Neuronale Netze

V 03/04

I – NNG.d

WS 03 / 04

Dr. Lorenz Müller

BFH, Hochschule für Technik und Informatik

Neuronale Netze

Inhalte

- **Einführung**
  Beispiel, biologische Basis, Modell**,** Systemaufbau, Neuron
- **Grundlagen und Konzepte**
  Topologien, Gewichtsmatrix, Lernkonzepte
- **Wichtigste Netze**
  Matrixspeicher, Gradientenabstieg, Backpropagation, Kohonen Netze
- **Semesterarbeit**

Neuronale Netze

Themen: Einführung

- Simples Beispiel
- Biologisches Vorbild
- Systemaufbau
- Neuronen Modell
- Informationsspeicherung
- XOR - Problem
- Anwendungspotential

Lernziele *Einführung*

Kennen der

- Grundlagen und Grundbegriffe
- Bedeutung von Eingabe, Ausgabe und Gewichten
- Neuronmodell
- einfachen Lernregeln (verstehen was *Lernen aus Beispielen* heisst)
- der geeigneten Anwendungsbereiche

Ein simples Beispiel

„ Der Wetterinterpreter“

aus Neuronale Netzwerke
M&T; E. Schönberger et al.

Eingabe:

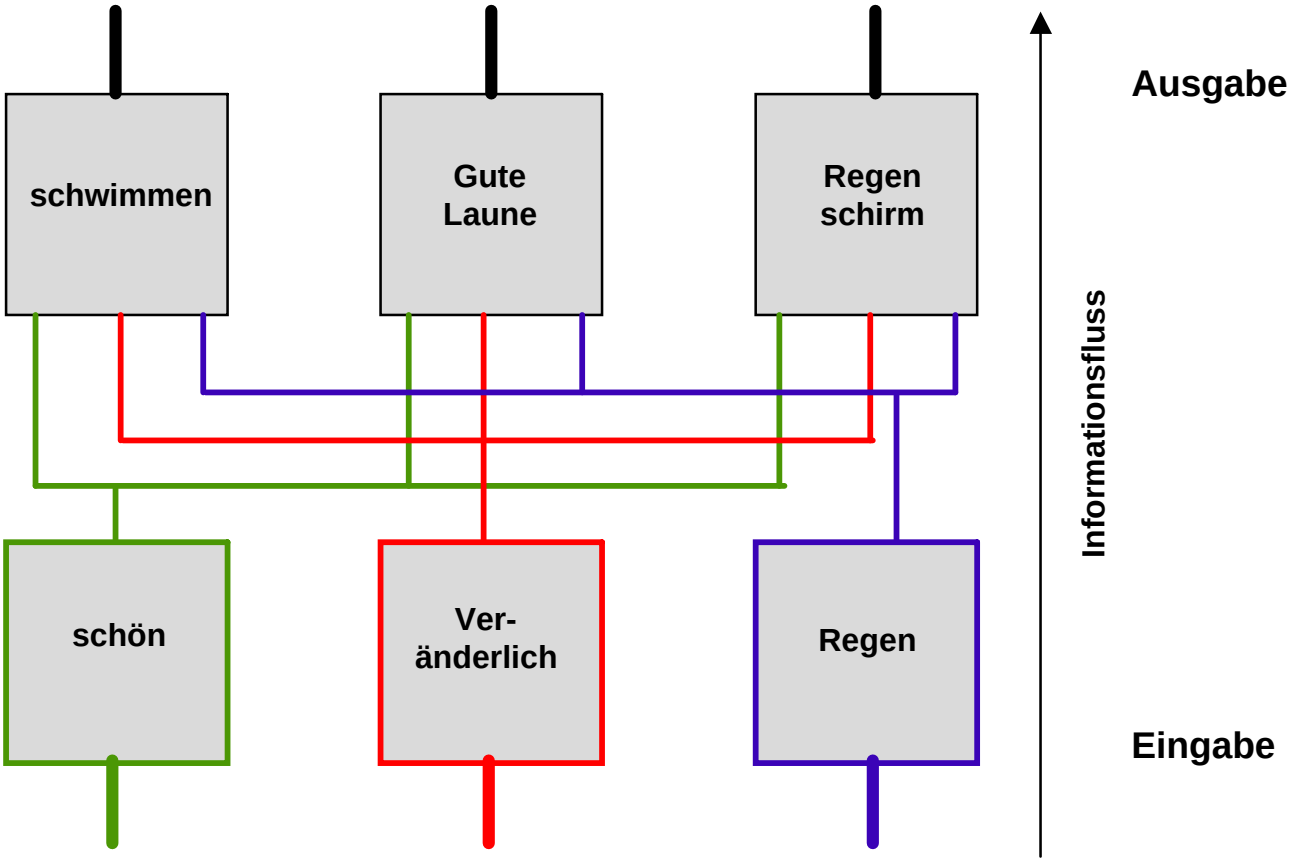
Wetterinformation
schön
veränderlich
Regen

Ausgabe:

Handlung und Laune
schwimmen gehen
gute Laune
Regenschirm nehmen

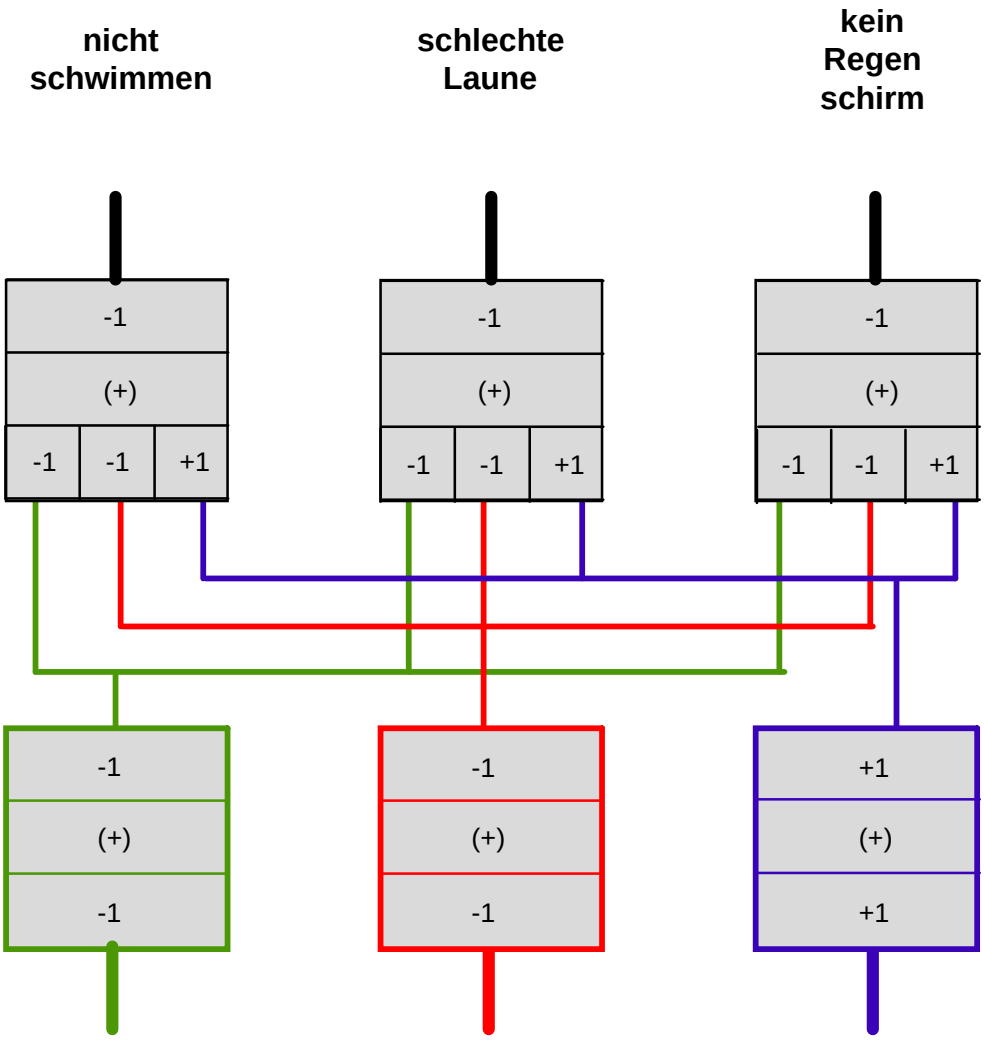
Gesucht ist ein lernfähiges Entscheidungssystem

Verarbeitungsmodell

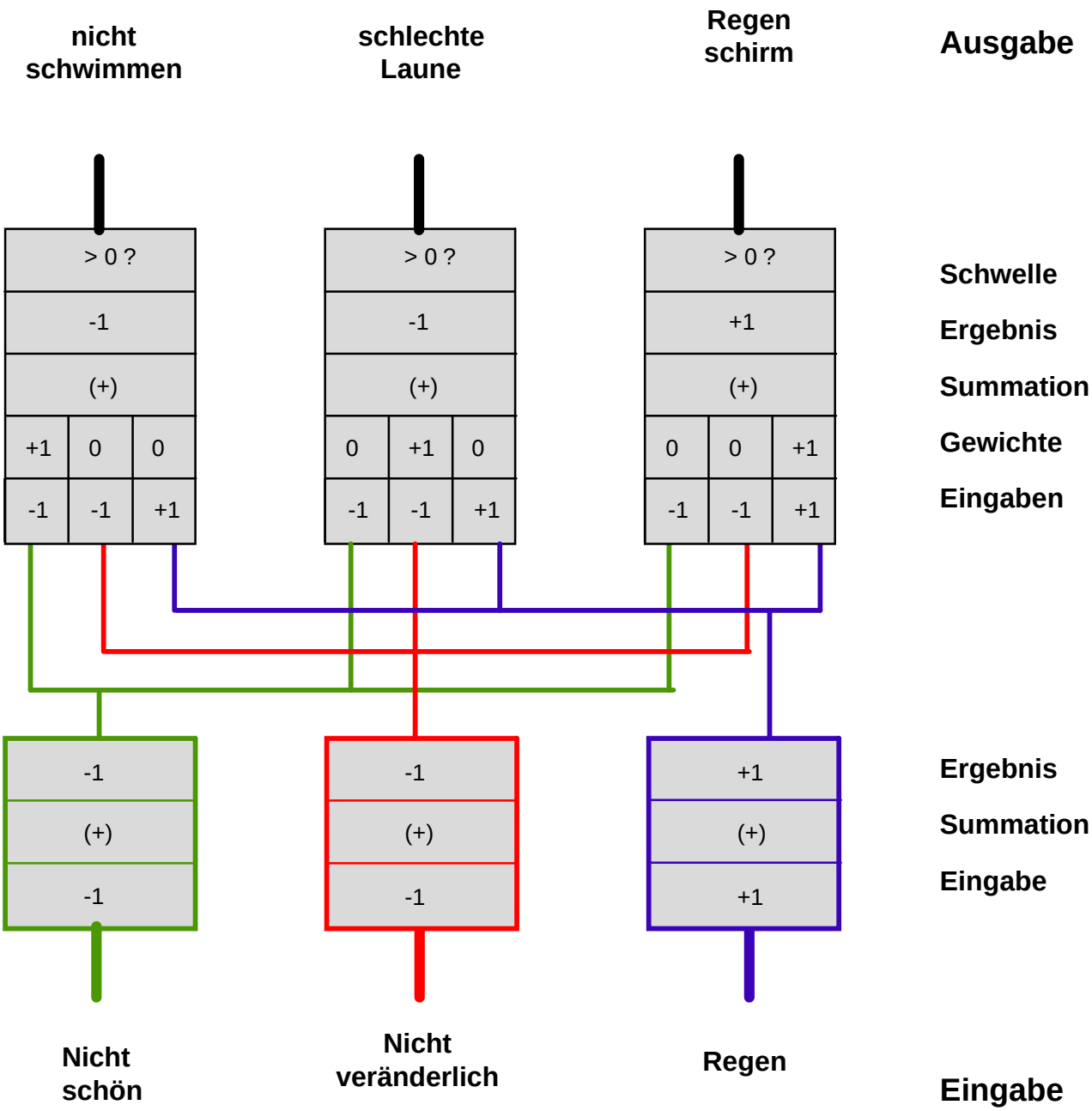


Codierung: **+1** **aktiv, Ereignis tritt ein**
-1 **inaktiv, Gegenteil tritt ein**

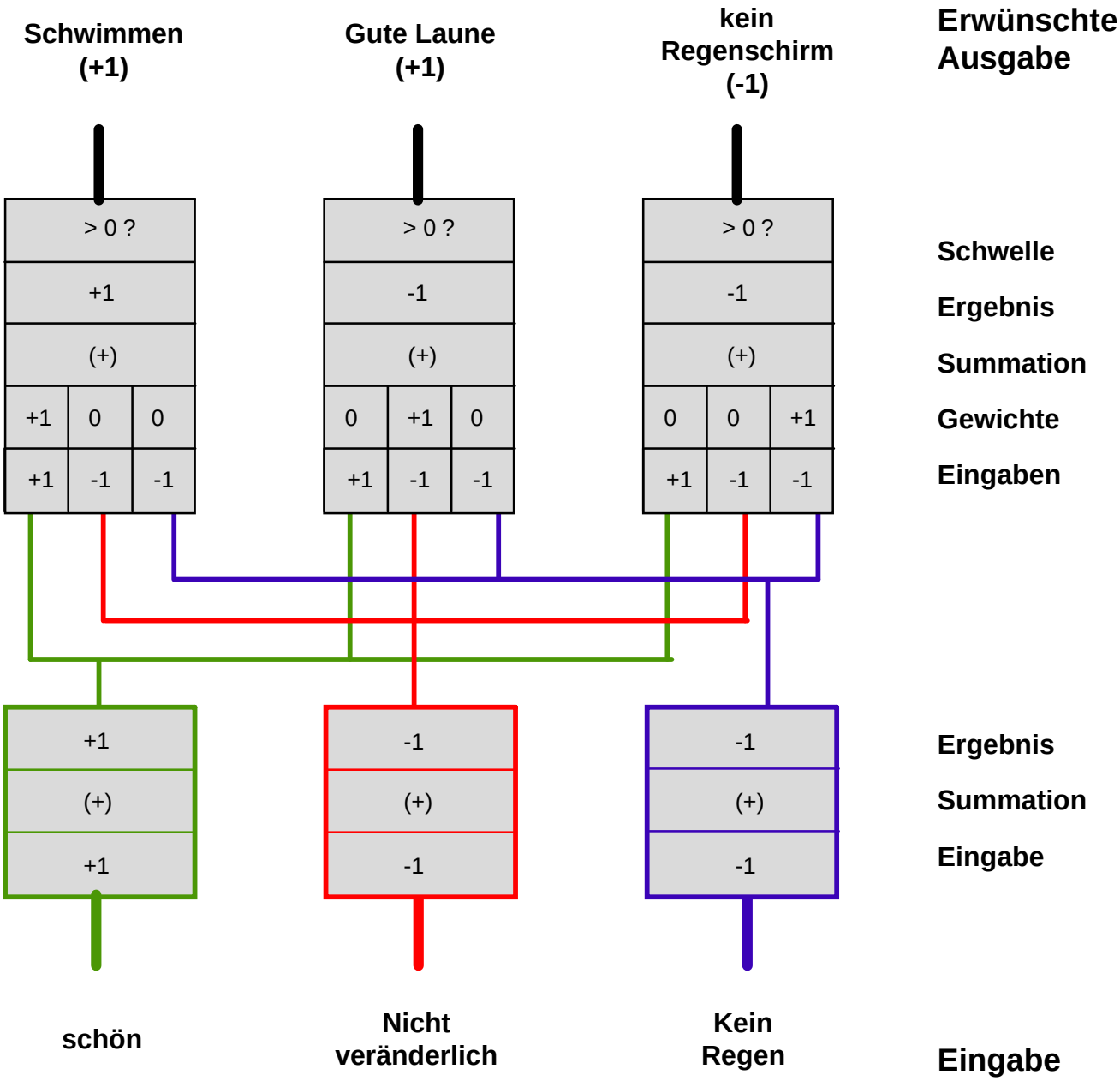
Verarbeitung eines Musters



Gewichtung der Verbindungen

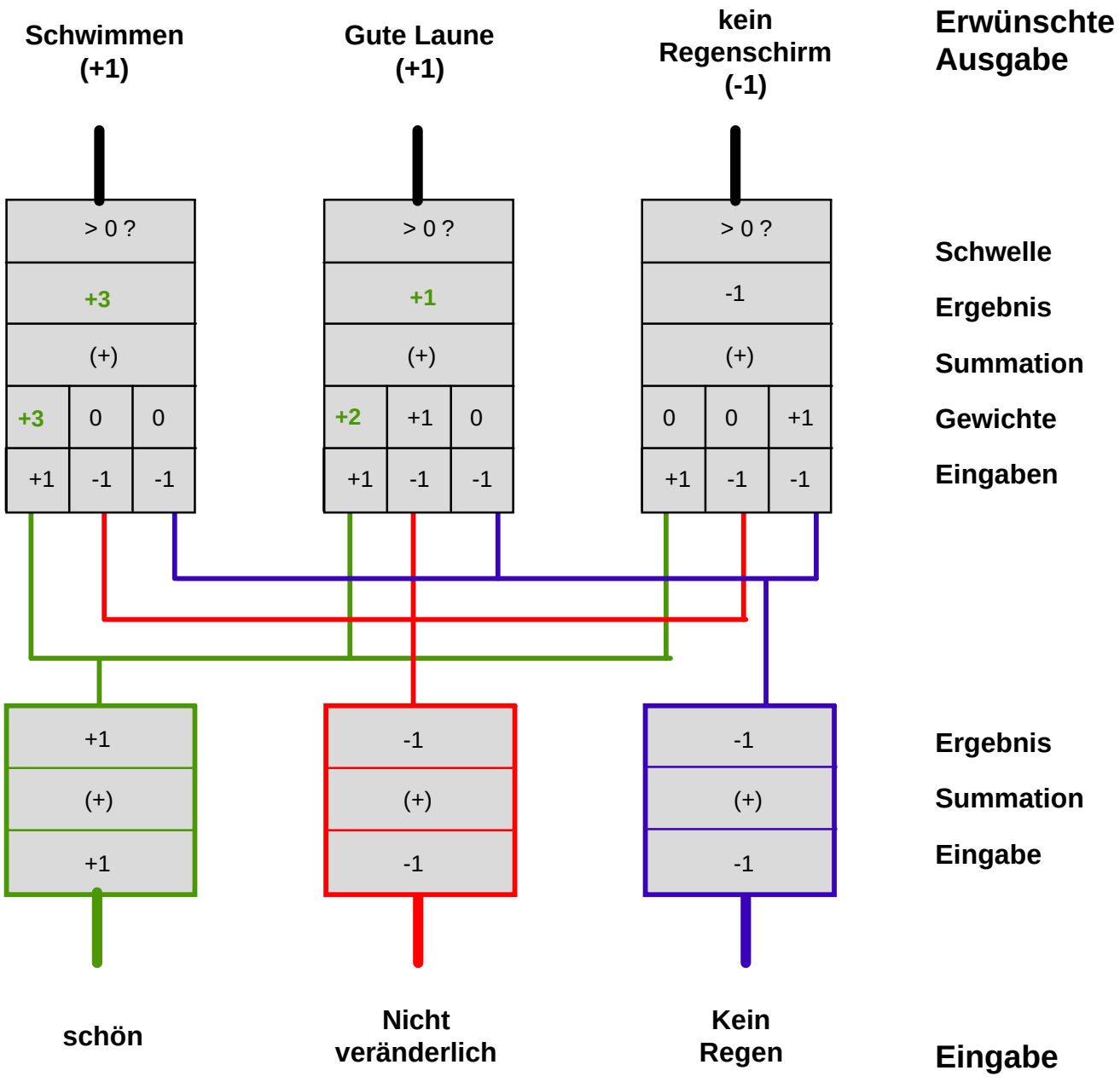


Anpassen der Gewichte
Lernregel



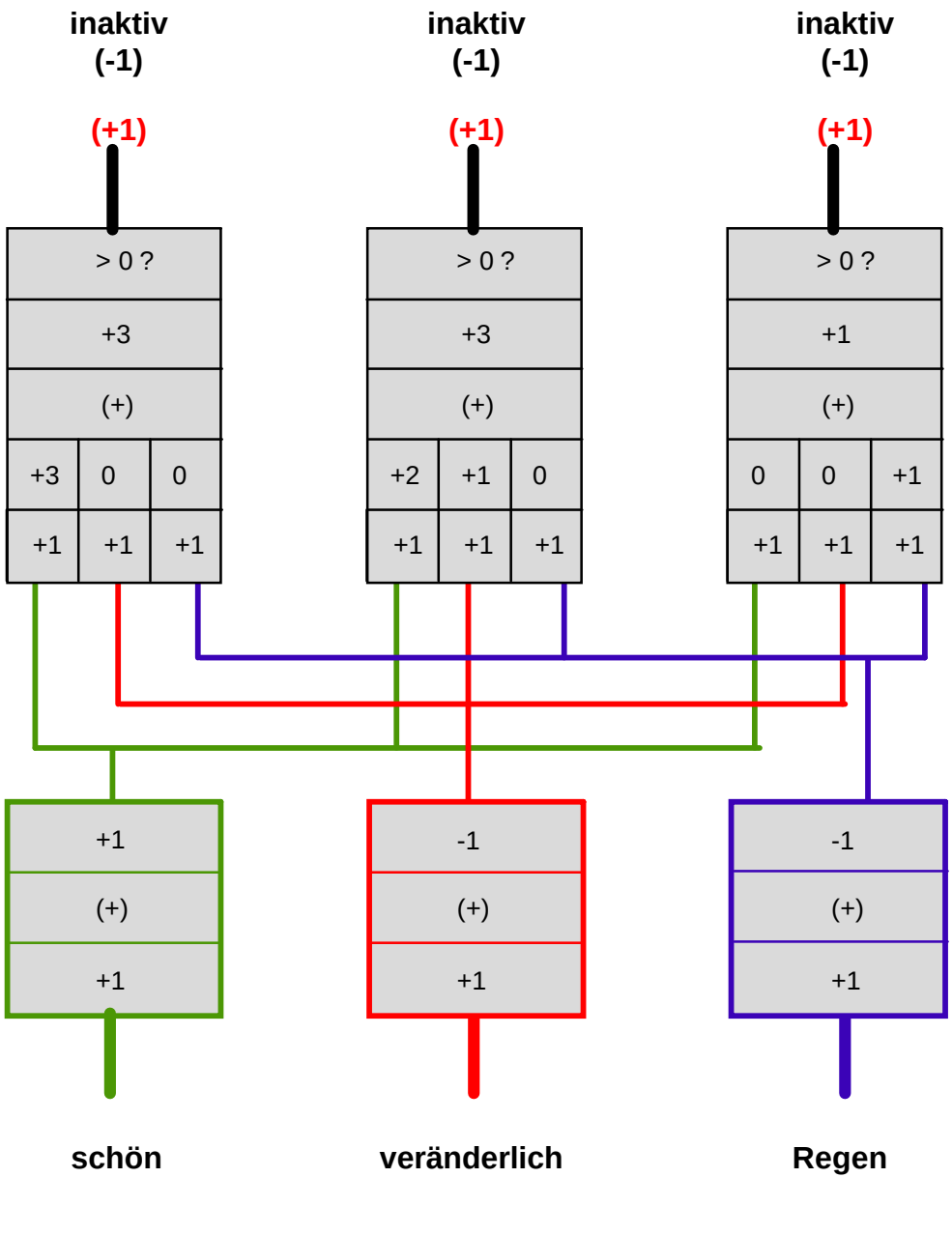
Hebbsche Lernregel:
„Verstärkung der Verbindung von zwei gleichzeitig aktiven Neuronen“

Anwendung Hebbsche Lernregel



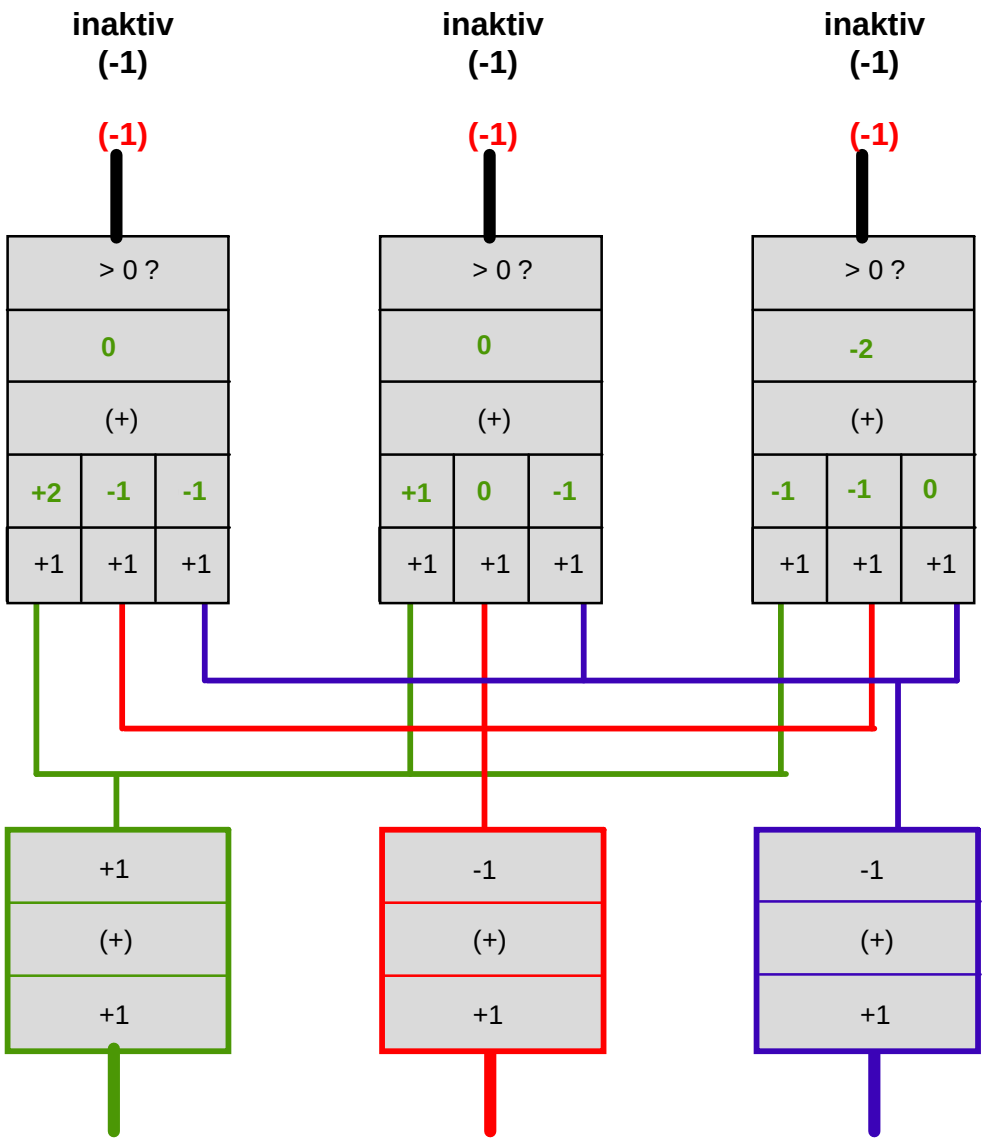
Hebbsche Lernregel:
Erhöhe Gewicht zwischen Neuron i und j, wenn Ein- und Ausgabe aktiv sind bzw. sein sollten. Verstärkungsfaktor ist Lernrate.

Grenzen des Hebbschen Lernens



Hebbsche Lernregel:
Erhöhe Gewicht zwischen Neuron i und j, wenn Ein- und Ausgabe aktiv sind bzw. sein sollten. Verstärkungsfaktor ist Lernrate.

Fehlerkorrigierende Lernregel



Erwünschte Ausgabe

(tatsächlich)

Schwelle

Ergebnis

Summation

Gewichte

Eingaben

Ergebnis

Summation

Eingabe

Nonsense Eingabe

Fehlerkorrigierende Lernregel: Delta Lernen

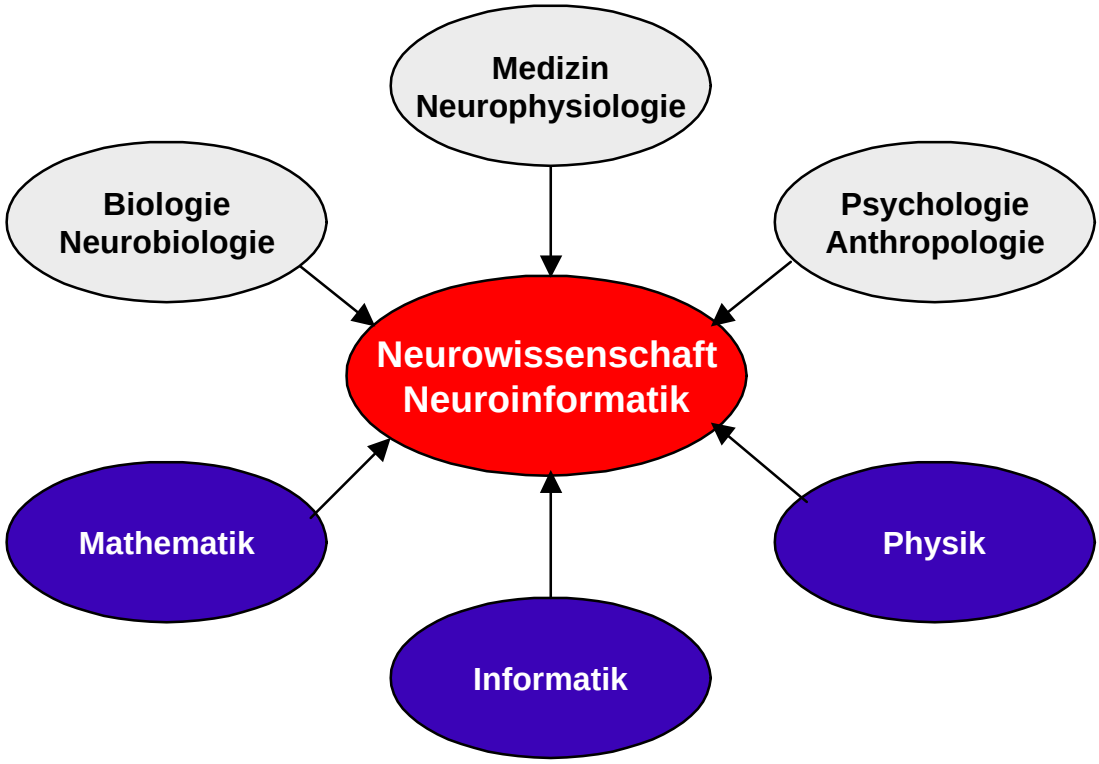
Fehler	:=	(gew. Zustand z) – (tats. Zustand y)
Gewichtskorrektur	:=	(altes Gewicht) + Lernrate q * Fehler
w_{ij}(neu)	=	w_{ij} (alt) + q *(z_i – y_i)

Was lernt uns das Beispiel

- Ein- und Ausgaben müssen codiert werden (numerisches Modell)
- Neuronen sind simple Prozessoren (Summation, Gewichtung der Eingänge, Schwelle)
- „Wissen“ ist in den Gewichten (Verbindungen, präsentierte Beispiele)
- Einfache Regeln führen zu Anpassung an Beispiele
- Lernfähigkeit hängt von Lernregel und Lernrate ab
- Ein neuronales Netz kann mehrere Beispiele lernen, ohne das früher gelernte zu vergessen

Neuroinformatik

Neuroinformatik ist ein interdisziplinäres Wissensgebiet



Zwei Ausrichtungen

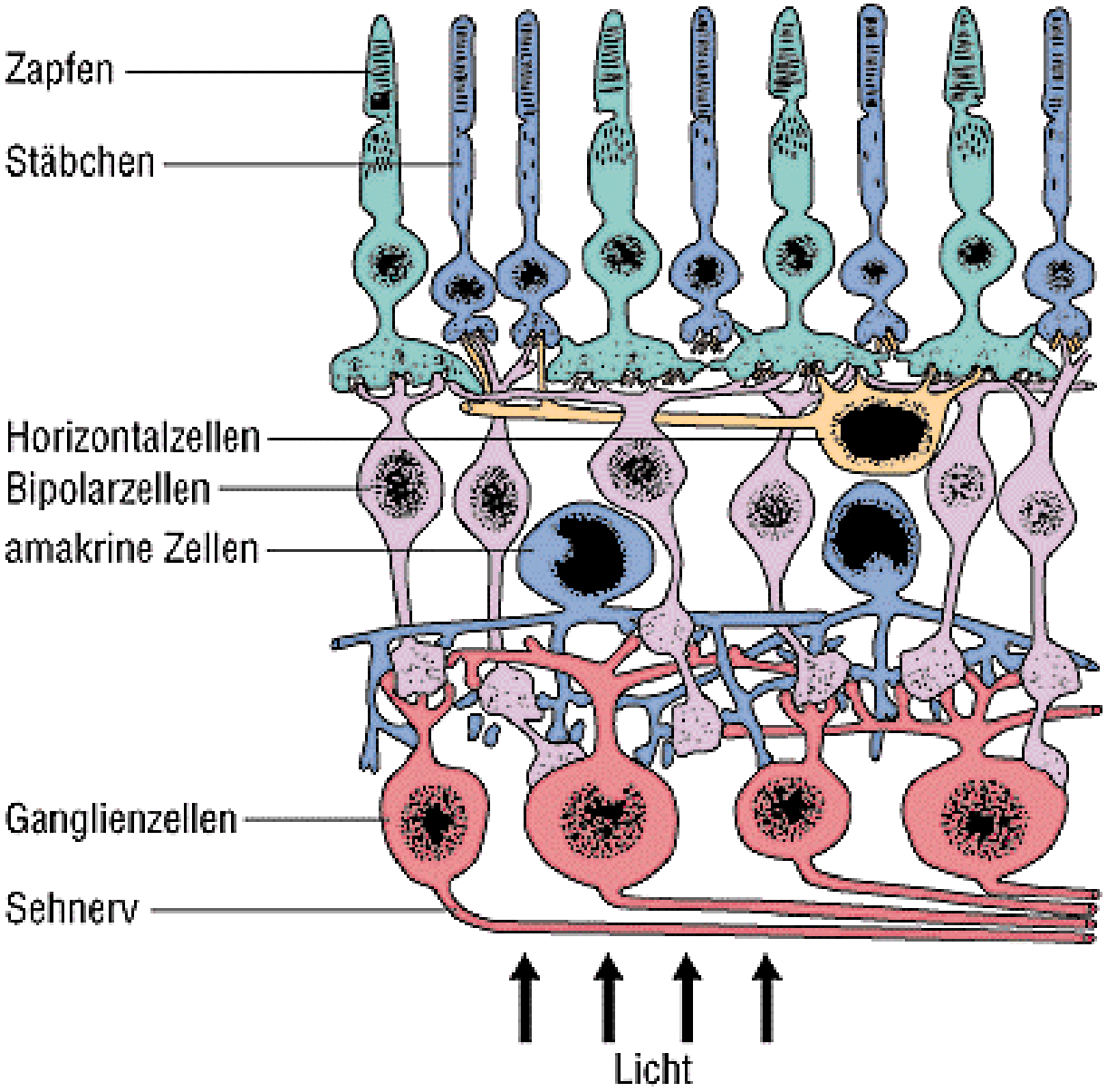
- Neurowissenschaft ==> Biologie, Physiologie
- Neuroinformatik ==> Informatik, KI

Was sind neuronale Netze

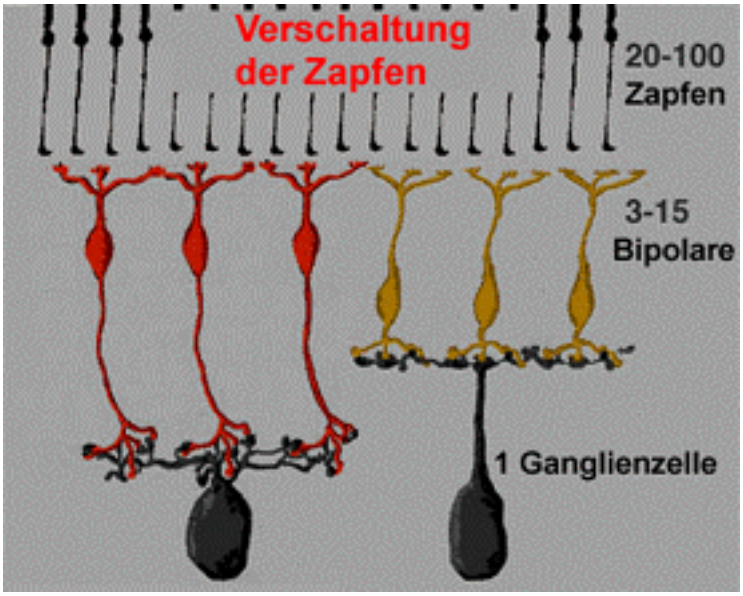
Biologisch inspirierte Informations-verarbeitungssysteme.

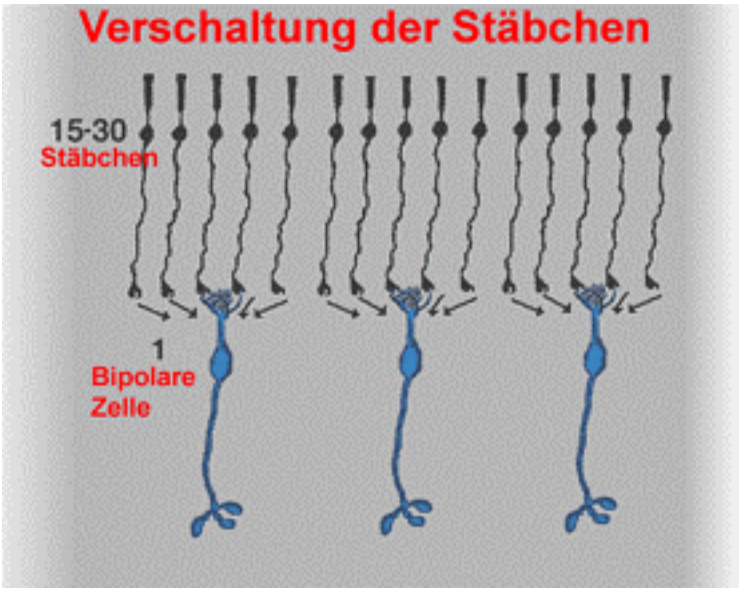
Zahlreiche miteinander verbundene Prozesseinheiten

Beispiel: Netzhaut

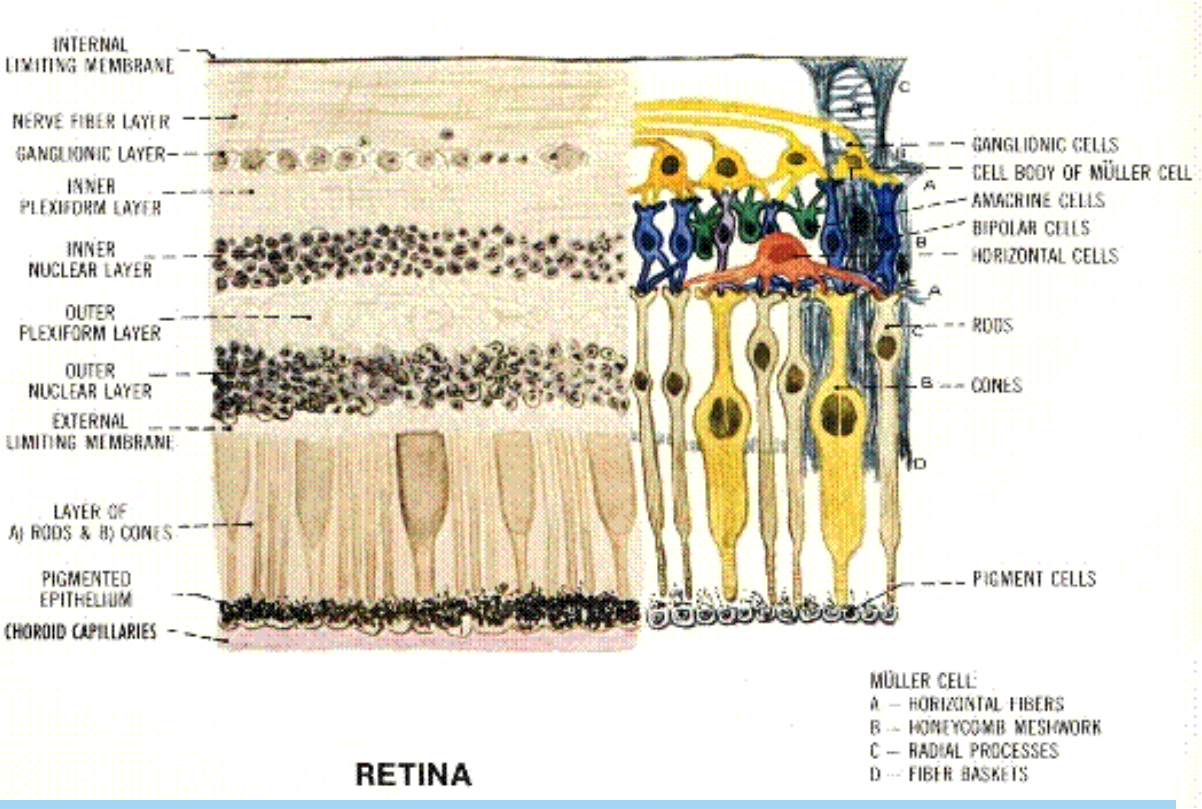


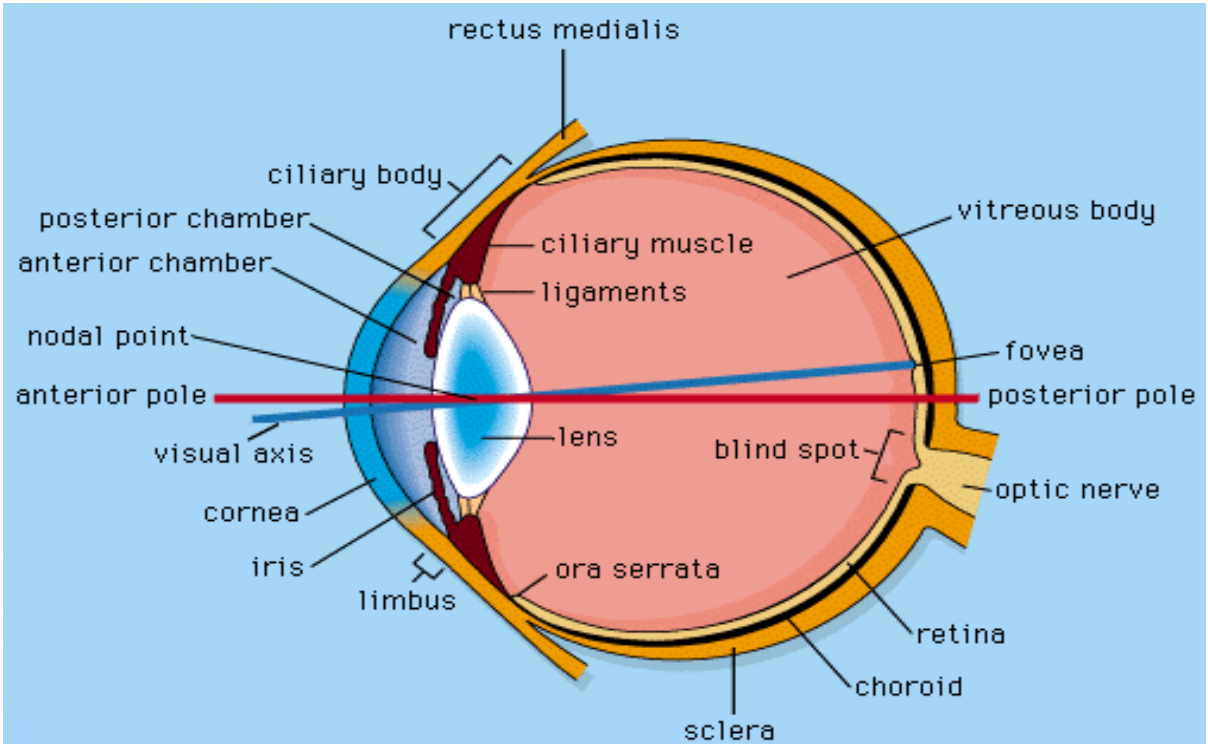
Lokale Vernetzung in der Retina



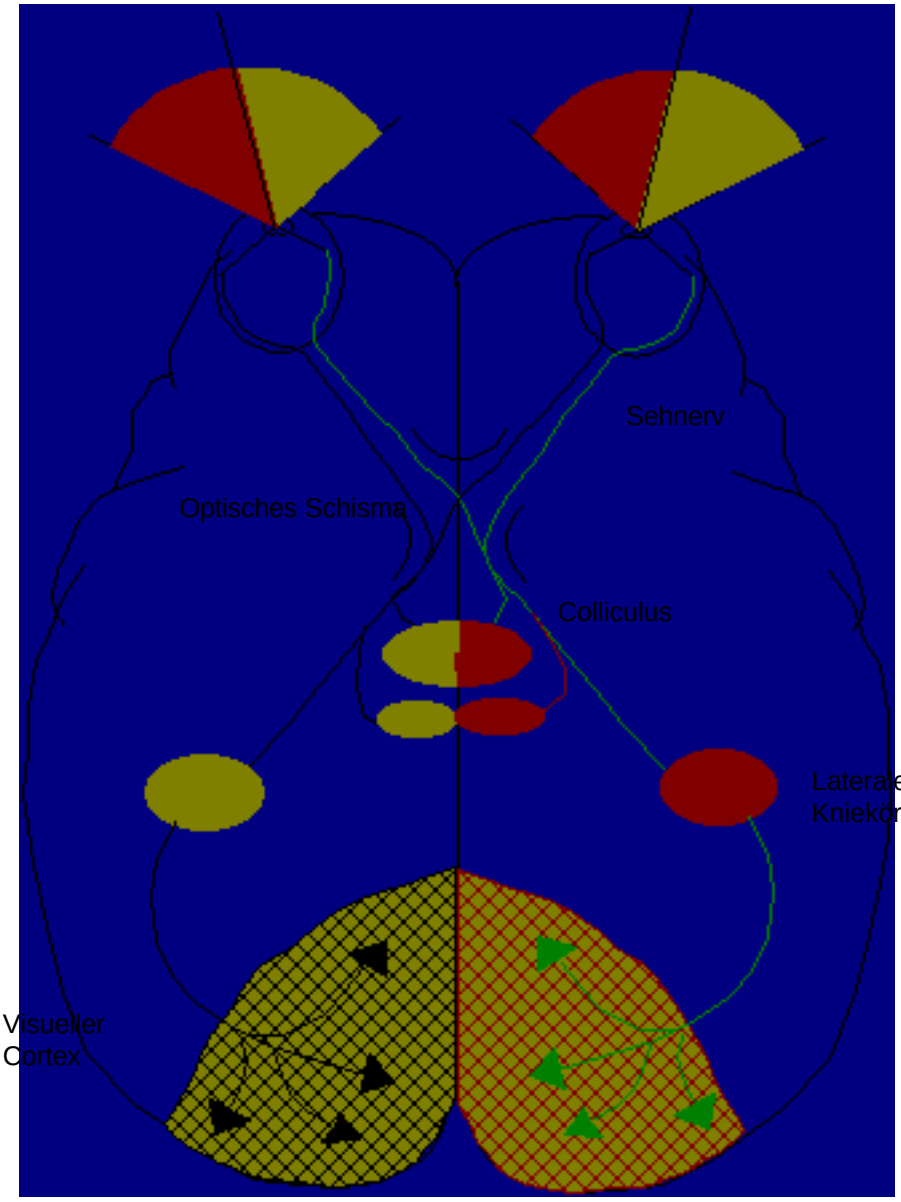


Sensorsystem Auge

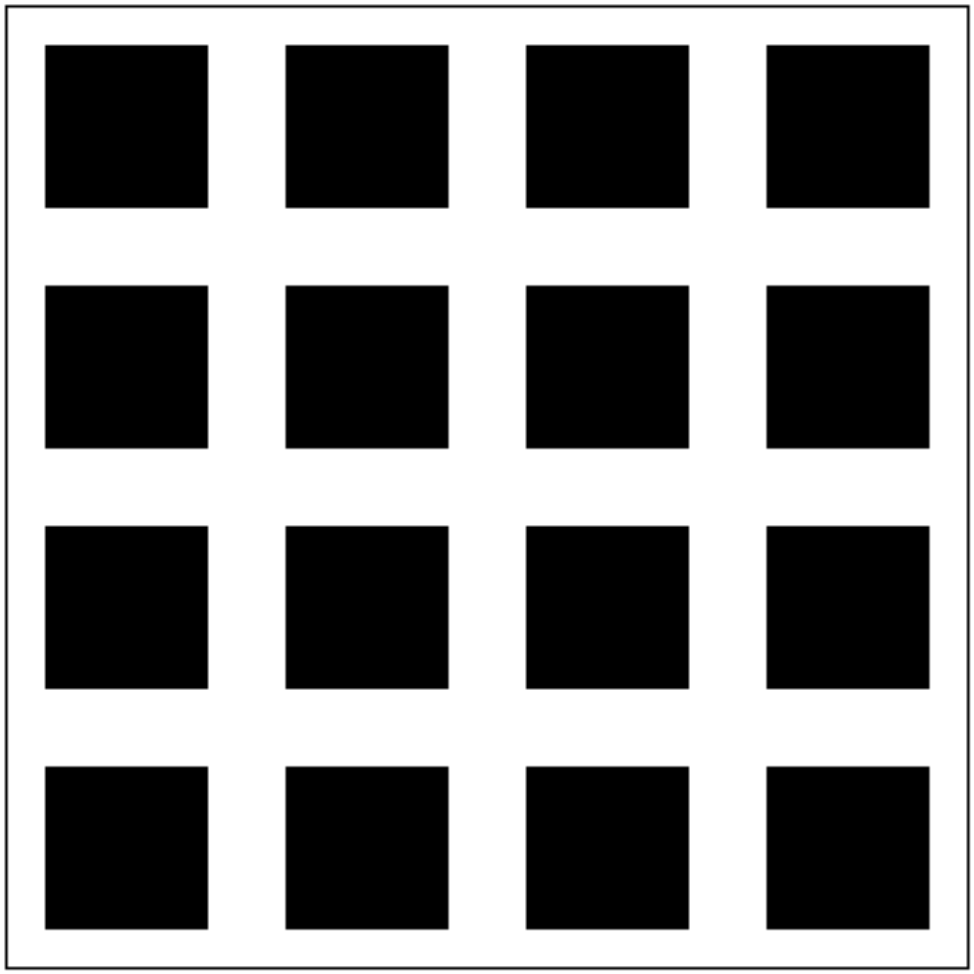




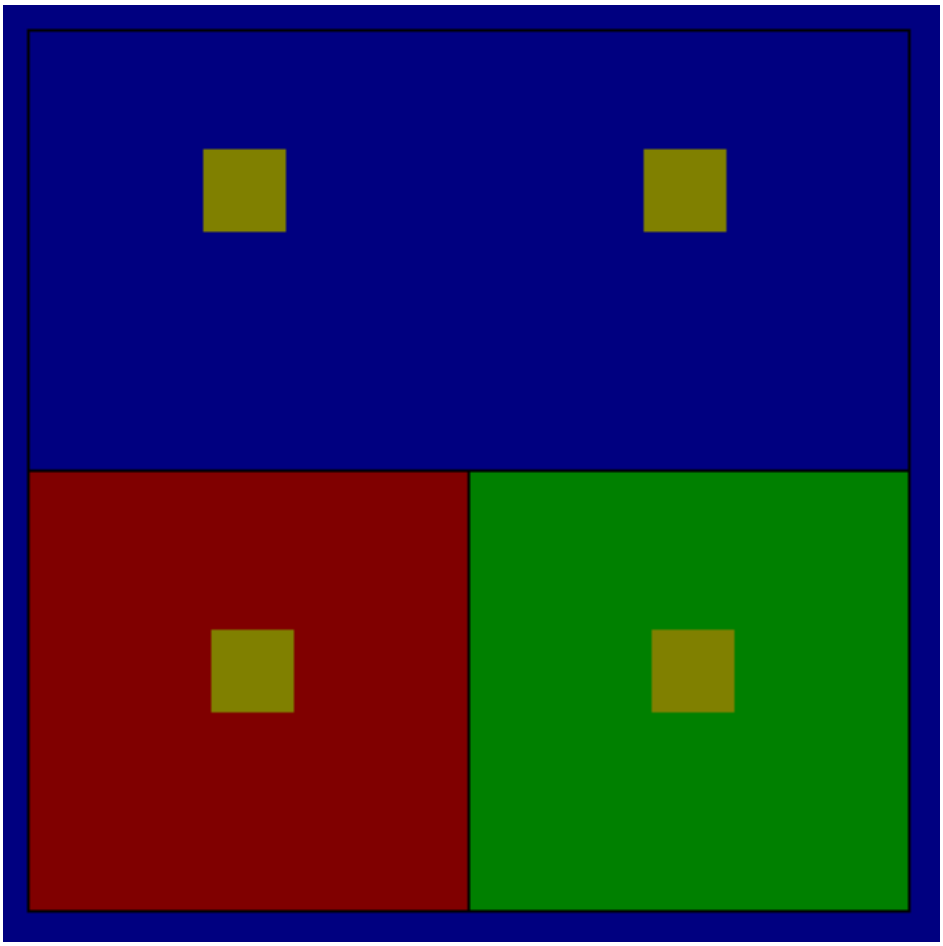
Visueller Pfad

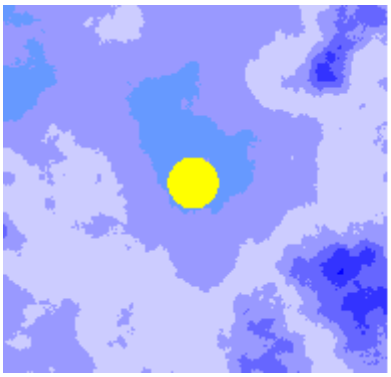


Optische Täuschung



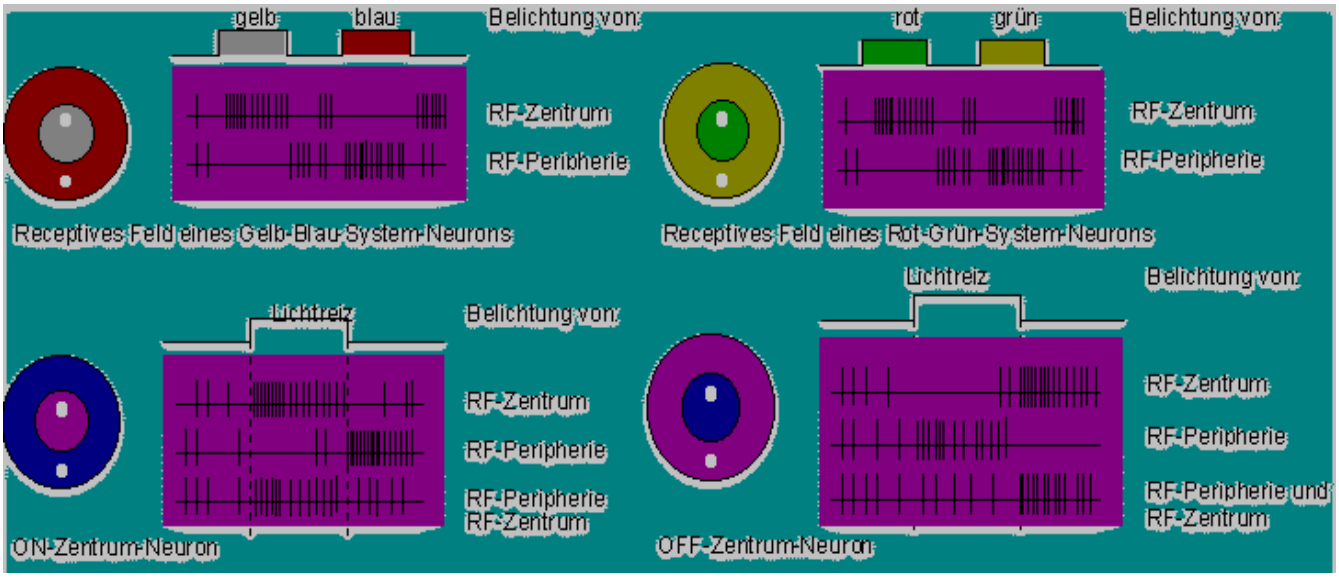
Farbensehen, Seitenschärfe





1 2 3 4 5 6

Vorverarbeitung in Retina



Die optische Hemisphäre im Auge ist in runde sich naheüberlagernde rezeptive Felder aufgeteilt.

Jedes Feld besteht aus einem zentralen und einem peripheren Bereich.
Es gibt Felder die werden aktiviert, wenn der zentrale Bereich beleuchtet wird und der periphere überwiegend im Dunkeln liegt (On-Neuron) und solche die gerade durch die umgekehrte Beleuchtung aktiviert werden (Off-Neuron).

Analog funktionierende Felder gibt es in der Makula für das Farbensehen.

Charakteristik

Struktur:

- Einfache Prozesseinheiten
- hochgradige Vernetzung
- funktionale Strukturen (Schichten,Ganglien)

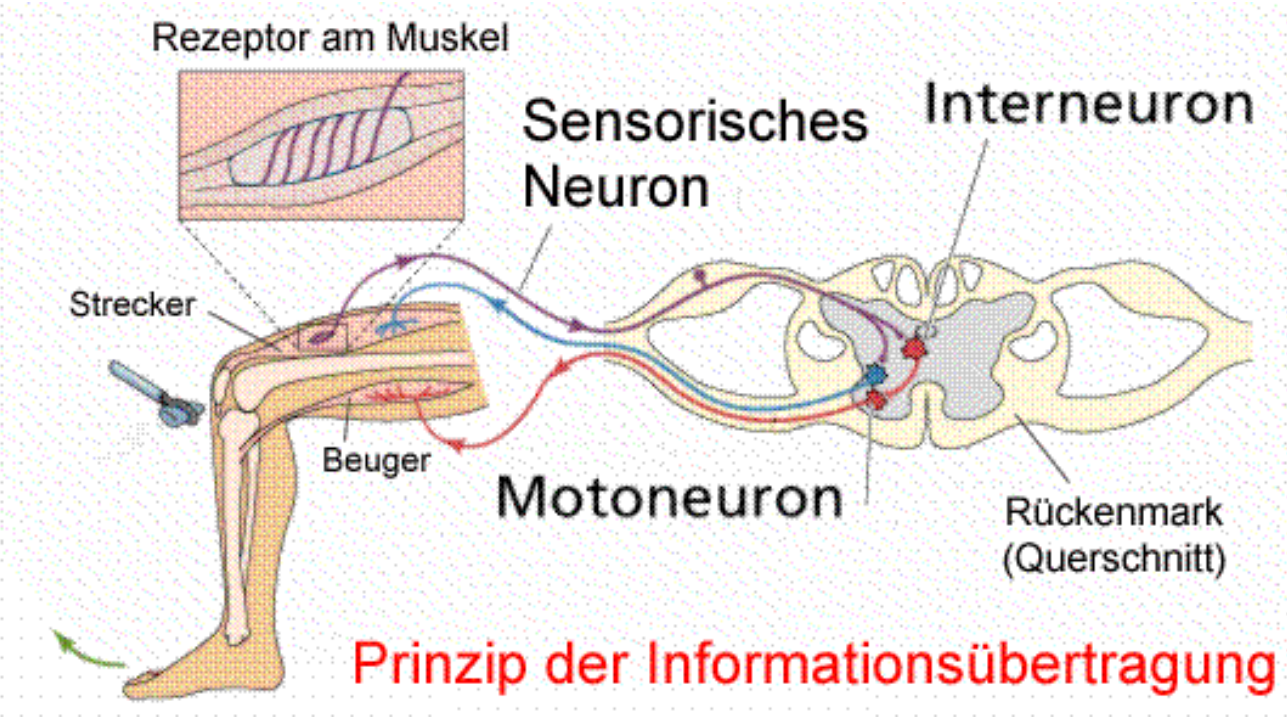
Informationsverarbeitung:

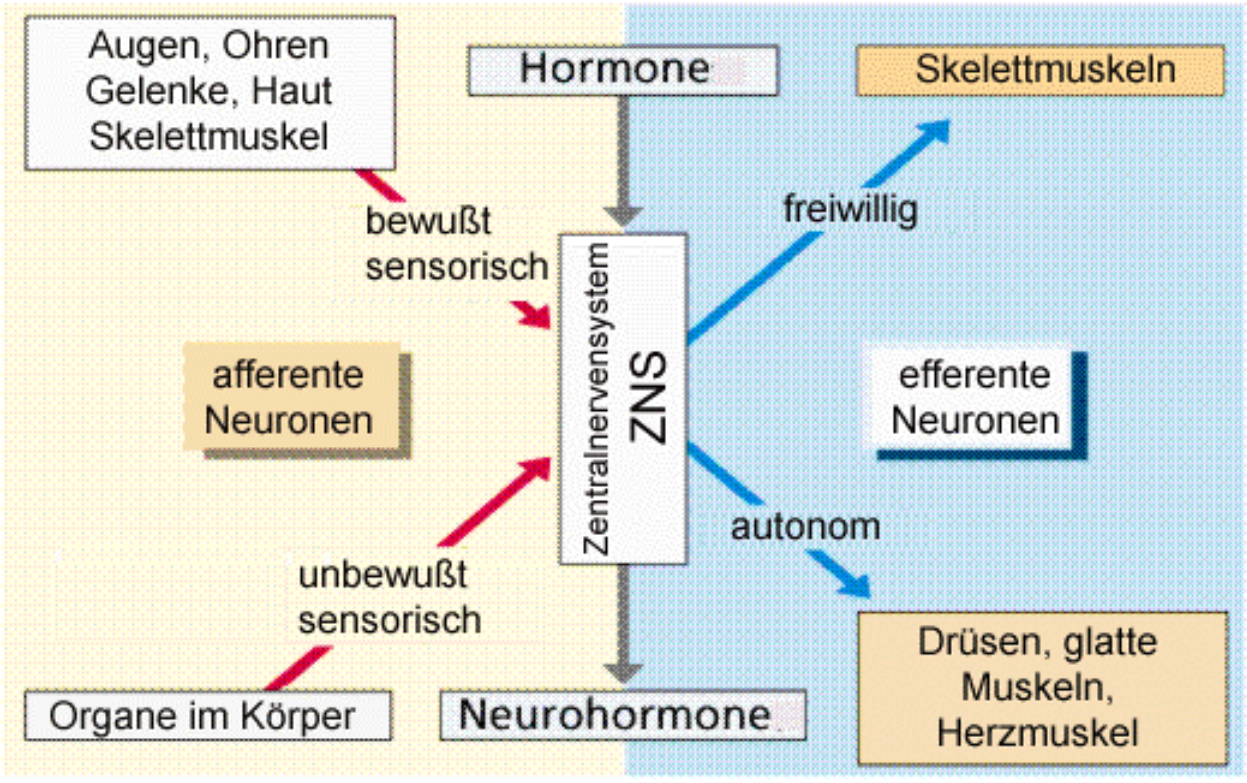
- subsymbolische Elementarinformationen
- massiv parallel und verteilt (MIMD)
- analog, nicht linear

Qualität:

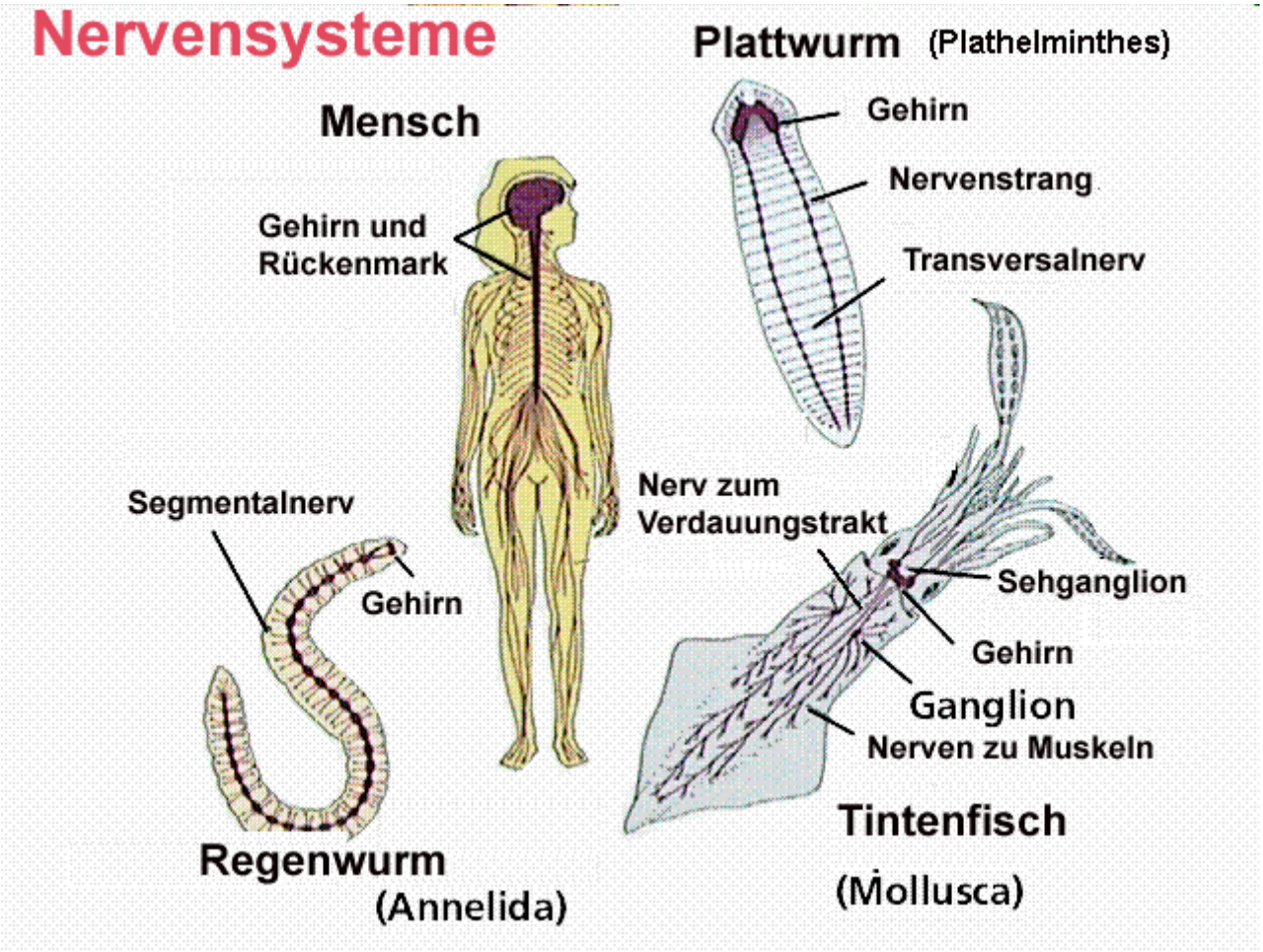
- induktiv und adaptiv
- fehlertolerant
- gut in Musterverarbeitung
- schlecht im Rechnen

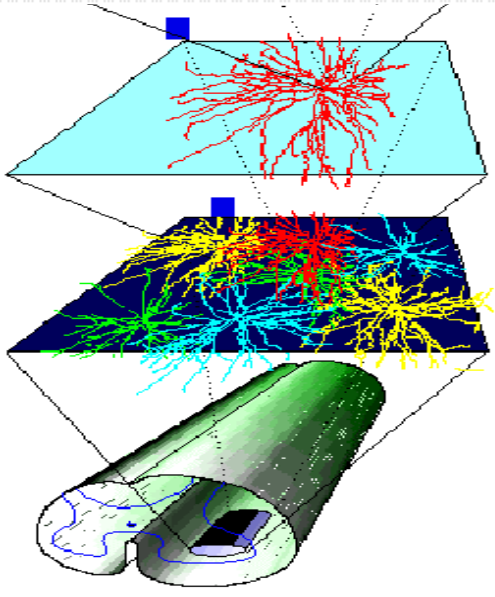
Biologische Neuronensysteme: System



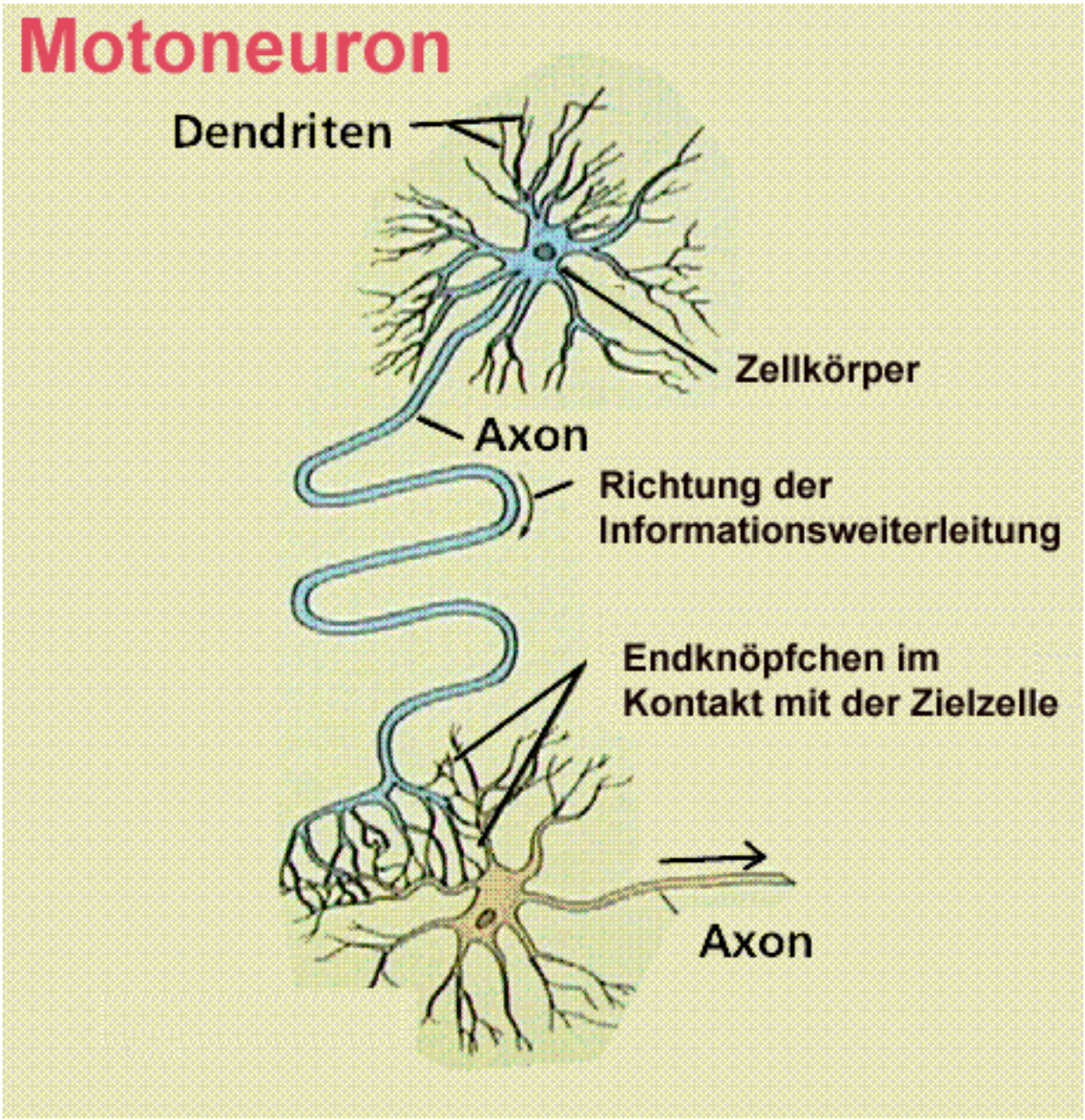


Biologische Neuronensysteme: Netz

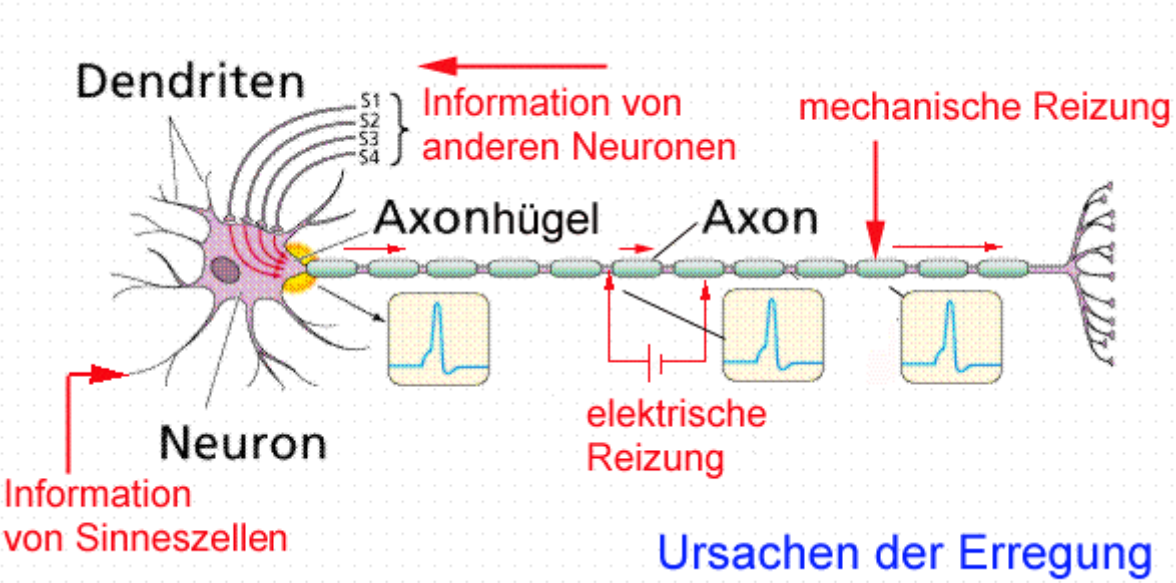




Biologische Neuronensysteme: Neuron

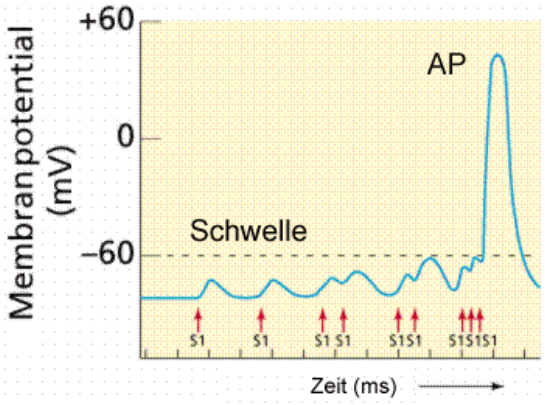


Signalerzeugung

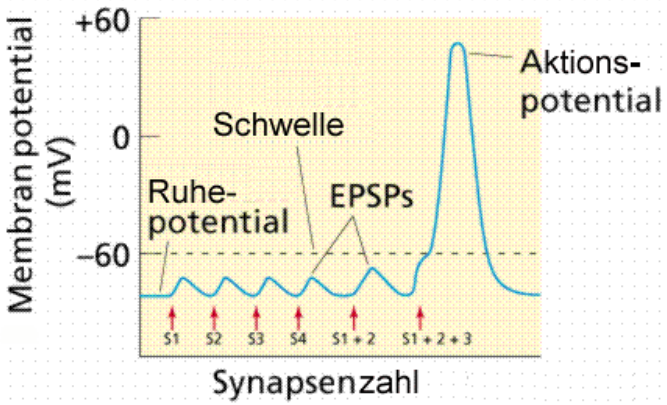


Ursachen der Erregung

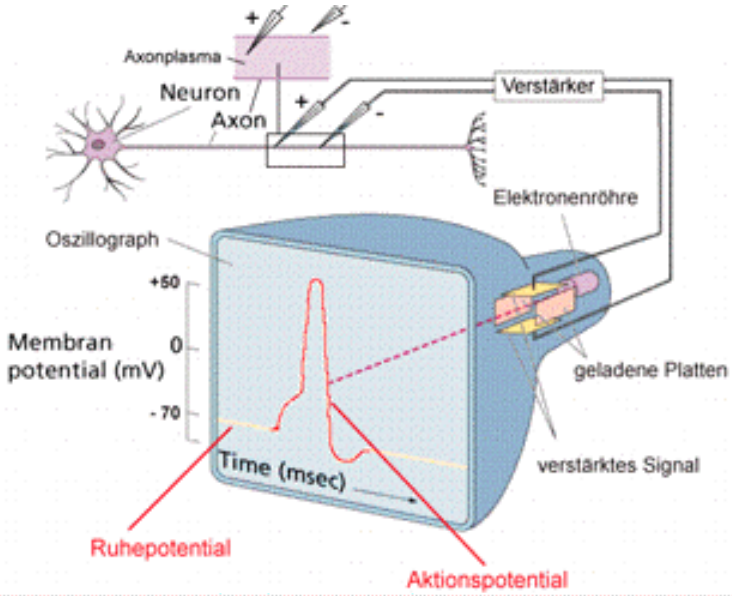
Zeitliche Summation

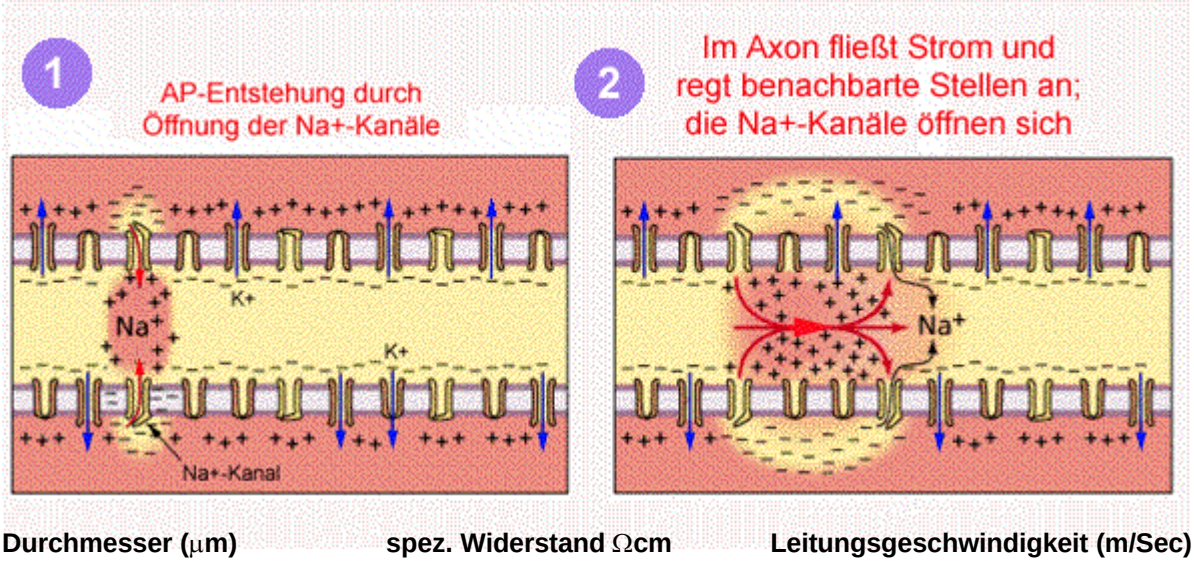


Räumliche Summation



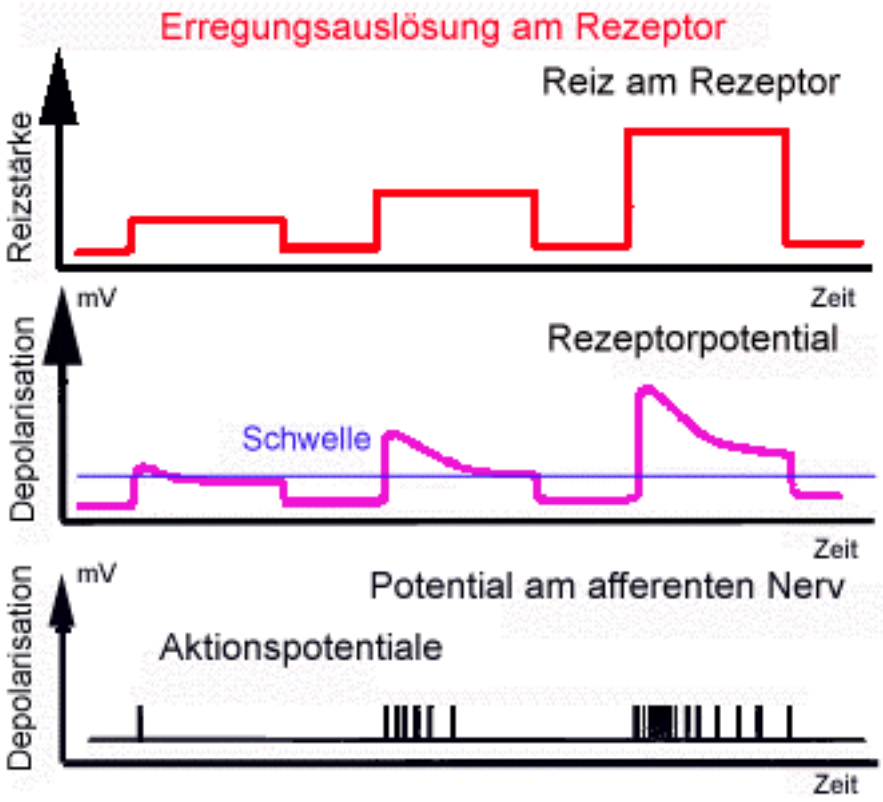
Signalpropagation

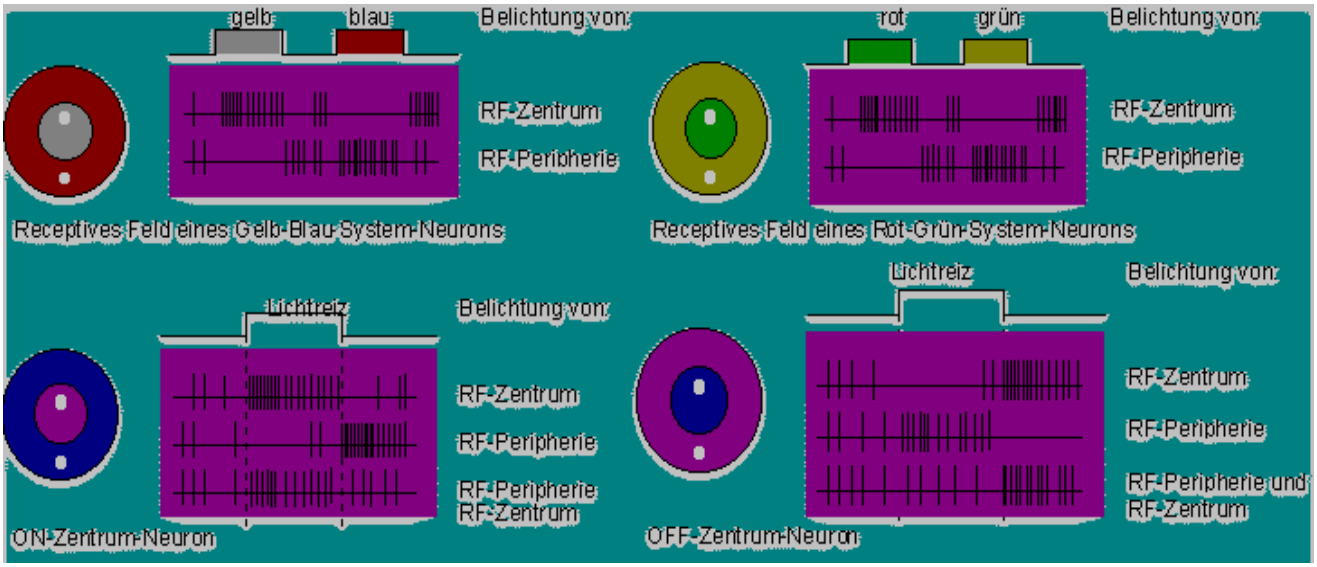




Durchmesser (µm)	spez. Widerstand Ωcm	Leitungsgeschwindigkeit (m/Sec)
A-Faser (peripher, afferent, markhaltig)		
1-20	ca. 125	5-120
B-Faser (visceral,afferent, markhaltig)		
< 4	ca. 125	3-15
C-Faser (Hautschmerz, afferent, marklos)		
0,3 - 1,5	ca. 125	0,6 -3
Riesenaxon Tintenfisch		
500-1000	ca. 30	max. 20

Informationscodierung





Modelle der Neuroinformatik

Systemhierarchie mit drei selbstähnlichen Verarbeitungsstufen:

- I/O Gesamtsystem mit Signalvorverarbeitung – Netzwerk – Resultatumsetzung bzw. –interpretation
- Verarbeitendes Netzwerk mit Eingabeschicht – interne Schichten – Ausgabeschicht
- Prozesseinheit mit Eingabe – Verarbeitung - Ausgabe

Biologisches System

System: Input-Output

==> Sensorsignale

<== Muskel- und Drüsenaktivierung

Netzwerk

==> Zentralnervensystem (Ganglien)

Prozesseinheit

==> Neuronen

Künstliche neuronale Netze

Künstliches Neuron als

==> **Prozesseinheit**

Künstliches neuronales Netzwerk für

==> **Informationsverarbeitung**

(Schichtstruktur, Verbindungstopologie)

Input-Output

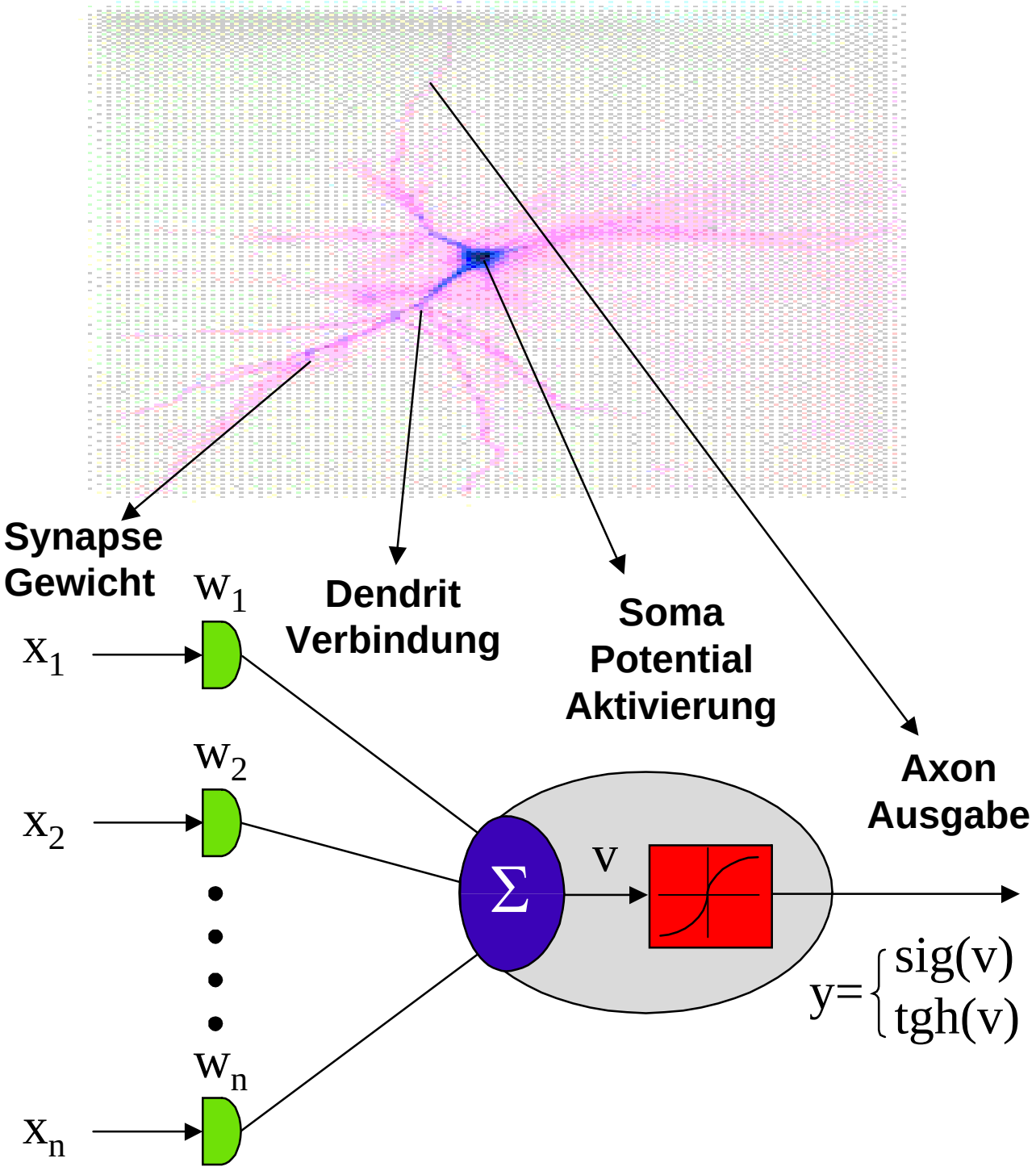
==> **Preprocessing**

Physikalische Daten in numerische Werte

==> **Postprocessing**

Numerische Ausgabe in symbolische Information/Steuerdaten

Prozesseinheit

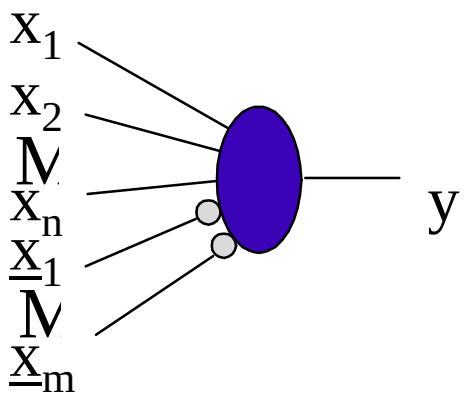


Prozesseinheit: Typen

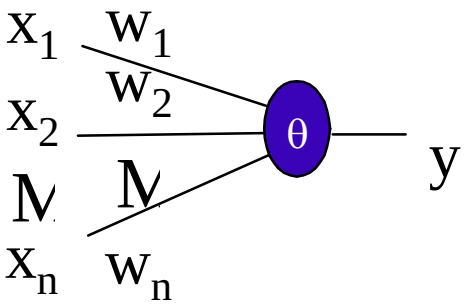
Grundbaustein neuronaler Netze

Neurontypen:

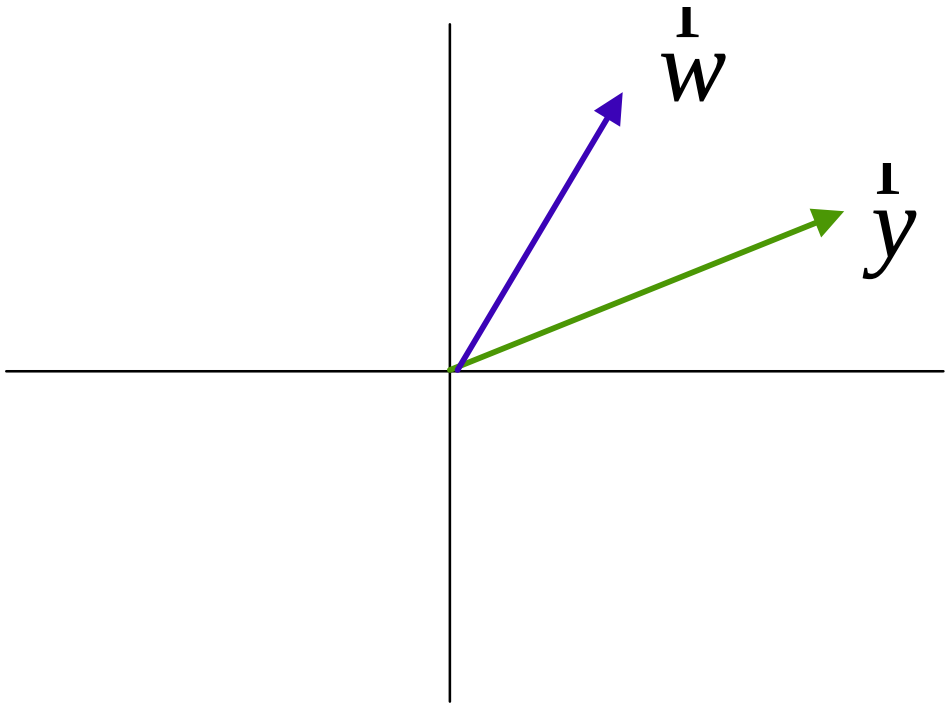
– Mc Culloch Pitts Neuron



– Perceptron Neuron: Gewichte, Bias



Potentialfunktion



Potentialfunktionen Beispiele

$$v_i = \sum_j w_{ij}\, y_j = \vec{w}_i \vec{y} \qquad \text{Summenfunktion}$$

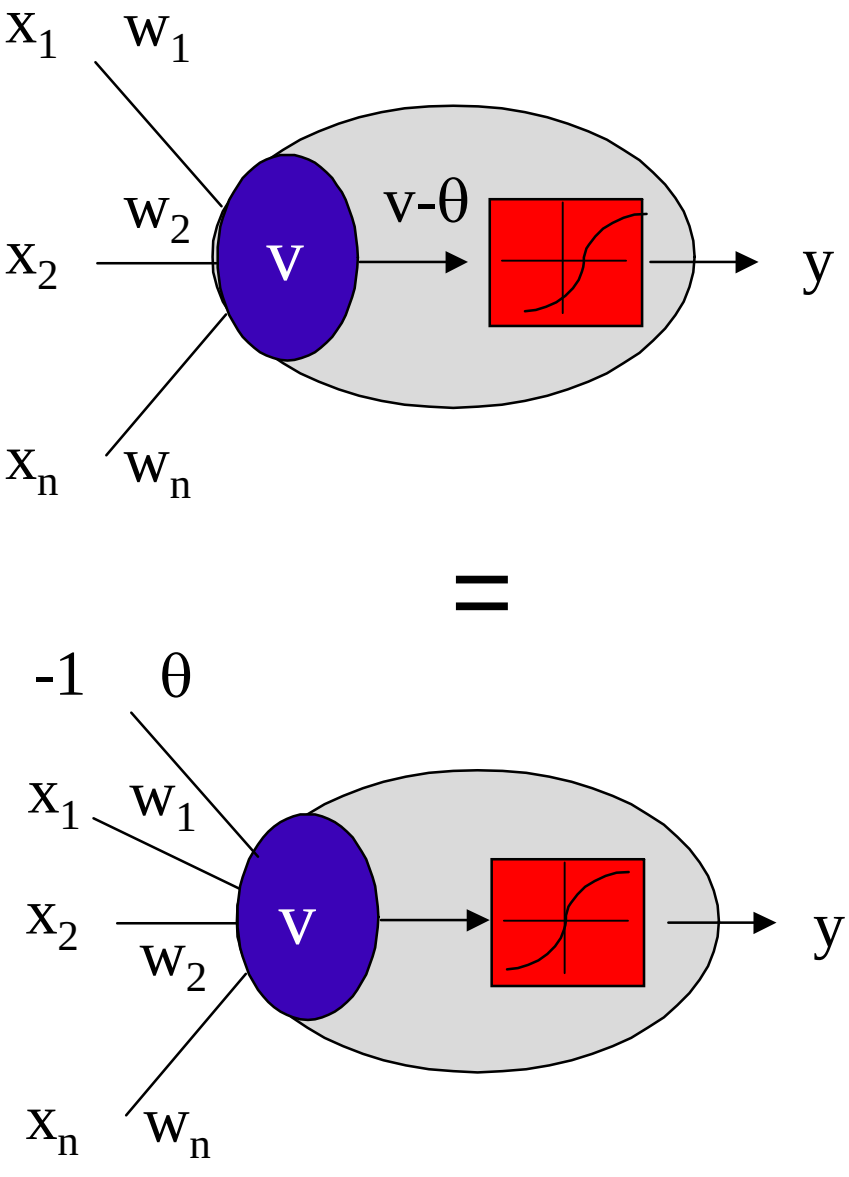
$$v_i = \sqrt{(\vec{y} - \vec{w}_i)^2} \qquad \text{Abstandsfunktion}$$

$$v_i = \frac{\vec{y}\,\vec{w}_i}{\|y\|\,\|w_i\|} = \cos\varphi \qquad \text{Winkelfunktion}$$

$$v_i = \frac{1}{2}\left[\|J\| - \sum w_{ij} y_j\right] \qquad \text{Hammingdistanz} \quad (w, y \quad \textit{binär})$$

Schwellwert (Bias)

Schwellwert ist ein lokaler Speicher
(individuelle Sensitivität des Neurons)



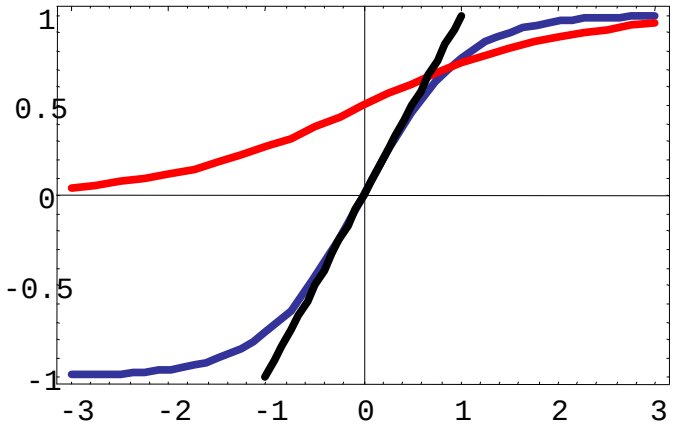
Aktivierungsfunktionen

Glatte Schwellenfunktionen

$$a_1(v) = tgh(v)$$

$$a_2(v) = 1/1 + e^{-v}$$

$$a_3(v) = v$$



Ableitung ist Funktion der Aktivierung a

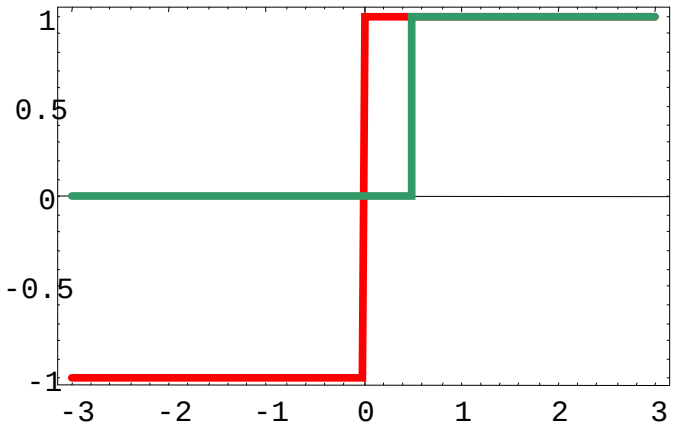
$$a_1'(v) = (1 - a^2) \quad , \quad a_2'(v) = a(1-a)$$

Harte Schwellenfunktionen

$$a(v) = Sig(v)$$

$$a(v) = \Theta(v - \theta)$$

Bsp: Bias



Informationscodierung

Amplitudencodierung:

Mc-Culloch-Pitt Neuron:

	Inaktiv	Aktiv
Binär:	0	1
Bipolar:	-1	1

Perceptron:

	Inaktiv	Aktiv
Linear:	->-◎	->+◎
Sigmoid:	->0	->+1
Tg-hyp:	->-1	->+1

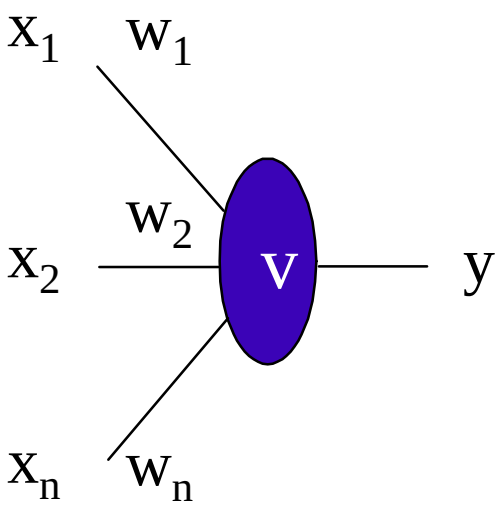
Zwischenzustände

Neurohardware: z.T. Frequenzcodierung

Informationsspeicherung

Nicht lokale verteilte Speicherung

==> Gewichte



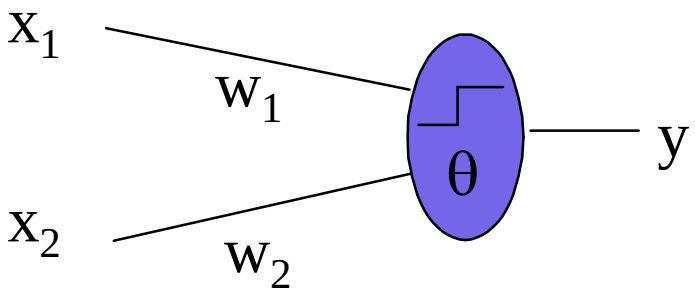
$$[x_1, x_2, \ldots x_n] \begin{bmatrix} w_1 \\ w_2 \\ \mathrm{M} \\ w_n \end{bmatrix} = x^{tr} w = \text{Potential} \rightarrow \text{Aktivierung}$$

Netze für boolsche Funktionen

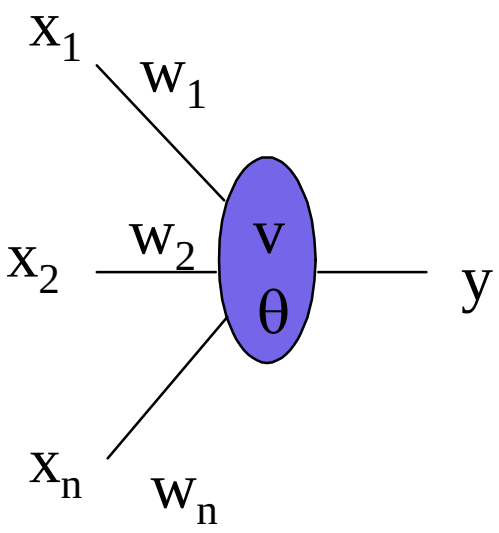
16 boolsche Funktionen

	Eingaben			
Eingang 1	1	1	0	0
Eingang 2	1	0	1	0
	Ausgaben			
0. Bool. Fkt.	0	0	0	0
1. Bool. Fkt.	0	0	0	1
2. Bool. Fkt.	0	0	1	0
3. Bool. Fkt.	0	0	1	1
4. Bool. Fkt.	0	1	0	0
5. Bool. Fkt.	0	1	0	1
6. Bool. Fkt.	0	1	1	0
7. Bool. Fkt.	0	1	1	1
8. Bool. Fkt.	1	0	0	0
9. Bool. Fkt.	1	0	0	1
A. Bool. Fkt.	1	0	1	0
B. Bool. Fkt.	1	0	1	1
C. Bool. Fkt.	1	1	0	0
D. Bool. Fkt.	1	1	0	1

Simples einschichtiges Netz

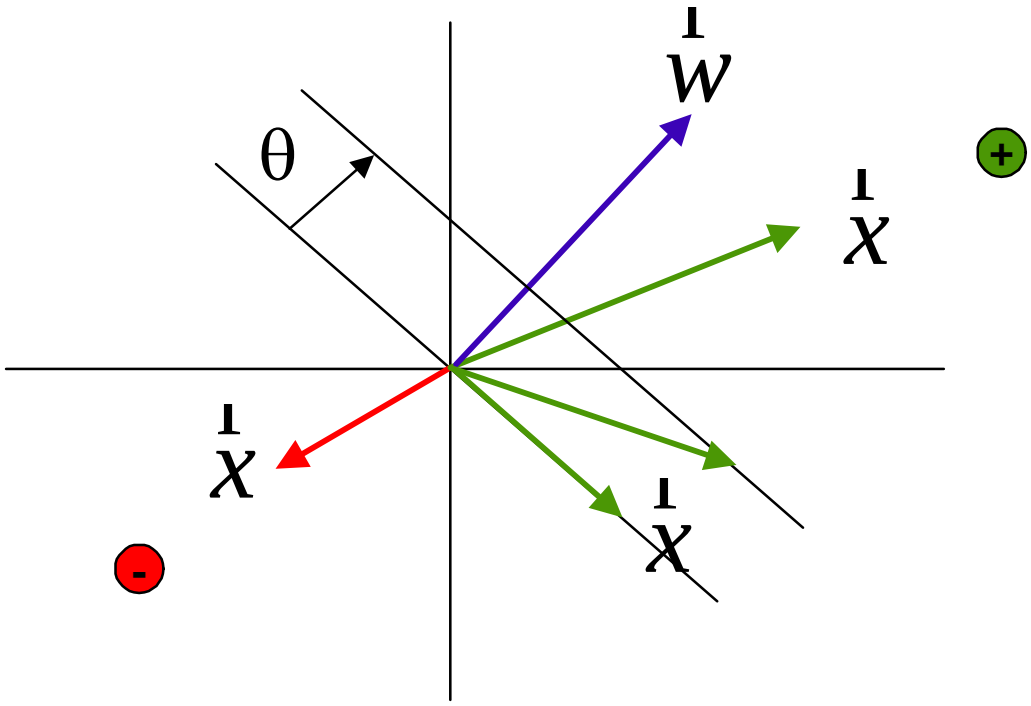


Verarbeitungspotential einer PE



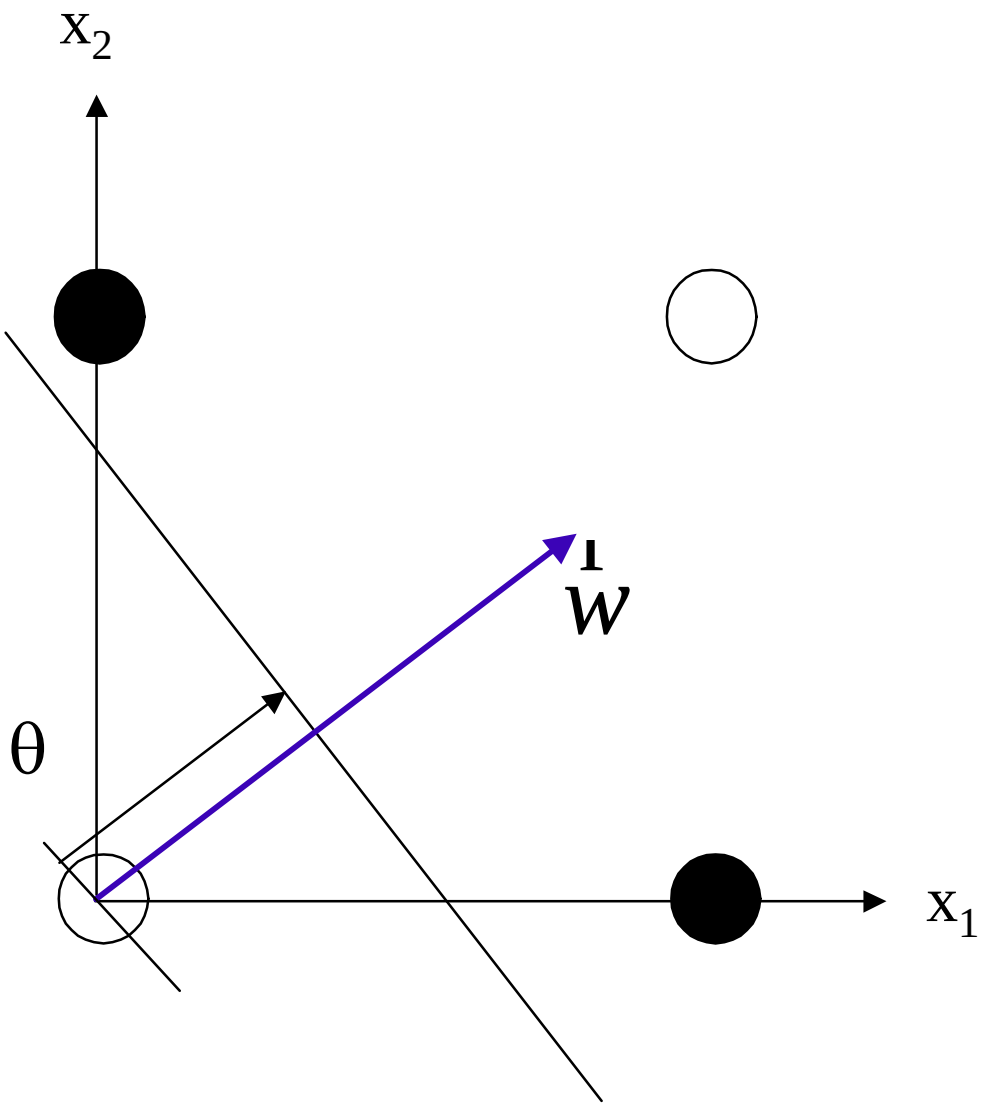
$$v = w_1 x_1 + w_2 x_2 - \theta = \vec{w}\vec{x}$$

Sprungpunkt: $v = 0$



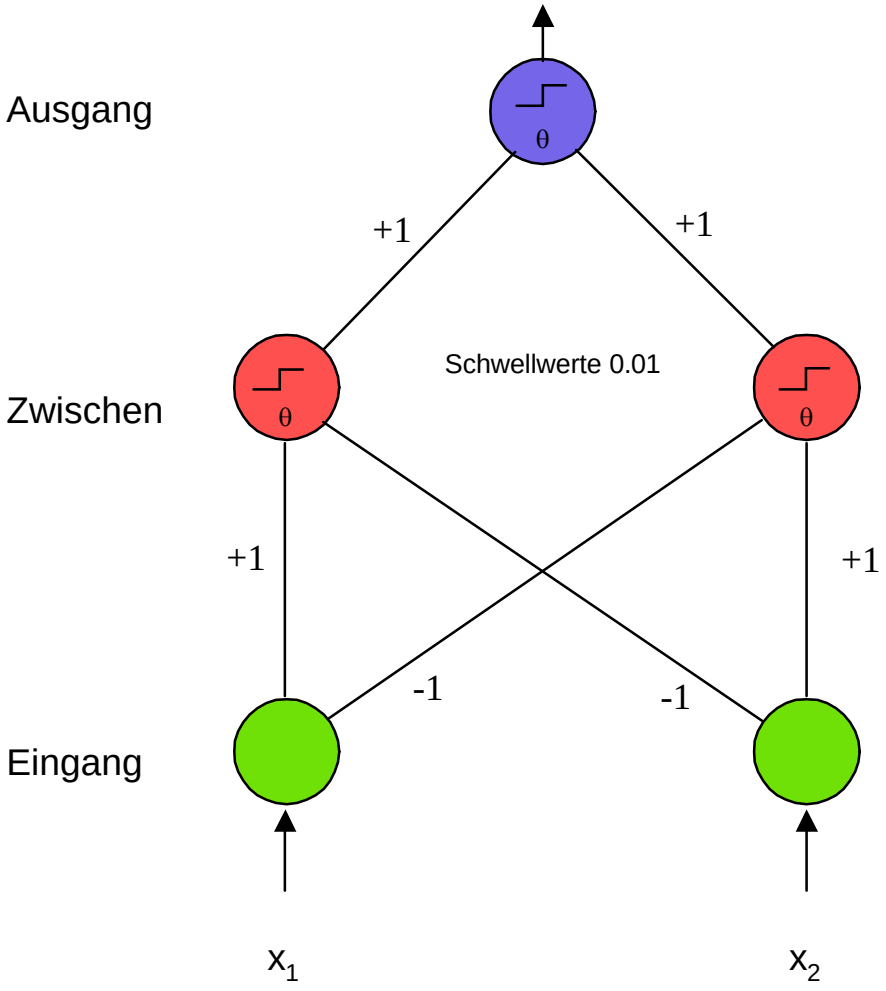
XOR - Problem

Geometrische Interpretation



Lösung

Mehrschichtiges Netzwerk



Problemkategorien

Komplexe Probleme, grosse Datenmenge, geringe Informationsdichte

- Voraussagen (Interpretation von Zeitreihen)
  Nichtlineare rückgekoppelte dynamische Systeme, Devisen/Börsenkurse, Sonnenflecken, Erdbeben, Stromverbrauch
- Klassifizierung, Kategorisierung (Merkmalsextraktion, qualitative Daten)
  Bild- und Tonverarbeitung (Objekt- und Spracherkennung), qualitative Daten (Bewerbungen, Kreditwürdigkeit, Psychologie)
- Assoziation (Datenmuster verbinden)
  Bildkompression, Informationssuche, Gesichts-, Handschrift-, Fingerabdruckerkennung
- Konzeptualisierung (Einteilung von unstrukturierten Daten)
  Versteckte Zusammenhänge in hochdimensionalen Daten, Roboterlernen, Autonome Agenten
- Filterung (Signalrekonstruktion)
  Intelligentes Netzwerkmanagement in der Telekommunikation, Hörgeräte, medizinische Analysen, Militärische Anwendungen
- Optimierung (Reduktion von Kostenfunktionen)
  TSP, Loaddistribution (Prozesse in verteilten Systemen), Ressourcenmanagement

Anwendungen

- Bildverarbeitung
- Signalverarbeitung, Mustererkennung
- Spracherkennung
- Robotik, Zielverfolgung
- Risikoanalyse und -bewertung
- Datenanalyse und Voraussagen
- Modellierung in Neurowissenschaft

Patente

Wirtschaftliche Bedeutung der NN Technologie illustriert durch Anzahl erteilte Patente

USA (92/93):

SW-Patente (insgesamt)	2700
AI und verwandte Gebiete	311
davon NN	83

Geschichtliche Entwicklung

- William James (1890): Assoziationsfähigkeit und neuronale Struktur des Gehirns
- McCulloch und Pitt (1943): Einführung des formalen **McCulloch Pitt Neurons** mit einer Stufenfunktion als Anregungsschwelle
- Donald Hebb (1949): Physiologisches Modell des Lernens (Hebbsche Lernregel), Verstärkung der aktiven Verbindungen
- Minsky, McCarthy, Rochester, Shannon (1956): Darthmouth Konferenz, Simulationen neuronaler Netze, Geburtsstunde der künstlichen computersimulierten Intelligenz
- Frank Rosenblatt (1958): Entwicklung des **Perceptrons**, erstes selbstorganisierendes System (1 Schicht Netz), Perceptron Lernregel
- Bernhard Widrow (1960): Adaline (Adaptive linear element) und Madaline (Multiple adaptive linear element) Netze, McCulloch Pitt Neuron mit Gradientenlernregel.
- Marvin Minsky, Seymour Papert (1969): Abbruch der Entwicklung nach Publikation des Buches *Perceptrons*, XOR Problem
- Anderson, Kohonen, Grossberg (1969-1980): Einige unentwegte arbeiten weiter und
- entwickeln **Brain-State-in-a-box, Assoziative Speicher** und **Adaptive resonance theory** (ART)
- John Hopfield (1982): Physikalische Analogie zu neuronalen Netzen, **Hopfield Modell** rekursives Netz mit konvergierender Iteration
- Hinton und Sejnowski (1985): Stochastische neuronale Netze, **Boltzmann Maschine** mit zufälligem Lernen
- Rumelhart, Hinton (1986): Ueberwindung der Minsky-Stagnation durch Einführung des **Backpropagation Algorithmus** für das Lernen in mehrschichtigen Netzen
- Fukushima, Hecht-Nielson, Kohonen, Grossberg uam (1987-heute): Neue Netzwerktypen und Lernalgorithmen (**ART II, Cogitron, Neocogitron, Counterpropagation, SOM, BAM, LVQ** etc)
- Cohen, Grossberg, Funahashi, Baum, Hecht-Nielsen, Kosko ua. (1988-1993): Vertiefte theoretische Einsichten in das Funktionieren von neuronalen Netzen (Kolmogorov Theorem, VC Dimension)
- ab 1988: Kommerzielle neuronale Netzwerksimulatoren und Netzwerkhardware
- ab 1990: Neuromorphe HW (Silicon Retina, Silicon Cochlea etc)
- 1998: Erste erfolgreiche Anwendung der Zeitcodierung in der Spracherkennung

Zukunft

Entwicklungsstufen neuronaler Netze

1. Generation:

   Neuronenmodelle, einschichtige Netze

2. Generation: (heute realisiert)

   Mehrschichtige Netze, Lernalgorithmen
   Neuronale HW

3. Generation: (Forschungsgebiet)

   Neuromorphe Netze, funktionale Substrukturen, biologisch inspiriert
   Biologie-Elektronik Schnittstellen

Neuronale Netze

Themen: Grundlagen und Konzepte

- Architektur, Topologien
- Gewichtsmatrix
- Netzdynamik
- Lernen
- Lernregel

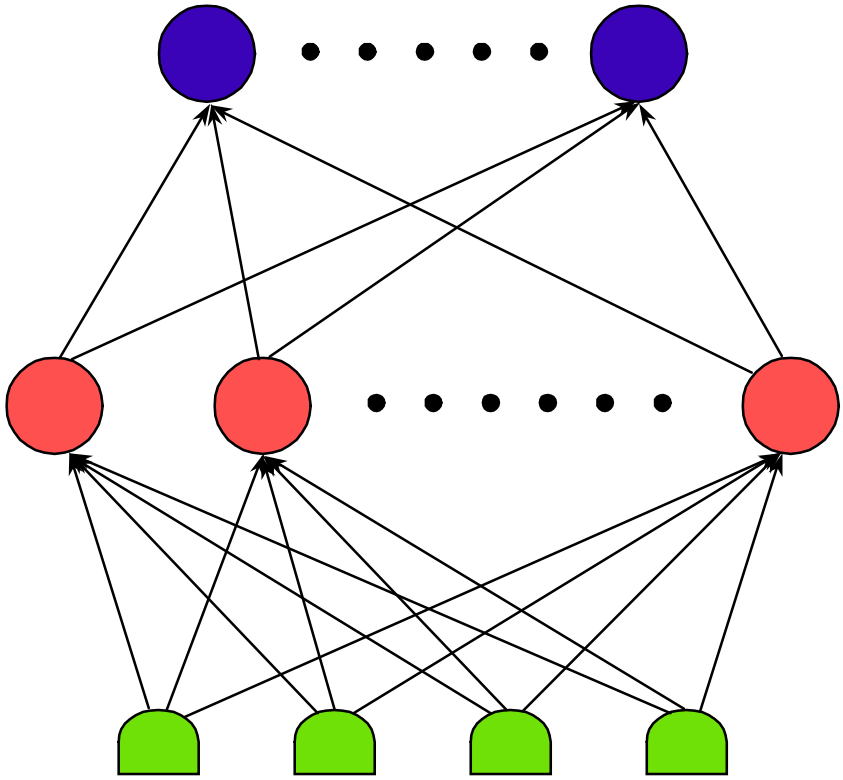
Lernziele *Grundlagen und Konzepte*

Kennen der

- Topologien
- Darstellung durch Gewichtsmatrix
- Lernregeln (verstehen was *Lernen als Algorithmus* heisst)
- Geometrische Bedeutung der Gewichte, Speicher

Aufbau neuronaler Netze

Schichtstruktur



Input	==>	Fanout
Hidden	==>	Fanin-Fanout
Output	==>	Fanin

Verbindungstopologie

Gewichtete Verbindungen

w>0	anregend
w<0	hemmend
w=0	nicht verbunden (zur Zeit)

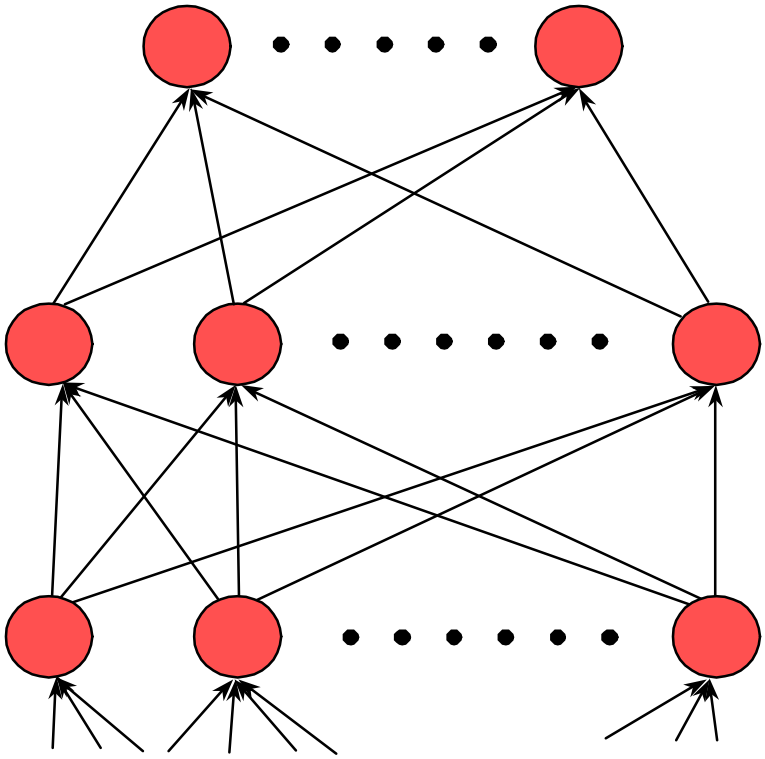
Datenflussrichtung relativ zu Schicht

- feedforward
- feedbackward

Datenflussrichtung relativ zu Neuron

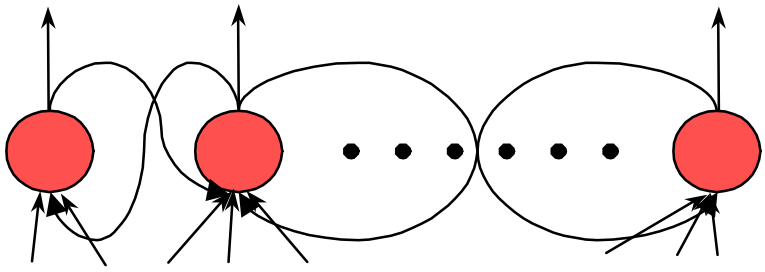
- forward
- recurrent
- lateral

Beispiele Topologien



Feedforward

Anzahl Berechnungsschritte definiert

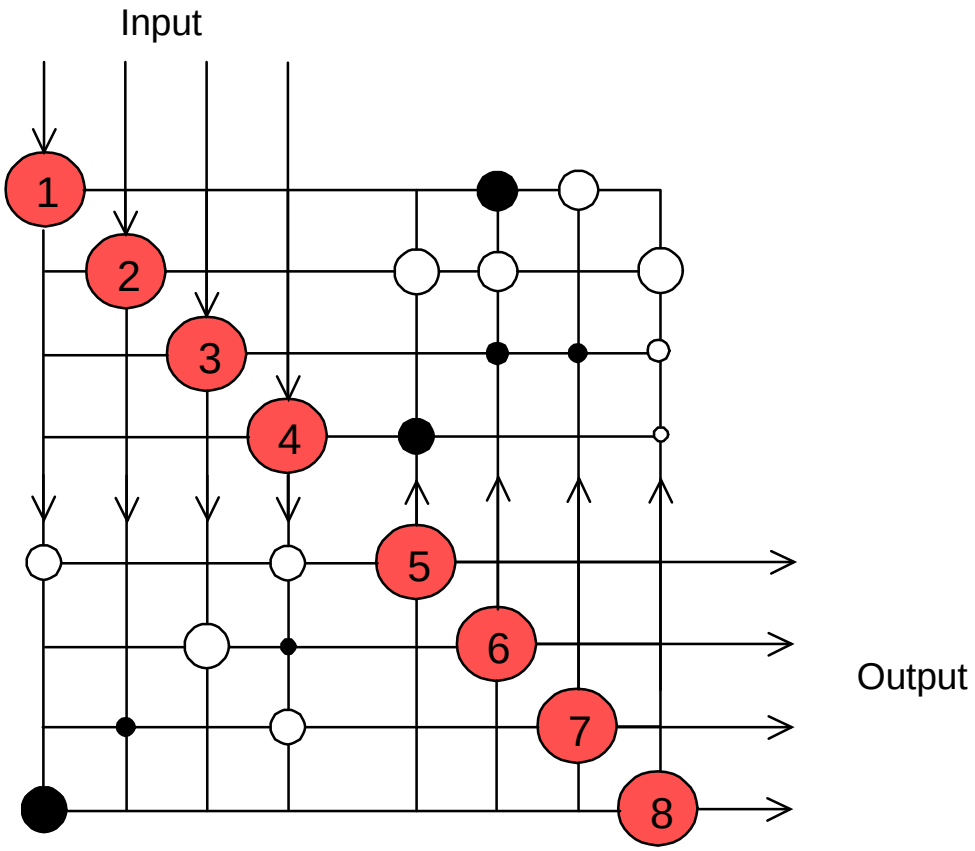


Rekurrent, lateral

Anzahl Berechnungsschritte variabel

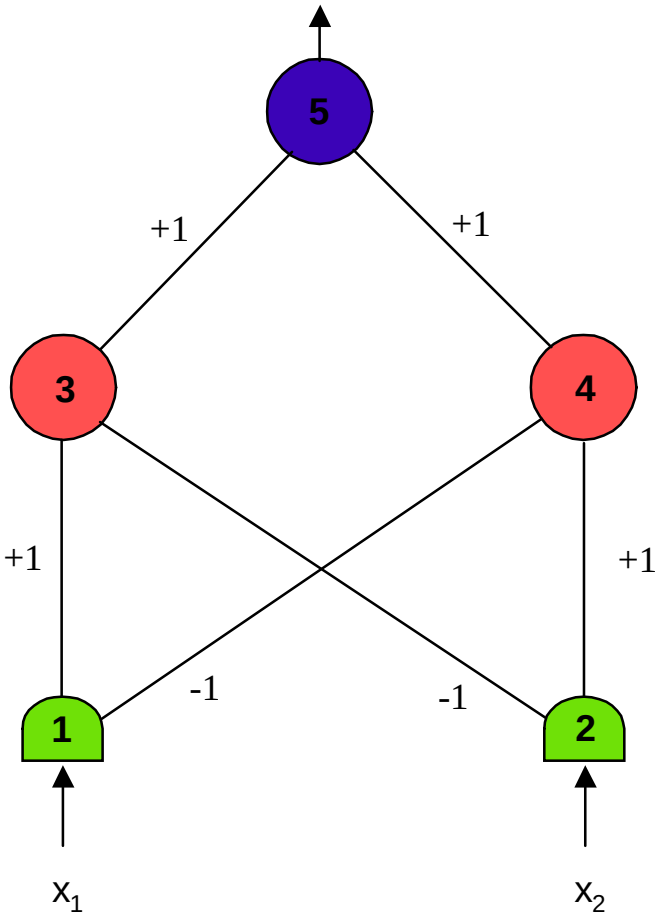
Darstellung

Hinton Diagramm



	PE1	PE2	PE3	PE4	PE5	PE6	PE7	PE8
PE1	0	0	0	0	0	-5	5	0
PE2	0	0	0	0	6	5	0	6
PE3	0	0	0	0	0	-3	-2	3
PE4	0	0	0	0	-4	0	0	1
PE5	4	0	0	4	0	0	0	0
PE6	0	0	6	-1	0	0	0	0
PE7	0	-2	0	4	0	0	0	0
PE8	-6	1	0	0	0	0	0	0

Gewichtsmatrix



Ausgang

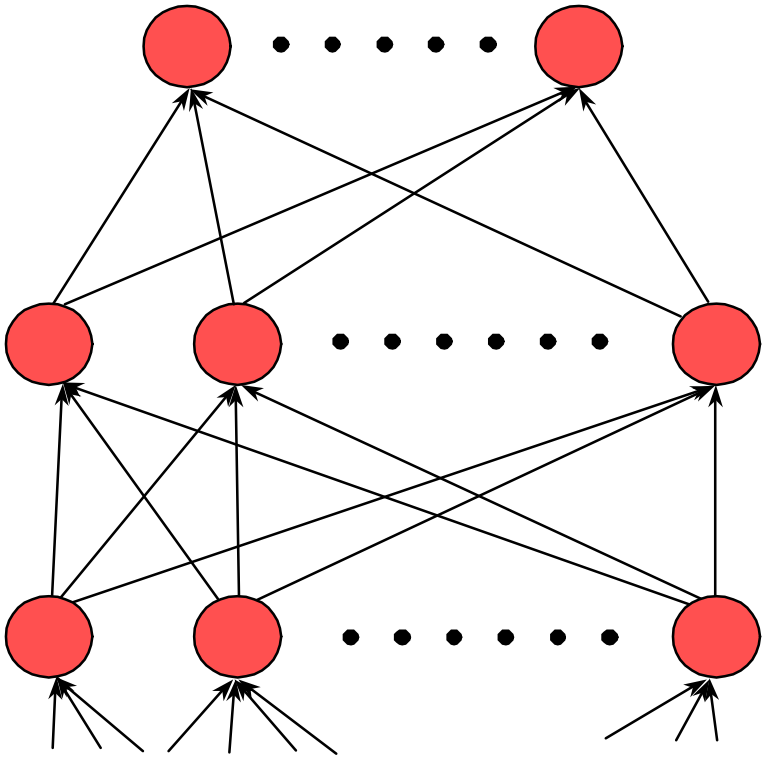
Zwischen

Eingang

	Eingang von				
zu	PE1	PE2	PE3	PE4	PE5
PE1	1	0	0	0	0
PE2	0	1	0	0	0
PE3	1	-1	0	0	0
PE4	-1	1	0	0	0
PE5	0	0	1	1	0

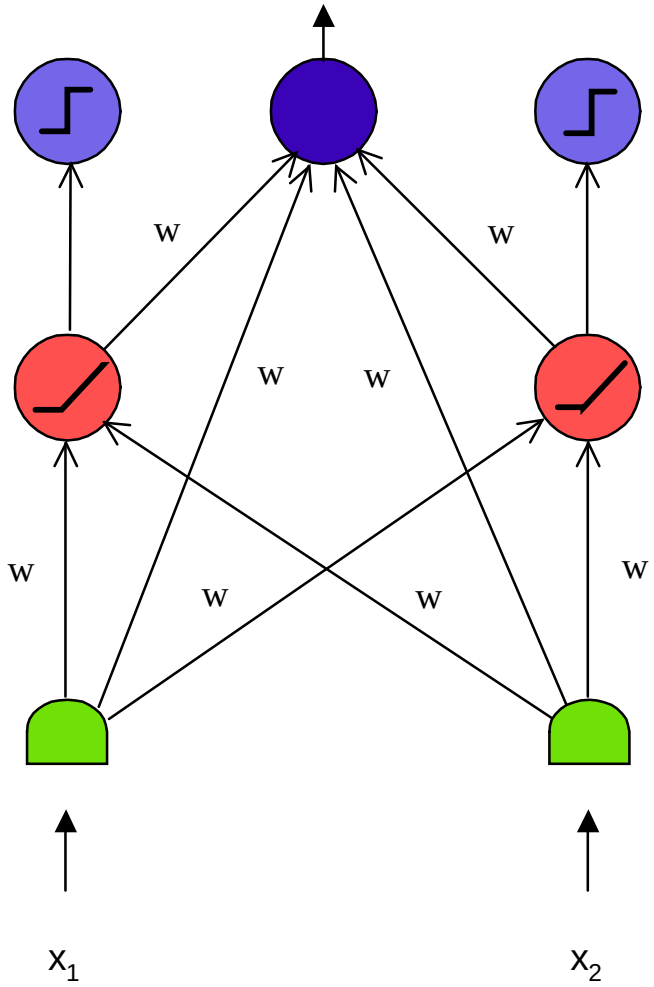
$= W$

Gewichtsmatrix Feedforward

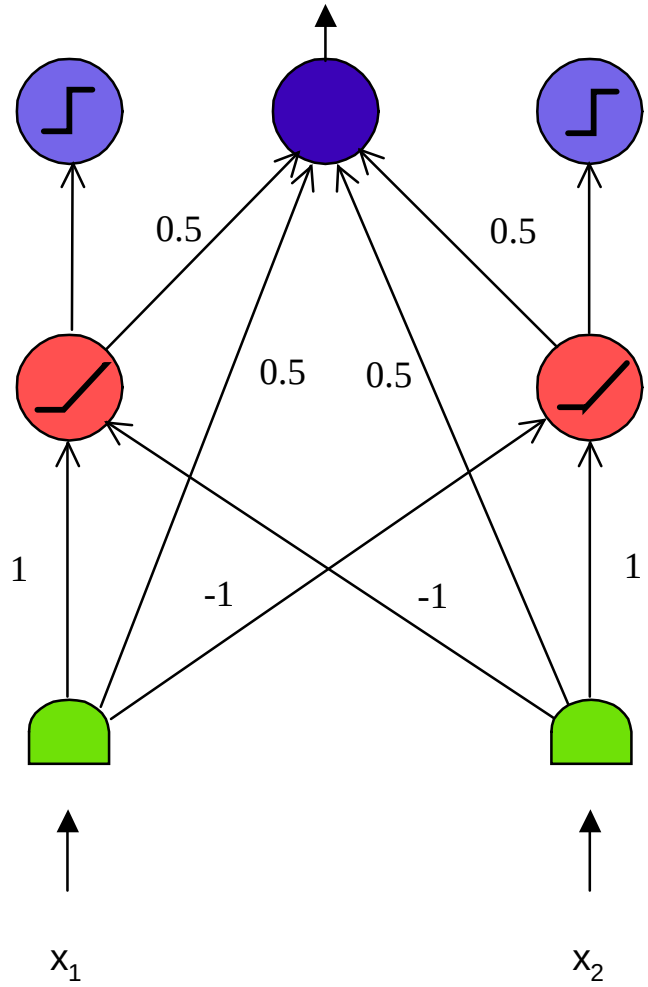


$$W = \begin{bmatrix} E & 0 & 0 & 0 \\ W^{[0]} & 0 & 0 & 0 \\ 0 & W^{[1]} & 0 & 0 \\ 0 & 0 & W^{[2]} & 0 \end{bmatrix}$$

Beispiel
Maximum Netz

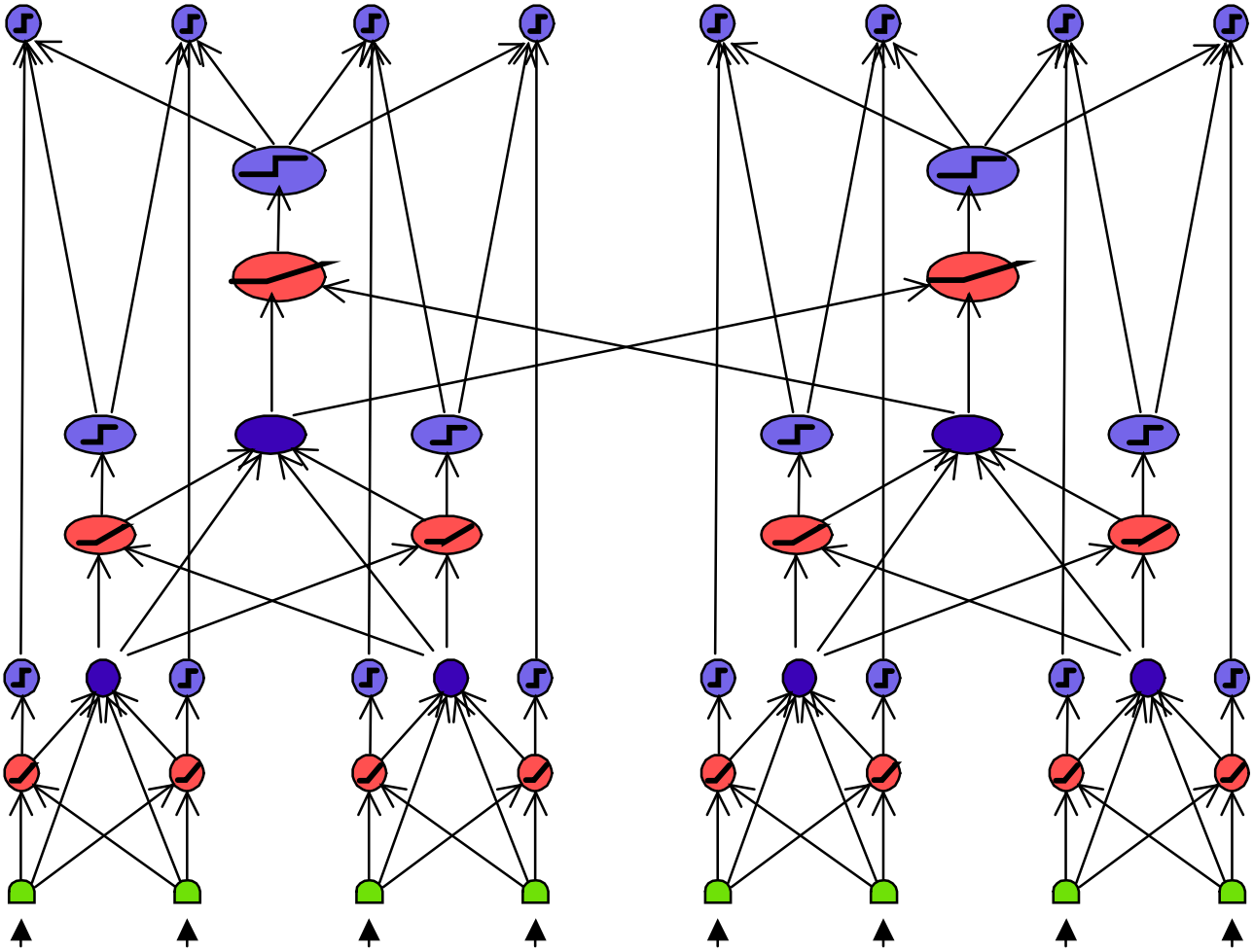


Beispiel
Maximum Netz



Segmentierte Netze

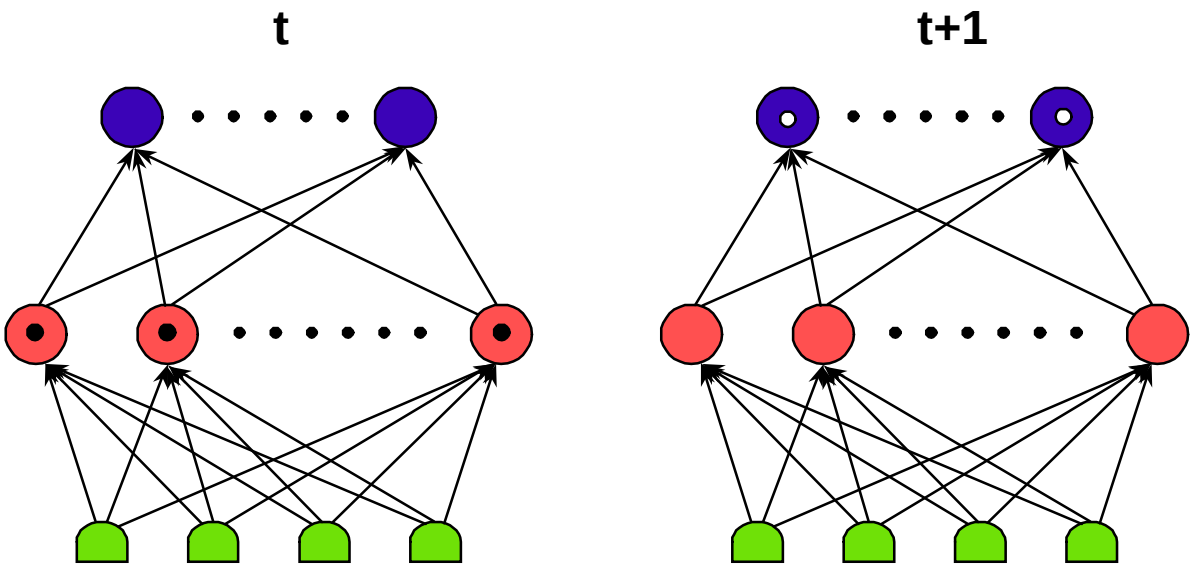
Beispiel: Maximum Net



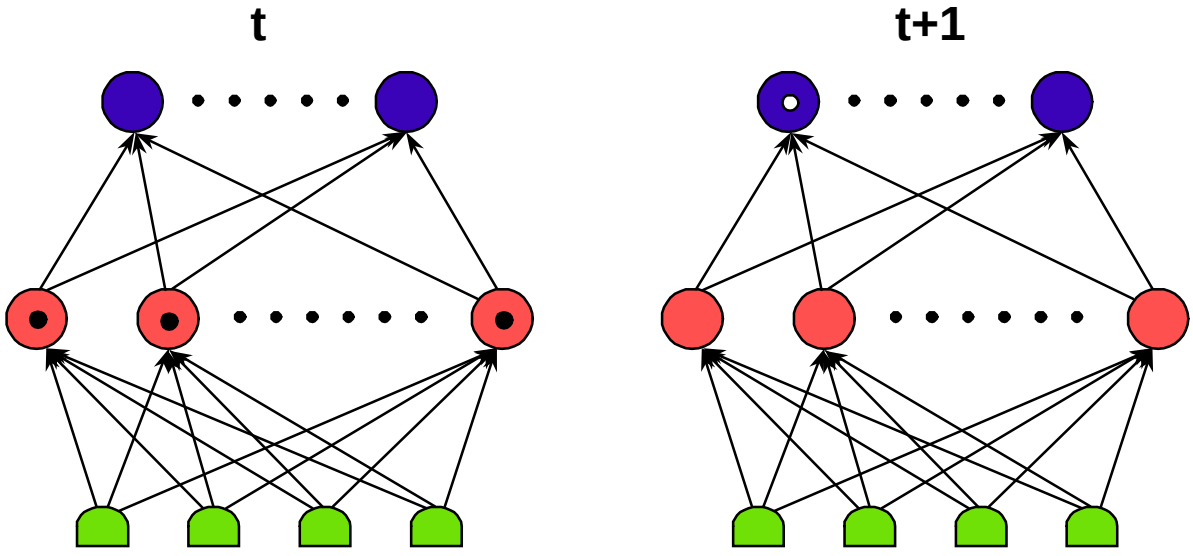
Netzdynamik

Datenflusskontrolle im Netz

==> (block)synchron



==> asynchron

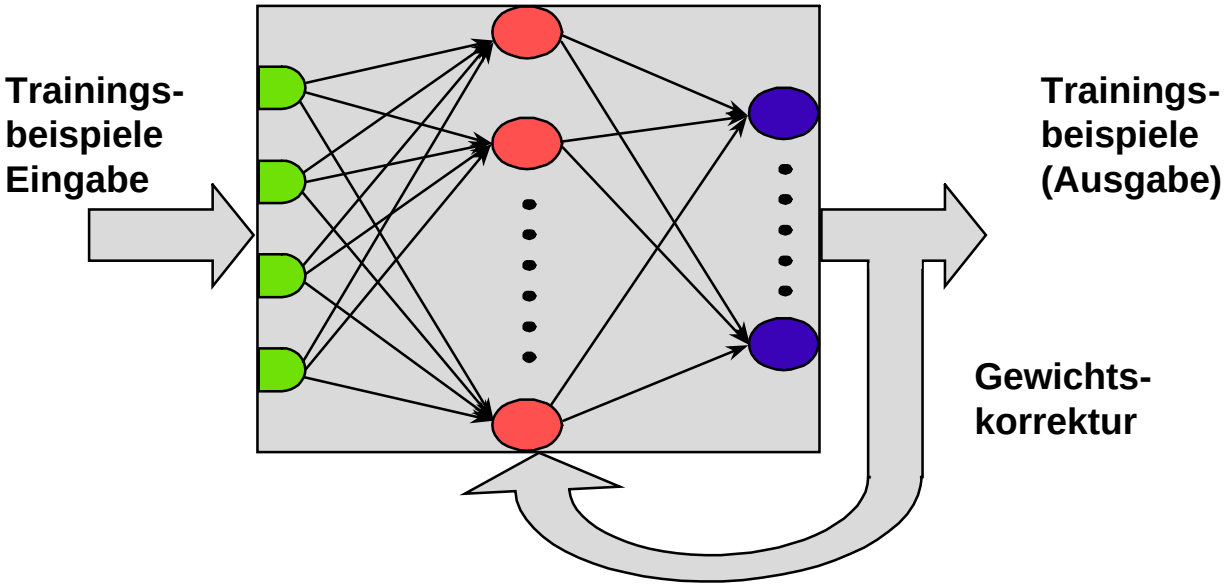


Netzbetrieb

Betriebsphasen

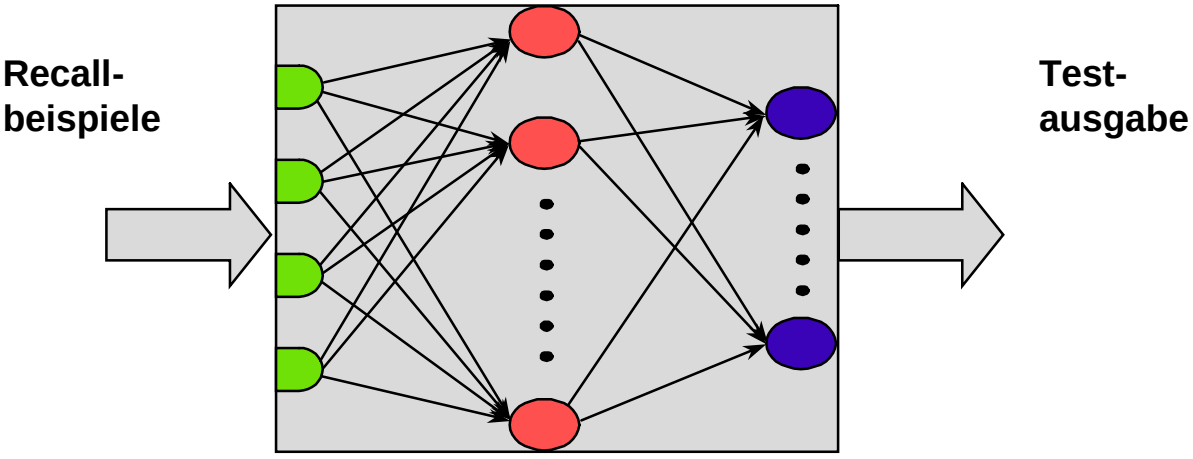
Training und Lernen

==> Anpassung der Gewichte



Recall

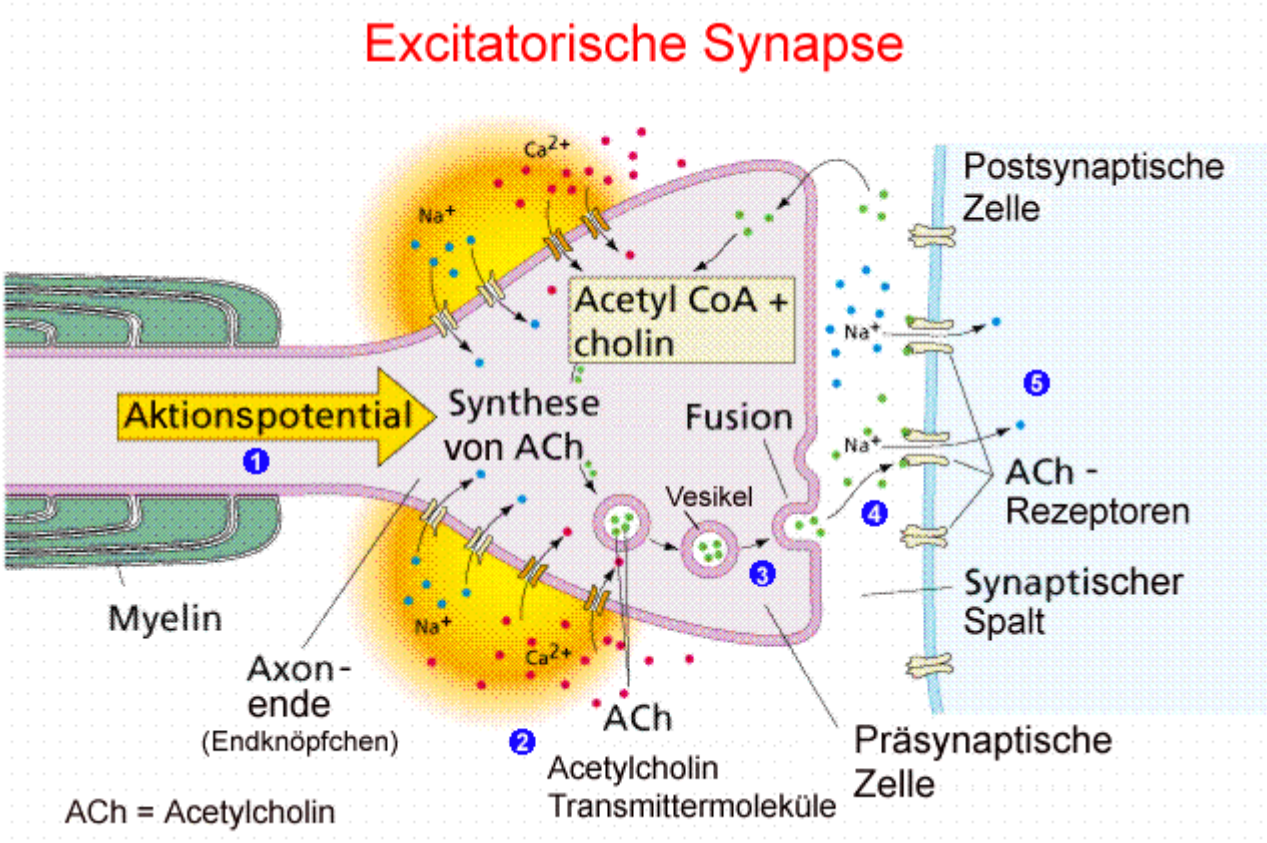
==> Feste Gewichte



Biologisches Lernen

Erzeugung / Elimination von Verbindungen

Veränderung der synaptischen Effizienz



Induktion, Lerntheorien

Prinzip der Induktion:
Gewinnung von allgemeinen Aussagen aus einzelnen Beobachtungen

Induktion ist Lernen aus Beispielen:

- negative Evidenz
- Varianz kausaler Faktoren und Resultate
- Wahrscheinlichkeit aus Korrelation
- einfache Aussagen (Occams razor)

Lerntheorien:
Analyse und Formalisierung des Lernens

- Verhaltenstheorie (S-R Theorien)
- Kognitive Lerntheorien (mentale Struktur)
- Quantitative Modelle (PAC, Neuroscience)
  ==> Stichprobenkomplexität
  ==> Trainingskomplexität

Lernhierarchie

- Verstärkungslernen (Dressur)
  ==> Bedingter Reflex

- Imitationslernen
  ==> Kopieren von Verhalten

- Lernen durch Versuch und Irrtum
  ==> Verstärkung, Dämpfung

- Konzeptlernen
  ==> Verallgemeinerung

- Lernen duch Strukturierung und Einsicht
  ==> Übertragen von Lösungen
  Zusammenhänge erkennen

Ziel des Lernens:

==> Verbesserung des Verhaltens
Verallgemeinerung

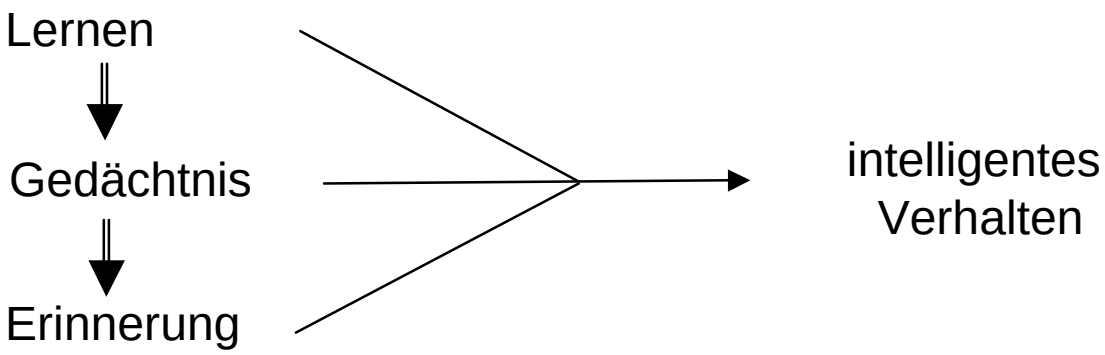
Lernphasen

Verhalten bestimmt durch:

- unbedingte Reflexe, Automatismen, Instinkt
  ==> angeborenes Verhalten

- Erfahrungen mit Lernen:
  ==> erfahrungsbedingtes Verhalten

Lernphasen



Lernen in neuronalen Netzen

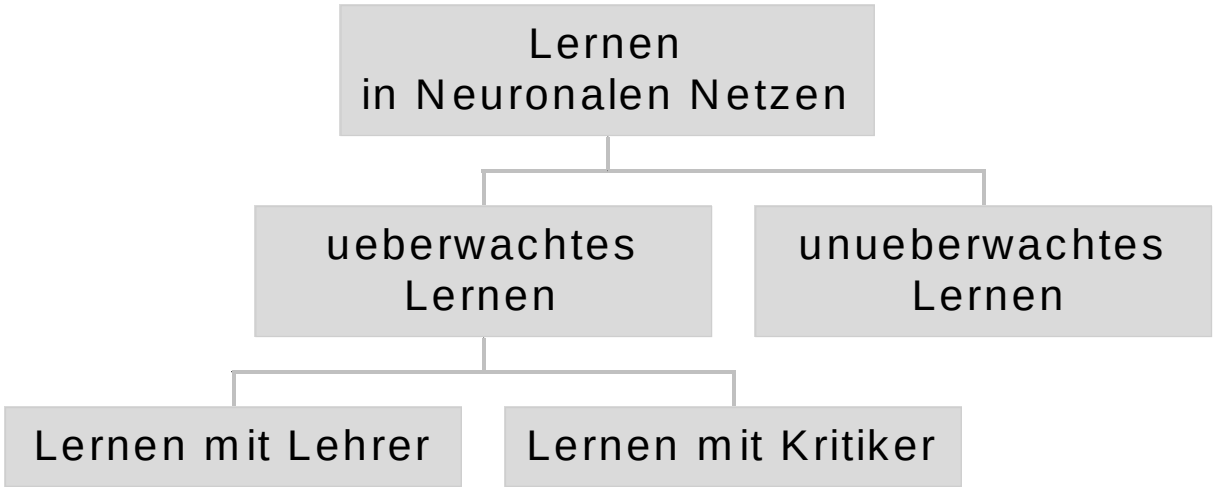
Induktives Lernen aus Beispielen

- **Lernen bedeutet:**
  Veränderung der Gewichte

$$\Delta W = Lernregel(x, y, z, \alpha, w...)$$

  - x Eingangsvektor
  - y Netzausgangsvektor
  - z Zielvektor
  - a Lernrate
  - w aktuelle Gewichte
- **Lernphasen**
  - Adaption mit Lernbeispielen: Gewichtsanpassung
  - Test der Verallgemeinerung: Testbeispiele
  - Anwendung: Trainiertes Netzwerk

Lernverfahren



Definitionen und Begriffe

- Lernen aus Beispielen, induktives Lernen

$$\#\,Beispiele \approx \frac{\#\,Gewichte}{Fehlerrate}$$

- Anpassung der Gewichte

$$w(t+1) = w(t) + \Delta w(t)$$

- Lernregel aus Fehlerfunktion, Lernrate

$$\Delta w(t) = L[B(\mathbf{x}, \mathbf{z}, \mathbf{w}, \eta..)] \overset{z.B.}{=} -\eta \frac{\partial B}{\partial w}(\mathbf{x}, \mathbf{z})$$

- Lerndauer
  ==> Lernoptimum, Verallgemeinerung

- Präsentation der Beispiele
  ==> Musterlernen
  Epochenlernen

- Lernerfolg
  ==> Fehlerrate (Trainingsbeispiele)
  Verallgemeinerung (Testbeispiele)

- Lernvermögen
  ==> Theoretisch lernbare Aufgaben

Lernkonzepte

Modellfreie Adaption an Musterbeispiele

- **Überwachtes Lernen**
  - mit Lehrer

$$x_p = (x_{1p}, \ldots x_{np}) \quad \Rightarrow \quad z_p = (z_{1p}, \ldots . z_{mp})$$

  - mit Kritiker (Bewertung)

$$x_p = (x_{1p}, \ldots x_{np}) \quad \Rightarrow \quad B(y_p)$$

- **Unüberwachtes Lernen**

$$x_p = (x_{1p}, \ldots x_{np}) \quad \Rightarrow \quad \text{Merkmalscluster}$$

Lernregeln

- Verstärkungslernen
  Hebbsches Lernen (1 Schicht)

$$w(t+1) = w(t) + \eta\, y_j x_i$$

  - Verbindungen zwischen Ein- und Ausgang, die beide aktiviert sind werden verstärkt (Korrelation)
  - Zusatz: Nichtaktivierte Verbindungen werden abgeschwächt

- Fehlerkorrigierendes Lernen
  Deltaregel, Backpropagation (1 Schicht)

$$w(t+1) = w(t) + \eta\,(z_j - y_j)\,x_i$$

  - Verbindungen zwischen Ein- und Ausgang werden verstärkt bzw. gehemmt, je nachdem ob der Netzausgang y dem Zielwert z entspricht
  - Die Korrektur erfolgt in Funktion der Abweichung (Fehler)
  - Definition einer Fehlerfunktion (Summe der quadratischen Abweichungen)
  - Korrektur der Gewichte, so dass Fehler minimiert wird
    Gradientenabstiegsverfahren

Neuronale Netze

Themen: Wichtigste Netze

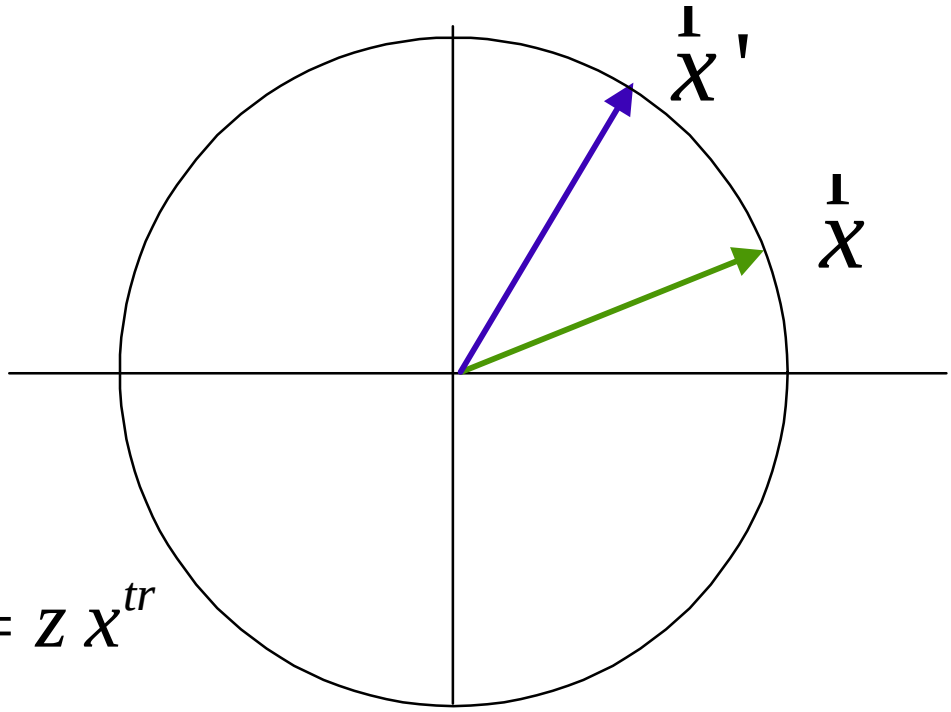
- Matrixspeicher
- Perceptron
- Gradientenabstieg
- Deltalernen
- Backpropagation
- PCA
- Hebbsches Lernen
- Kohonennetz
- Lernstrategie

Lernziele
Wichtigste Netze

Kennen der

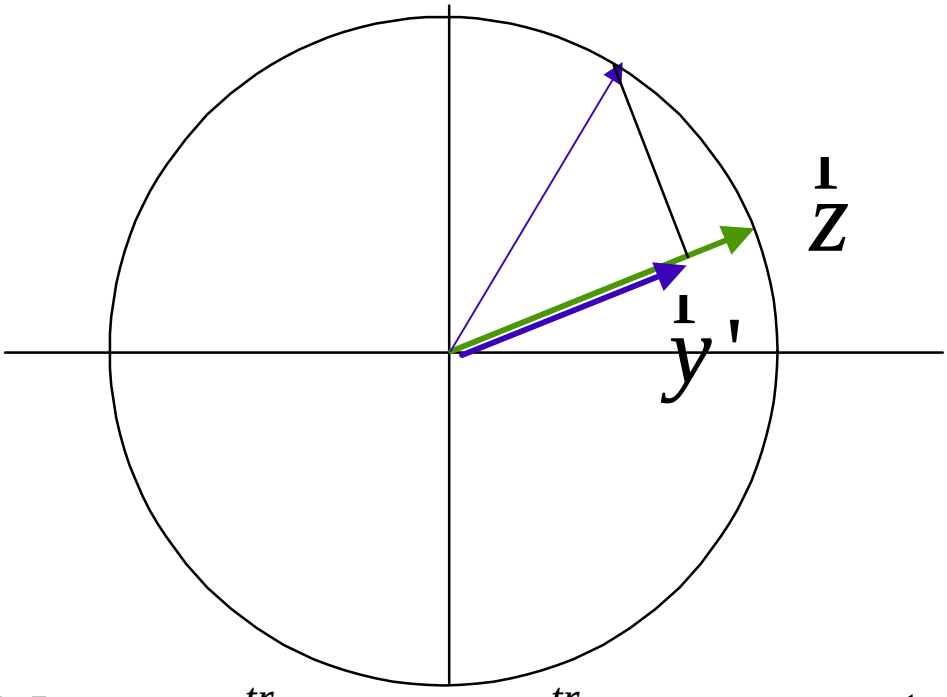
- Mathematischen Basis des Lernens
- Verstehen, was ein Gradientenabstieg ist
- Optimierung im Phasenraum, Komplementarität von genetischen Algorithmen
- Backpropagation und Kohonennetz kennen
- Idee der PCA-Analyse verstehen

Matrixspeicher
Direkte Berechnung



$$W = z\, x^{tr}$$

$$Wx = z\, x^{tr} x = z < x^{tr} x >= z$$



$$y' = Wx' = z\, x^{tr} x' = z < x^{tr} x' >= z\cos(x, x')$$

Beispiel Matrixspeicher

$$x = \begin{bmatrix} 1/\sqrt{30} \\ 2/\sqrt{30} \\ 5/\sqrt{30} \end{bmatrix} \quad z = \begin{bmatrix} 1/\sqrt{5} \\ 0 \\ 2/\sqrt{5} \end{bmatrix}$$

$$W = z\,x^{tr} = \begin{bmatrix} 1/\sqrt{5} \\ 0 \\ 2/\sqrt{5} \end{bmatrix} \begin{bmatrix} 1/\sqrt{30} & 2/\sqrt{30} & 5/\sqrt{30} \end{bmatrix} = \begin{bmatrix} 1/5\sqrt{6} & 2/5\sqrt{6} & 1/\sqrt{6} \\ 0 & 0 & 0 \\ 2/5\sqrt{6} & 4/5\sqrt{6} & 2/\sqrt{6} \end{bmatrix}$$

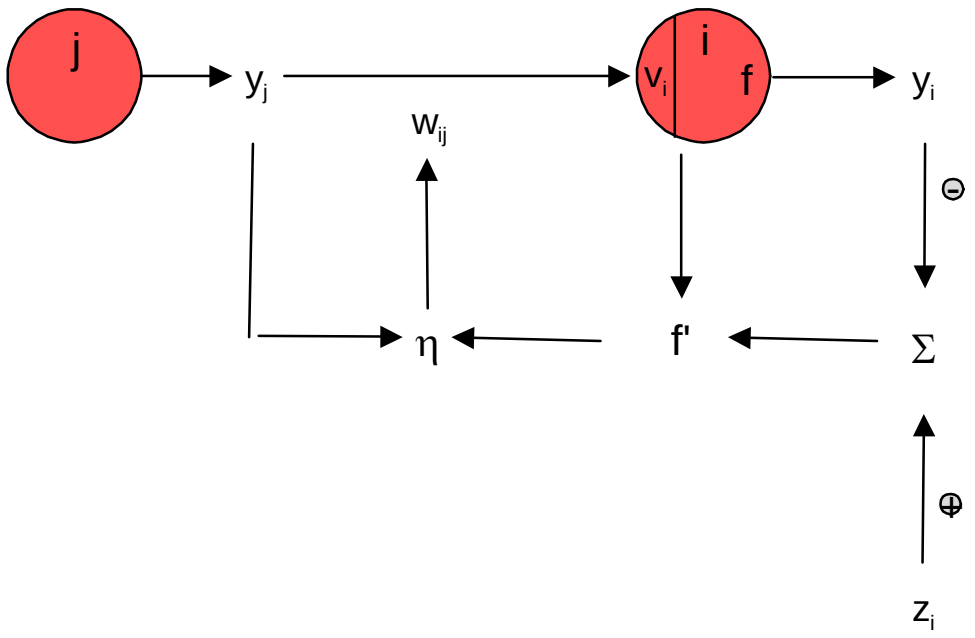
$$x' = \begin{bmatrix} 1/\sqrt{3} \\ -1/\sqrt{3} \\ 1/\sqrt{3} \end{bmatrix}$$

$$Wx = z\,x^{tr}x = \begin{bmatrix} 1/5\sqrt{6} & 2/5\sqrt{6} & 1/\sqrt{6} \\ 0 & 0 & 0 \\ 2/5\sqrt{6} & 4/5\sqrt{6} & 2/\sqrt{6} \end{bmatrix} \begin{bmatrix} 1/\sqrt{30} \\ 2/\sqrt{30} \\ 5/\sqrt{30} \end{bmatrix} = \begin{bmatrix} 1/\sqrt{5} \\ 0 \\ 2/\sqrt{5} \end{bmatrix}$$

Lernen mit Lehrer

Fehlerkorrigierendes Lernen

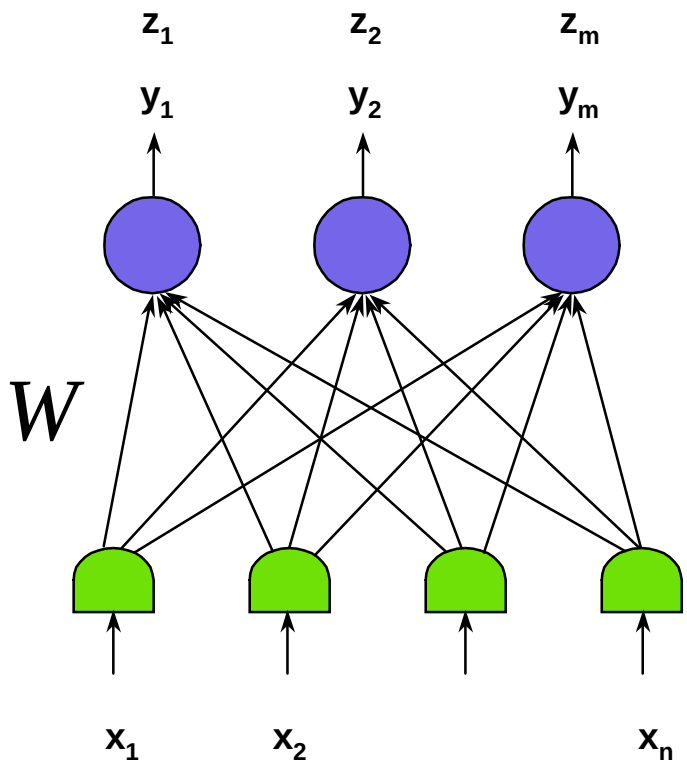
$$\Delta w_{ij} = \eta (z_i - y_i) f'(v_i) y_j$$



Perceptron

Einschichtiges Netzwerk

Perceptron Neuronen (binär, bipolar)



Zielmuster

Netzausgabe

Bias:

$$W' = \begin{bmatrix} \theta & W \end{bmatrix}$$

$$x' = \begin{bmatrix} -1 \\ x \end{bmatrix}$$

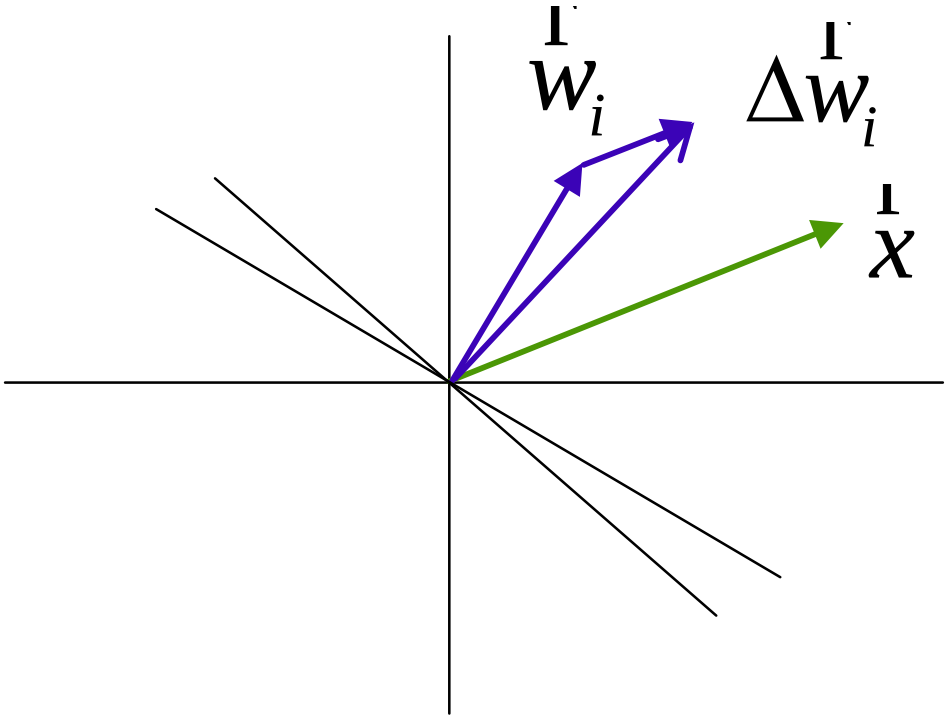
Eingabemuster

$$y = sig[Wx - \theta] = sig[W'x']$$

Deltalernregel:

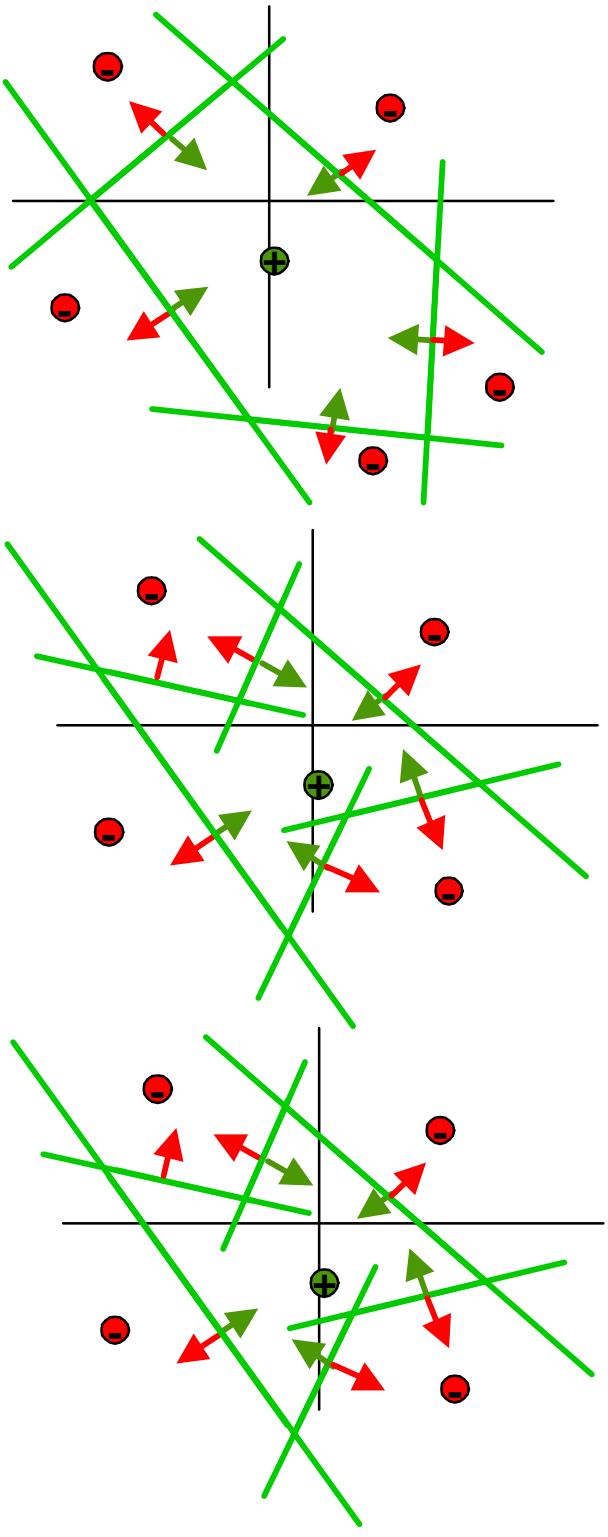
$$\Delta w_{ij} = \eta x_j (z_i - y_i)$$

Geometrische Interpretation des Lernens



$$\Delta\vec{w}_i = \eta(z_i - y_i)\vec{x}$$

Entscheidungsregionen



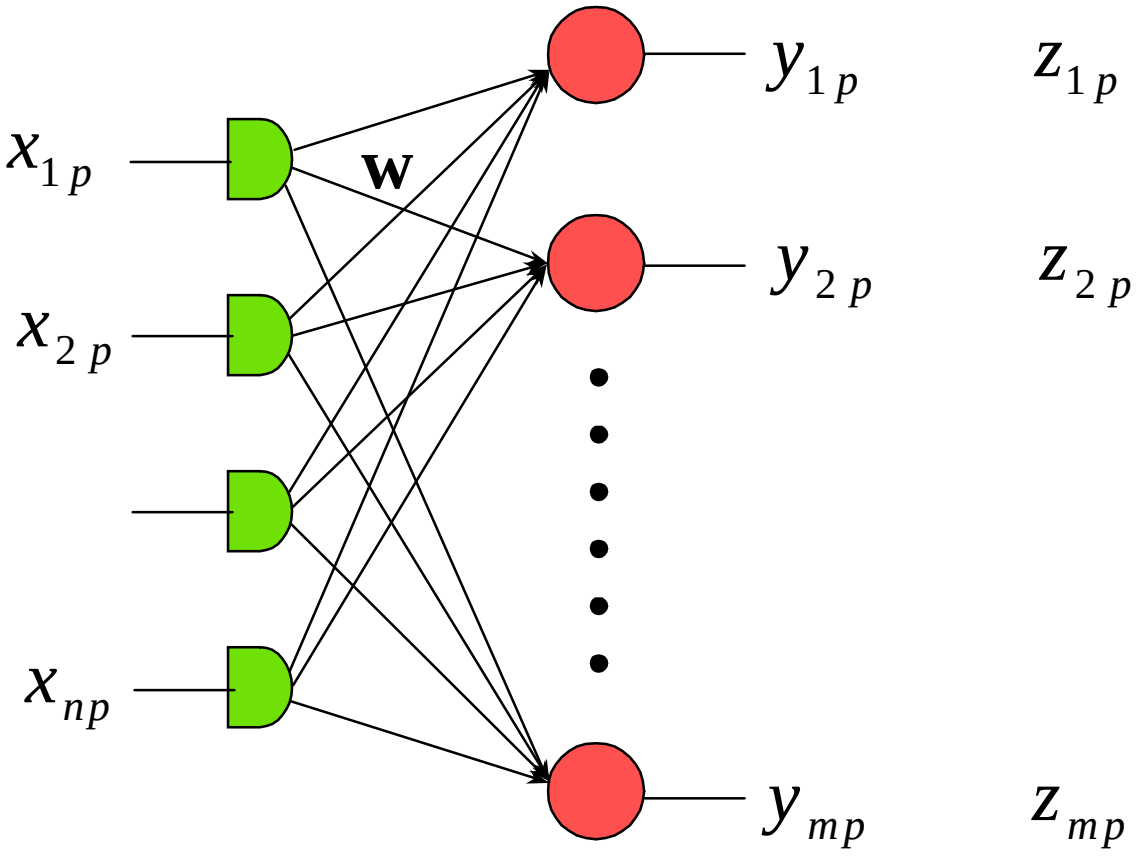
Ausgabe und Vorgabe

Minimierung des Fehlers am Ausgang

$$E(\mathbf{w}) = \frac{1}{2}\sum_{p=1}^{P}(\mathbf{z}_p - \mathbf{y}_p)^2$$

$$= \frac{1}{2}\sum_{p=1}^{P}(\mathbf{z}_p - \mathbf{f}(v)_p)^2$$

$$= \frac{1}{2}\sum_{p=1}^{P}(\mathbf{z}_p - \mathbf{f}(\mathbf{w}\,\mathbf{x}_p))^2$$



Fehlerfunktion

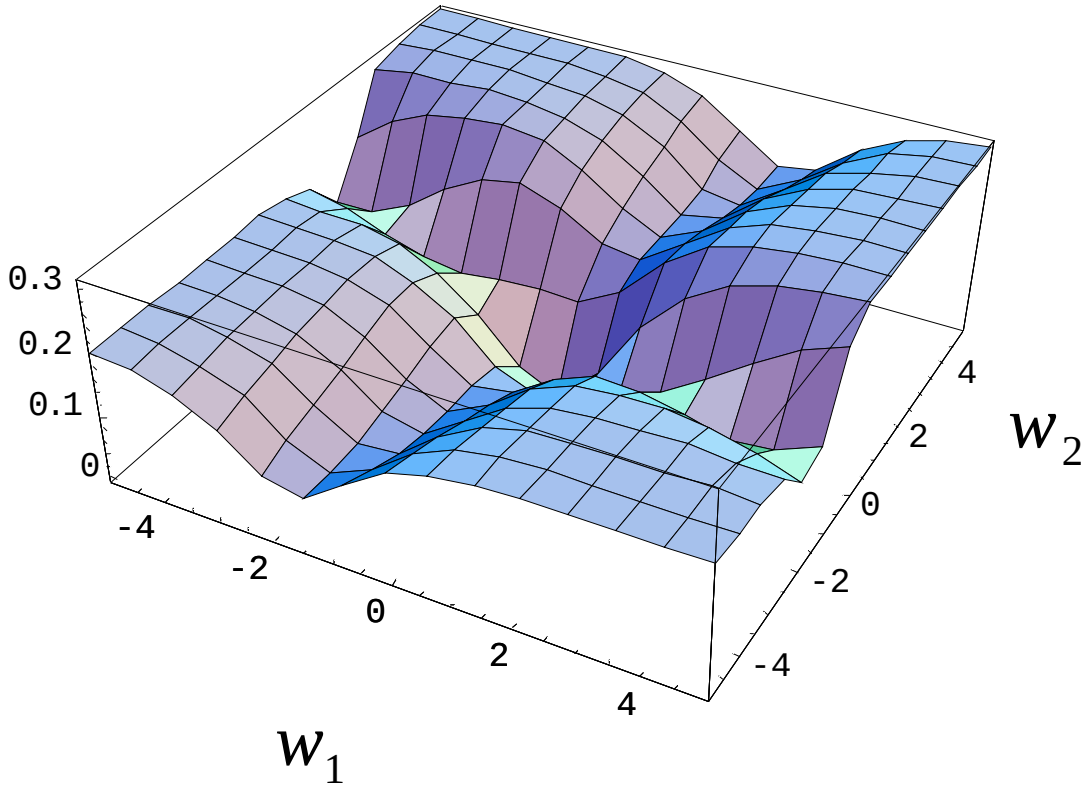
Beispiel: Netz mit 2 Neuronen am Ausgang und mit 2 Eingängen; 2 Beispiele

$$f(x) := \frac{1}{1 + e^{-x}}$$

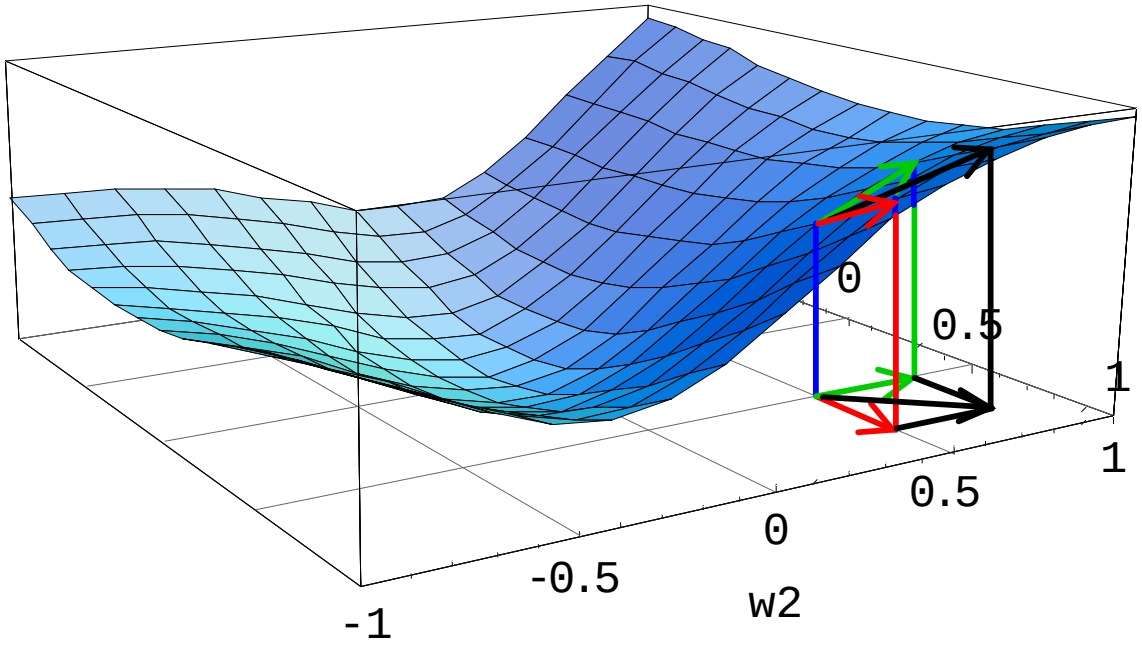
$$x = \begin{bmatrix} 1.5 & -1 \\ -0.5 & -3 \end{bmatrix}$$

$$z = \begin{bmatrix} 0.5 & 0.6 \end{bmatrix}$$

$$E(w) = E(w_1, w_2) = \frac{1}{2} \sum_{p=1}^{2} (\mathbf{z}_p - \mathbf{f}(\mathbf{w}\,\mathbf{x}_p))^2$$



Gradient



$$\nabla E(w) = \left(\frac{\partial E(w)}{\partial w_{ij}} \right) = \left(\frac{\partial \left(\frac{1}{2} \sum_{p=1}^{P} (\mathbf{z}_p - \mathbf{f}(\mathbf{w}\,\mathbf{x}_p))^2 \right)}{\partial w_{ij}} \right)$$

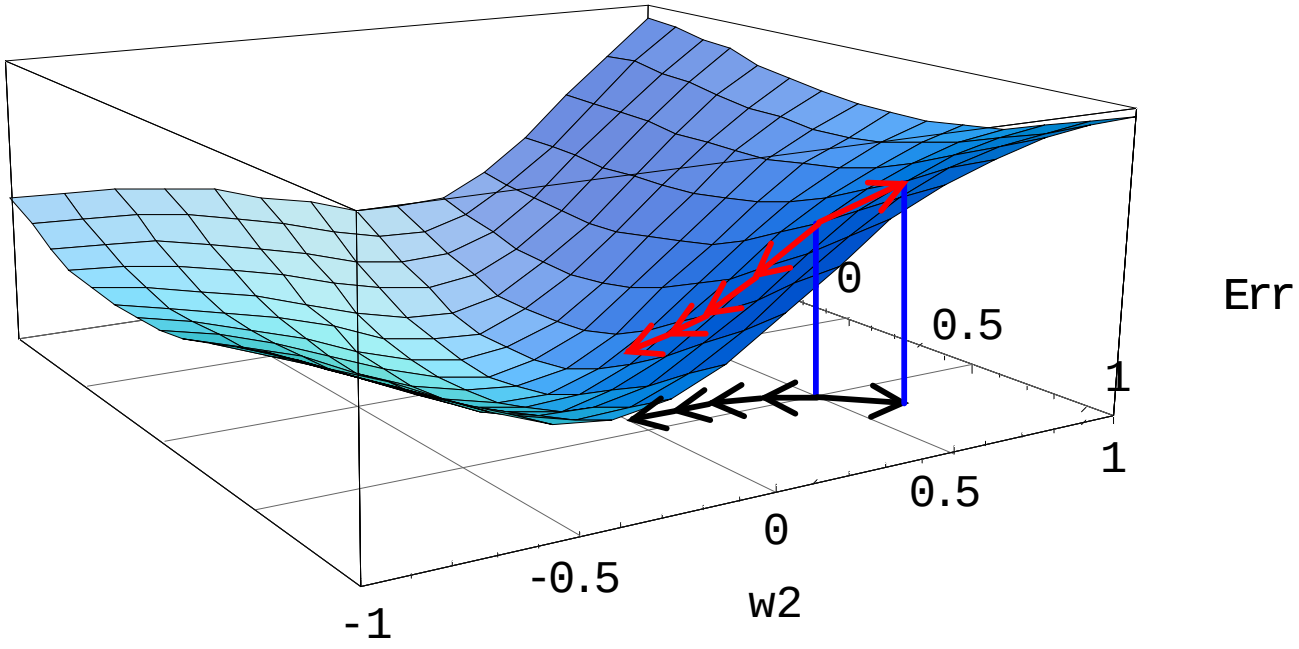
$$= \left(\sum_{p=1}^{P} \frac{\partial E}{\partial y_{ip}} \frac{\partial y_{ip}}{\partial v_{ip}} \frac{\partial v_{ip}}{\partial w_{ij}} \right)$$

$$= \left(-\sum_{p=1}^{P} (z_{ip} - f(v_{ip})) f'(v_{ip}) x_{jp} \right)$$

$$= \left(-\sum_{p=1}^{P} (z_{ip} - y_{ip}) \underbrace{y_{ip}(1 - y_{ip})}_{f'} x_{jp} \right)$$

Gradientenabstieg

Gradient: Richtung des steilsten Anstiegs auf der Fläche E(w)



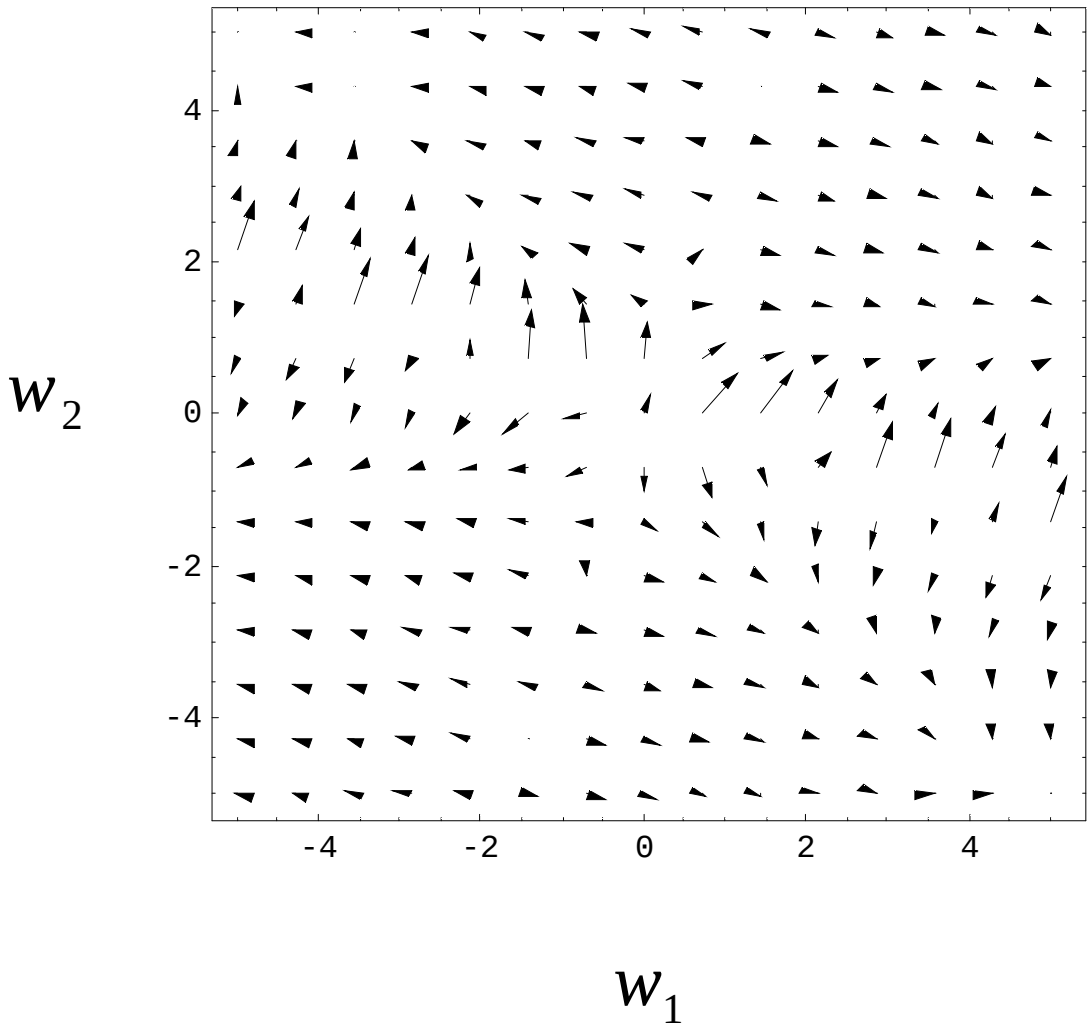
$$\Delta w(t) = -\eta \nabla E(w(t))$$

$$= \eta \sum_{p=1}^{P} (\mathbf{z}_p - \mathbf{y}_p) \underbrace{y_p(1-y_p)}_{\mathbf{f}'} \mathbf{x}_p$$

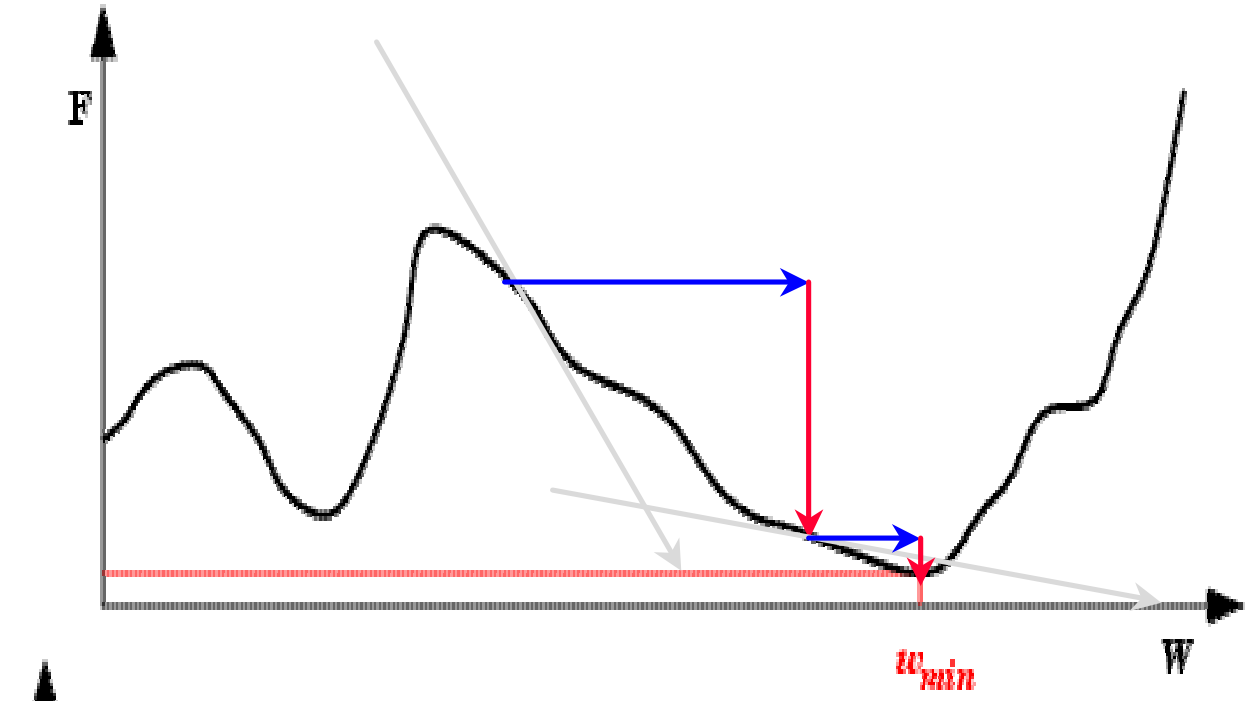
Berechnung des Gradienten an der aktuellen Stelle im Gewichtsraum w(t)

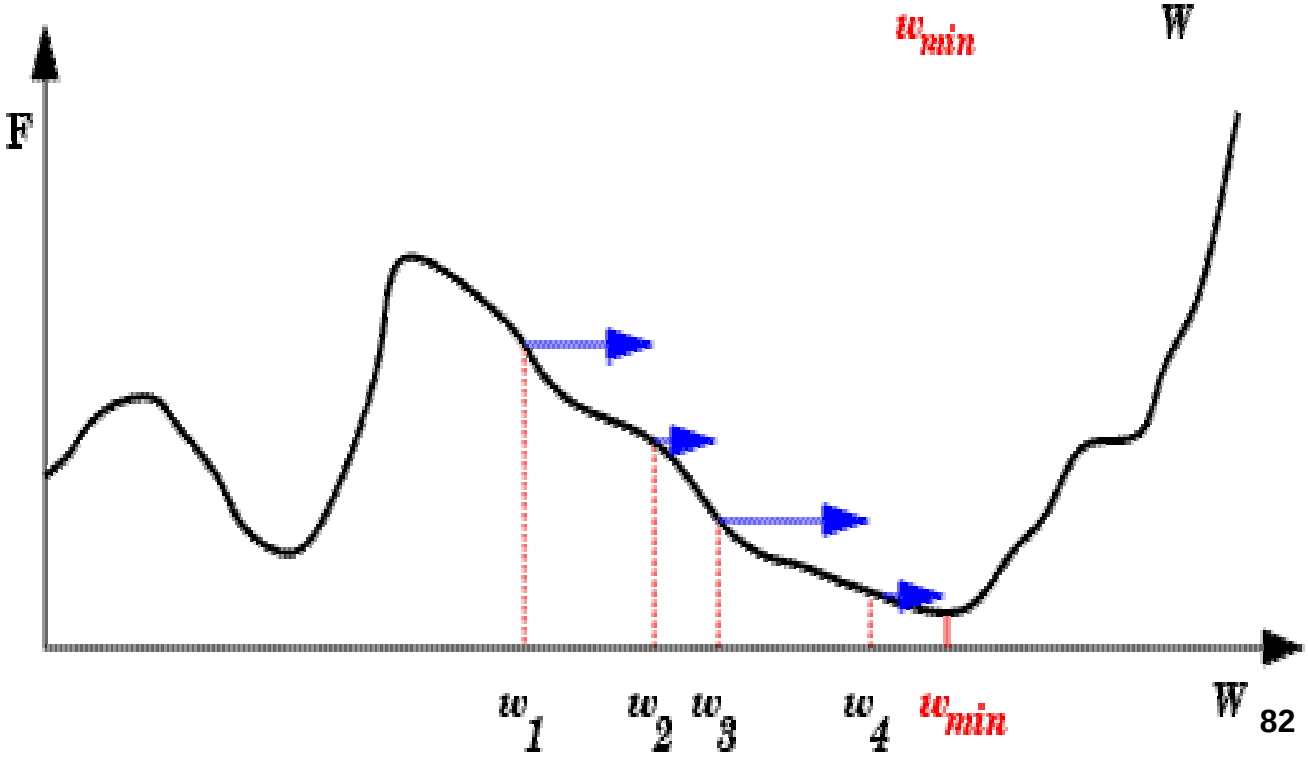
Gewichtskorrektur = schrittweiser Abstieg entlang des negativen Gradienten bis zu einem Minimum in der Fehlerfunktion E(w)

Gradientenfeld im Gewichtsraum

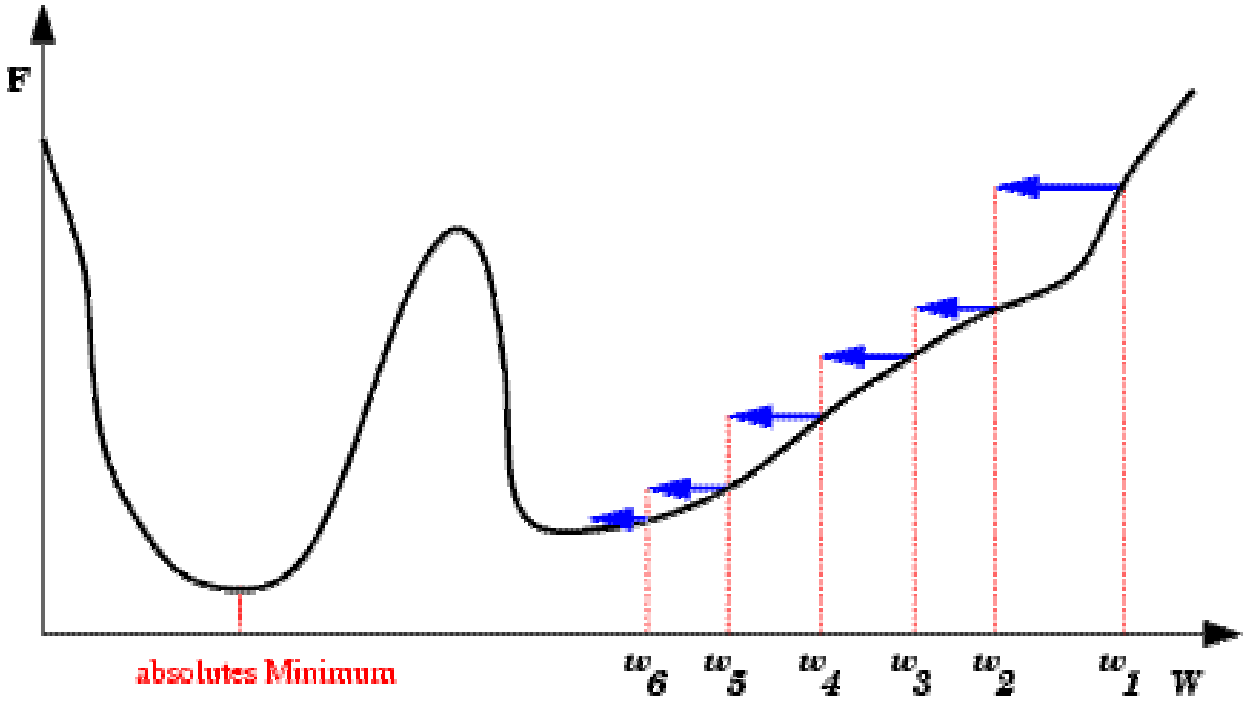


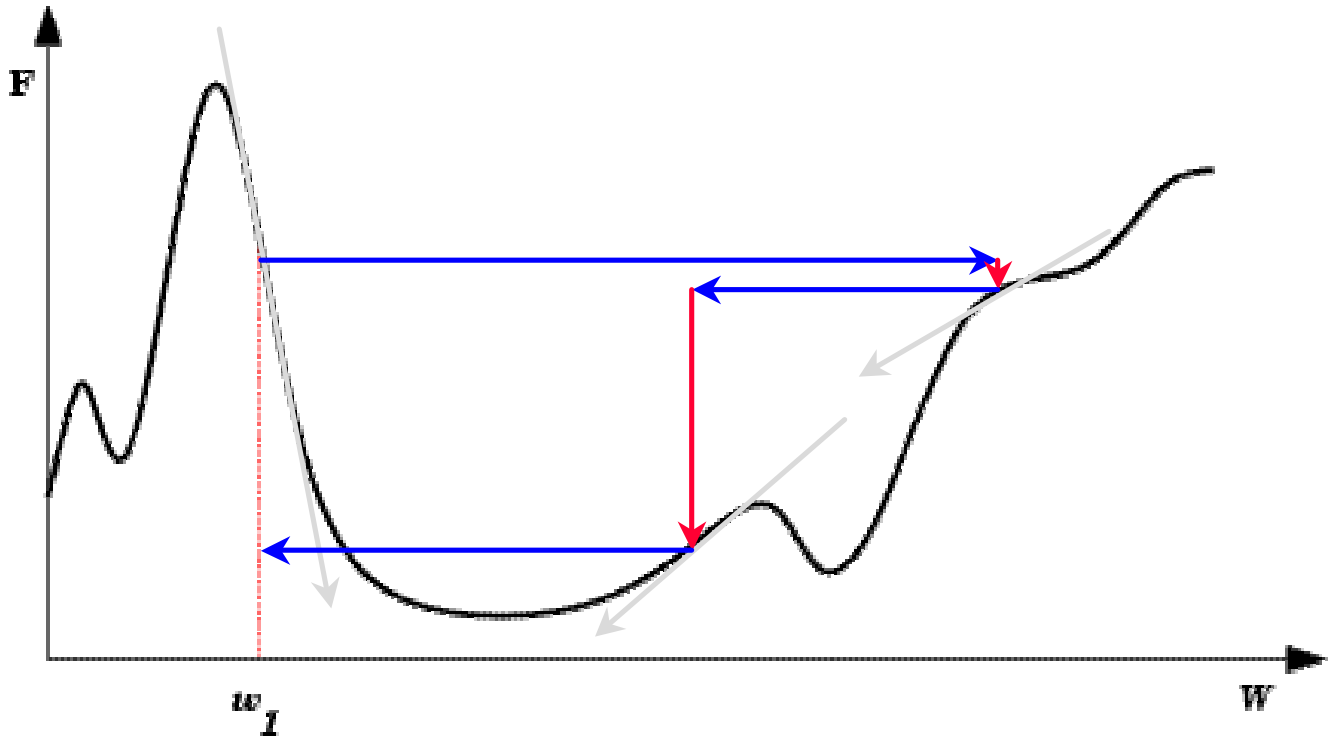
Fehlerkorrektur



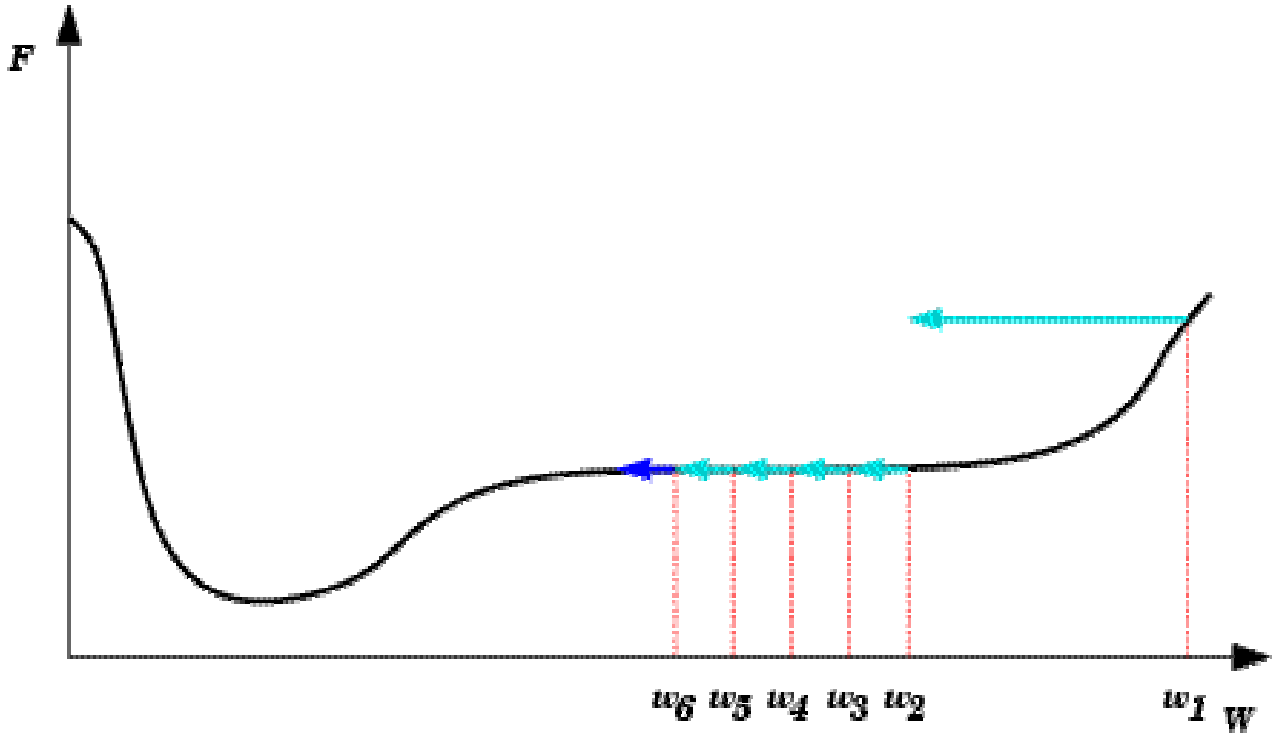


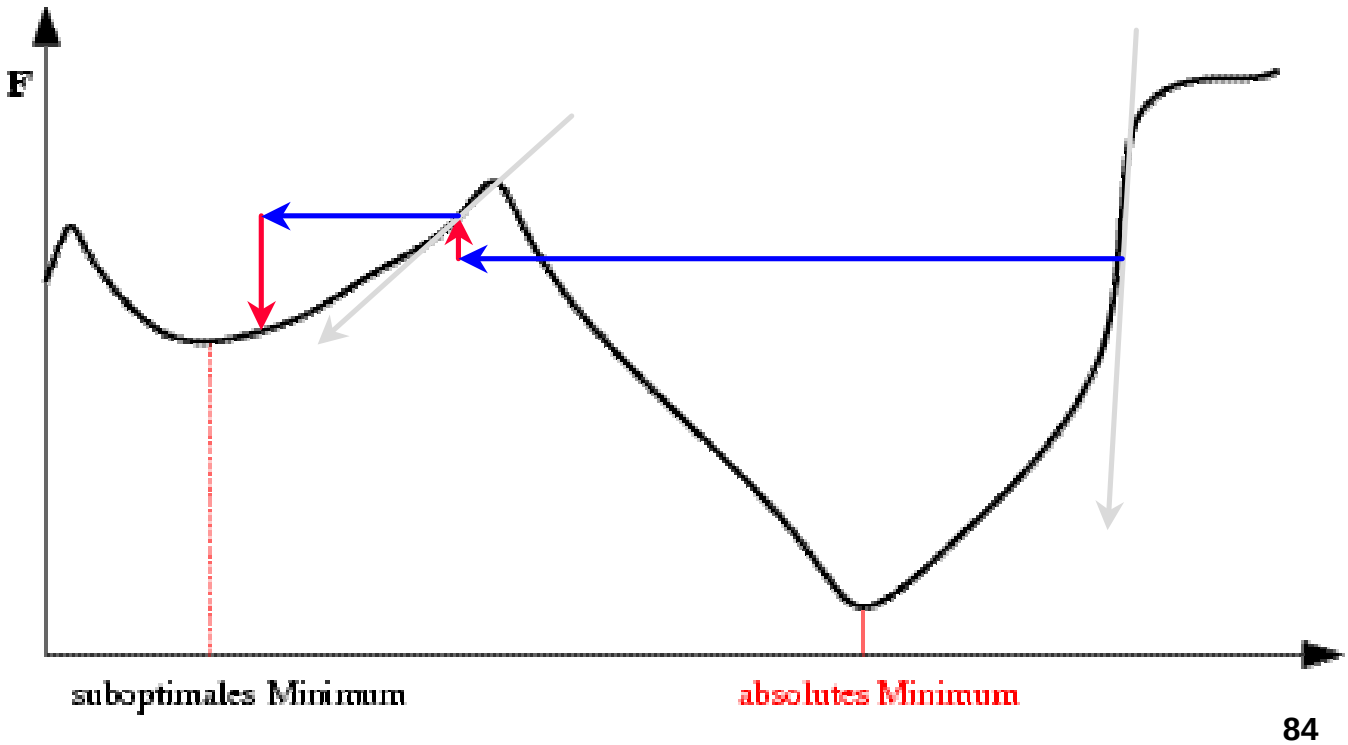
Problem der lokalen Optimierung



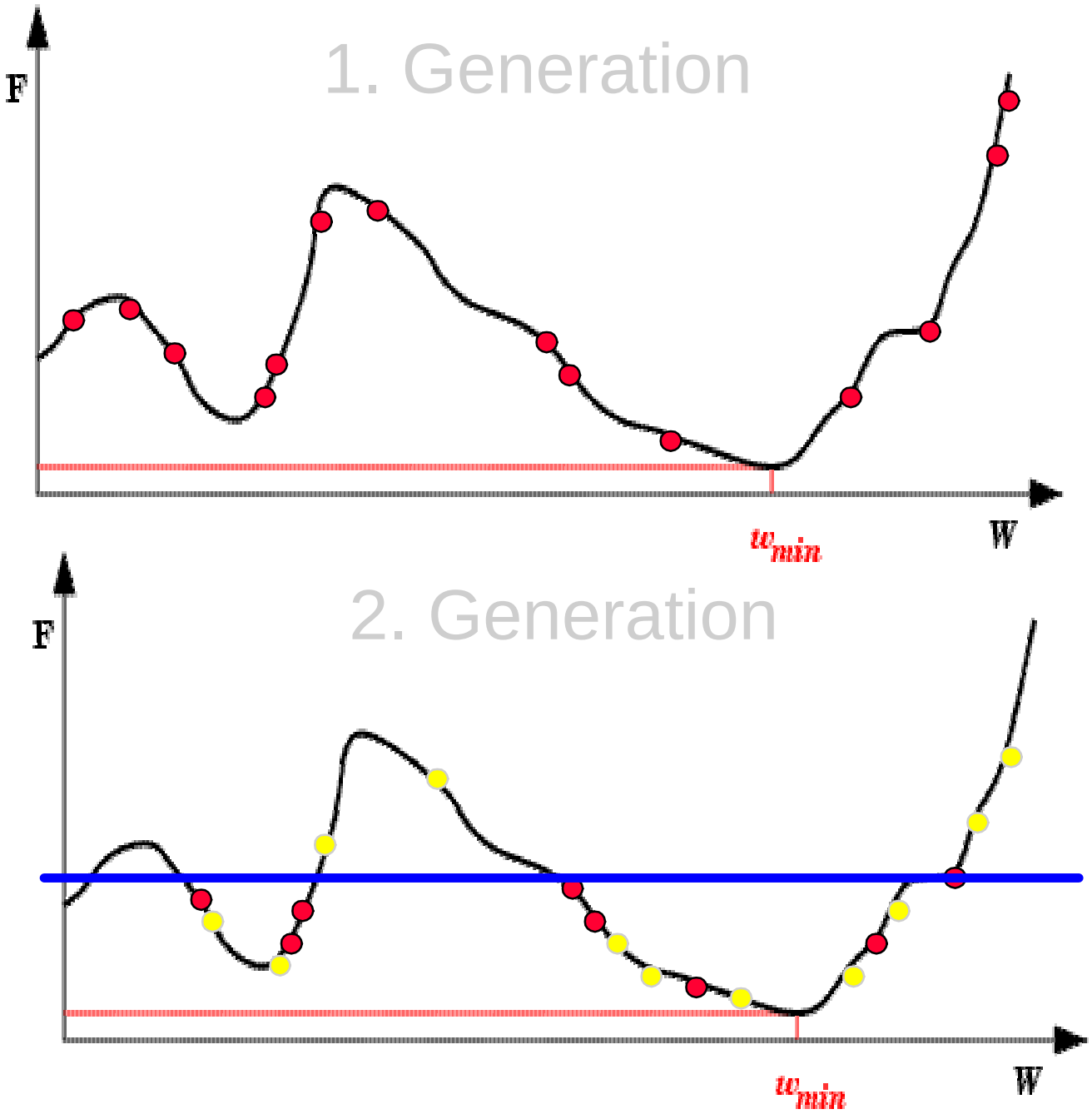


Auswirkung der Lernrate

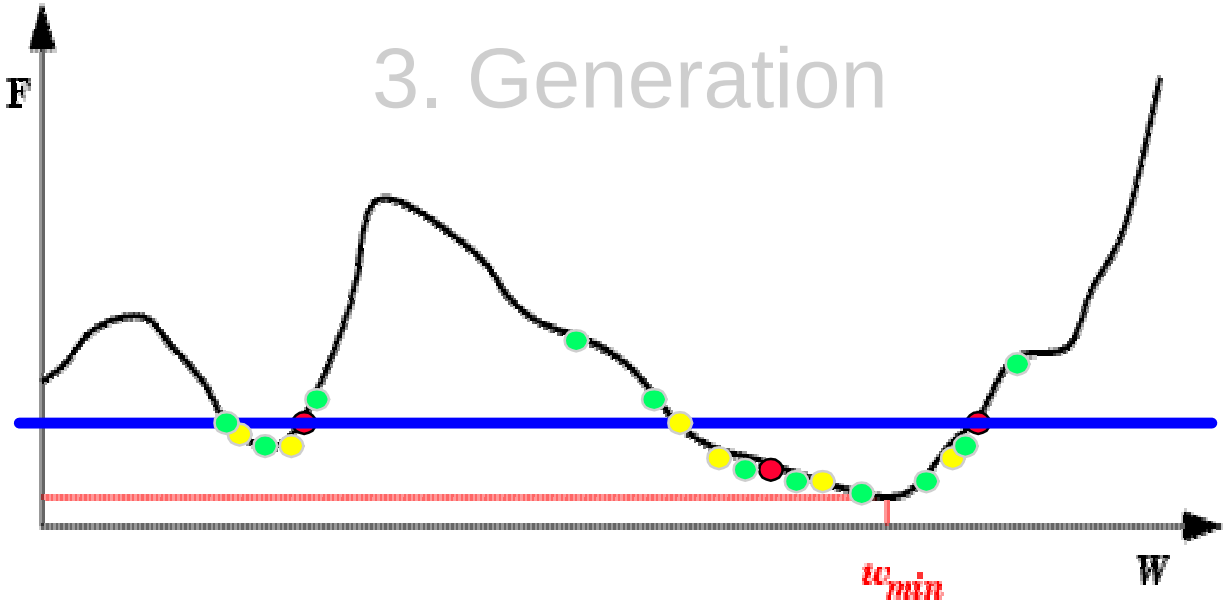


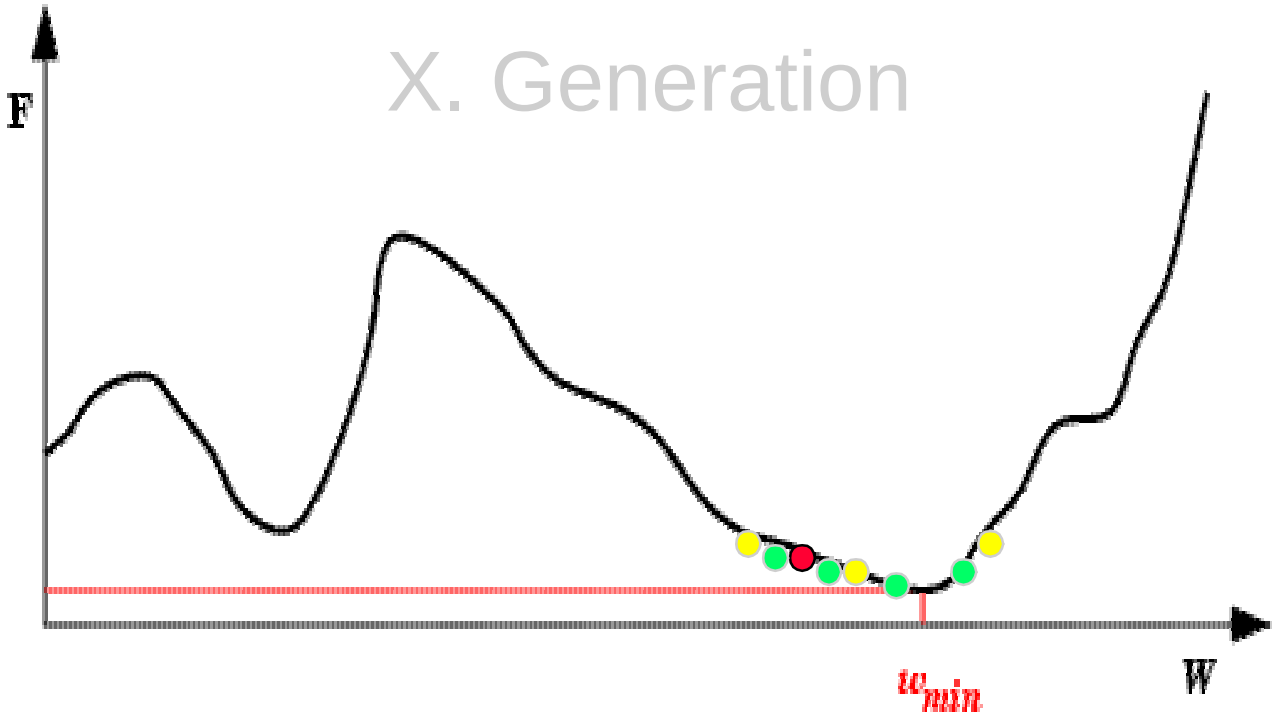


Kombination mit genetischen Algorithmen



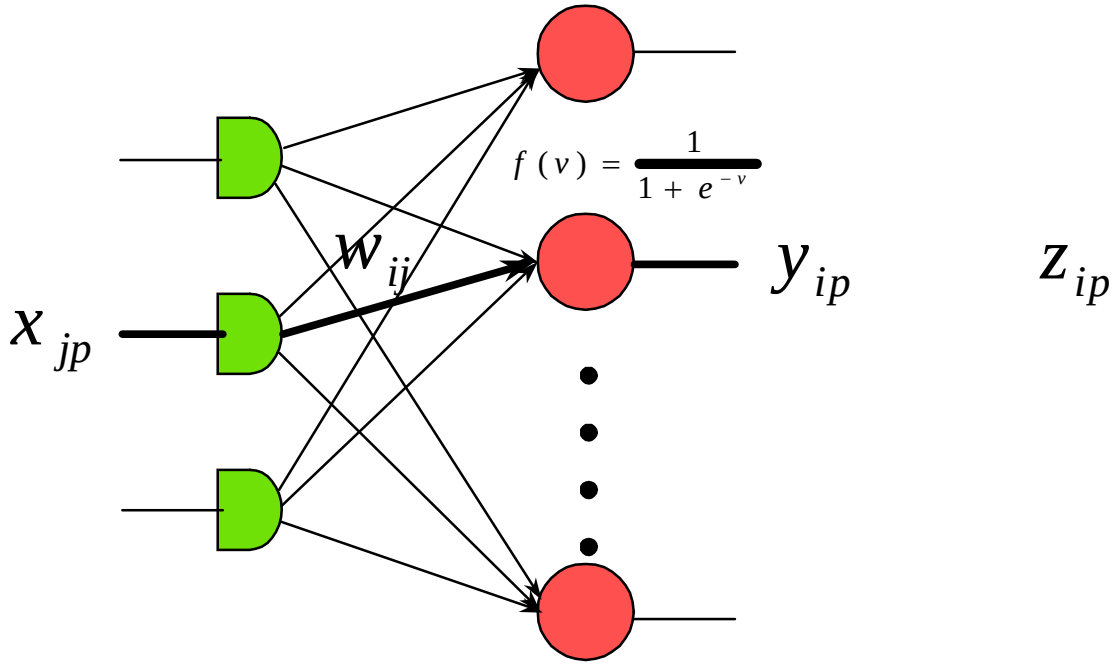
Kombination mit genetischen Algorithmen





Deltaregel musterweises Lernen

Musterweises Lernen: Lernschritt nach jedem präsentierten Beispiel



$$\Delta_p w_{ij} = \eta (z_{ip} - f(v_{ip})) f'(v_{ip}) x_{jp}$$

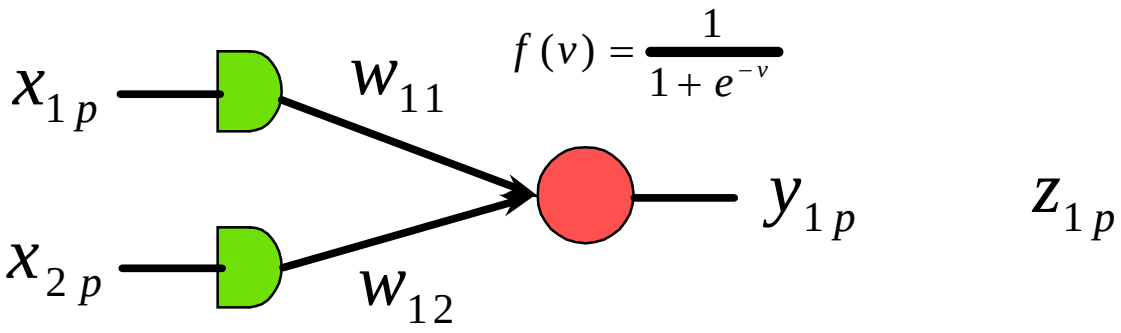
$$= \eta (z_{ip} - y_{ip}) \underbrace{y_{ip}(1 - y_{ip})}_{f'} x_{jp}$$

$$= \eta \underbrace{\delta_{ip}}_{\text{Deltafehler}} x_{jp}$$

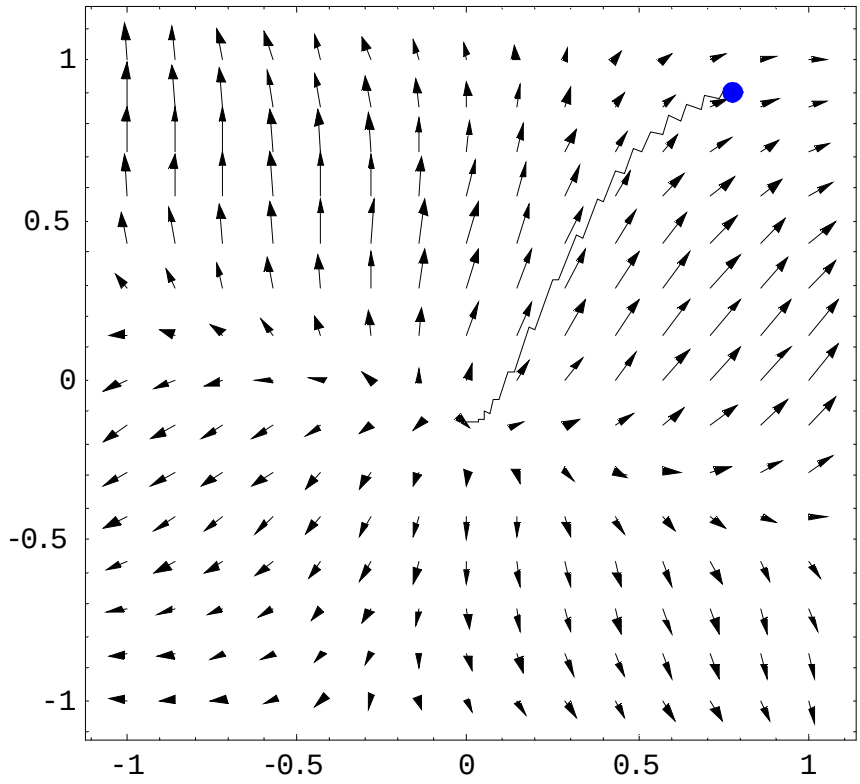
Beispiel

2 Eingänge, 1 Ausgang, 2 Musterbeispiele zum Lernen

$$x = \begin{bmatrix} 1.5 & -1 \\ -0.5 & -3 \end{bmatrix} \qquad z = \begin{bmatrix} 0.5 & 0.6 \end{bmatrix}$$

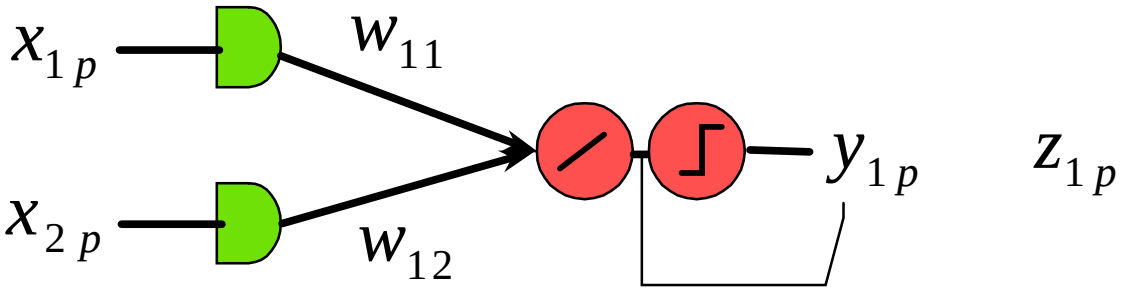


eta = 0.8 W(t=0) = {0.8, 0.9}



Spezialfall Lineares Neuron

$$f(v) = v;\ f'(v) = 1$$



$$\Delta_p w_{ij} = \eta\ (z_{ip} - f(v_{ip}))\ f'(v_{ip})\ x_{jp}$$

$$= \eta\ (z_{ip} - \mathbf{w}_{i.}\ \mathbf{x}_p)\ x_{jp}$$

Beispiel: Adaline (linear bzw. Signumfunktion)

$$\mathbf{w} = [1, 3]$$

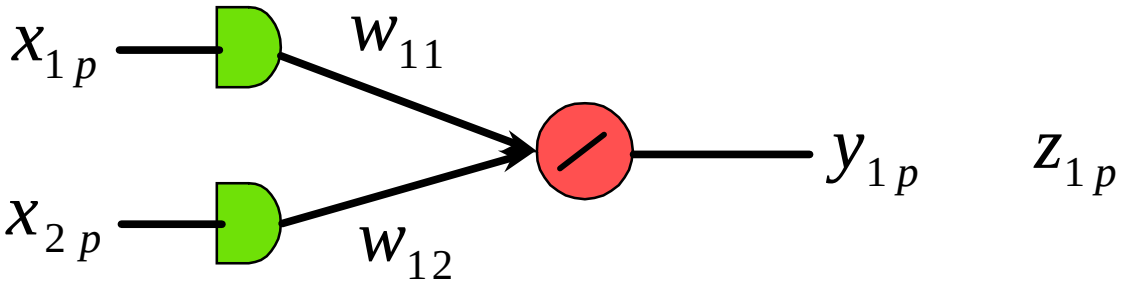
$$\mathbf{x}_1 = \begin{bmatrix} 0 \\ -1 \end{bmatrix};\quad \mathbf{x}_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix};\quad \mathbf{x}_3 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$z_1 = 1;\qquad z_2 = 1;\qquad z_3 = 1;$$

$$\eta = 0.5$$

Beispiel: lineares Neuron

$$\Delta_p w_{ij} = \eta \, (z_{ip} - \mathbf{w}_{i.} \, \mathbf{x}_p) \, x_{jp}$$



Musterweises Lernen mit obigem Beispiel

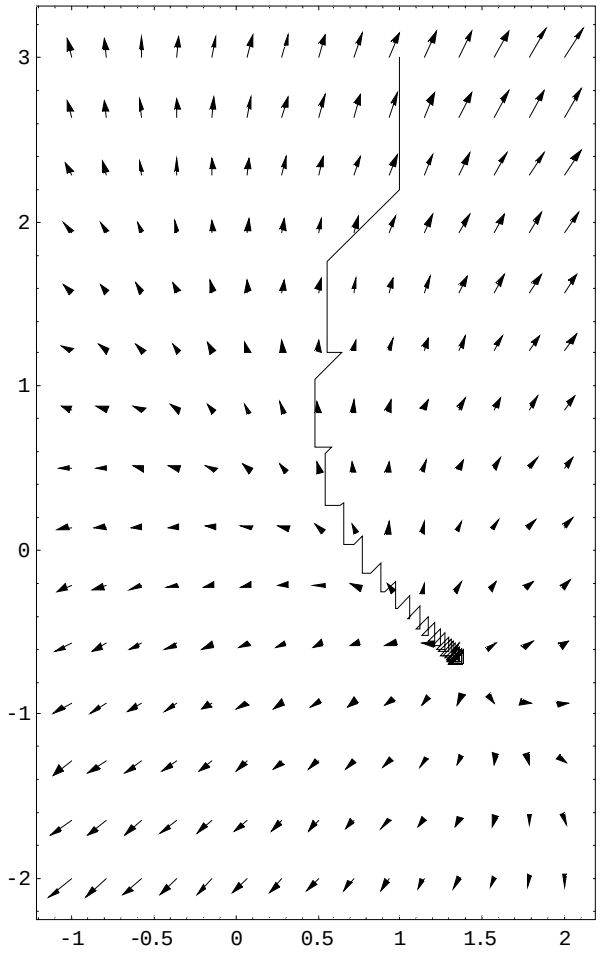
$$\mathbf{w} = [1, 3]$$

$$\mathbf{x}_1 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}; \; z_1 = 1;$$

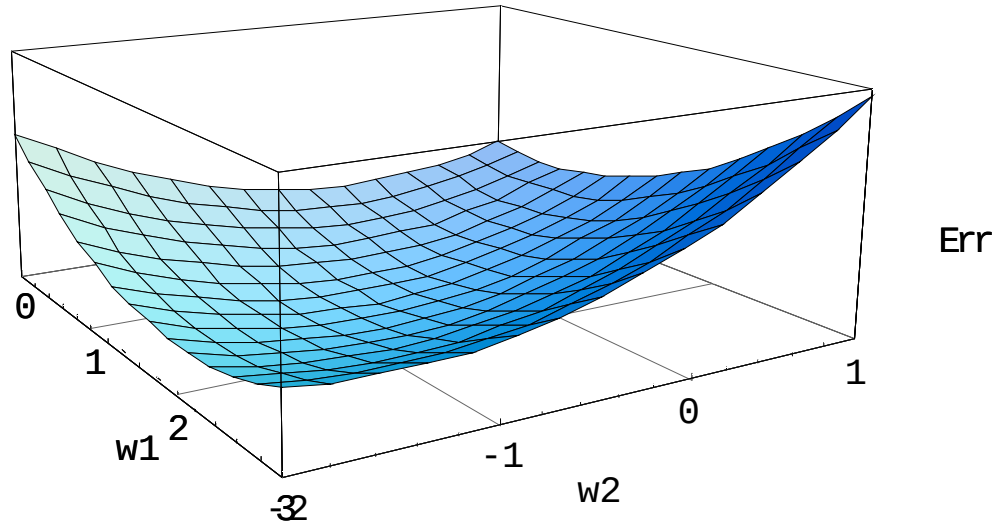
$$\mathbf{x}_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}; \; z_2 = 1;$$

$$\mathbf{x}_3 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}; \; z_3 = 1;$$

$$\eta = 0.2$$



Direkte Berechnung aus Fehlerfunktion



$$E(W) = \|Z - WX\|^2 = \min; \quad \|W\|^2 = \min$$

$$(= Sp\left[(Z - WX)(Z - WX)^{tr} + \alpha WW^{tr}\right] = \min)$$

$$\nabla_W E(W) = -2(Z - WX)X^{tr} + 2\alpha W$$

$$= 2W\underbrace{(XX^{tr} + \alpha I)}_{regulär} - ZX^{tr}$$

$$= 0$$

$$0 = W(XX^{tr} + \alpha I) - ZX^{tr}$$

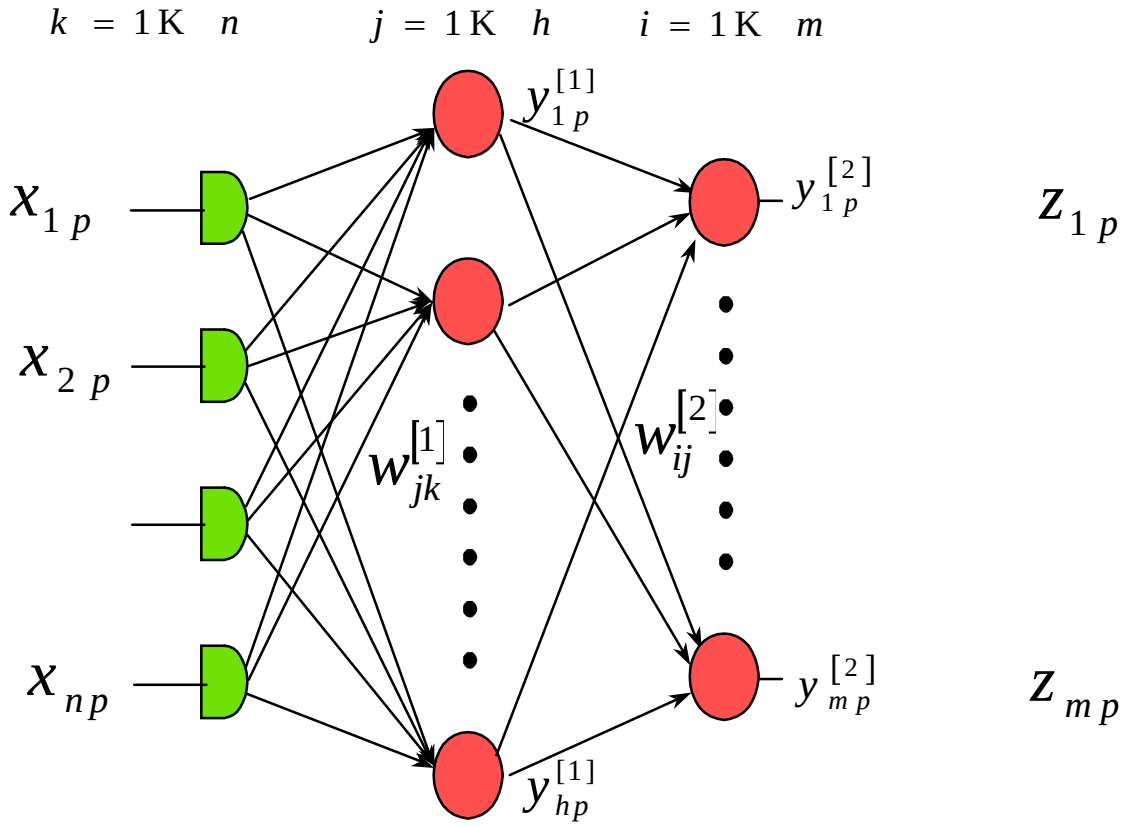
$$W = ZX^{tr}\underbrace{(XX^{tr} + \alpha I)^{-1}}_{existiert}$$

$$= ZX^{+}$$

$$X^{+} = \lim_{\alpha \to 0} X^{tr}(XX^{tr} + \alpha I)^{-1}$$

Pseudo-Inverse
Moore-Penrose Inverse

Backpropagation Algorithmus



Minimierung des Fehlers am Ausgang

$$E(W^{[1]}, W^{[2]}) = \frac{1}{2}\sum_{p=1}^{P}(\mathbf{z}_p - \mathbf{y}_p^{[2]})^2 = \frac{1}{2}\sum_{p=1}^{P}\sum_{m}(z_{ip} - \mathbf{f}(v_{ip}^{[2]}))^2$$

$$= \frac{1}{2}\sum_{p=1}^{P}\sum_{m}(z_{ip} - f(\sum_{h} w_{ij}^{[2]} y_{jp}^{[1]}))^2$$

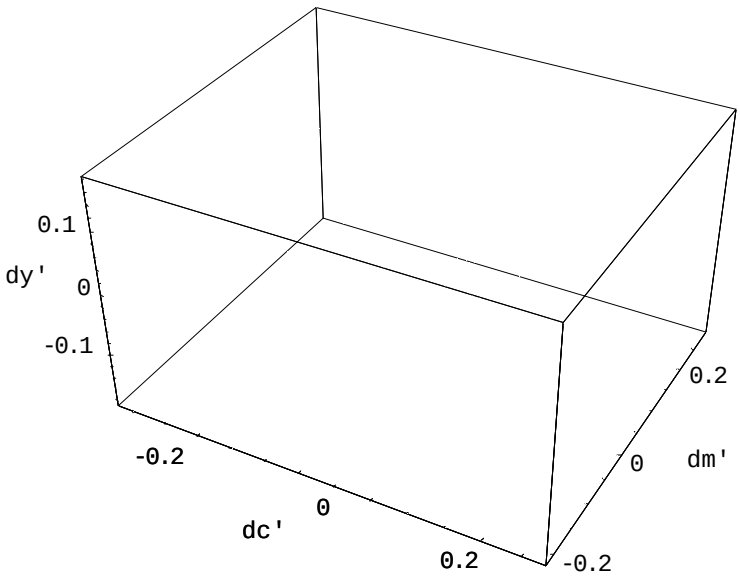
$$= \frac{1}{2}\sum_{p=1}^{P}\sum_{m}(z_{ip} - f(\sum_{h} w_{ij}^{[2]} f(v_{jp}^{[1]}))^2$$

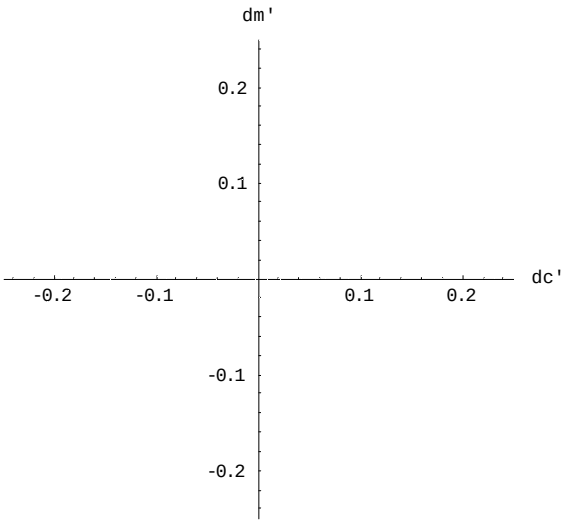
$$= \frac{1}{2}\sum_{p=1}^{P}\sum_{m}(z_{ip} - f(\sum_{h} w_{ij}^{[2]} f(\sum_{k} w_{jk}^{[1]} x_{kp})))^2$$

Principle Component Analysis (PCA)

p-Datenpunkte mit je n-Merkmalen

- **X(roh) n x p Matrix der Datenpunkte**
- **Bestimme X durch zentrieren von X(roh):**
  x-> x-<x> für alle n Merkmale
- **Dispersionsmatrix T = XX^{tr}** ;
  symmetrische n x n Matrix; Rang(T) <= n
- **V Eigensystem von T ;**
  geordnet nach Grösse der Eigenwerte
  (normierte Eigenvektoren als Kolonnen)
- **Basistransformation: X‘ = V^{tr} X**
  X darstellen in neuer Basis der Eigenvektoren



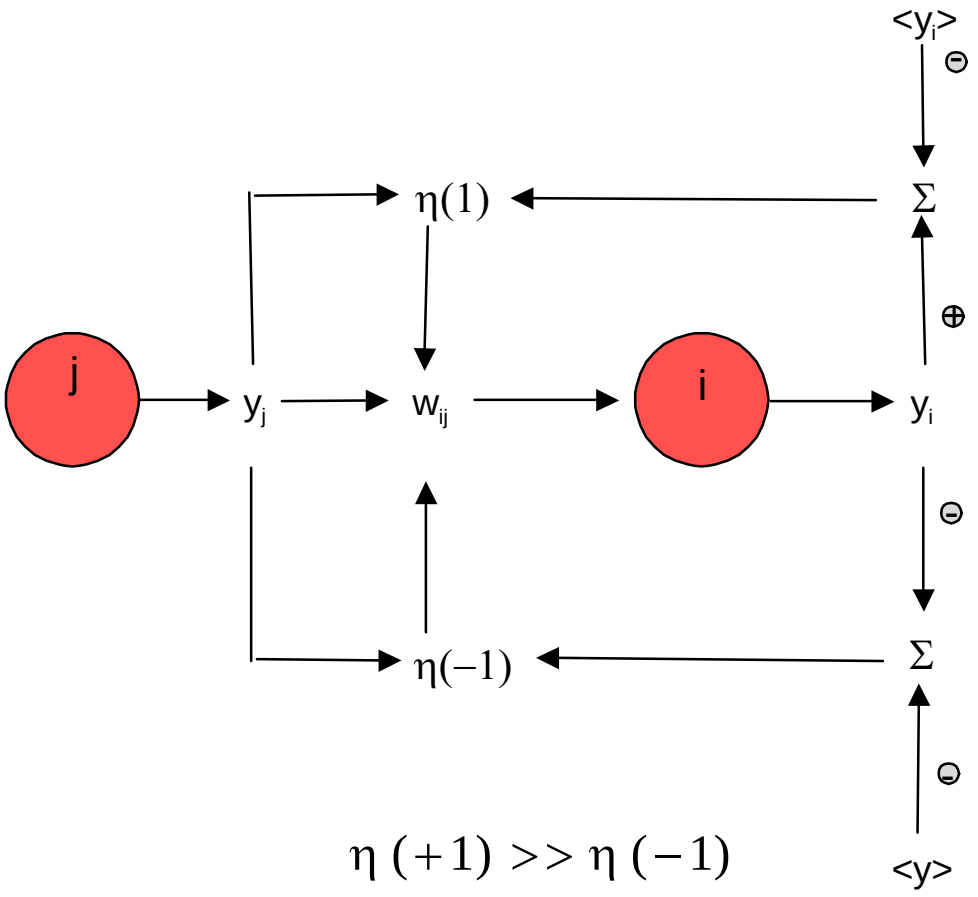


Lernen mit Kritiker

Verstärkungslernen

– Kritik: r(<y>)=+1 oder
r(<y>)=-1

$$\Delta w_{ij} = \begin{cases} \eta\,(r=+1)\,y_j\,(y_i - < y_i >) \\ \eta\,(r=-1)\,y_j\,(-y_i - < y_i >) \end{cases}$$

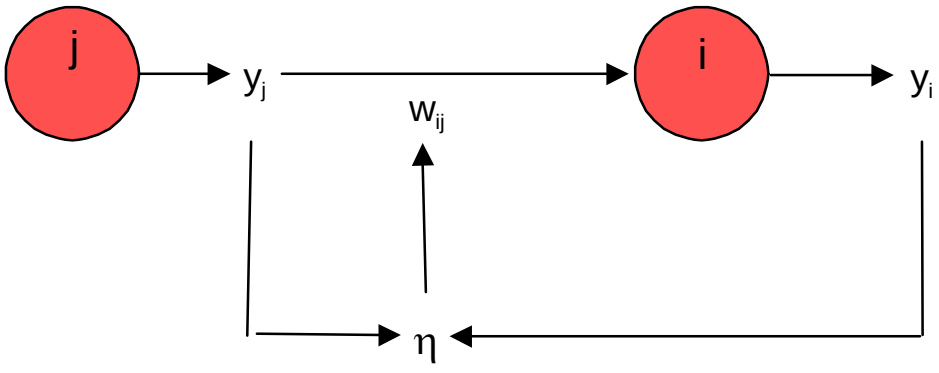


Hebbsches Lernen

Neurophysiologische Lernregel

– synaptische Verstärkung (Korrelation)

$$\Delta w_{ij} = \eta \, y_j \, y_i$$



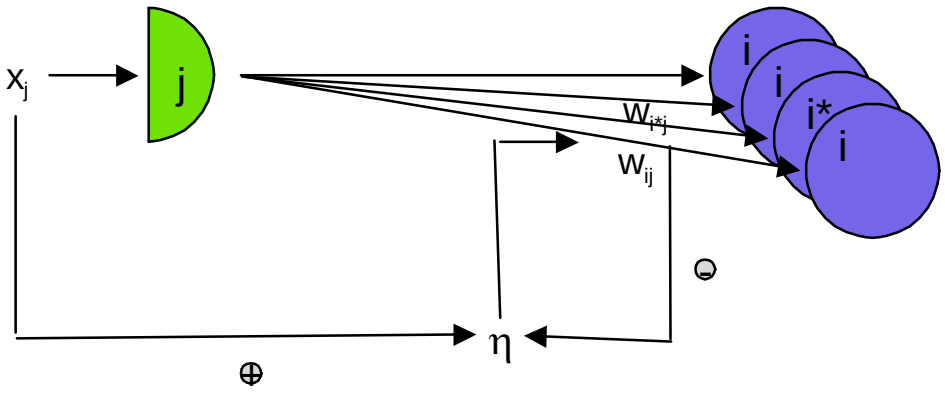
– Binäre Neuronen: stärkend
– Bipolare Neuronen: stärkend, dämpfend

Kompetitives Lernen

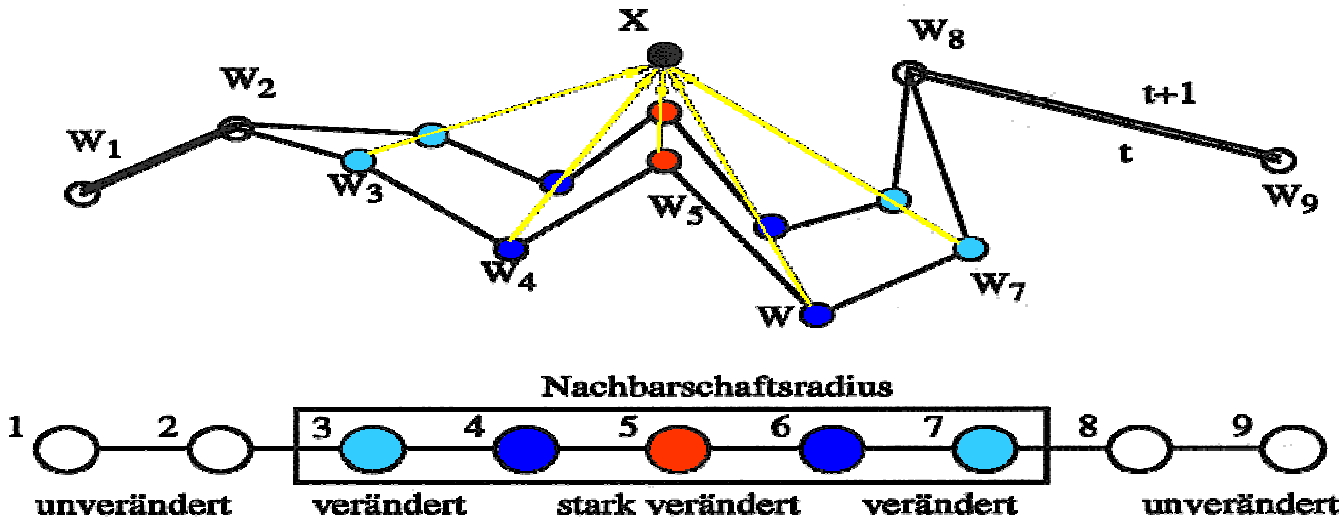
Topologische Lernregel

- PEs haben räumlich Anordnung (z.B. Gitterebene)
- Lernen in Umgebung N des Gewinners i*

$$\Delta w_{ij} = \begin{cases} \eta(i^*, i)(x_j - w_{ij}) & i \in N(i^*) \\ 0 & sonst \end{cases}$$



Kohonen Algorithmus



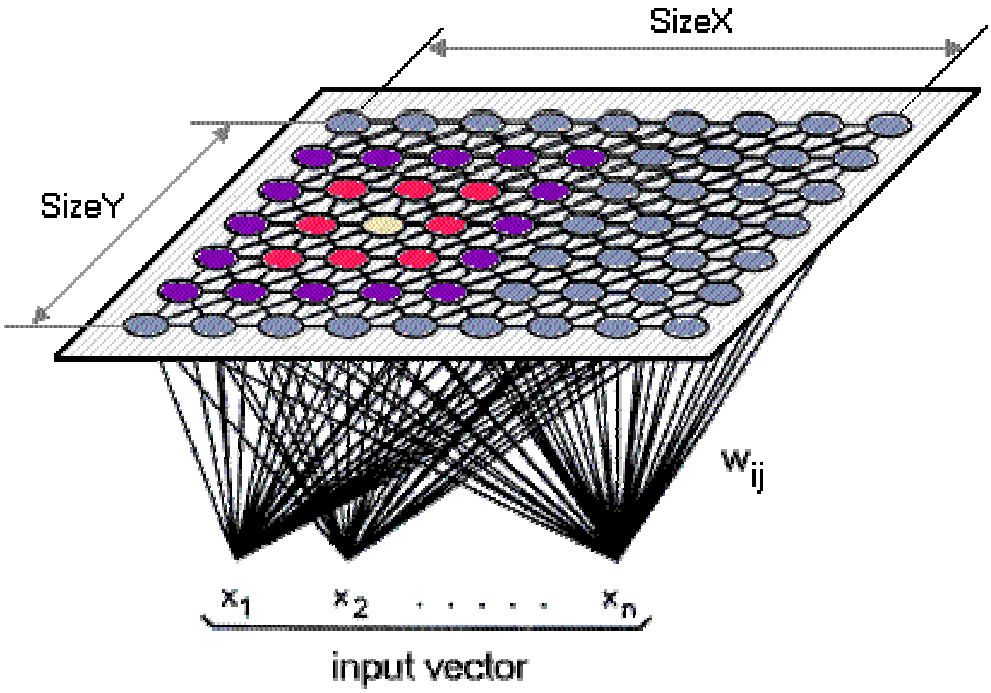
Der Gewichtsvektor W5 von Neuron 5 ist dem Eingabevektor X am nächsten, W5 wird am stärksten zu X hingezogen

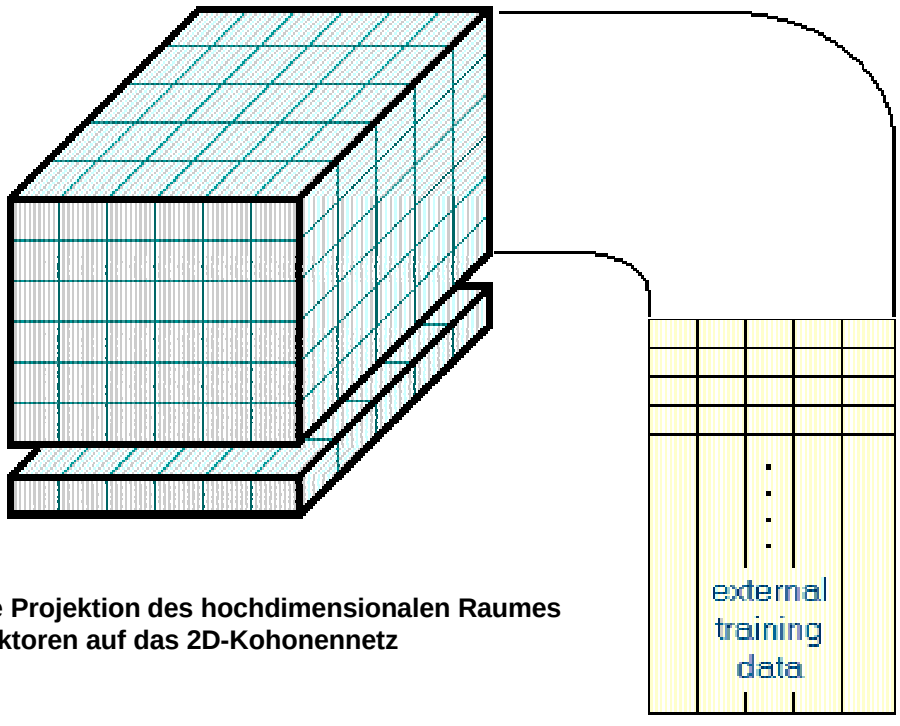
die Gewichtsvektoren W4 und W6 am zweitstärksten

W3 und W7 werden ebenfalls noch berücksichtigt.

Die Neuronen 1, 2, 8 und 9 liegen ausserhalb dieses Nachbarschaftsradius und werden nicht verändert.

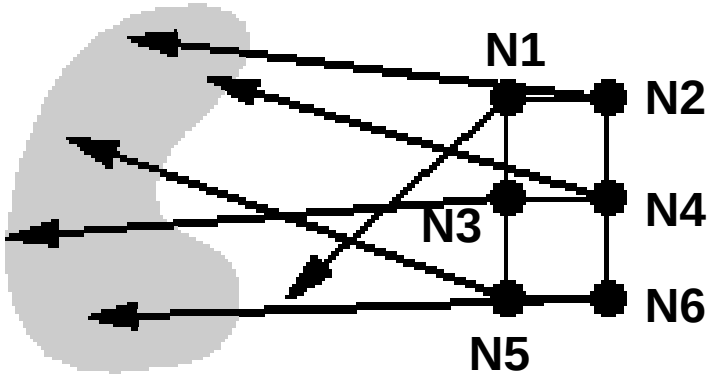
2D-Kohonen

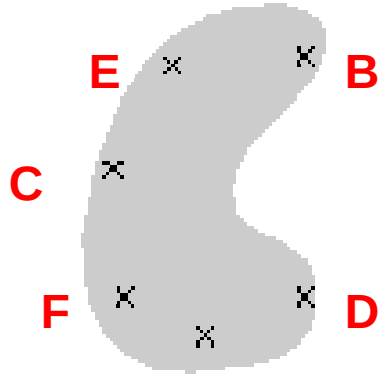


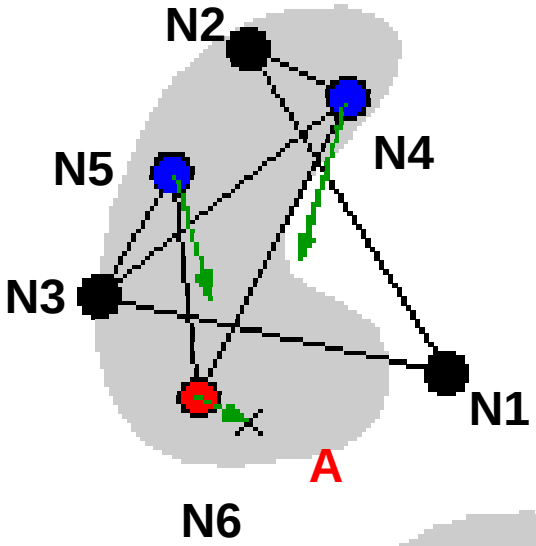


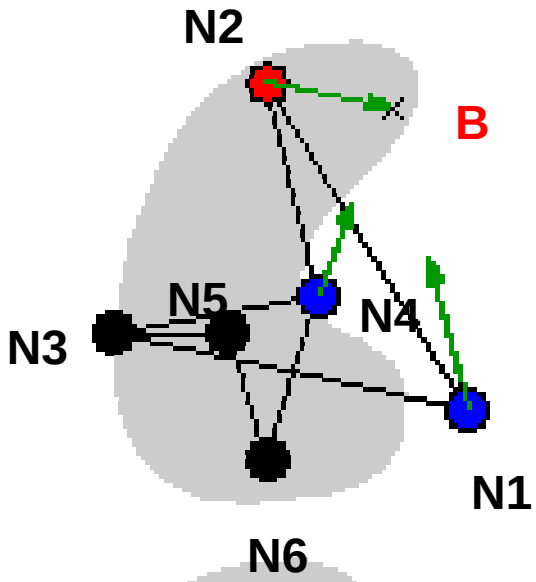
Nichtlineare Projektion des hochdimensionalen Raumes
Der Inputvektoren auf das 2D-Kohonennetz

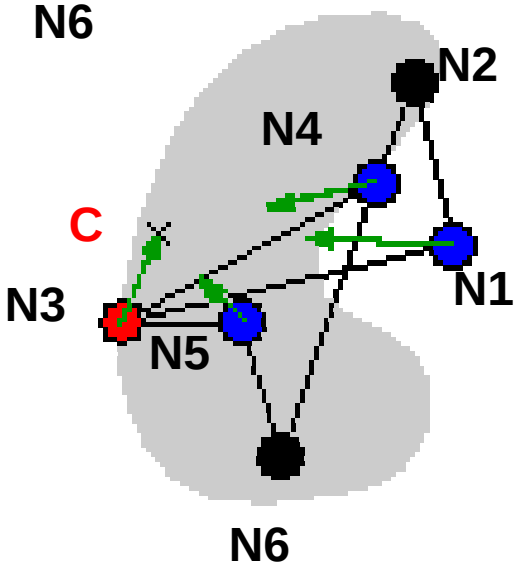
Netzentfaltung im Gewichtsraum

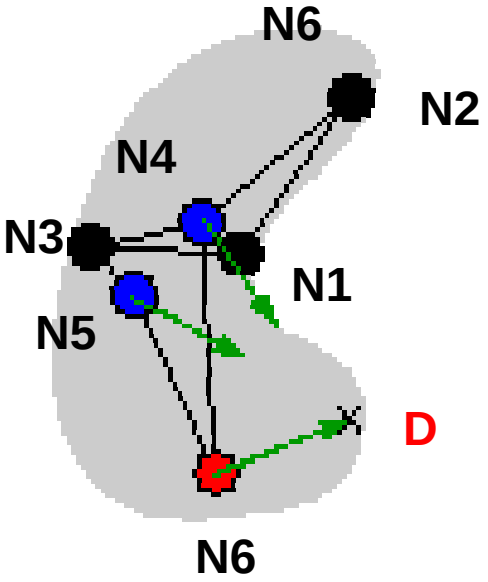




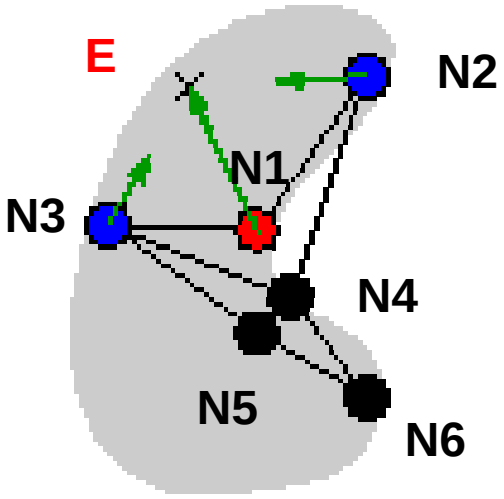


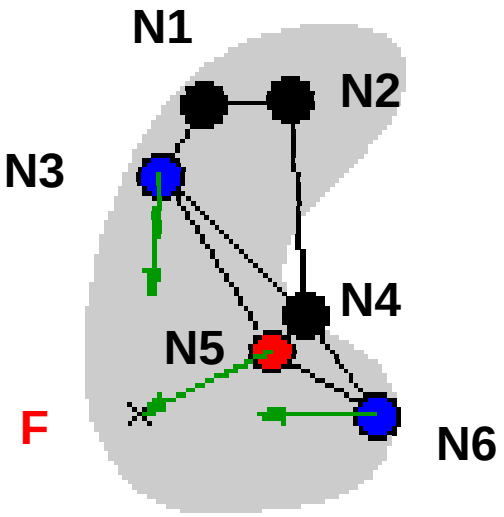


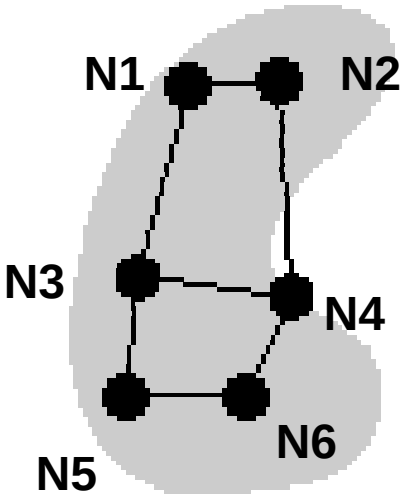


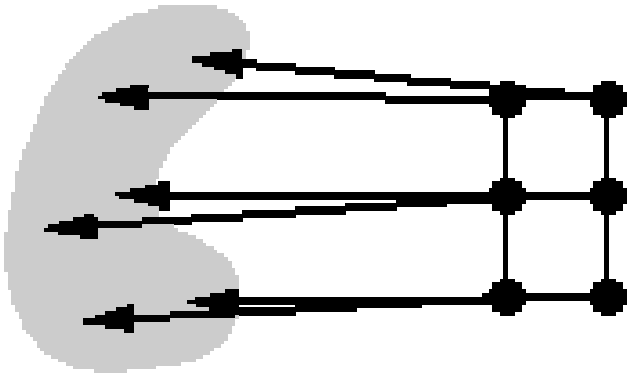


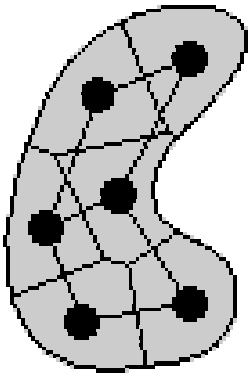
Topologische Abbildung











Lernalgorithmus Epochenlernen

1. Initialisierung der Gewichte mit zufälligen Werten
2. Berechnung der Ausgangsvektoren y und der Gewichtskorrektur für alle Muster

$$(\vec{x}_p, \vec{z}_p) \rightarrow \vec{y}_p \rightarrow \Delta_p W$$

3. Berechnung der Gesamtkorrektur

$$\Delta W = \sum_p \Delta_p W$$

4. Anpassung der Gewichte

$$W(t+1) = W(t) + \Delta W$$

5. Wiederholung ab Schritt 2 bis der Fehler genügend klein ist

Lernalgorithmus Musterlernen

1. Initialisierung der Gewichte mit zufälligen Werten
2. (Zufällige) Auswahl eines Musters
3. Berechnung des Ausgangsvektors y_p und der Gewichtskorrektur für das Muster p

$$(\vec{x}_p, \vec{z}_p) \rightarrow \vec{y}_p \rightarrow \Delta_p W$$

4. Anpassung der Gewichte

$$W(t+1) = W(t) + \Delta_p W$$

5. Wiederholung ab Schritt 2 bis der Fehler genügend klein ist