	0	3	0
0	1	2	0
1	1	5	1

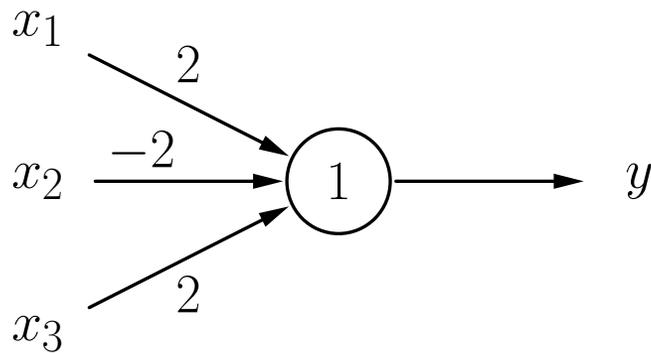
Schwellenwertelement für die Implikation $x_2 \rightarrow x_1$.



x_1	x_2	$2x_1 - 2x_2$	y
0	0	0	1
1	0	2	1
0	1	-2	0
1	1	0	1

Schwellenwertelemente: Beispiele

Schwellenwertelement für $(x_1 \wedge \overline{x_2}) \vee (x_1 \wedge x_3) \vee (\overline{x_2} \wedge x_3)$.



x_1	x_2	x_3	$\sum_i w_i x_i$	y
0	0	0	0	0
1	0	0	2	1
0	1	0	-2	0
1	1	0	0	0
0	0	1	2	1
1	0	1	4	1
0	1	1	0	0
1	1	1	2	1

Rückblick: Geradendarstellungen

Geraden werden typischerweise in einer der folgenden Formen dargestellt:

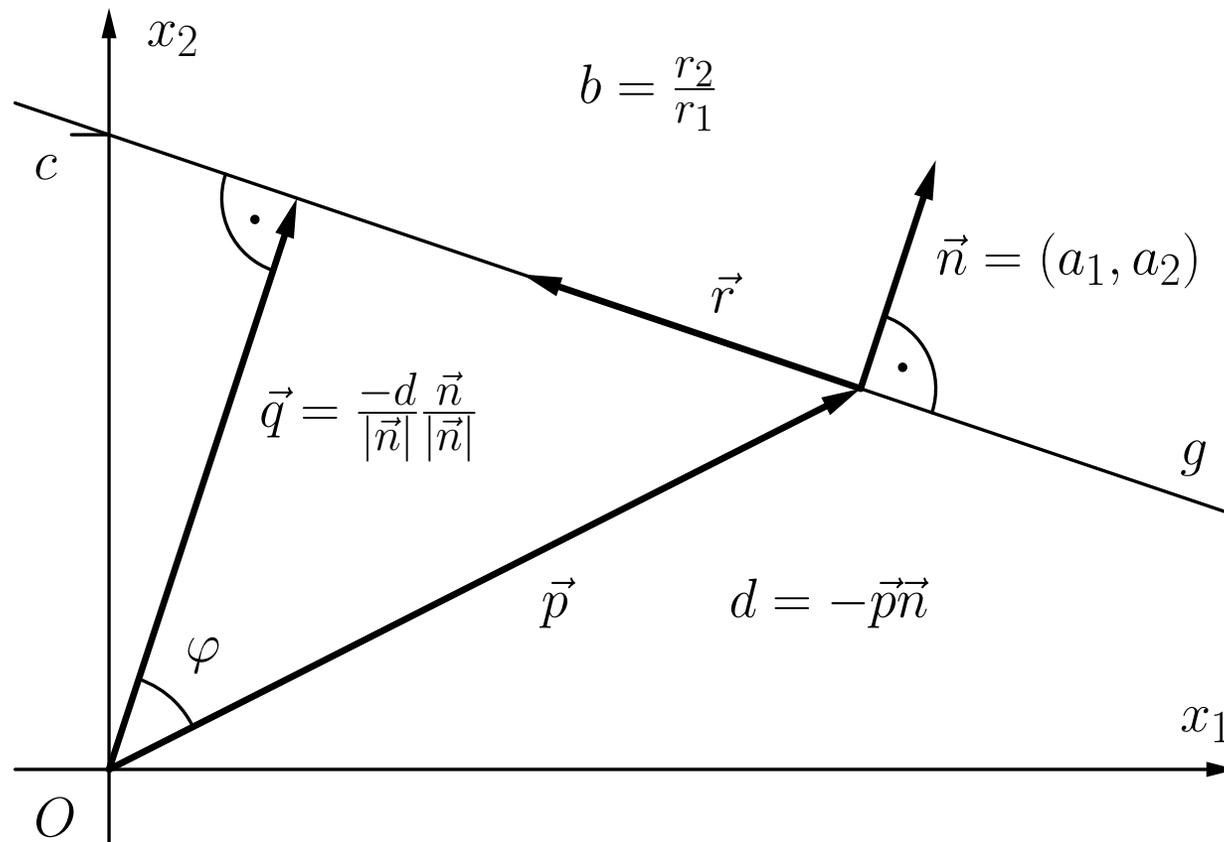
Explizite Form:	$g \equiv x_2 = bx_1 + c$
Implizite Form:	$g \equiv a_1x_1 + a_2x_2 + d = 0$
Punkt-Richtungs-Form:	$g \equiv \vec{x} = \vec{p} + k\vec{r}$
Normalform	$g \equiv (\vec{x} - \vec{p})\vec{n} = 0$

mit den Parametern

- b : Anstieg der Geraden
- c : Abschnitt der x_2 -Achse
- \vec{p} : Vektor zu einem Punkt auf der Gerade (Ortsvektor)
- \vec{r} : Richtungsvektor der Gerade
- \vec{n} : Normalenvektor der Gerade

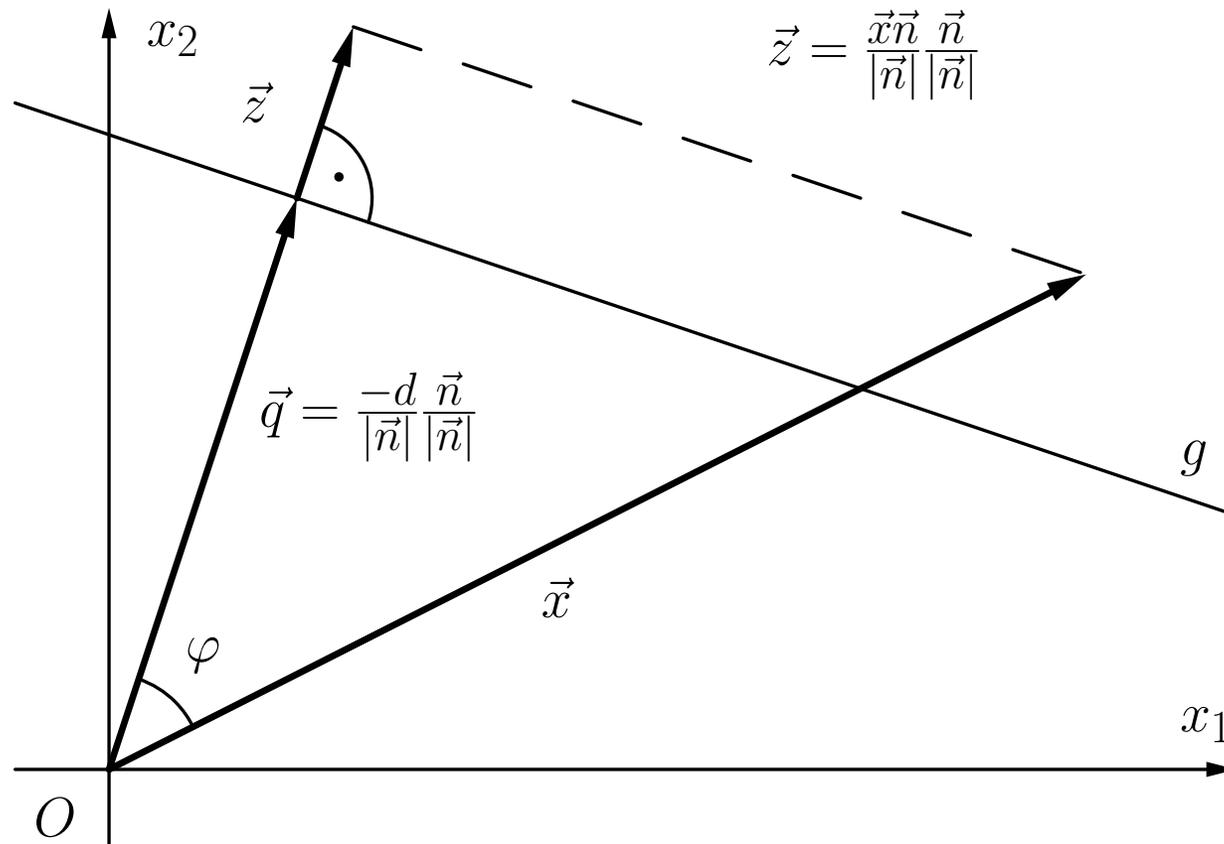
Schwellenwertelemente: Geometrische Interpretation

Eine Gerade und ihre definierenden Eigenschaften.



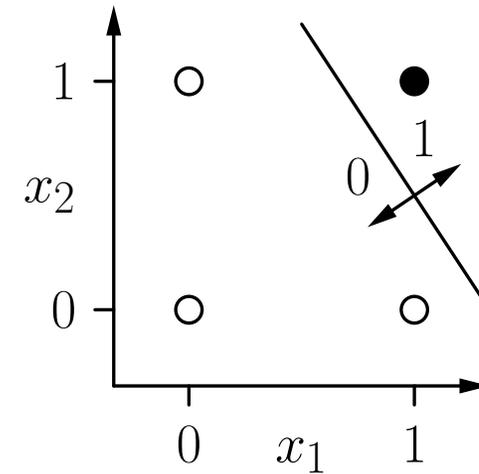
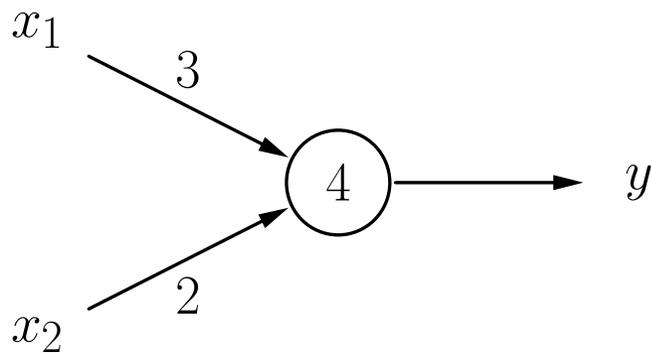
Schwellenwertelemente: Geometrische Interpretation

Bestimmung, auf welcher Seite ein Punkt \vec{x} liegt.

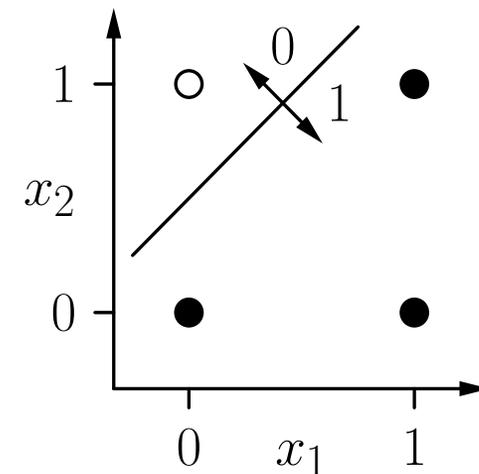
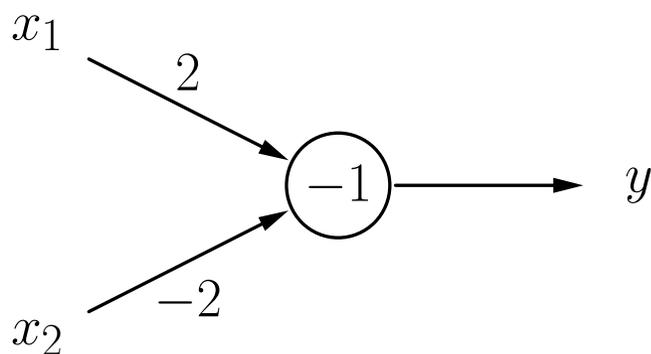


Schwellenwertelemente: Geometrische Interpretation

Schwellenwertelement für $x_1 \wedge x_2$.

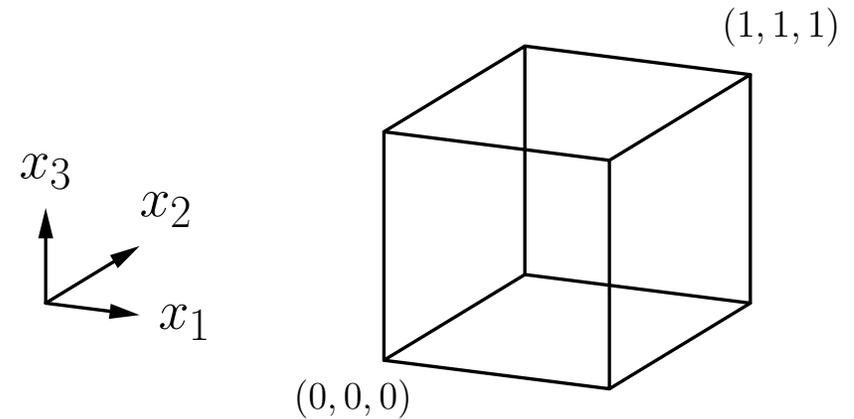


Ein Schwellenwertelement für $x_2 \rightarrow x_1$.

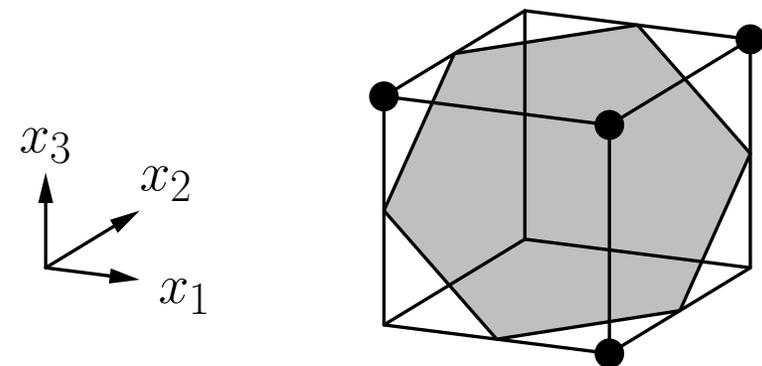
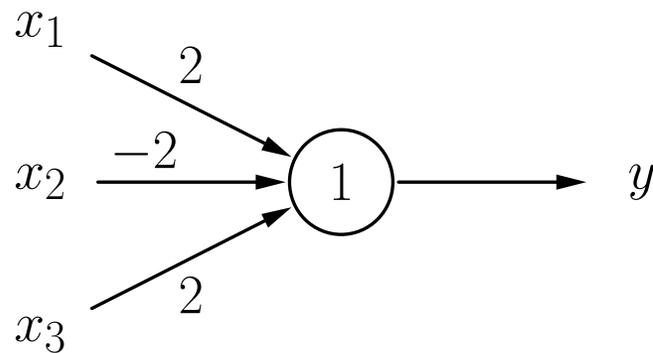


Schwellenwertelemente: Geometrische Interpretation

Darstellung 3-dimensionaler
Boolescher Funktionen:



Schwellenwertelement für $(x_1 \wedge \overline{x_2}) \vee (x_1 \wedge x_3) \vee (\overline{x_2} \wedge x_3)$.



Schwellenwertelemente: lineare Separabilität

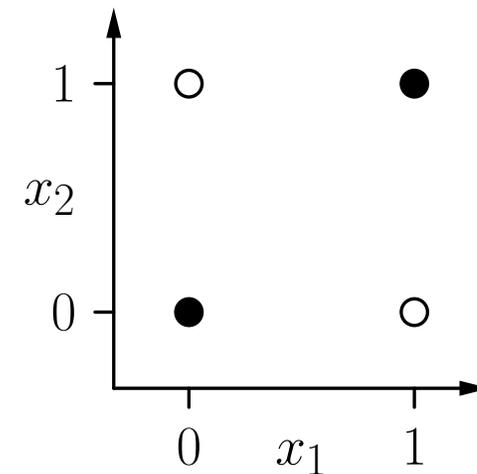
Zwei Punktmenge in einem n -dimensionalen Raum heißen linear separabel, wenn sie durch eine $(n-1)$ -dimensionale Hyperebene getrennt werden können. Die Punkte der einen Menge dürfen dabei auch auf der Hyperebene liegen.

Eine Boolesche Funktion heißt linear separabel, falls die Menge der Urbilder von 0 und die Menge der Urbilder von 1 linear separabel sind.

Schwellenwertelemente: Grenzen

Das Biimplikationsproblem $x_1 \leftrightarrow x_2$: Es gibt keine Trenngerade.

x_1	x_2	y
0	0	1
1	0	0
0	1	0
1	1	1



Formaler Beweis durch *reductio ad absurdum*:

$$\text{da } (0, 0) \mapsto 1: \quad 0 \geq \theta, \quad (1)$$

$$\text{da } (1, 0) \mapsto 0: \quad w_1 < \theta, \quad (2)$$

$$\text{da } (0, 1) \mapsto 0: \quad w_2 < \theta, \quad (3)$$

$$\text{da } (1, 1) \mapsto 1: \quad w_1 + w_2 \geq \theta. \quad (4)$$

(2) und (3): $w_1 + w_2 < 2\theta$. Mit (4): $2\theta > \theta$, oder $\theta > 0$. Widerspruch zu (1).

Schwellenwertelemente: Grenzen

Vergleich zwischen absoluter Anzahl und der Anzahl linear separabler Boolescher Funktionen.

([Widner 1960] zitiert in [Zell 1994])

Eingaben	Boolesche Funktionen	linear separable Funktionen
1	4	4
2	16	14
3	256	104
4	65536	1774
5	$4.3 \cdot 10^9$	94572
6	$1.8 \cdot 10^{19}$	$5.0 \cdot 10^6$

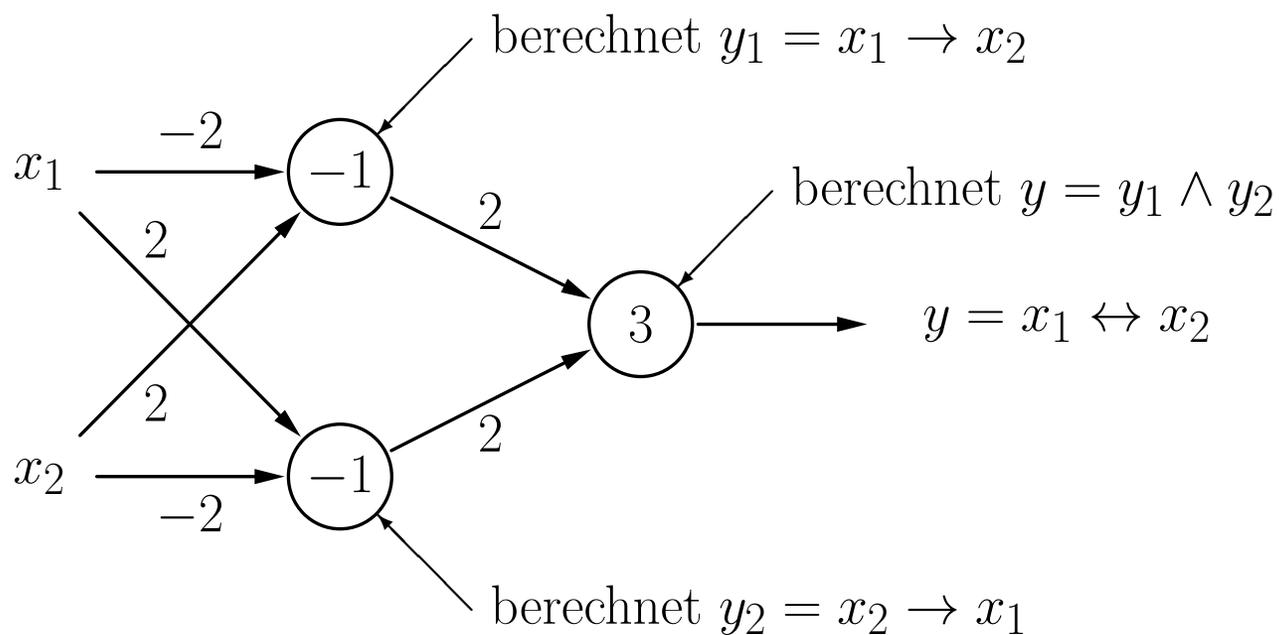
Für viele Eingaben kann ein SWE fast keine Funktion berechnen.

Netze aus Schwellenwertelementen sind notwendig, um die Berechnungsfähigkeiten zu erweitern.

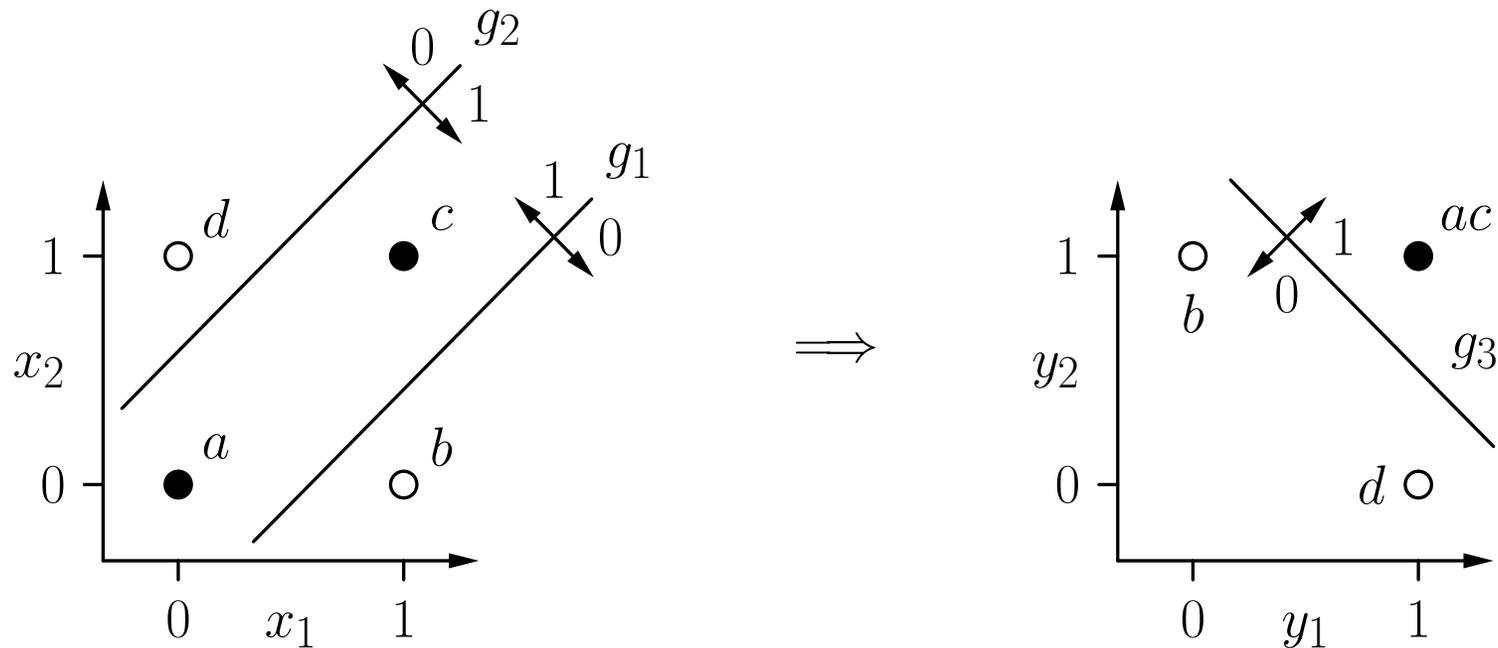
Netze aus Schwellenwertelementen

Biimplikationsproblem, Lösung durch ein Netzwerk.

Idee: logische Zerlegung $x_1 \leftrightarrow x_2 \equiv (x_1 \rightarrow x_2) \wedge (x_2 \rightarrow x_1)$



Lösung des Biimplikationsproblems: Geometrische Interpretation



Die erste Schicht berechnet neue Boolesche Koordinaten für die Punkte.
Nach der Koordinatentransformation ist das Problem linear separabel.

Darstellung beliebiger Boolescher Funktionen

Sei $y = f(x_1, \dots, x_n)$ eine Boolesche Funktion mit n Variablen.

(i) Stelle $f(x_1, \dots, x_n)$ in disjunktiver Normalform dar. D.h. bestimme $D_f = K_1 \vee \dots \vee K_m$, wobei alle K_j Konjunktionen von n Literalen sind, d.h., $K_j = l_{j1} \wedge \dots \wedge l_{jn}$ mit $l_{ji} = x_i$ (positives Literal) oder $l_{ji} = \neg x_i$ (negatives Literal).

(ii) Lege ein Neuron für jede Konjunktion K_j der disjunktiven Normalform an (mit n Eingängen — ein Eingang pro Variable), wobei

$$w_{ji} = \begin{cases} 2, & \text{falls } l_{ji} = x_i, \\ -2, & \text{falls } l_{ji} = \neg x_i, \end{cases} \quad \text{und} \quad \theta_j = n - 1 + \frac{1}{2} \sum_{i=1}^n w_{ji}.$$

(iii) Lege ein Ausgabeneuron an (mit m Eingängen — ein Eingang für jedes Neuron, das in Schritt (ii) angelegt wurde), wobei

$$w_{(n+1)k} = 2, \quad k = 1, \dots, m, \quad \text{und} \quad \theta_{n+1} = 1.$$

Trainieren von Schwellenwertelementen

Trainieren von Schwellenwertelementen

Die geometrische Interpretation bietet eine Möglichkeit, SWE mit 2 und 3 Eingängen zu konstruieren, aber:

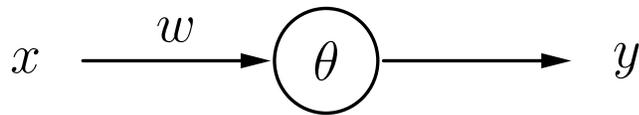
- Es ist keine automatische Methode (Visualisierung und Begutachtung ist nötig).
- Nicht möglich für mehr als drei Eingabevariablen.

Grundlegende Idee des automatischen Trainings:

- Beginne mit zufälligen Werten für Gewichte und Schwellenwert.
- Bestimme den Ausgabefehler für eine Menge von Trainingsbeispielen.
- Der Fehler ist eine Funktion der Gewichte und des Schwellenwerts:
 $e = e(w_1, \dots, w_n, \theta)$.
- Passe Gewichte und Schwellenwert so an, dass der Fehler kleiner wird.
- Wiederhole diese Anpassung, bis der Fehler verschwindet.

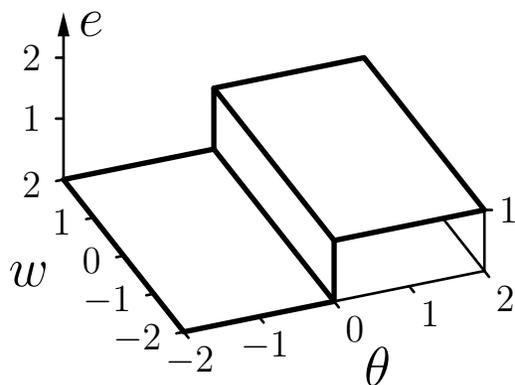
Trainieren von Schwellenwertelementen

Schwellenwertelement mit einer Eingabe für die Negation $\neg x$.

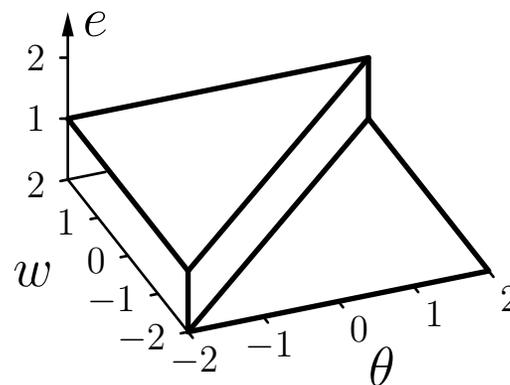


x	y
0	1
1	0

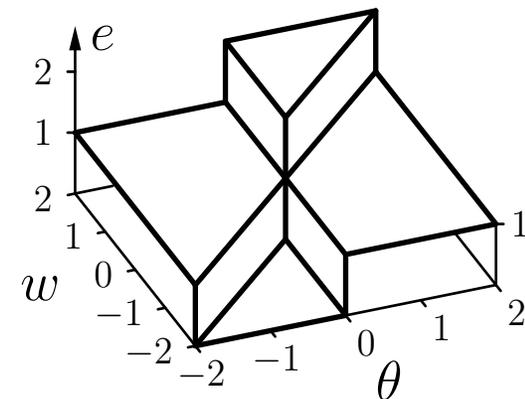
Ausgabefehler als eine Funktion von Gewicht und Schwellenwert.



Fehler für $x = 0$



Fehler für $x = 1$



Summe der Fehler

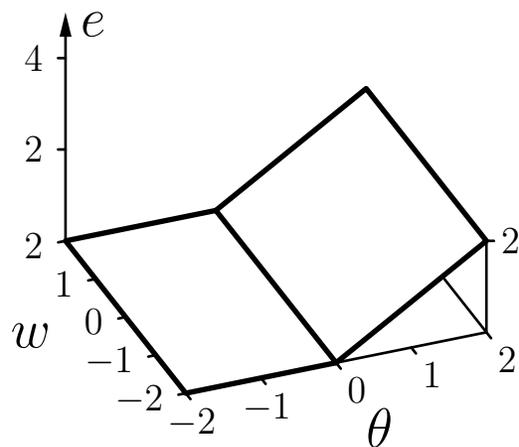
Trainieren von Schwellenwertelementen

Die Fehlerfunktion kann nicht direkt verwendet werden, da sie aus Plateaus besteht.

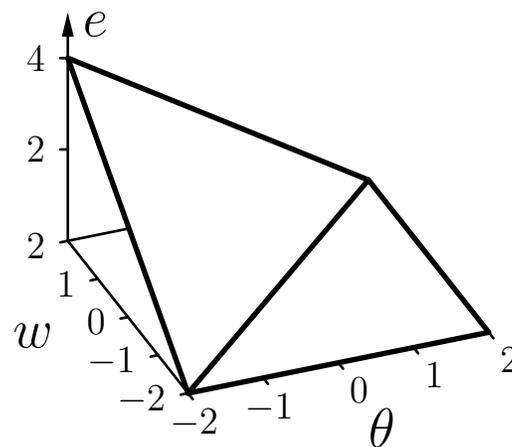
Lösung: Falls die berechnete Ausgabe falsch ist, dann berücksichtige, wie weit θ überschritten (für $x = 0$) oder unterschritten ist (für $x = 1$).

anschaulich: Berechnung ist „umso falscher“, je weiter θ überschritten (für $x = 0$) bzw. unterschritten ist (für $x = 1$).

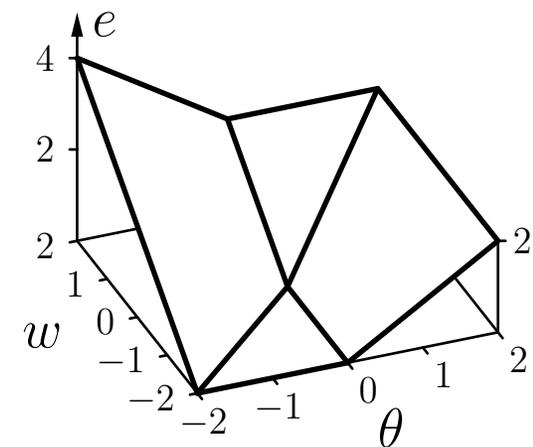
Modifizierter Ausgabebefehler als Funktion von \vec{w} und θ .



Fehler für $x = 0$



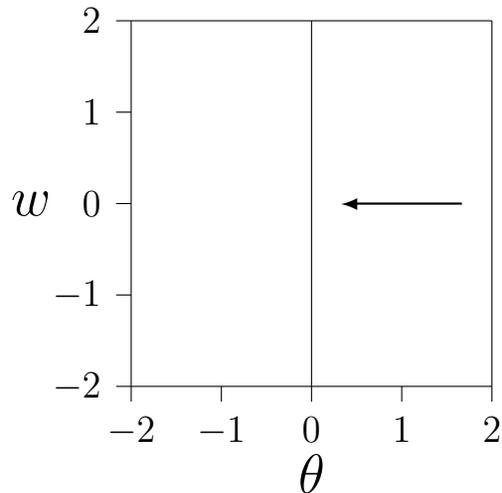
Fehler für $x = 1$



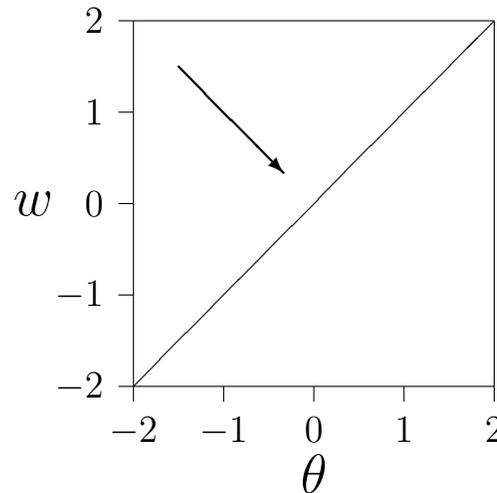
Summe der Fehler

Trainieren von Schwellenwertelementen

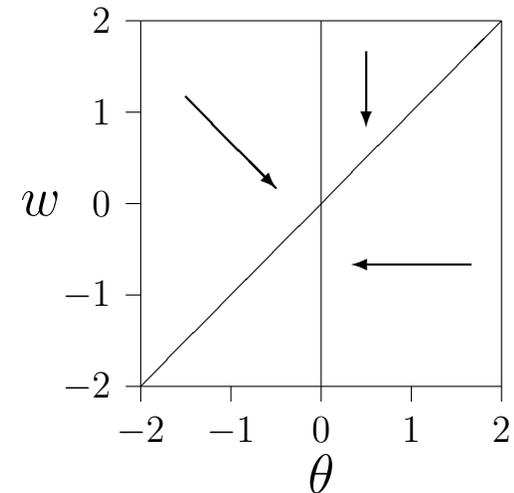
Schema der resultierenden Richtungen der Parameteränderungen.



Änderungen für $x = 0$



Änderungen für $x = 1$



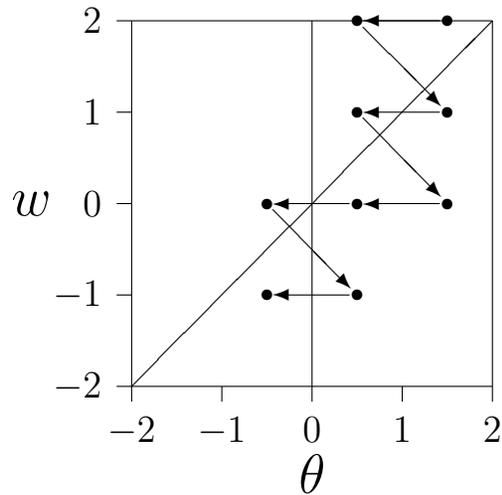
Summe der Änderungen

Beginne an zufälligem Punkt.

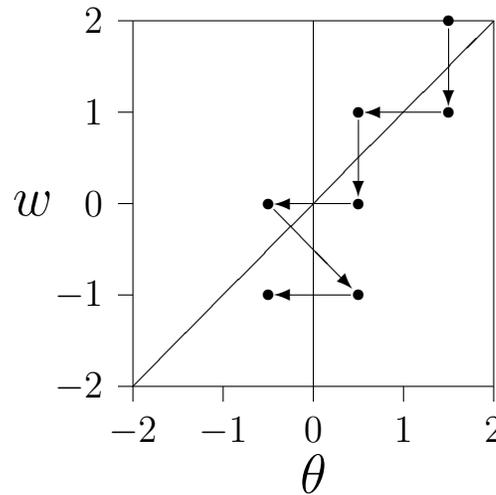
Passe Parameter iterativ an,
entsprechend der zugehörigen Richtung am aktuellen Punkt.

Trainieren von Schwellenwertelementen

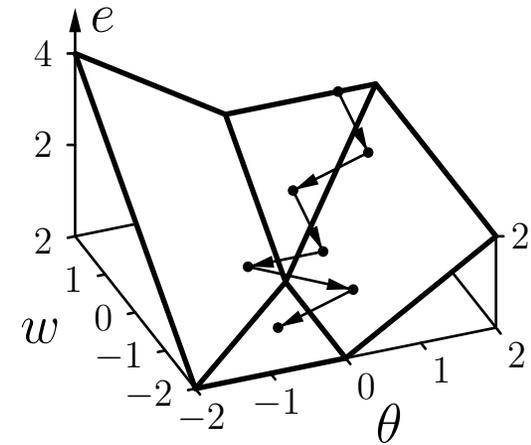
Beispieltrainingsprozedur: Online- und Batch-Training.



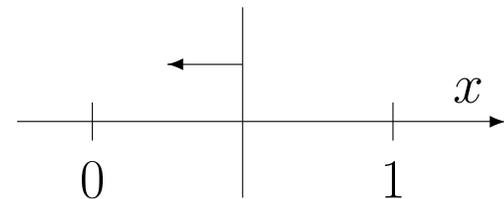
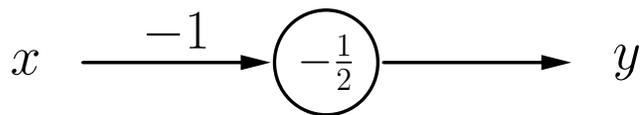
Online-Lernen



Batch-Lernen



Batch-Lernen



Trainieren von Schwellenwertelementen: Delta-Regel

Formale Trainingsregel: Sei $\vec{x} = (x_1, \dots, x_n)$ ein Eingabevektor eines Schwellenwertelements, o die gewünschte Ausgabe für diesen Eingabevektor, und y die momentane Ausgabe des Schwellenwertelements. Wenn $y \neq o$, dann werden Schwellenwert θ und Gewichtsvektor $\vec{w} = (w_1, \dots, w_n)$ wie folgt angepasst, um den Fehler zu reduzieren:

$$\begin{aligned} \theta^{(\text{neu})} &= \theta^{(\text{alt})} + \Delta\theta & \text{wobei } \Delta\theta &= -\eta(o - y), \\ \forall i \in \{1, \dots, n\} : w_i^{(\text{neu})} &= w_i^{(\text{alt})} + \Delta w_i & \text{wobei } \Delta w_i &= \eta(o - y)x_i, \end{aligned}$$

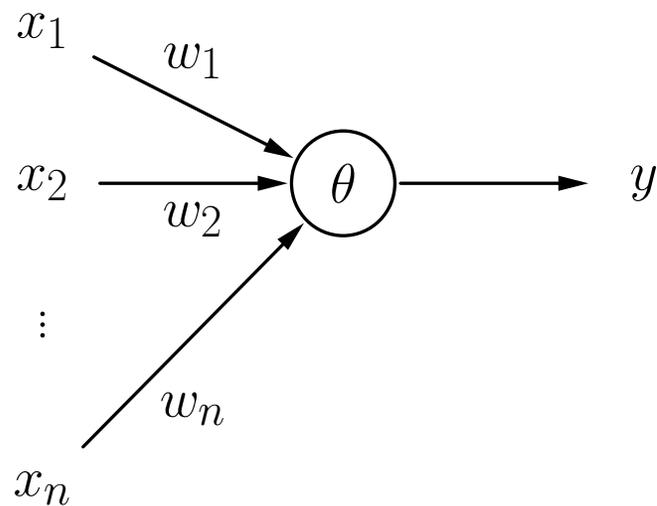
wobei η ein Parameter ist, der **Lernrate** genannt wird. Er bestimmt die Größenordnung der Gewichtsänderungen. Diese Vorgehensweise nennt sich **Delta-Regel** oder **Widrow–Hoff–Procedure** [Widrow and Hoff 1960].

Online-Training: Passe Parameter nach jedem Trainingsmuster an.

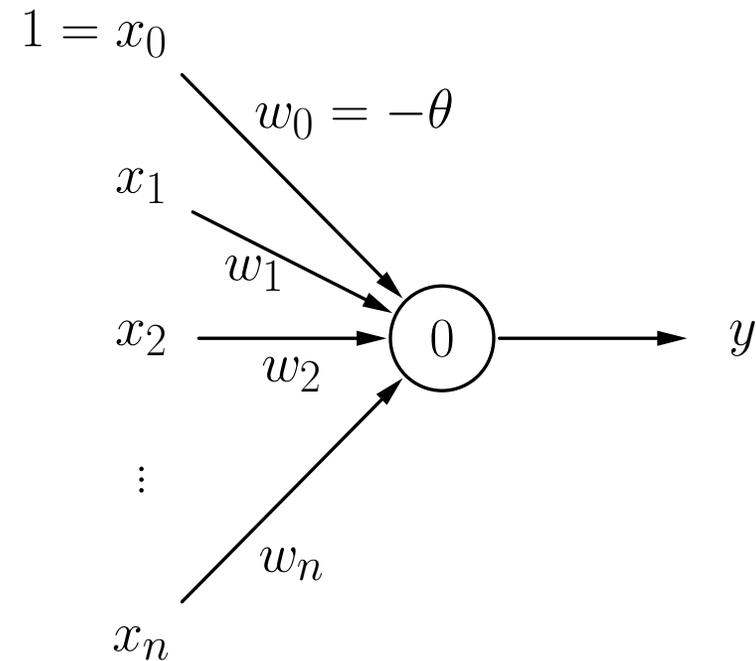
Batch-Training: Passe Parameter am Ende jeder **Epoche** an, d.h. nach dem Durchlaufen aller Trainingsbeispiele.

Trainieren von Schwellenwertelementen: Delta-Regel

Ändern des Schwellenwerts in ein Gewicht:



$$\sum_{i=1}^n w_i x_i \geq \theta$$



$$\sum_{i=1}^n w_i x_i - \theta \geq 0$$

Trainieren von Schwellenwertelementen: Delta-Regel

```
procedure online_training (var  $\vec{w}$ , var  $\theta$ ,  $L$ ,  $\eta$ );  
var  $y$ ,  $e$ ; (* Ausgabe, Fehlersumme *)  
begin  
  repeat  
     $e := 0$ ; (* initialisiere Fehlersumme *)  
    for all  $(\vec{x}, o) \in L$  do begin (* durchlaufe Trainingsmuster*)  
      if  $(\vec{w}\vec{x} \geq \theta)$  then  $y := 1$ ; (* berechne Ausgabe*)  
      else  $y := 0$ ; (* des Schwellenwertelements *)  
      if  $(y \neq o)$  then begin (* Falls Ausgabe falsch *)  
         $\theta := \theta - \eta(o - y)$ ; (* passe Schwellenwert *)  
         $\vec{w} := \vec{w} + \eta(o - y)\vec{x}$ ; (* und Gewichte an *)  
         $e := e + |o - y|$ ; (* summiere die Fehler*)  
      end;  
    end;  
  until  $(e \leq 0)$ ; (* wiederhole die Berechnungen*)  
end; (* bis der Fehler verschwindet*)
```

Trainieren von Schwellenwertelementen: Delta-Regel

```
procedure batch_training (var  $\vec{w}$ , var  $\theta$ ,  $L$ ,  $\eta$ );  
var  $y$ ,  $e$ , (* Ausgabe, Fehlersumme *)  
     $\theta_c$ ,  $\vec{w}_c$ ; (* summierte Änderungen *)  
begin  
  repeat  
     $e := 0$ ;  $\theta_c := 0$ ;  $\vec{w}_c := \vec{0}$ ; (* Initialisierungen *)  
    for all  $(\vec{x}, o) \in L$  do begin (* durchlaufe Trainingsbeispiele*)  
      if  $(\vec{w}\vec{x} \geq \theta)$  then  $y := 1$ ; (* berechne Ausgabe *)  
        else  $y := 0$ ; (* des Schwellenwertelements *)  
      if  $(y \neq o)$  then begin (* Falls Ausgabe falsch*)  
         $\theta_c := \theta_c - \eta(o - y)$ ; (* summiere die Änderungen von*)  
         $\vec{w}_c := \vec{w}_c + \eta(o - y)\vec{x}$ ; (* Schwellenwert und Gewichten *)  
         $e := e + |o - y|$ ; (* summiere Fehler*)  
      end;  
    end;  
     $\theta := \theta + \theta_c$ ; (* passe Schwellenwert*)  
     $\vec{w} := \vec{w} + \vec{w}_c$ ; (* und Gewichte an *)  
  until  $(e \leq 0)$ ; (* wiederhole Berechnungen *)  
end; (* bis der Fehler verschwindet*)
```

Trainieren von Schwellenwertelementen: Online

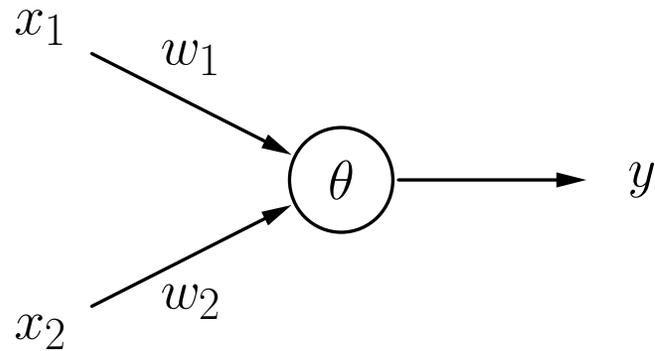
Epoche	x	o	$\vec{x}\vec{w}$	y	e	$\Delta\theta$	Δw	θ	w
								1.5	2
1	0	1	-1.5	0	1	-1	0	0.5	2
	1	0	1.5	1	-1	1	-1	1.5	1
2	0	1	-1.5	0	1	-1	0	0.5	1
	1	0	0.5	1	-1	1	-1	1.5	0
3	0	1	-1.5	0	1	-1	0	0.5	0
	1	0	0.5	0	0	0	0	0.5	0
4	0	1	-0.5	0	1	-1	0	-0.5	0
	1	0	0.5	1	-1	1	-1	0.5	-1
5	0	1	-0.5	0	1	-1	0	-0.5	-1
	1	0	-0.5	0	0	0	0	-0.5	-1
6	0	1	0.5	1	0	0	0	-0.5	-1
	1	0	-0.5	0	0	0	0	-0.5	-1

Trainieren von Schwellenwertelementen: Batch

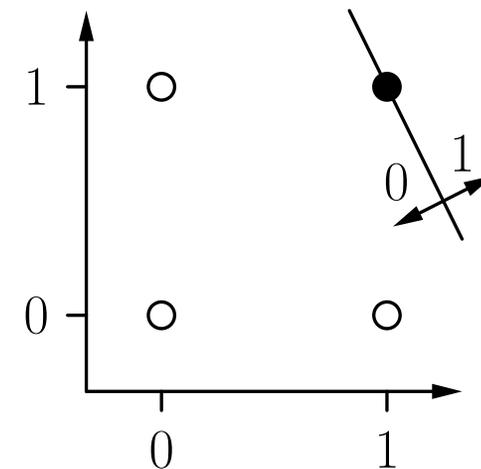
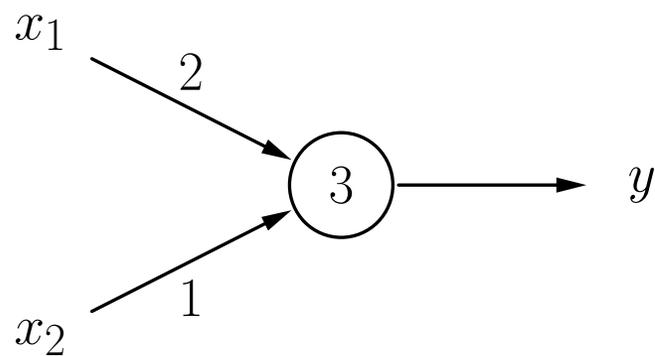
Epoche	x	o	$\vec{x}\vec{w}$	y	e	$\Delta\theta$	Δw	θ	w
								1.5	2
1	0	1	-1.5	0	1	-1	0		
	1	0	0.5	1	-1	1	-1	1.5	1
2	0	1	-1.5	0	1	-1	0		
	1	0	-0.5	0	0	0	0	0.5	1
3	0	1	-0.5	0	1	-1	0		
	1	0	0.5	1	-1	1	-1	0.5	0
4	0	1	-0.5	0	1	-1	0		
	1	0	-0.5	0	0	0	0	-0.5	0
5	0	1	0.5	1	0	0	0		
	1	0	0.5	1	-1	1	-1	0.5	-1
6	0	1	-0.5	0	1	-1	0		
	1	0	-1.5	0	0	0	0	-0.5	-1
7	0	1	0.5	1	0	0	0		
	1	0	-0.5	0	0	0	0	-0.5	-1

Trainieren von Schwellenwertelementen: Konjunktion

Schwellenwertelement mit zwei Eingängen für die Konjunktion.



x_1	x_2	y
0	0	0
1	0	0
0	1	0
1	1	1



Trainieren von Schwellenwertelementen: Konjunktion

Epoche	x_1	x_2	o	$\vec{x}\vec{w}$	y	e	$\Delta\theta$	Δw_1	Δw_2	θ	w_1	w_2
										0	0	0
1	0	0	0	0	1	-1	1	0	0	1	0	0
	0	1	0	-1	0	0	0	0	0	1	0	0
	1	0	0	-1	0	0	0	0	0	1	0	0
	1	1	1	-1	0	1	-1	1	1	0	1	1
2	0	0	0	0	1	-1	1	0	0	1	1	1
	0	1	0	0	1	-1	1	0	-1	2	1	0
	1	0	0	-1	0	0	0	0	0	2	1	0
	1	1	1	-1	0	1	-1	1	1	1	2	1
3	0	0	0	-1	0	0	0	0	0	1	2	1
	0	1	0	0	1	-1	1	0	-1	2	2	0
	1	0	0	0	1	-1	1	-1	0	3	1	0
	1	1	1	-2	0	1	-1	1	1	2	2	1
4	0	0	0	-2	0	0	0	0	0	2	2	1
	0	1	0	-1	0	0	0	0	0	2	2	1
	1	0	0	0	1	-1	1	-1	0	3	1	1
	1	1	1	-1	0	1	-1	1	1	2	2	2
5	0	0	0	-2	0	0	0	0	0	2	2	2
	0	1	0	0	1	-1	1	0	-1	3	2	1
	1	0	0	-1	0	0	0	0	0	3	2	1
	1	1	1	0	1	0	0	0	0	3	2	1
6	0	0	0	-3	0	0	0	0	0	3	2	1
	0	1	0	-2	0	0	0	0	0	3	2	1
	1	0	0	-1	0	0	0	0	0	3	2	1
	1	1	1	0	1	0	0	0	0	3	2	1

Trainieren von Schwellenwertelementen: Biimplikation

Epoch	x_1	x_2	o	$\vec{x}\vec{w}$	y	e	$\Delta\theta$	Δw_1	Δw_2	θ	w_1	w_2
										0	0	0
1	0	0	1	0	1	0	0	0	0	0	0	0
	0	1	0	0	1	-1	1	0	-1	1	0	-1
	1	0	0	-1	0	0	0	0	0	1	0	-1
	1	1	1	-2	0	1	-1	1	1	0	1	0
2	0	0	1	0	1	0	0	0	0	0	1	0
	0	1	0	0	1	-1	1	0	-1	1	1	-1
	1	0	0	0	1	-1	1	-1	0	2	0	-1
	1	1	1	-3	0	1	-1	1	1	1	1	0
3	0	0	1	0	1	0	0	0	0	0	1	0
	0	1	0	0	1	-1	1	0	-1	1	1	-1
	1	0	0	0	1	-1	1	-1	0	2	0	-1
	1	1	1	-3	0	1	-1	1	1	1	1	0

Trainieren von Schwellenwertelementen: Konvergenz

Konvergenztheorem: Sei $L = \{(\vec{x}_1, o_1), \dots, (\vec{x}_m, o_m)\}$ eine Menge von Trainingsmustern, jedes bestehend aus einem Eingabevektor $\vec{x}_i \in \mathbb{R}^n$ und einer gewünschten Ausgabe $o_i \in \{0, 1\}$. Sei weiterhin $L_0 = \{(\vec{x}, o) \in L \mid o = 0\}$ und $L_1 = \{(\vec{x}, o) \in L \mid o = 1\}$. Falls L_0 und L_1 linear separabel sind, d.h., falls $\vec{w} \in \mathbb{R}^n$ und $\theta \in \mathbb{R}$ existieren, so dass

$$\begin{aligned} \forall (\vec{x}, 0) \in L_0 : \quad & \vec{w}\vec{x} < \theta \quad \text{und} \\ \forall (\vec{x}, 1) \in L_1 : \quad & \vec{w}\vec{x} \geq \theta, \end{aligned}$$

dann terminieren sowohl Online- als auch Batch-Training.

Für nicht linear separable Probleme terminiert der Algorithmus nicht.

Trainieren von Netzwerken aus Schwellenwertelementen

Einzelne Schwellenwertelemente haben starke Einschränkungen:
Sie können nur linear separable Funktionen berechnen.

Netzwerke aus Schwellenwertelementen können beliebige Boolesche Funktionen berechnen.

Das Trainieren einzelner Schwellenwertelemente mit der Delta-Regel ist schnell und findet garantiert eine Lösung, falls eine existiert.

Netzwerke aus Schwellenwertelementen können nicht mit der Delta-Regel trainiert werden.