

Computational Intelligence

1. Einführung cont.

Prof. Dr. Sven Behnke

Webseite zur Vorlesung

- <http://wwwais.uni-bonn.de/SS20/CI.html>
- Folien und Übungsaufgaben werden kurz vor der Vorlesung online gestellt

User: **AIS**

Passwort: **Cosero**



**AUTONOME
INTELLIGENTE
SYSTEME**



Übungen

- Tutoren: Jacobus Conradi, Peer Schütt und Moritz Zappel

	Mo	Di	Mi	Do	Fr
8(c.t.) – 10		Übung Schütt			
10 (c.t.) – 12	Übung Zappel				
12 (c.t.) – 14	Übung Zappel	Übung Schütt			
14 (c.t.) – 16	Übung Conradi	Übung Conradi		Vorlesung	

Montags 10 (c.t.) – 12, <https://bbbzentral.informatik.uni-bonn.de/b/zap-reu-x2t>

Montags 12 (c.t.) – 14, <https://bbbzentral.informatik.uni-bonn.de/b/zap-rwn-ren>

Montags 14 (c.t.) – 16, <https://bbbzentral.informatik.uni-bonn.de/b/con-xyd-py7>

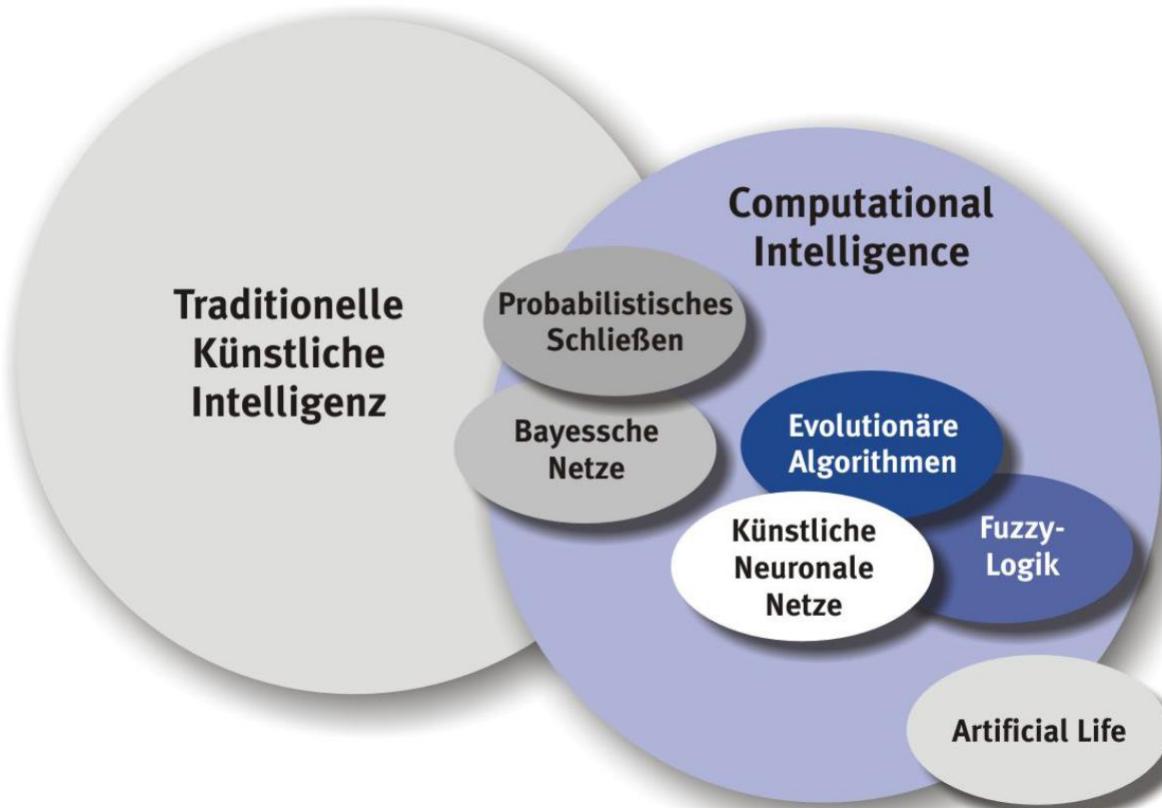
Dienstags 8 (c.t.) – 10, <https://bbbzentral.informatik.uni-bonn.de/b/sch-wm2-77x> Code 674947

Dienstags 12 (c.t.) – 14, <https://bbbzentral.informatik.uni-bonn.de/b/sch-4p3-f2p> Code 002455

Dienstags 14 (c.t.) – 16, <https://bbbzentral.informatik.uni-bonn.de/b/con-xyd-py7>

- Anmeldung war bis 26.4. via **TVS** <https://puma.cs.uni-bonn.de>
- Erste Übung war am 27./28. April

Letzte Vorlesung: Bausteine der Computational Intelligence



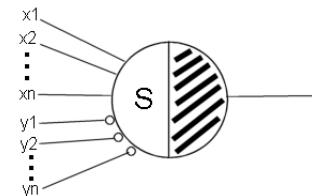
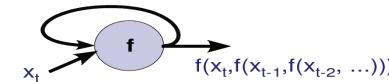
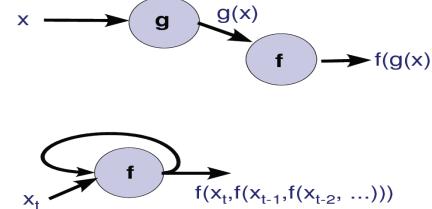
Letzte Vorlesung

■ Struktur neuronaler Netze

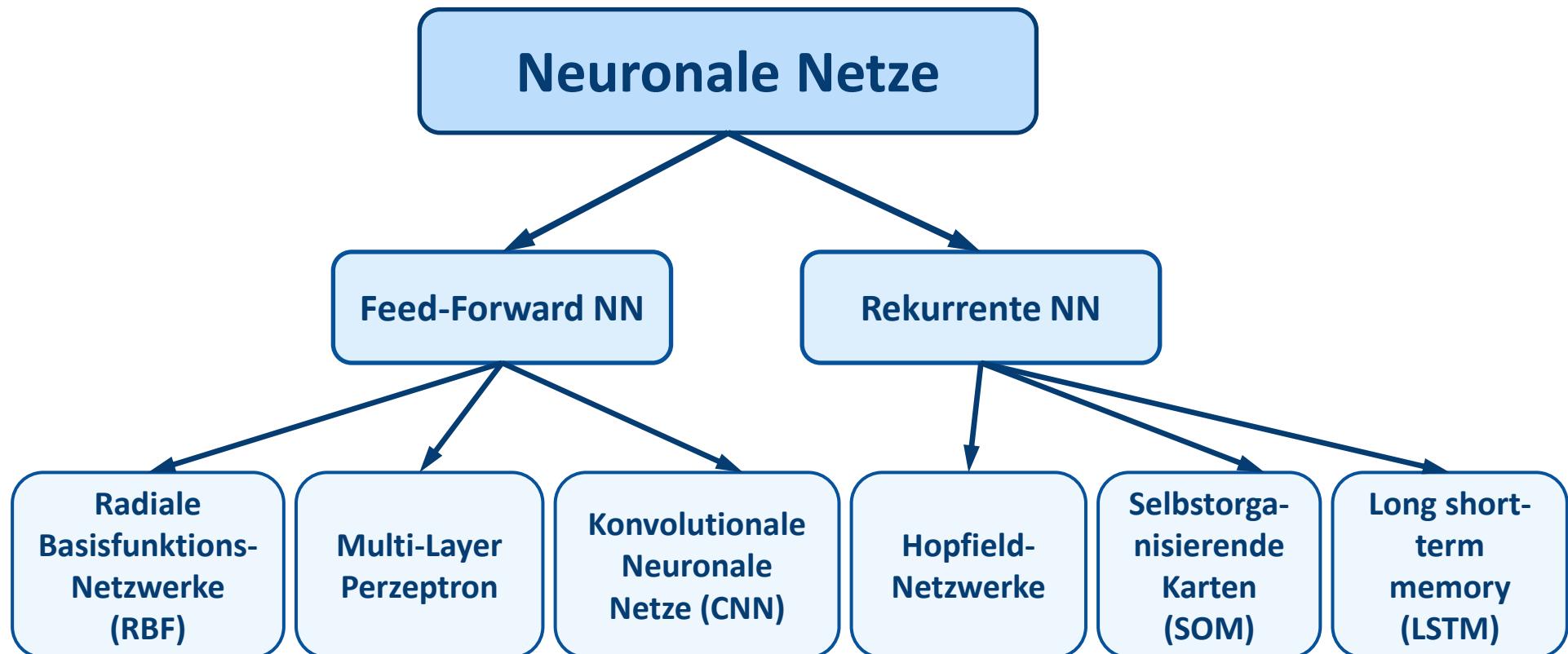
- Funktionen-Graph
- Vorwärtsgerichtet vs. rekurrent

■ McCulloch-Pitts-Neuronen

- Schwellenwertelemente
- Kombinatorische Funktionen
- Endliche Automaten



Künstliche Neuronale Netze



Evolutionäre Algorithmen

■ Motiviert durch die natürliche Evolution

- Hat angepasste Arten hervorgebracht
- Ist der Ursprung der Vielfalt allen Lebens

■ Prinzipien & Charakteristika

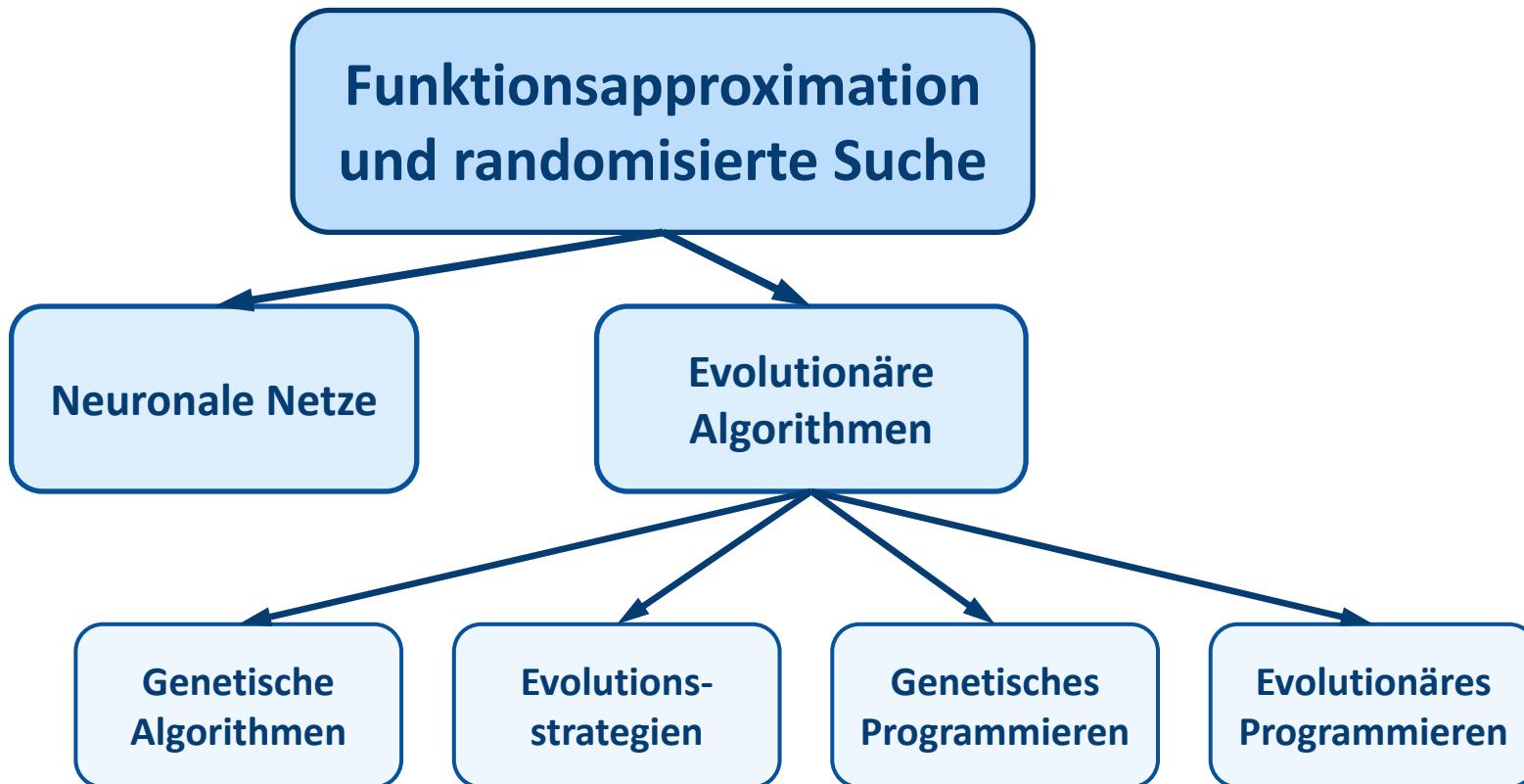
- Populationsbasiert
- Veränderung der Individuen mittels genetischer Operatoren (Rekombination und Mutation)
- Unterschiedliche Überlebens- und Reproduktionsrate (Selektion) gemäß Anpassungsgrad an die Umwelt
- Optimiererverfahren 0. Ordnung, d.h. ableitungsfrei

■ Anwendungsgebiete

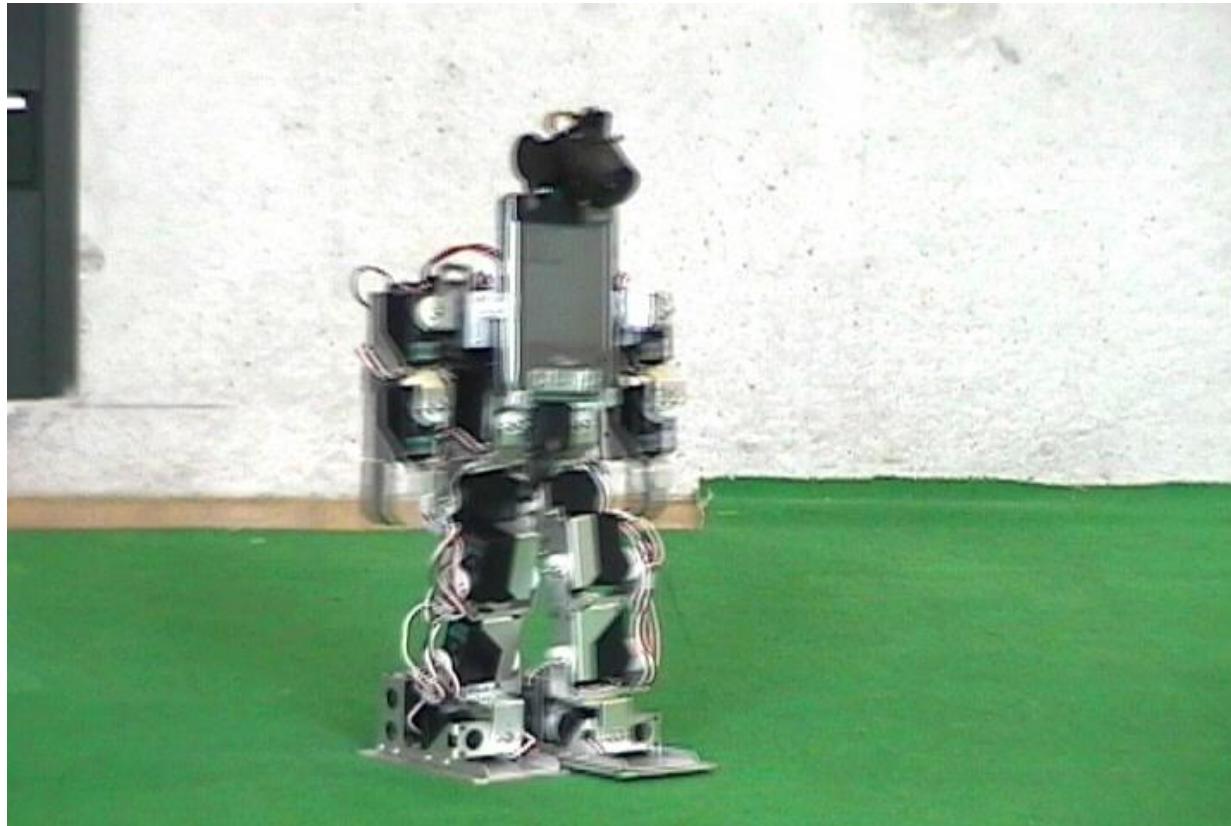
- Optimierung analytisch unbekannter Zusammenhänge
- Optimierung von Problemen mit nicht berechenbaren Ableitungen
- Optimierung von 'schweren' Problemen



Evolutionäre Algorithmen



Beispiel: Optimierung des Gangs



[Julio Pastrana, Masterarbeit, 2005]

Fuzzy-Logik

■ Motiviert durch Unschärfe in der realen Welt

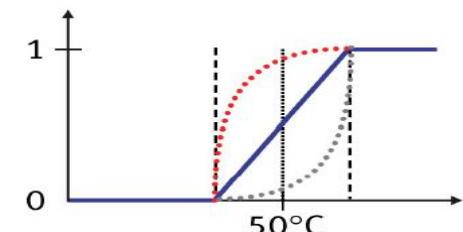
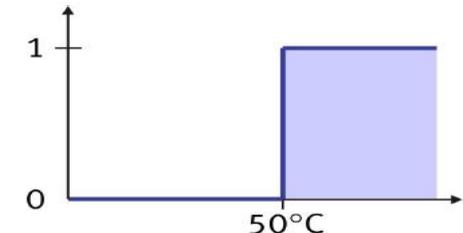
- Beobachtungen in der Umwelt lassen sich mit klassischen mathematischen Mitteln nur schwer beschreiben

■ Prinzipien & Charakteristika

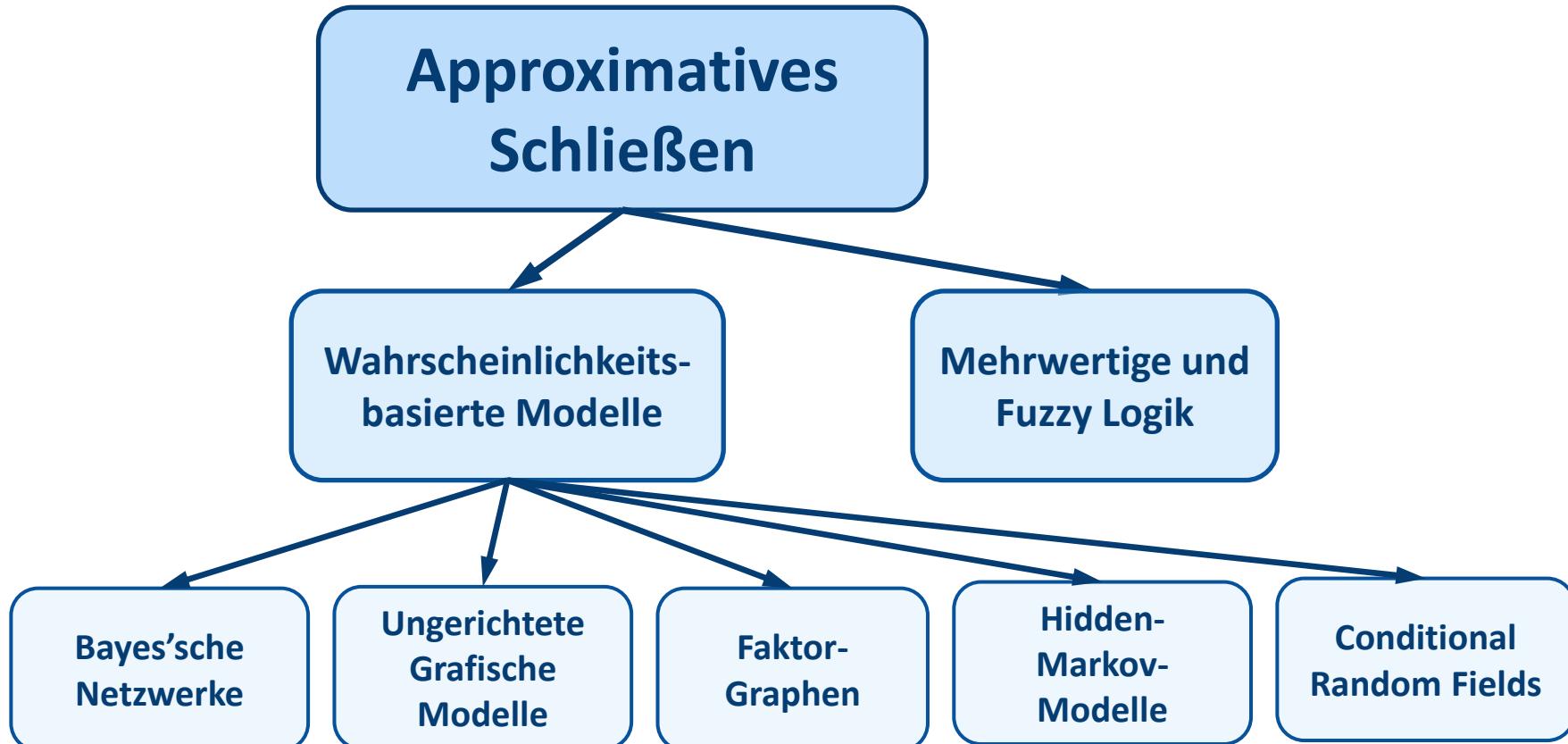
- Erweiterung der traditionellen Logik um graduelle Wahrheitswerte
- Beliebige Werte zwischen wahr und falsch
- => Erweiterung der logischen Operatoren notwendig
 - Konjunktion (Und) -> tNorm (z.B. minimum)
 - Disjunktion (Oder) -> tConorm (z.B. maximum)
 - Negation (Nicht) -> FuzzyNegation (z.B. $N(x)=1-x$)
- Verbale Beschreibung von Informationen
- Klassische Logik ist als Spezialfall enthalten

■ Anwendungsgebiete

- Steuerungs- und Regelungstechnik (Konsumerelektronik, Maschinenbau, Automobilbau, Medizintechnik, . . .)
- Bildverarbeitung, Sprachverarbeitung, . . .

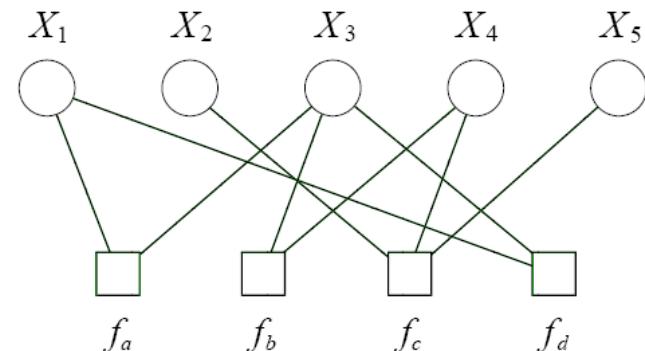
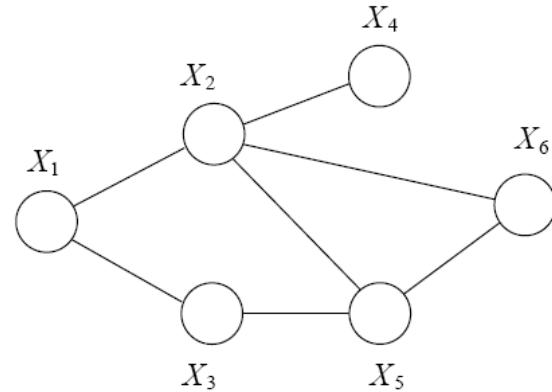
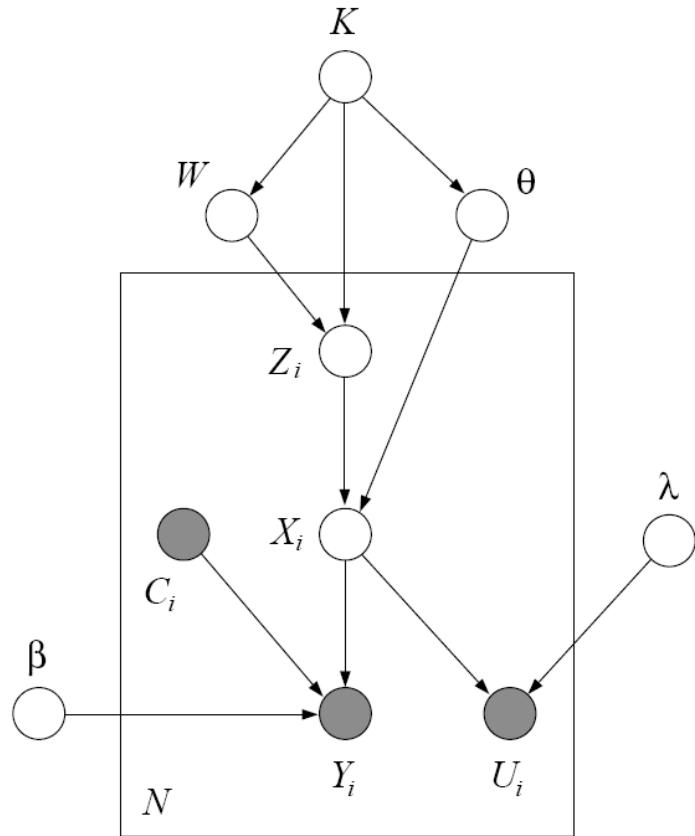


Approximativer Schliessen

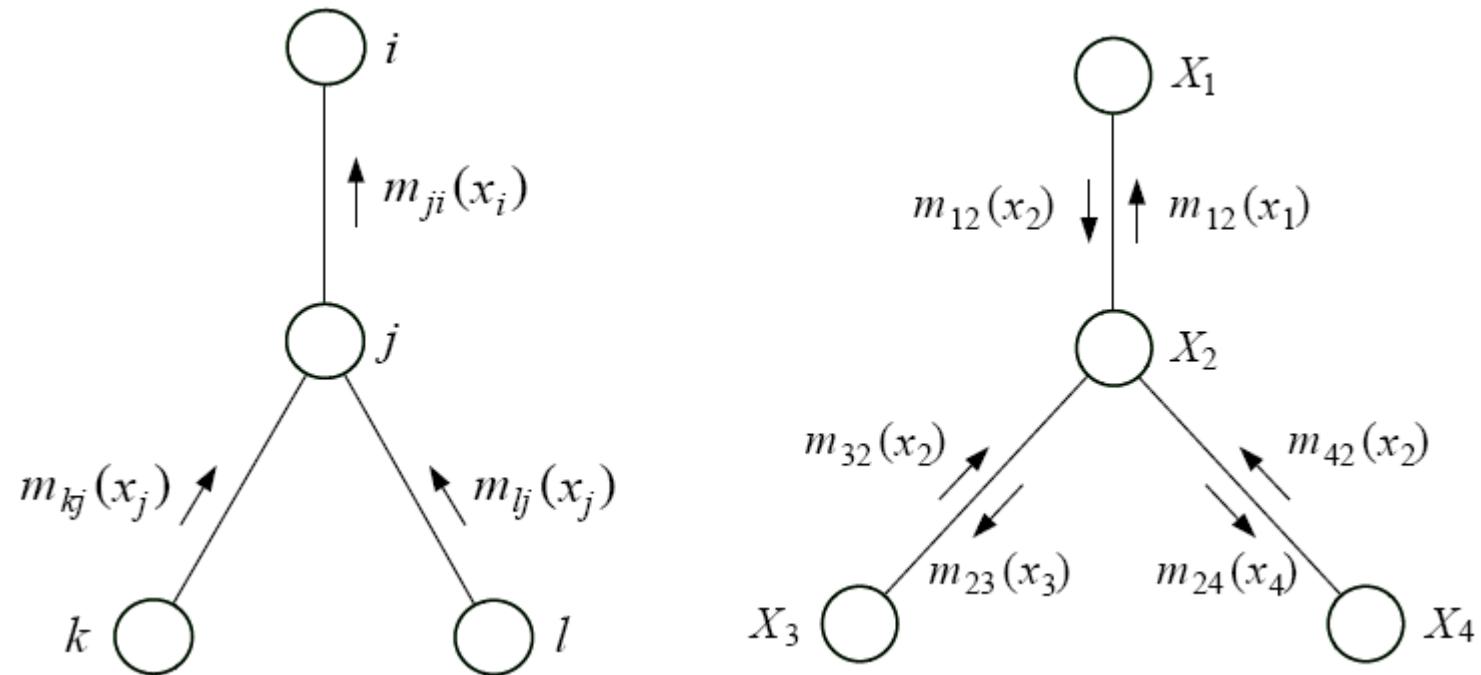


Grafische Modelle

■ Modellierung lokaler bedingter Wahrscheinlichkeiten



Inferenz durch Nachrichtenaustausch



Schwarmintelligenz

Motiviert durch „Superorganismen“

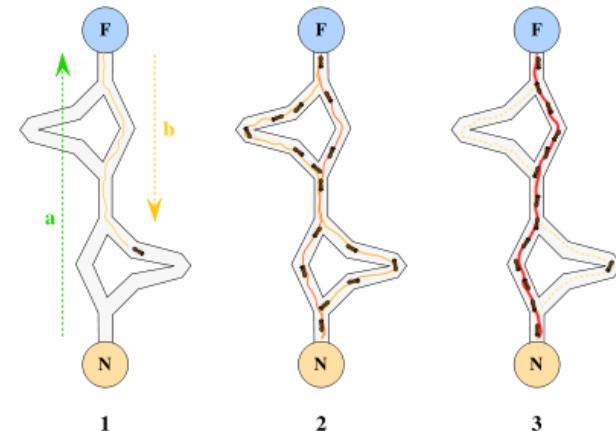
- Staatenbildende Insekten, Fisch- und Vogelschwärme sowie Herden
- Kommunikation und spezifische Handlungen von Individuen können intelligente Verhaltensweisen des betreffenden „Superorganismus“, d.h. der sozialen Gemeinschaft, hervorrufen

Prinzipien & Charakteristika

- Selbstorganisation
- Keine zentrale Kontrolle
- Lokale Interaktion
- Teilweise Aufgabe der Autonomie

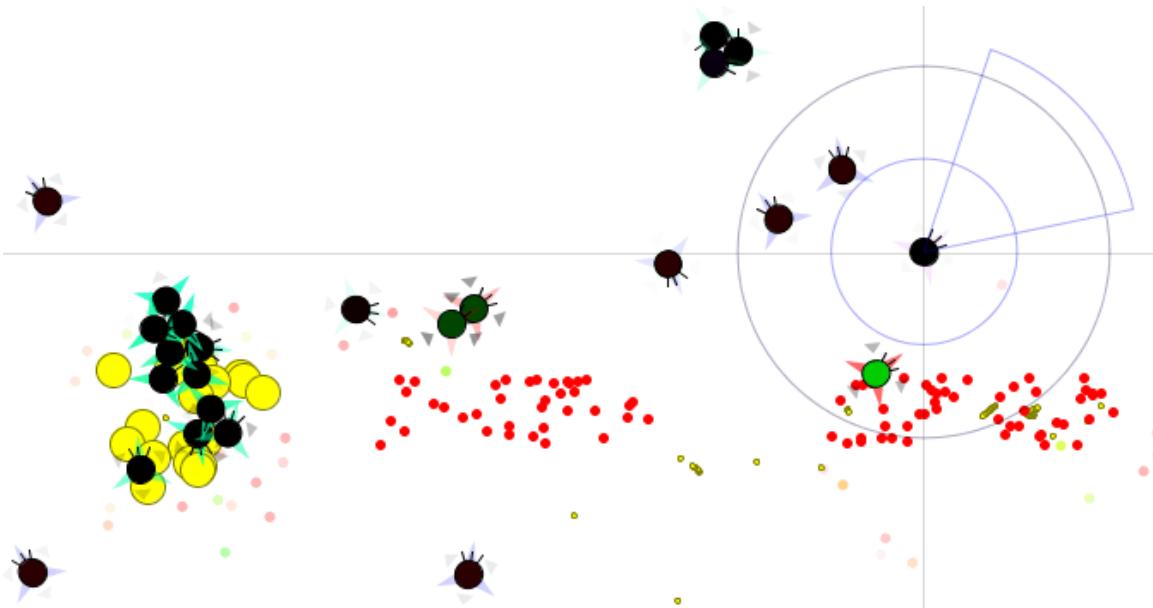
Anwendungsgebiete

- Kombinatorische Optimierung (z.B. kürzeste Wege)
- Kollektive Entscheidungsfindung (z.B. Particle Swarm Optimization)
- Multi-Agenten-Systeme (z.B. RoboCup-Fußball)



Beispiel: Verteilte, evolutionäre Optimierung von Schwärmen

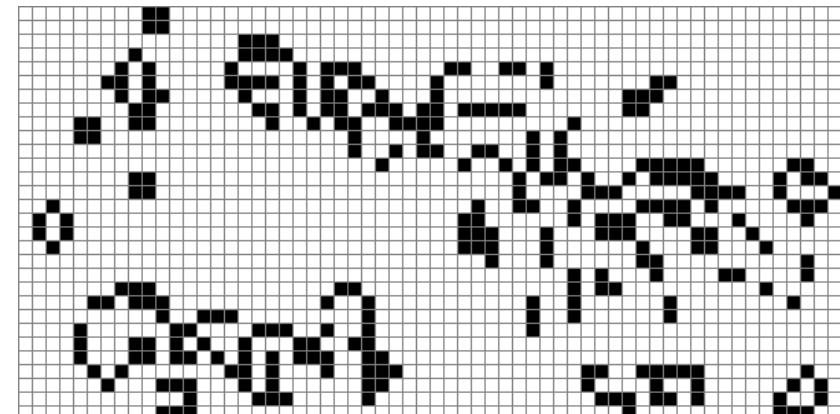
[David Kriesel, Diplomarbeit, 2009]



Artificial Life

■ Untersuchung einfacher künstlicher Lebensformen

- Realisierung von Eigenschaften von Lebewesen
 - Reproduktion
 - Evolution
 - Informationsaustausch
 - Entscheidungsfreiheit
- Beispiele:
 - Game-of-Life (Conway)
 - Computerviren
 - Synthetische Biologie



Künstliche Immunnetzwerke

Inspiriert vom Immunsystem der Wirbeltiere

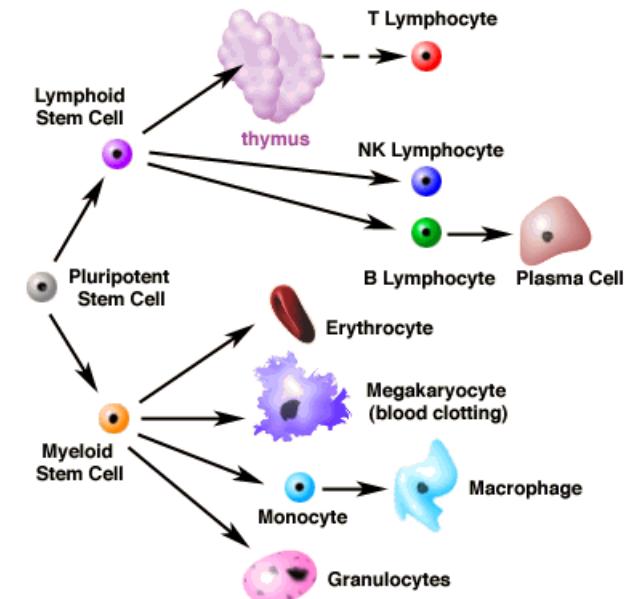
- Immunsystem erlaubt Überleben in feindlicher Umwelt

Prinzipien & Charakteristika

- Mustererkennung
- Vielfalt
- Selbst/Fremd – Unterscheidung
- Autonomie
- Lernen und Gedächtnis

Anwendungen

- Anomalie-Erkennung
- Lernende Virenscanner



Computational Intelligence vs. Soft Computing

- “Soft computing differs from conventional (hard) computing in that, unlike hard computing, it is tolerant of imprecision, uncertainty and partial truth.
- In effect, the role model for soft computing is the human mind. The guiding principle of soft computing is: Exploit the tolerance for imprecision, uncertainty and partial truth to achieve tractability, robustness and low solution cost.
- The principal constituents of soft computing (SC) are
 - Fuzzy logic (FL),
 - Neural network theory (NN) and
 - Probabilistic reasoning (PR),
with the latter subsuming
 - belief networks, genetic algorithms,
 - chaos theory and
 - parts of learning theory. . . .



Lotfi Zadeh

Computational Intelligence vs. Soft Computing II

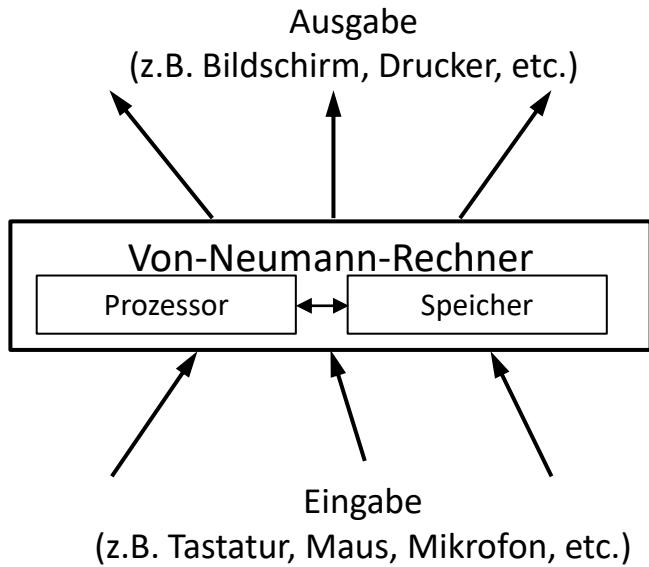
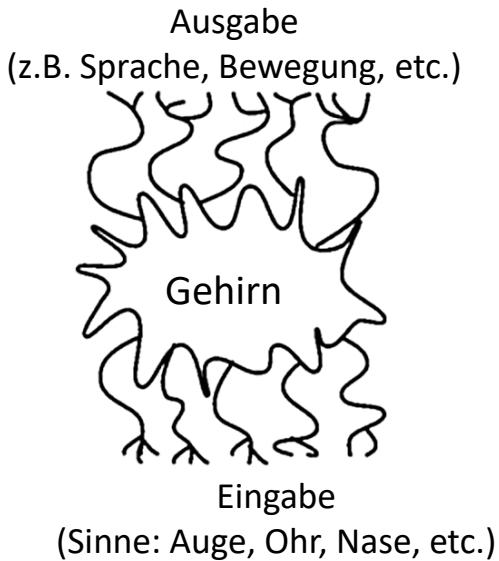
- ... What is important to note is that SC is not a melange of FL, NN and PR.
 - Rather, it is a partnership in which each of the partners contributes a distinct methodology for addressing problems in its domain.
 - In this perspective, the principal contributions of FL, NN and PR are **complementary** rather than competitive." [Lotfi Zadeh]
-
- Entspricht der CI bzgl. der betrachteten Teilgebiete
 - Unterschied liegt in der Betonung synergistischer Effekte durch Fusion von zwei oder mehr Modellen / Techniken

COMPUTATIONAL INTELLIGENCE

2. NEURONALE NETZE

Prof. Dr. Sven Behnke

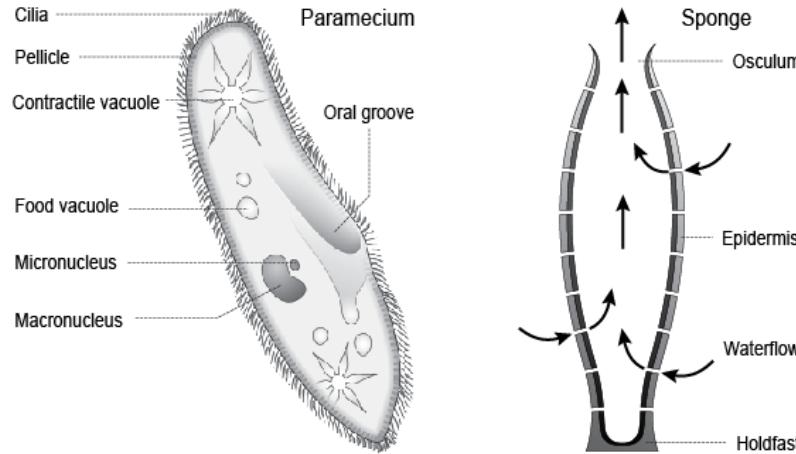
Gehirn vs. Computer



	Gehirn	Computer
Anz. Recheneinheiten	$\approx 10^{11}$	$\approx 10^9$
Art Recheneinheiten	Neuronen	Transistoren
Art der Berechnung	Massiv parallel	i.d.R. seriell
Datenspeicherung	assoziativ	adressbasiert
Schaltzeit	$\approx 10^{-3}s$	$\approx 10^{-9}s$
Theoretische Schaltvorgänge	$\approx 10^{14}s^{-1}$	$\approx 10^{18}s^{-1}$
Tatsächliche Schaltvorgänge	$\approx 10^{12}s^{-1}$	$\approx 10^{10}s^{-1}$

Warum Nervensysteme?

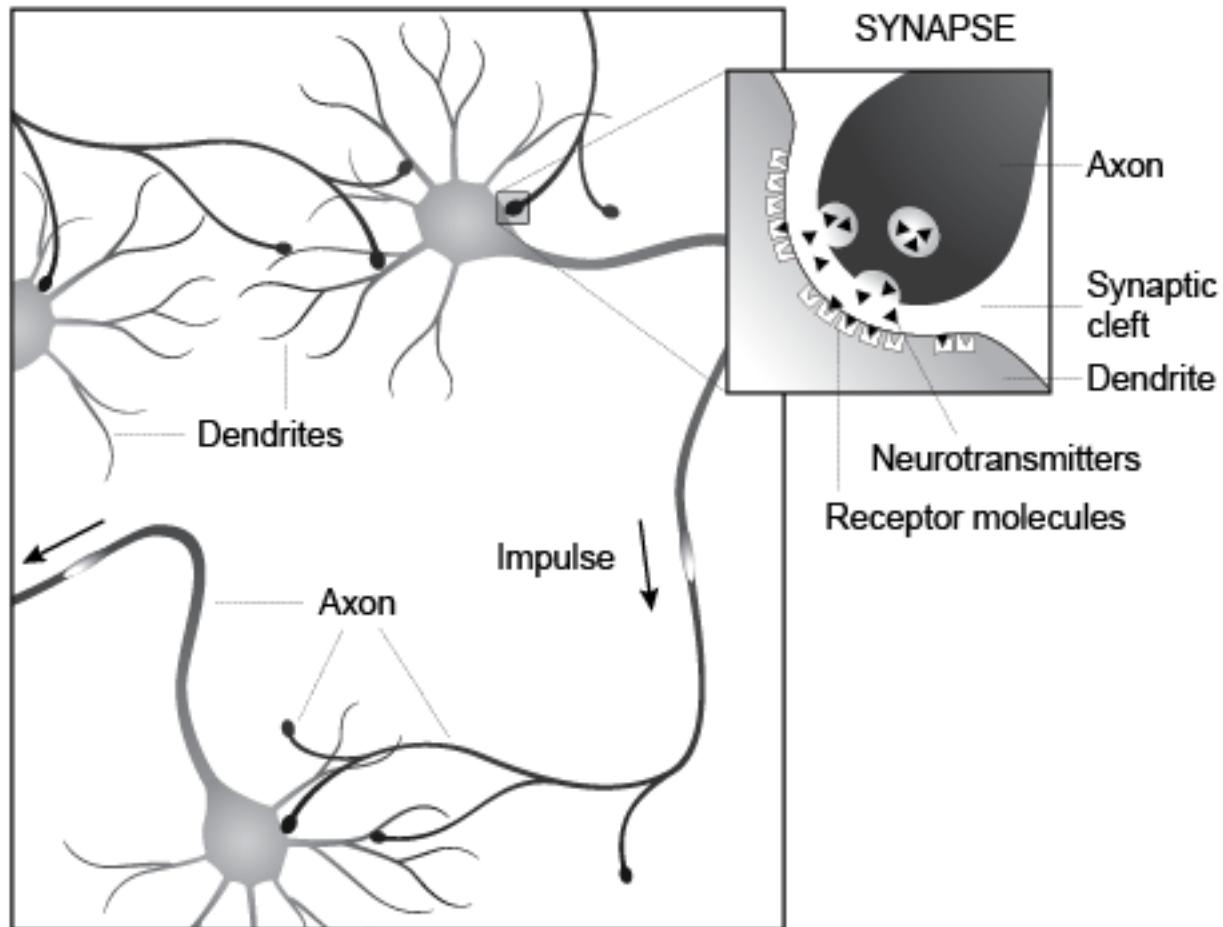
- Nicht alle Tiere haben Nervensysteme. Manche nutzen chemische Reaktionen.
- Pantoffeltierchen und Schwämme bewegen sich, fressen, zeigen Habituation.



■ Vorteile von Nervensystemen:

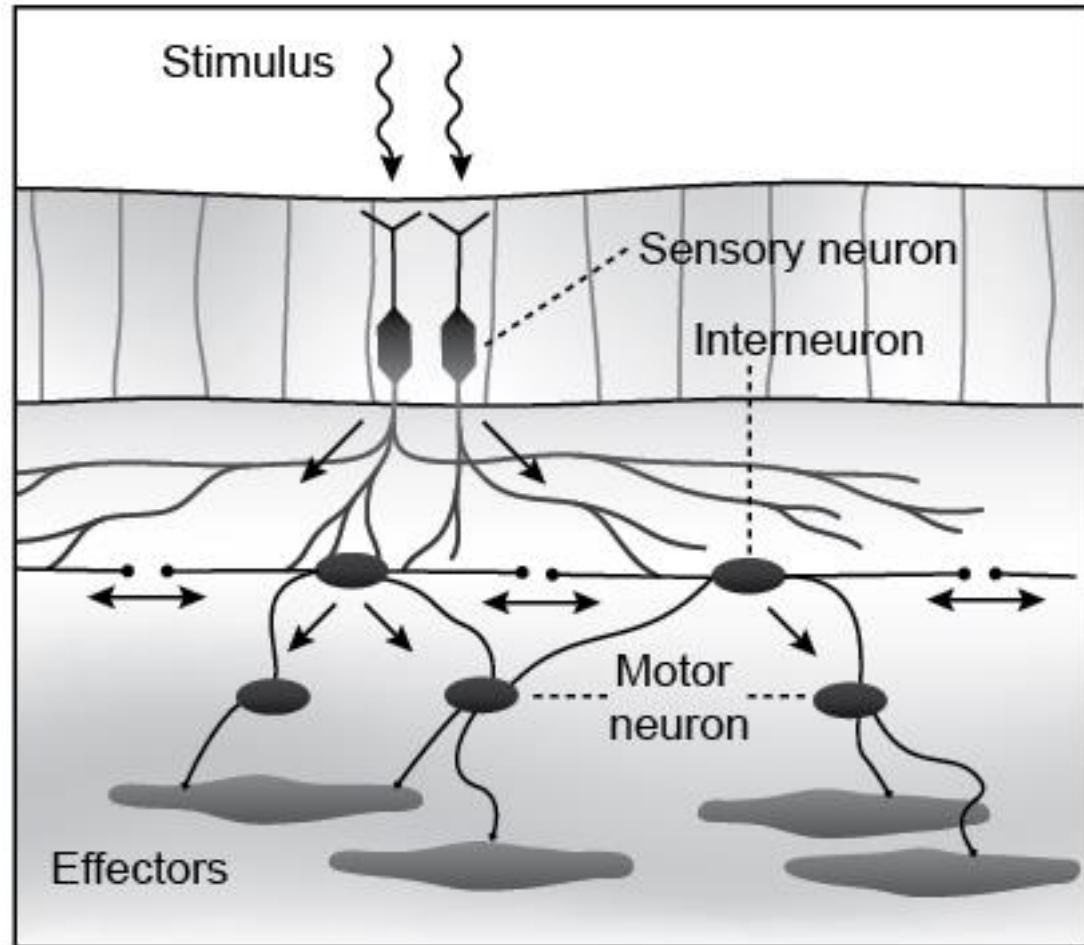
- Selektive Signalübertragung über weite Distanzen => komplexere Körper
- Komplexe Adaptation => Überleben in veränderlicher Umwelt

Biologische Neuronen



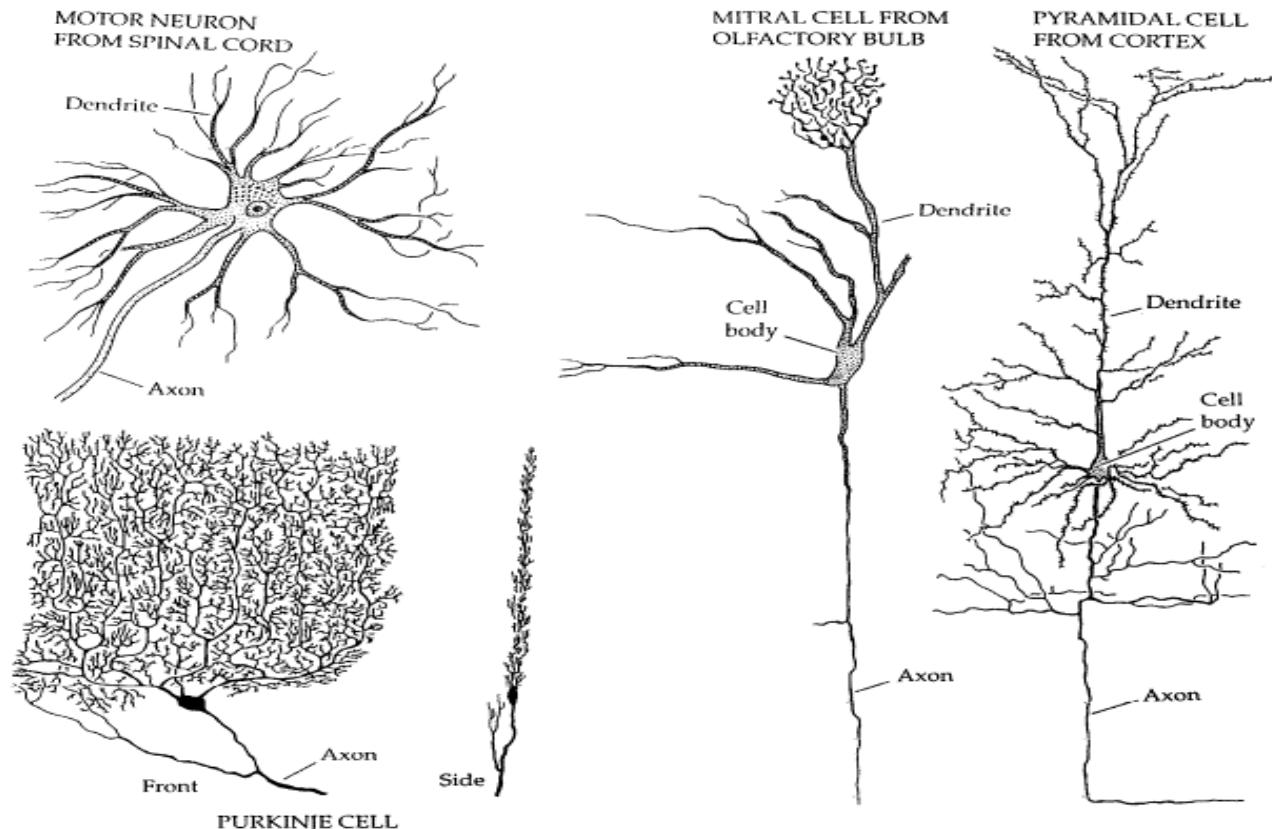
Neuronentypen

- Sensorische Neuronen
 - Nehmen Reize auf und wandeln diese in neuronalen Code um
- Interneurone
 - Erregend oder hemmend
 - Komplexe Informationsverarbeitung
- Motoneuronen
 - Steuern Muskeln an



Konnektivität von Neuronen

- Neuronen sind immer gleich aufgebaut und haben die gleiche Funktionsweise (Parker, 1919)
- Unterschiede liegen vor allem in der Konnektivität
- Klassifikation durch Cajal, 1911



[Cajal, 1911]

Signalübertragung an Synapsen

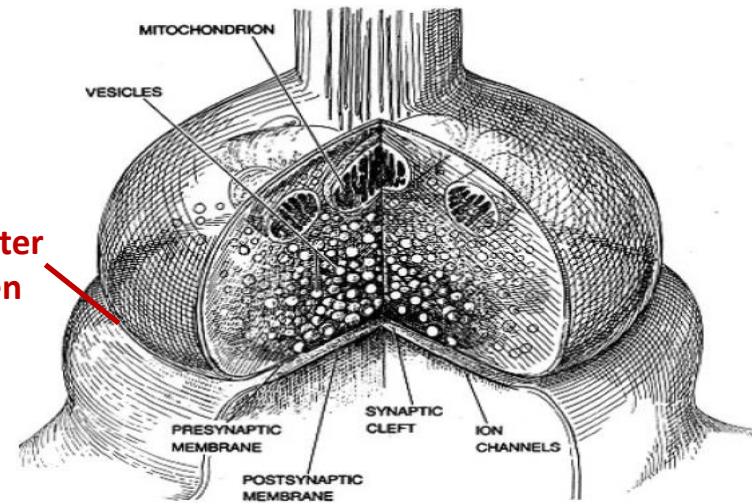
■ Zwischen Axon und Dendrit

- Signalübertragung zum nächsten Neuron
- Elektrischer Impuls => chemisches Signal => analoges elektrisches Signal
- Hier findet das Lernen statt

■ Hohe Flexibilität der Übertragung

- Verschiedene Neurotransmitter
- Verschiedene Rezeptoren
- Einfluss von Neuromodulatoren
- Vorzeichen: Verstärkung (Excitation) oder Hemmung (Inhibition)
- Zeitverhalten bei kurz aufeinander folgenden Pulsen
 - Facilitation (zweiter Puls hat stärkere Wirkung) oder
 - Depression (zweiter Puls hat geringere Wirkung)

Neurotransmitter
im synaptischen
Spalt



Aktionspotential
auf Axon

Postsynaptisches
Potential im
Dendrit

Informationsverarbeitung im Neuron

■ Inaktives Neuron

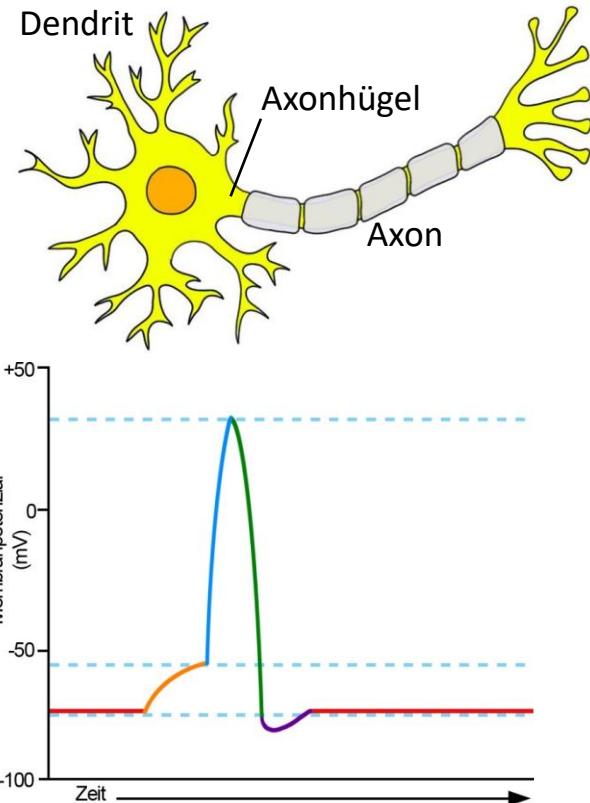
- Ruhepotential ohne äußere Erregung: ca. -70 mV
- Zellkörper integriert erregende und hemmende analoge Signale aus Dendriten
- Potentialfluktuationen bleiben unter Aktivierungsschwellwert

■ Generierung eines Aktionspotentials (Feuern)

- Erregende Reize überwiegen
- Potential überschreitet den Schwellwert von ca. -50 mV
- Selbstverstärkender Prozess erzeugt Potentialspitze von ca. 30 mV, Dauer ca. 1 ms
- Aktionspotential (Impuls, Spike) wird verlustfrei durch Axon zu den Nachfolgezellen geleitet

■ Erholungsphase (Refraktärzeit)

- Direkt nach dem Feuern sinkt Potential unter Ruhepotential ab
- Das Neuron ist nicht erregbar (ca. 5 ms)
- Die Erregbarkeit nimmt erst langsam wieder zu



Ausbreitung von Aktionspotentialen

■ Weiterleitung des Aktionspotentials

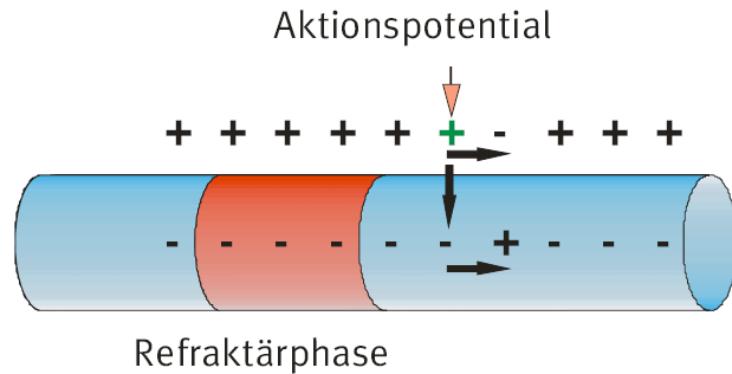
- Lokale Depolarisation führt zu Depolarisation benachbarter Membranbereiche
 - Wenn Schwellwert (ca. -50 mV) überschritten wird, entsteht lokal wieder Depolarisation

■ Definierte Ausbreitungsrichtung

- Ausgleichsströme sind ungerichtet
 - Richtung aus der Aktionspotential kommt, ist refraktär

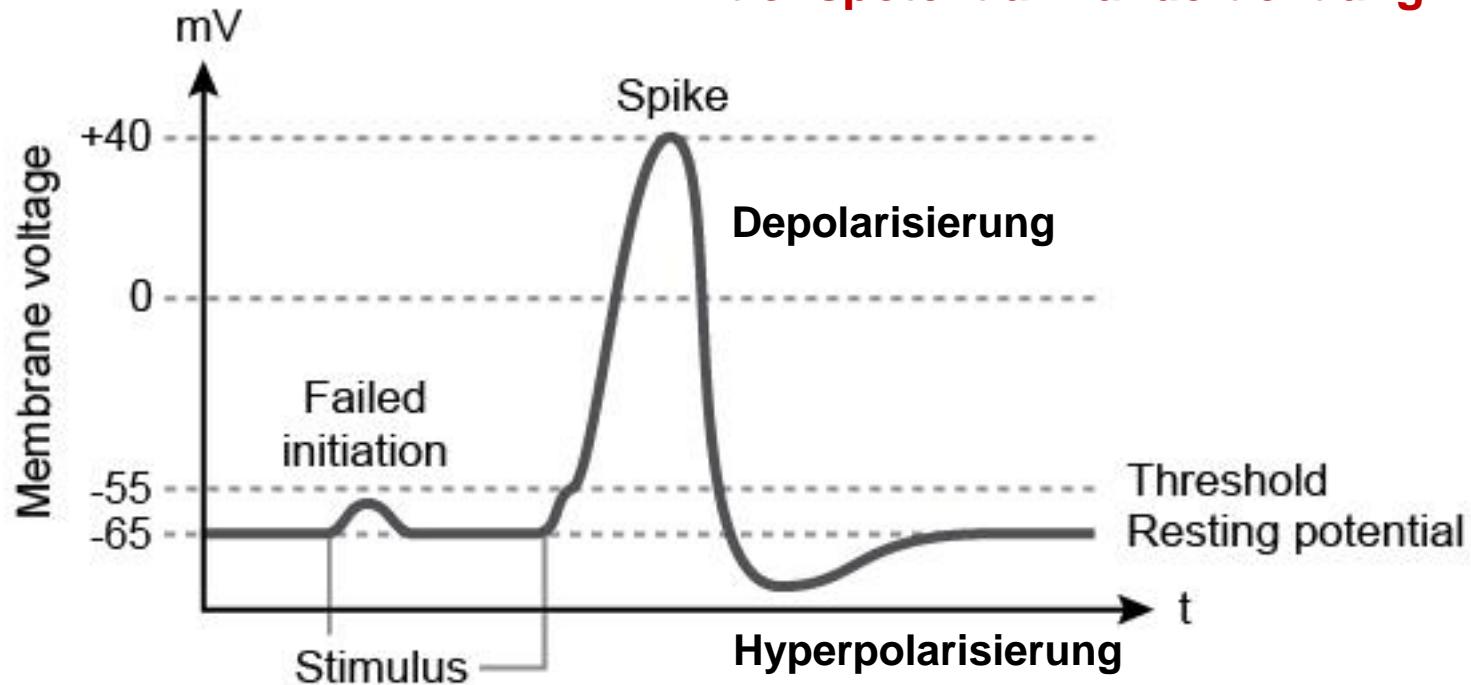
■ Selbst-Regenerierung des Aktionspotentials

- Keine Abschwächung
 - Vergleich: umfallende Reihe von Dominosteinen
 - Signalform und -höhe hat keine Bedeutung
 - Alles-oder-Nichts-Prinzip



Membrandynamik

Aktionspotential wandert entlang Axon



- Zyklus dauert 3-50 ms, abhängig von den beteiligten Ionenkanälen
(Hodgkin and Huxley, 1952)

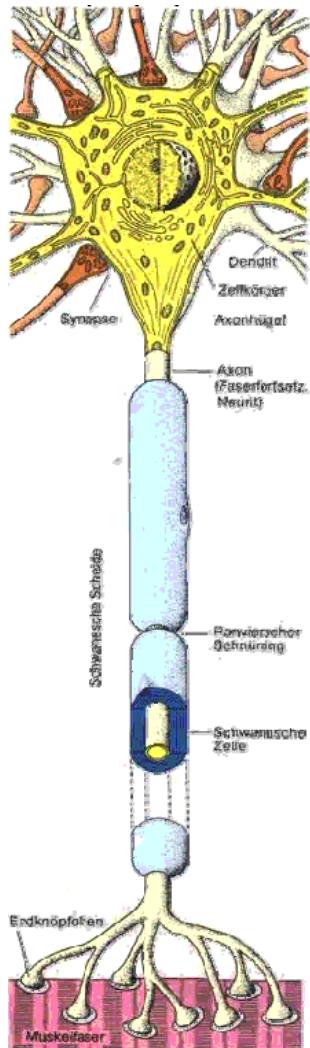
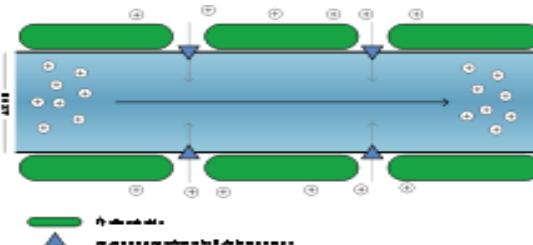
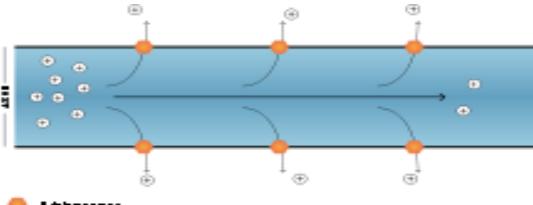
Signalausbreitung

■ Signalausbreitung

- Spannung längs zum Axon
- Strom quer zum Axon
- Relativ langsam

■ Ranviersche Schnürringe

- Elektrische Isolierung der Nervenfaser mit regelmäßigen Einschnürungen
- Myelinisierung (1 µm dick)
- Signal springt von Einschnürung zu Einschnürung
- Sehr viel schneller

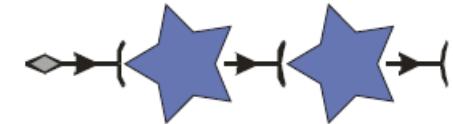


Einfache Grundschaltungen

■ Mehrstufige Weiterleitung

- Erregung durchläuft Neuronen sequentiell
- Verzögerung pro Neuron ca. 1 ms
- Nach 4 ms wieder erregbar

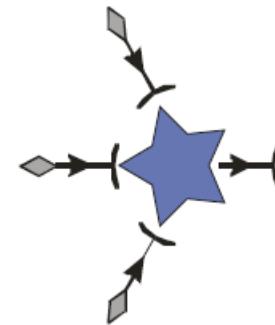
=> Funktion ähnlich zu Verzögerungsleitung & Tiefpassfilter



■ Konvergenz

- Mehrere Neuronen wirken gemeinsam auf ein Neuron
- Summe der Reize muss Schwellwert übersteigen

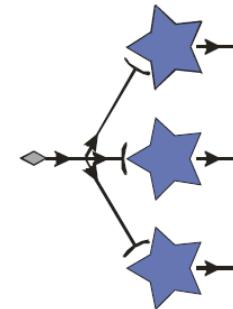
=> Merkmalsextraktion



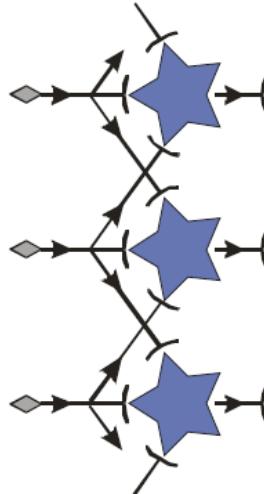
Einfache Grundschaltungen

■ Divergenz

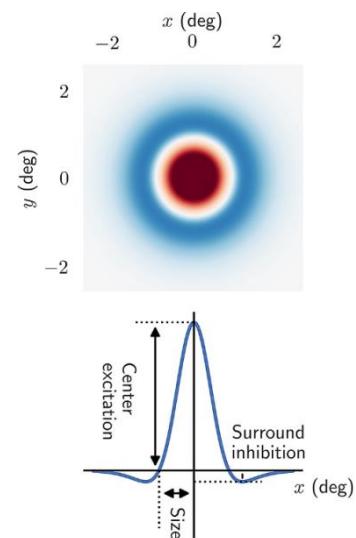
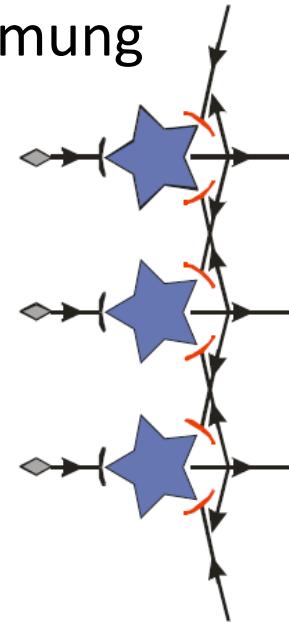
- Ein Neuron wirkt auf mehrere Neuronen
=> Verteilung von Information



■ Lokale Konnektivität

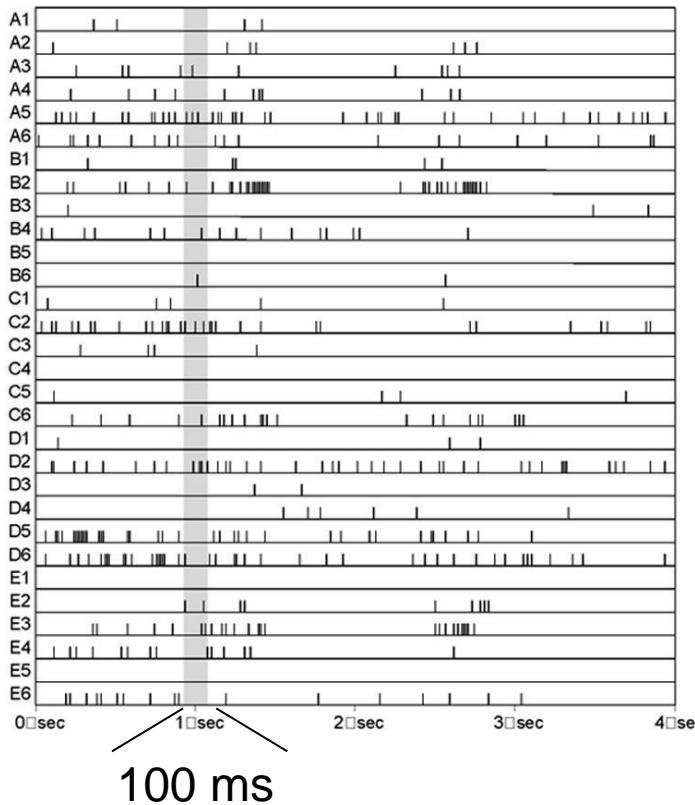


■ Laterale Hemmung



Neuronaler Code

■ Folgen von Aktionspotenzialen



Feuer-**Rate**

McCulloch-Pitts

Konnektionismus

Feuer-**Zeitpunkt**

Spikende Neuronen

Computational
Neuroscience

Neuronales Lernen

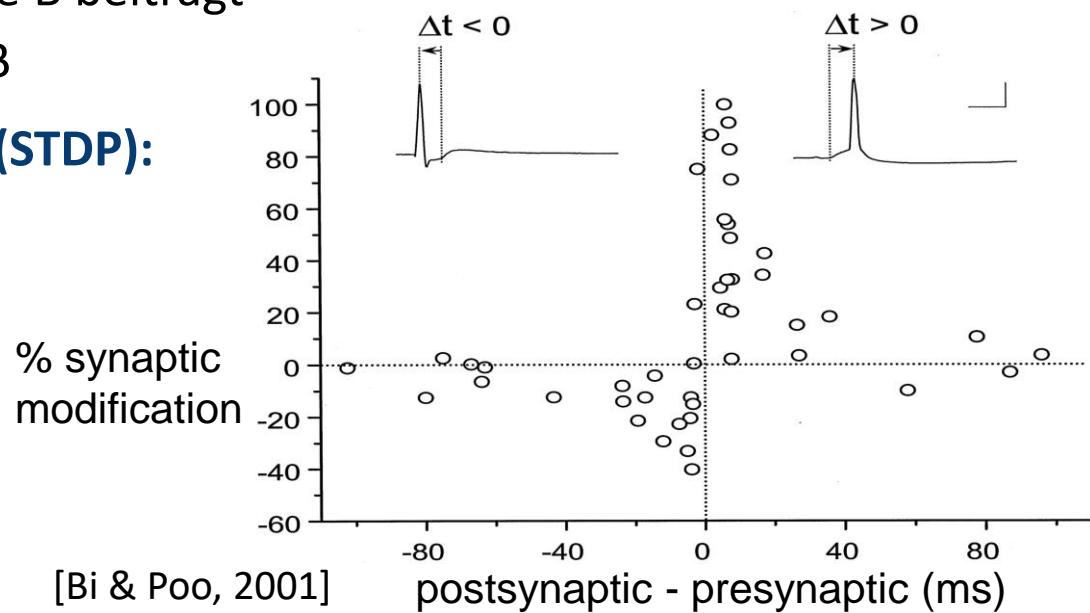
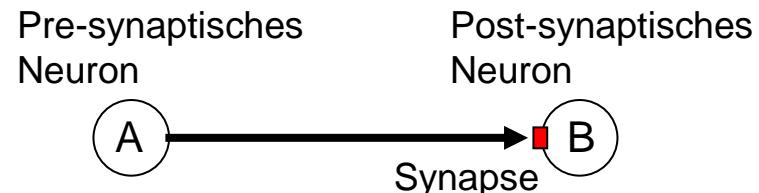
■ Lernen durch Veränderung der Synapsen

■ Hebb'sche Regel (1949):

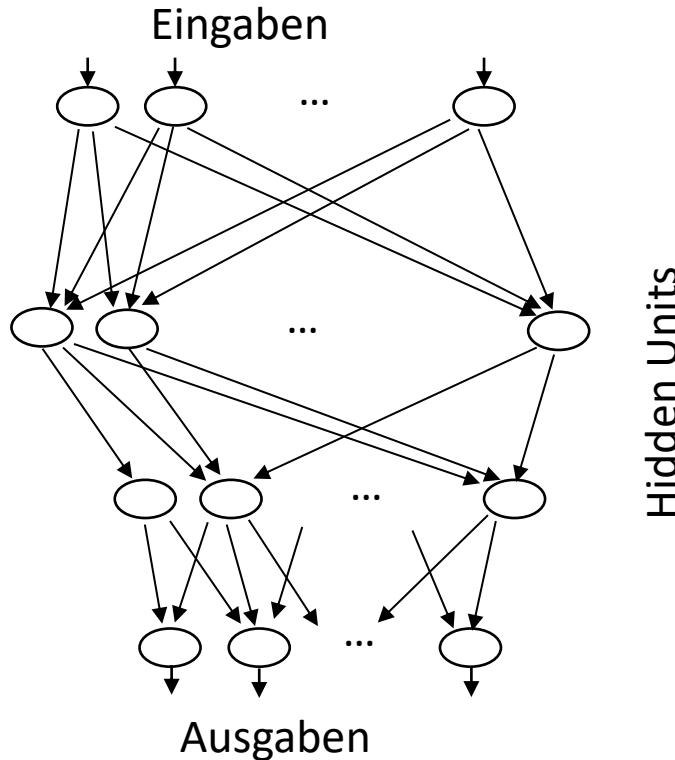
- Synaptische Stärke wird erhöht, wenn Zelle A wiederholt zum Feuern von Zelle B beiträgt
- Zeitliche Relation: A feuert vor B

■ Spike-Time Dependent Plasticity (STDP):

- Kleines Zeitfenster
- Verstärkung (LTP) bei positiver Zeitdifferenz
- Abschwächung (LTD) bei negativer Zeitdifferenz



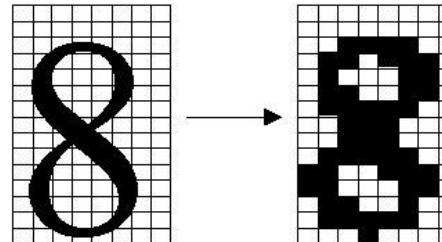
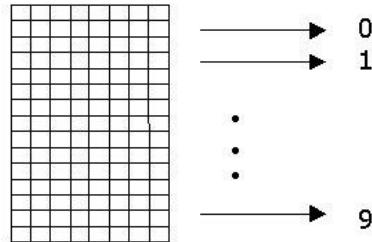
Künstliche Neuronale Netze



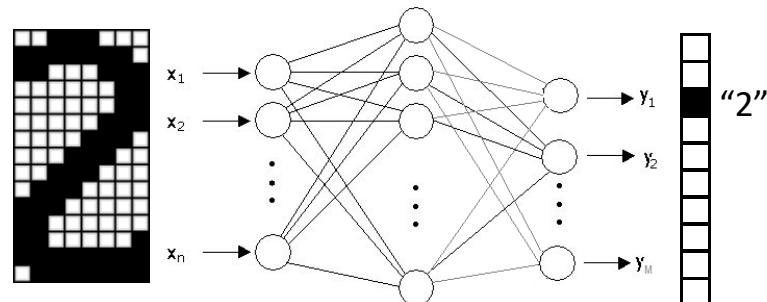
- Kommunikation mit der Umwelt durch Eingabe- und Ausgabe-Knoten
- Alle anderen Knoten werden Hidden Units genannt
- Knoten verbunden durch gerichtete Verbindungen
- Verbindungen haben Gewichte (vorzeichenbehaftet)

Einfaches Anwendungsbeispiel

- Neuronales Netz soll Ziffern von 0 bis 9 erkennen
 - Eingabe: Vektor x mit 8×15 Elementen
 - Ausgabe: Vektor y mit 10 Elementen
1-aus-M-Codierung
- Geschriebene Ziffer wird in eine Folge von Nullen und Einsen umgewandelt, wobei 0 für leere und 1 für belegte Rasterpunkte steht:



- Nach erfolgreichem Training schafft es das Neuronale Netz, geschriebene Ziffern zu erkennen und diese als Ausgabe y auszugeben.



Ein Bild sagt mehr als tausend Worte



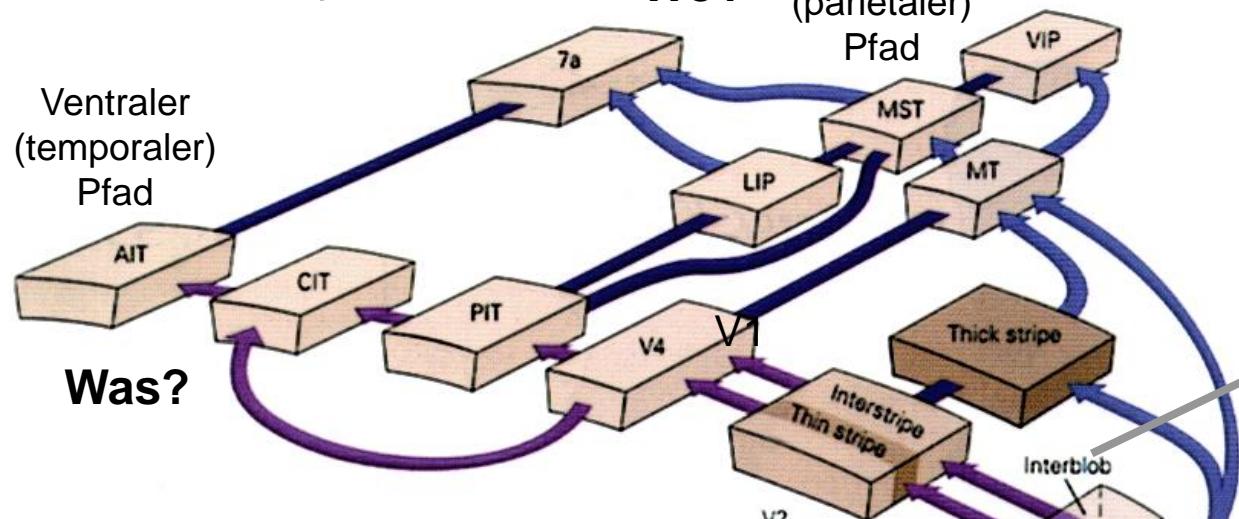
[Vinyals et al. 2014]

Leistungsfähigkeit des visuellen Systems

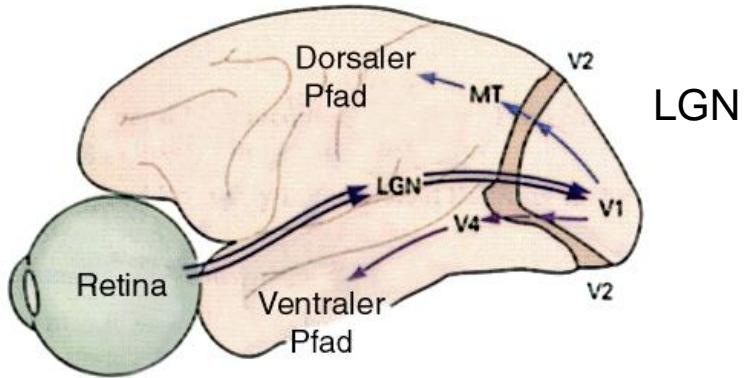


Visuelles System

Wo?



Was?



Dorsaler
(parietaler)
Pfad

MST

VIP

LIP

MT

V1

V2

Thick stripe

Interstripe

Thin stripe

Interblob

2

3

4B

4C α

4C β

V1

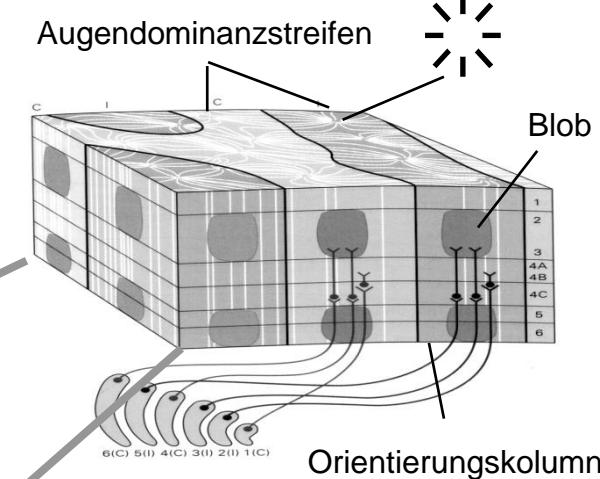
LGN

Parvo

Magno

P

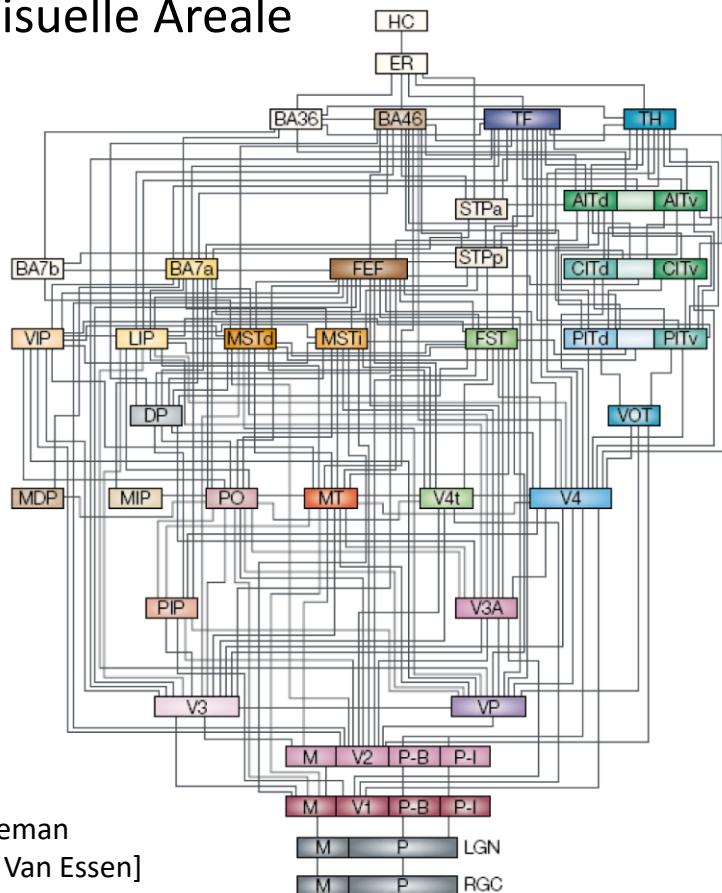
M



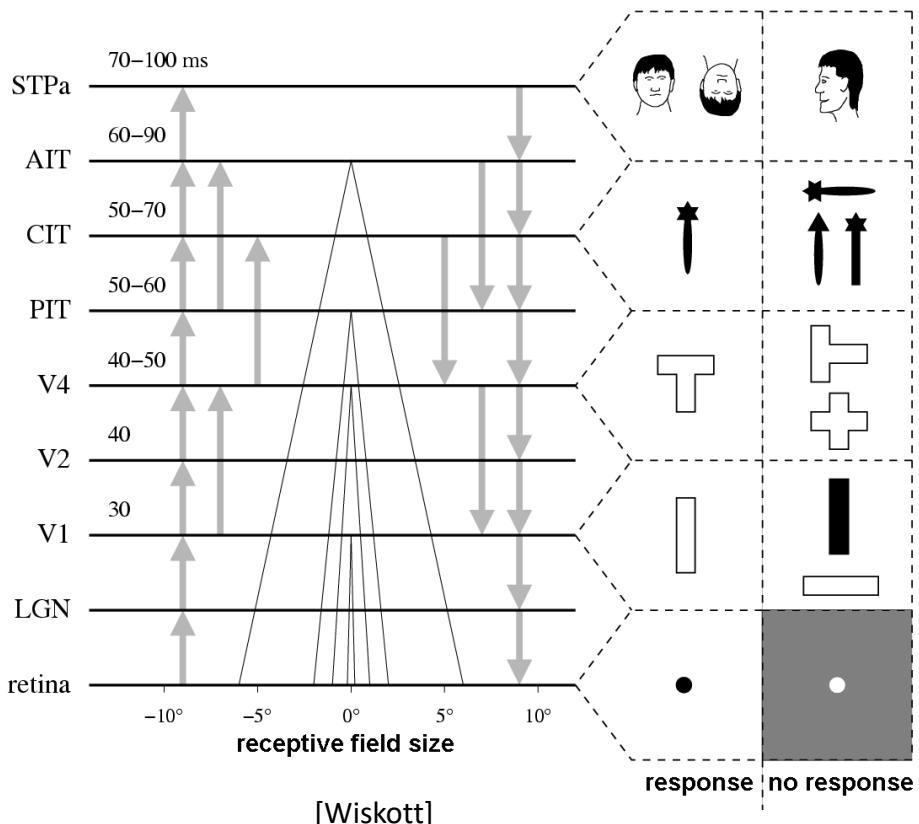
[Kandel et al. 2000]

Visuelle Verarbeitungshierarchie

■ Visuelle Areale

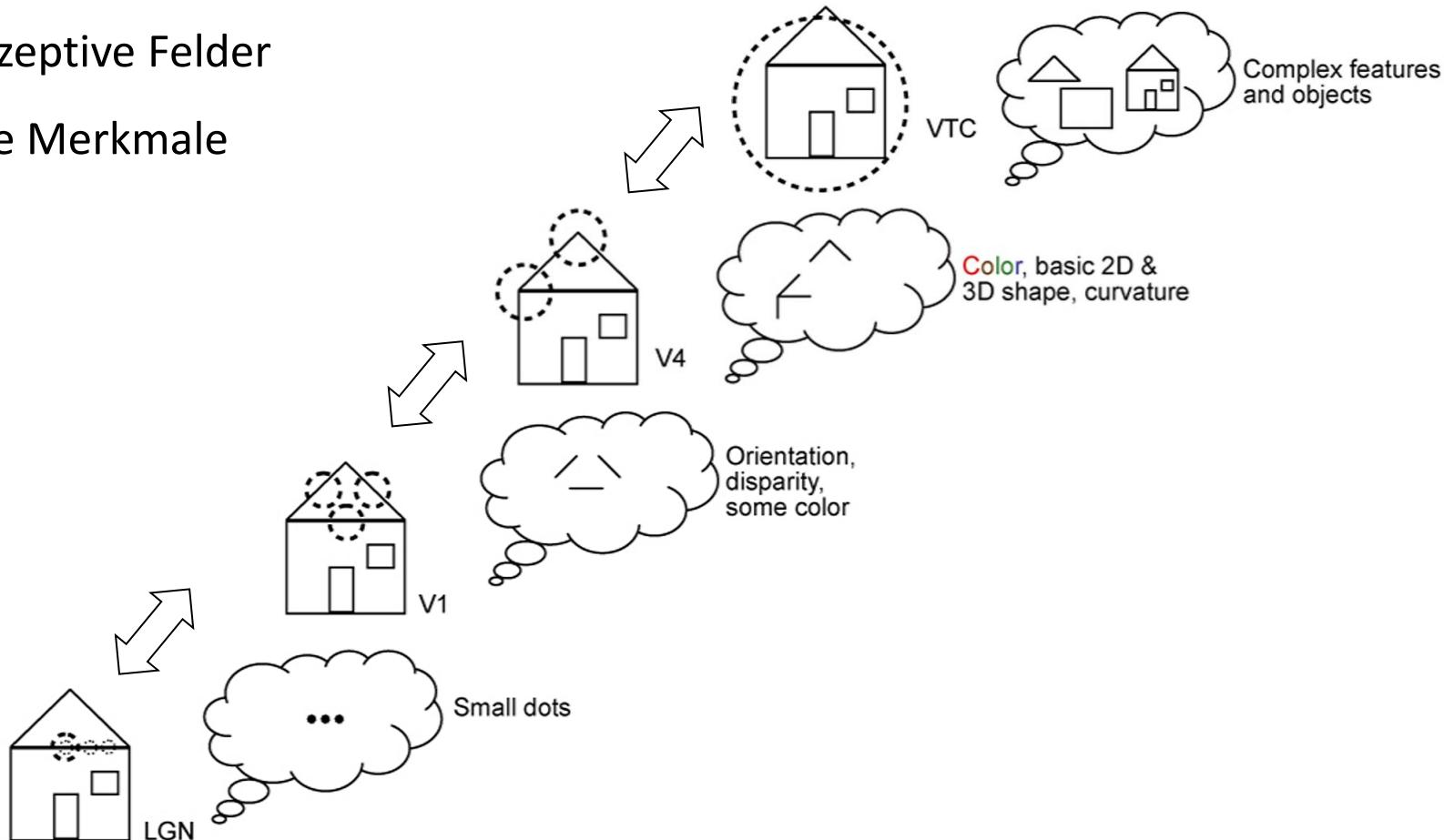


■ Repräsentierte Merkmale



Visuelle Verarbeitungshierarchie

- Größere rezeptive Felder
- Komplexere Merkmale



Kategorisierung von Bildern



GT: horse cart

- 1: horse cart
- 2: minibus
- 3: oxcart
- 4: stretcher
- 5: half track



GT: birdhouse

- 1: birdhouse
- 2: sliding door
- 3: window screen
- 4: mailbox
- 5: pot



GT: forklift

- 1: forklift
- 2: garbage truck
- 3: tow truck
- 4: trailer truck
- 5: go-kart



GT: letter opener

- 1: drumstick
- 2: candle
- 3: wooden spoon
- 4: spatula
- 5: ladle



GT: coucal

- 1: coucal
- 2: indigo bunting
- 3: lorikeet
- 4: walking stick
- 5: custard apple



GT: komondor

- 1: komondor
- 2: patio
- 3: llama
- 4: mobile home
- 5: Old English sheepdog



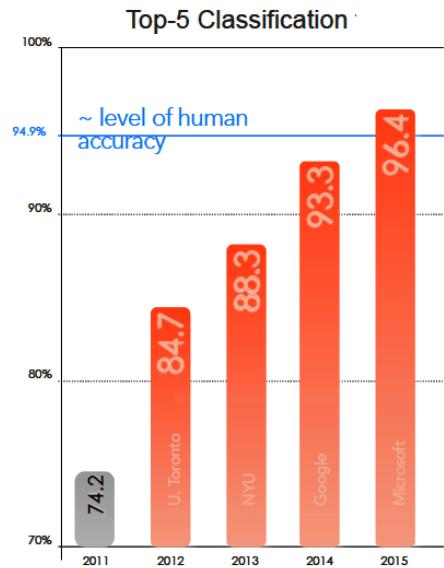
GT: yellow lady's slipper

- 1: yellow lady's slipper
- 2: slug
- 3: hen-of-the-woods
- 4: stinkhorn
- 5: coral fungus



GT: spotlight

- 1: grand piano
- 2: folding chair
- 3: rocking chair
- 4: dining table
- 5: upright piano



[He et al. 2015]

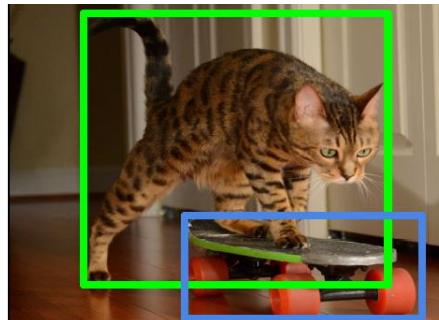
Dichte Bildbeschreibung

■ Kategorisierung



Cat

■ Objektdetektion



Cat

Skateboard

■ Bildbeschreibung



A cat
riding a
skateboard

■ Dichte Beschreibung



Orange spotted cat

Skateboard with
red wheels

Cat riding a
skateboard

Brown hardwood
flooring

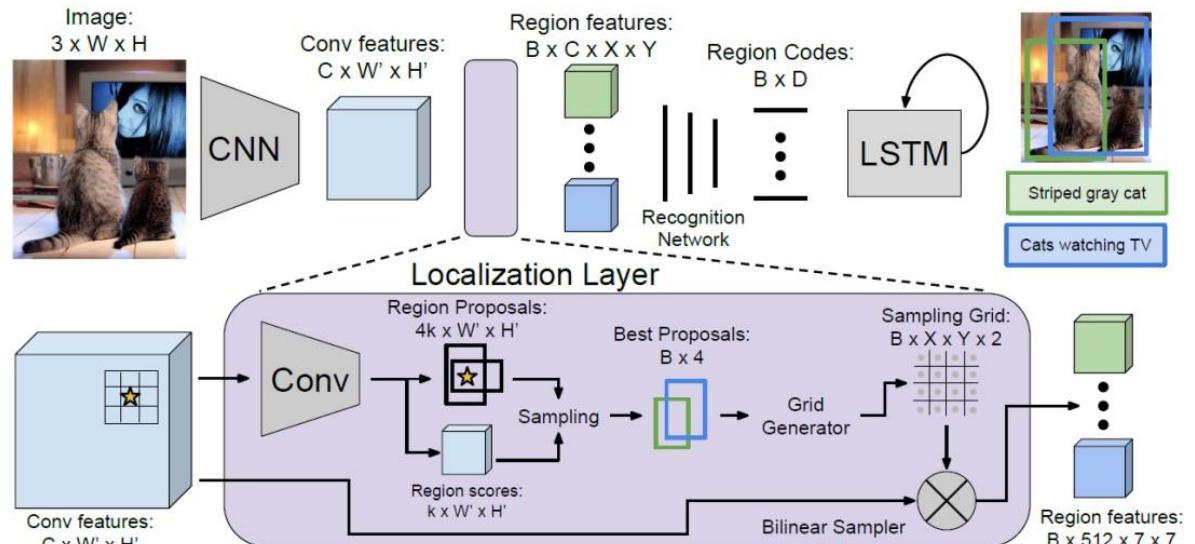
Mehr Regionen

[Johnson et al. 2016]

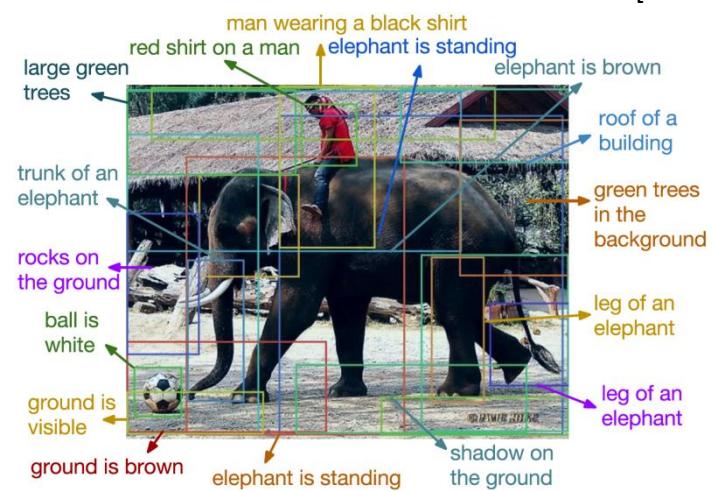
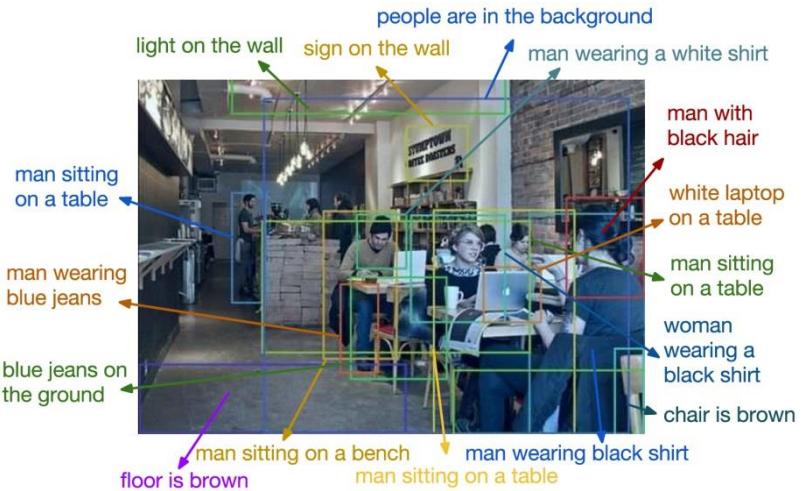
Komplexere Label

DenseCap

- Tiefe konvolutionale Netze
- Generierung zahlreicher Objekthypothesen
- Ausschneiden von Regionen
- Generierung von Regionenbeschreibungen

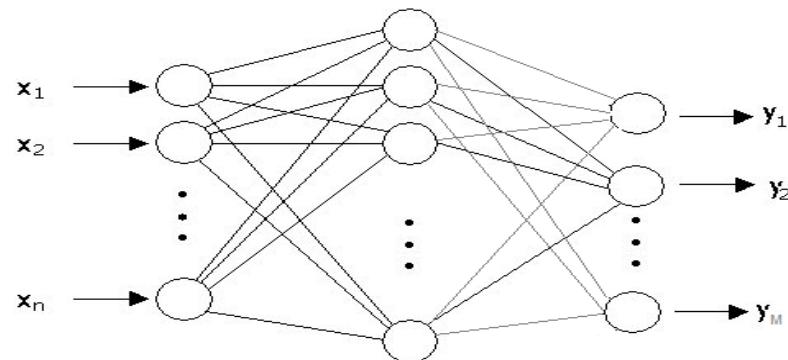


[Johnson et al. 2016]



Eigenschaften Neuronaler Netze

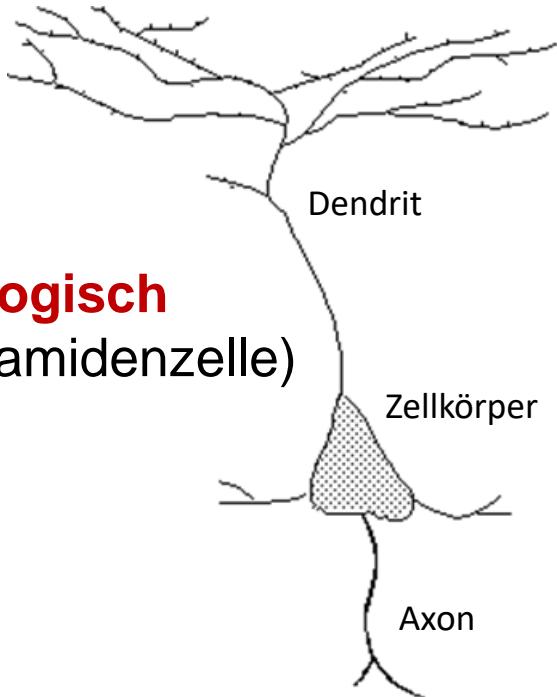
- Lernfähigkeit
- Generalisierungsfähigkeit
- Fehlertoleranz (etwa bei verrauschten Daten)
- Parallelität



Biologische vs. Künstliche Neuronen

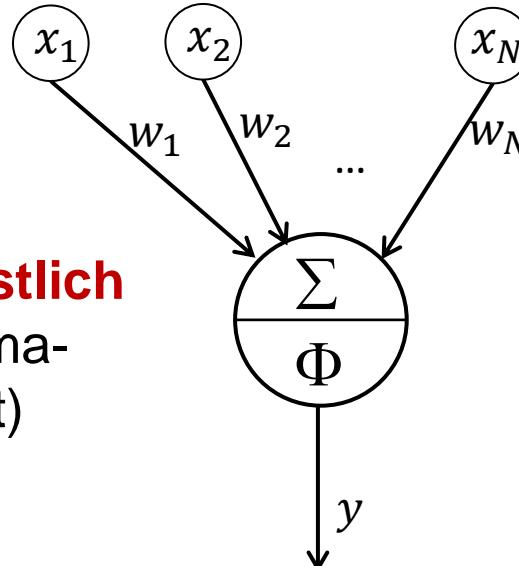
Biologisch

(Pyramidenzelle)



Informationsfluss

Künstlich
(Sigma-
Unit)



Eingaben

Gewichte

Integrationsfunktion

Transferfunktion

Ausgabe

$$y = \Phi\left(\sum_j^N w_j x_j\right)$$

Die Integrationsfunktion

■ Die Integrationsfunktion berechnet aus dem Eingabevektor $x = (x_1, x_2, \dots, x_N)$ und dem Gewichtsvektor $w = (w_1, w_2, \dots, w_N)$ den Nettoinput a des Neurons

■ Übliche Integrationsfunktionen:

- **Summe:** $a = \sum_{i=1}^N w_i x_i$

- **Distanz:** $a = \sum_{i=1}^N (w_i - x_i)^2$

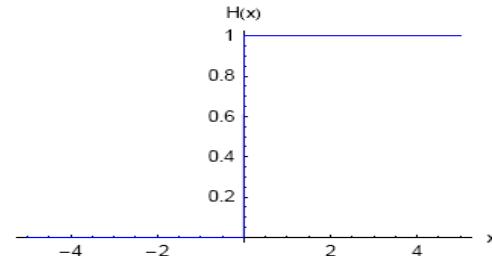
- **Produkt:** $a = \prod_{i=1}^N x_i^{w_i}$

- **Maximum:** $a = \max_i w_i x_i$

AKTIVIERUNGSFUNKTIONEN II

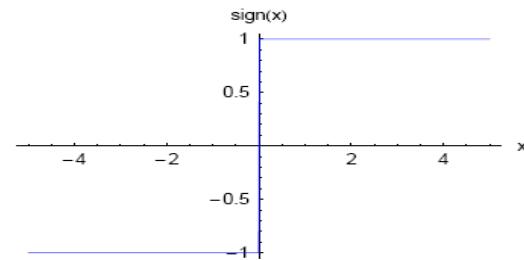
■ Unipolare Schwellwertfunktion

$$H(x) = \begin{cases} 0 & \text{falls } x < 0 \\ 1 & \text{sonst} \end{cases}$$



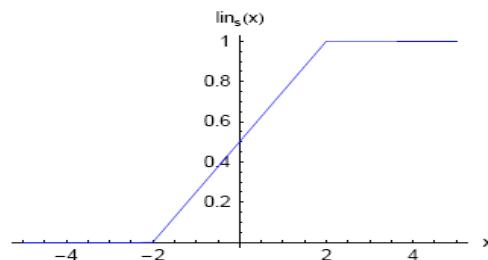
■ Bipolare Schwellwertfunktion

$$\text{sign}(x) = \begin{cases} -1 & \text{falls } x < 0 \\ 1 & \text{sonst} \end{cases}$$



■ Linear bis Sättigung

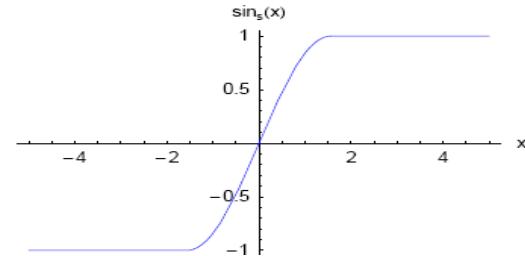
$$\text{lin}_s(x) = \begin{cases} 0 & \text{falls } x < \frac{a}{s} \\ s \cdot x - a & \text{falls } \frac{a}{s} \leq x \leq \frac{1+a}{s} \\ 1 & \text{sonst} \end{cases}$$



AKTIVIERUNGSFUNKTIONEN

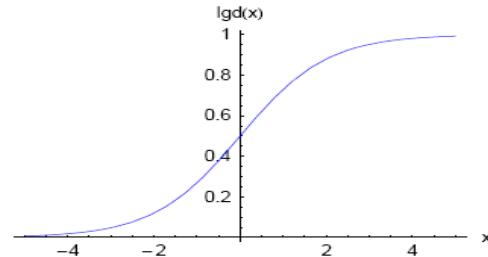
■ Sinus bis Sättigung

$$\sin_s(x) = \begin{cases} -1 & \text{falls } x < -\frac{\pi}{2} \\ \sin(x) & \text{falls } -\frac{\pi}{2} \leq x \leq \frac{\pi}{2} \\ 1 & \text{sonst} \end{cases}$$



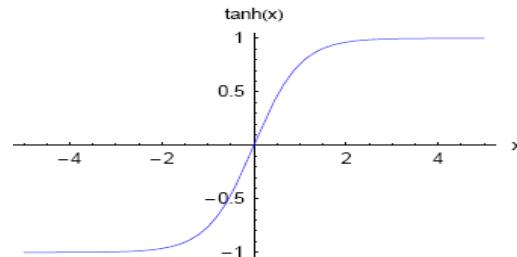
■ Logistische Funktion

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



■ Tangens Hyperbolicus

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$



AKTIVIERUNGSFUNKTIONEN III

■ Rektifizierend

● Stückweise linear

$$f_{\text{lt}}(x) = \begin{cases} 0 & : x \leq 0 \\ \alpha x & : x > 0 \end{cases}$$

● Glatt

$$f_{\text{st}}(x) = \frac{\log(1 + e^{\beta x})}{\beta}$$

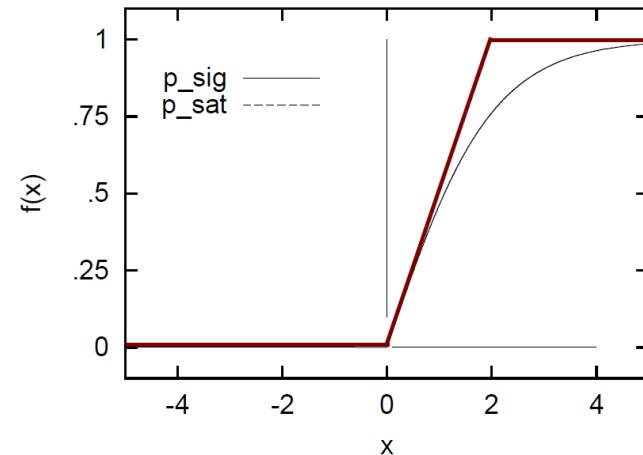
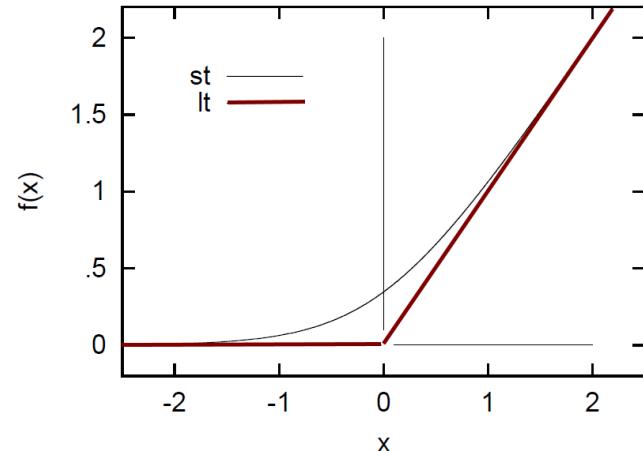
■ Rektifizierend mit Sättigung

● Stückweise linear

$$f_{\text{p-sat}}(x) = \begin{cases} 0 & : x \leq 0 \\ \frac{x}{\alpha} & : 0 < x < \alpha \\ 1 & : x \geq \alpha \end{cases}$$

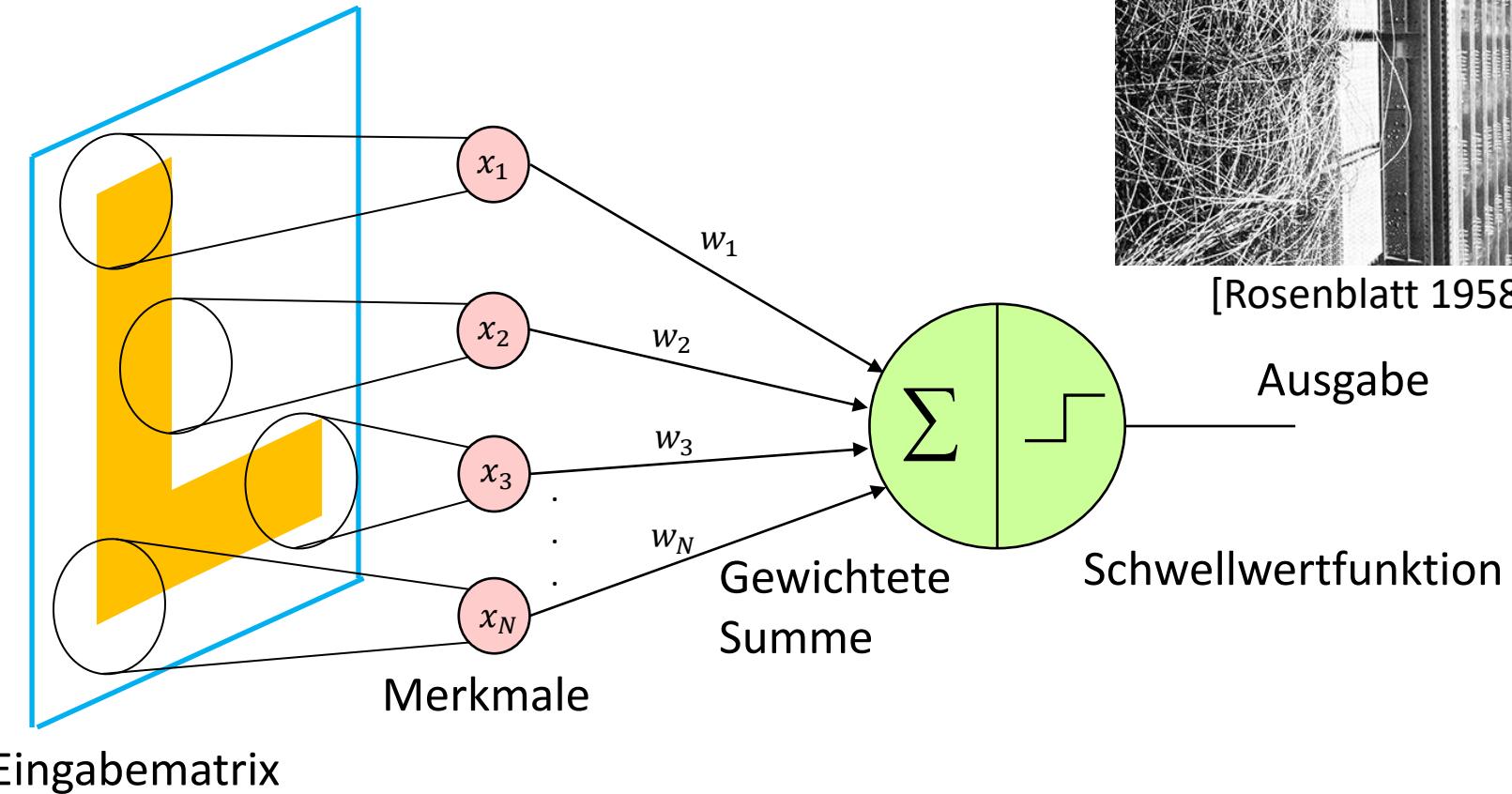
● Glatt

$$f_{\text{p-sig}}(x) = \max(0, \frac{2}{1 + e^{-\beta x}} - 1)$$



Perzepton

- Merkmalsextraktion => Mustererkennung



Skalarprodukt berechnet Ähnlichkeitsmaß

- Skalarprodukt berechnet Ähnlichkeit zwischen Gewichtsvektor und Eingabevektor

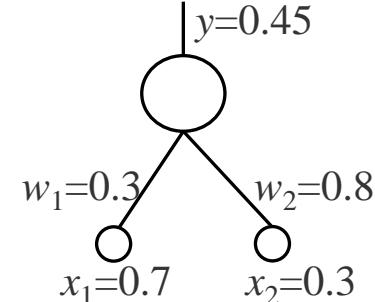
$$y = \Phi\left(\sum_{i=1}^N w_i x_i\right), \Phi(a) = a$$

$$\longrightarrow y = \mathbf{w} \cdot \mathbf{x}$$

- Winkel zwischen Vektoren

$$\|\mathbf{x}\| = \sqrt{\mathbf{x} \cdot \mathbf{x}} = \sqrt{x_1^2 + x_2^2 + \dots + x_n^2}$$

$$\cos \alpha = \frac{\mathbf{w} \cdot \mathbf{x}}{\|\mathbf{w}\| \|\mathbf{x}\|}, 0 \leq \alpha \leq \pi$$



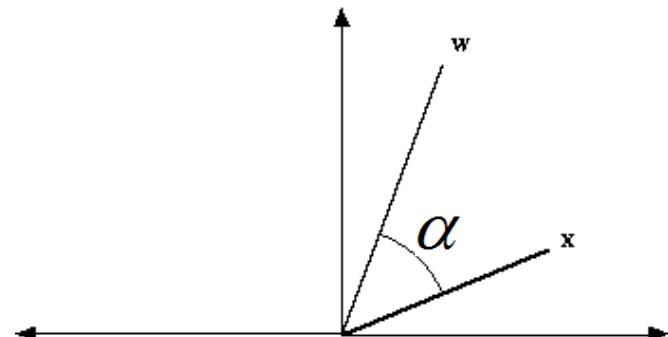
- Ausgabe signalisiert Ähnlichkeit

$$\alpha = 0^\circ \rightarrow \cos \alpha = 1,$$

$$\mathbf{w} \cdot \mathbf{x} = \|\mathbf{w}\| \|\mathbf{x}\| \cos \alpha$$

$$\alpha = 90^\circ \rightarrow \cos \alpha = 0,$$

$$\alpha = 180^\circ \rightarrow \cos \alpha = -1$$

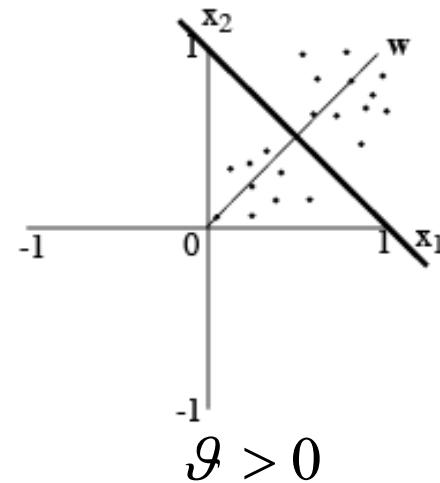
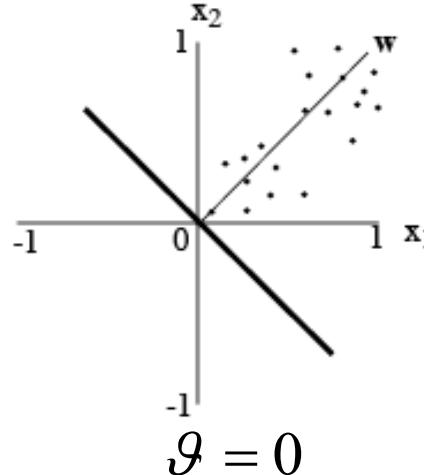


Trennung von Eingabemustern

- Skalarprodukt teilt Eingaberaum in zwei Regionen:
Eine mit Wert ≥ 0 und eine mit Wert < 0
- Trennlinie durch Gewichte definiert (orthogonal zu Gewichtsvektor)

$$w_1x_1 + w_2x_2 - \vartheta = 0$$

$$x_2 = \frac{\vartheta}{w_2} - \frac{w_1}{w_2} x_1$$



Bias

- Gewichtete Summe nicht vollständig

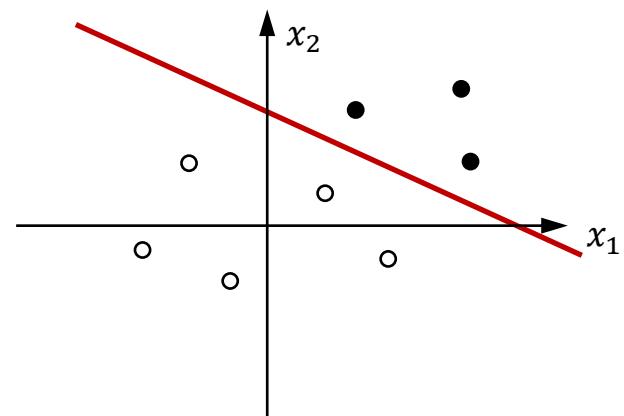
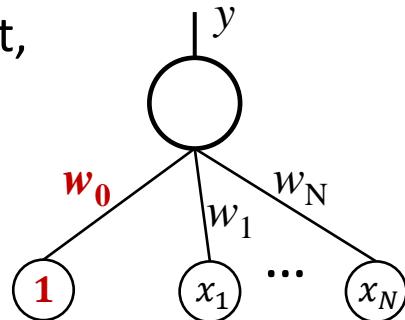
$$y_j = \sum_{i=1}^N w_i x_i$$

$x_1 = x_2 = \dots = x_n = 0 \Rightarrow y = 0$, d.h. Treنngerade geht immer durch Ursprung

- Konstante notwendig

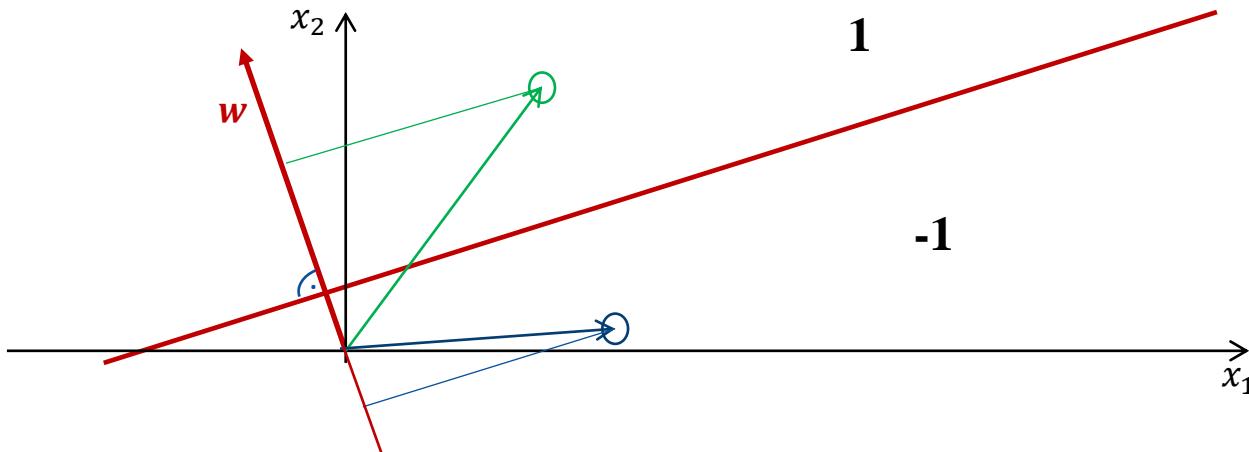
$$y = \sum_{i=1}^N w_i x_i + w_0$$

- Realisierung: Zusätzliche Unit, die immer auf 1 gesetzt wird (Bias Unit)



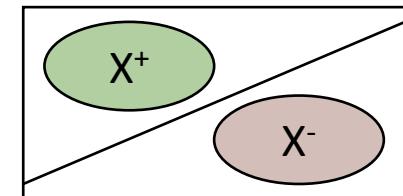
Geometrische Interpretation

- Die Gleichung $wx = \theta$ beschreibt eine Hyperebene im N-dimensionalen Raum, auf der der Gewichtsvektor w senkrecht steht.
- Der Eingaberaum wird dadurch in zwei Hälften geteilt, auf dessen einer Hälfte ($wx < \theta$) bei Schwellwertelementen -1 oder 0 ausgegeben wird, auf der anderen Hälfte Ausgabe 1.
- Netto-Aktivität = Projektion der Eingabevektoren auf den Gewichtsvektor



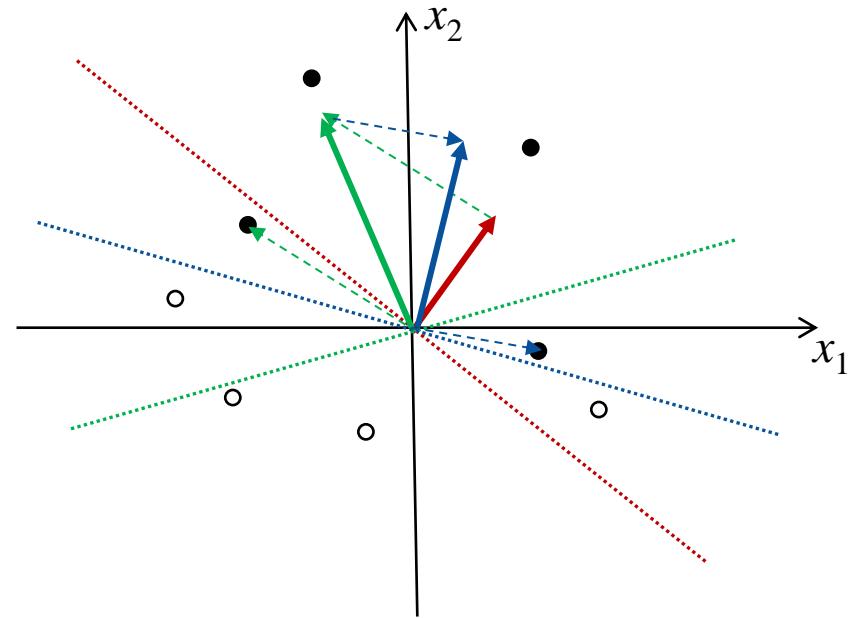
Binäre Klassifikation

- Das zu lernende Problem besteht aus zwei Mengen X^+ , X^- von erweiterten Eingabevektoren, wobei X^+ auf 1 und X^- auf 0 abgebildet werden soll
- Gesucht ist also ein (erweiterter) Gewichtsvektor \mathbf{w} mit $\mathbf{x}\mathbf{w} > 0$ wenn $\mathbf{x} \in X^+$ und $\mathbf{x}\mathbf{w} < 0$ (d.h. $-\mathbf{x}\mathbf{w} > 0$) für alle $\mathbf{x} \in X^-$
- In der geometrischen Interpretation bedeutet das:
Wir suchen eine Hyperebene, welche die Mengen X^+ und X^- voneinander trennt
- Als weitere Vereinfachung können wir $Z = X^+ \cup -X^-$ betrachten, für das nur noch die Bedingung $\mathbf{z}\mathbf{w} > 0$ für alle $\mathbf{z} \in Z$ erfüllt werden muss



Perzeptron-Lernalgorithmus

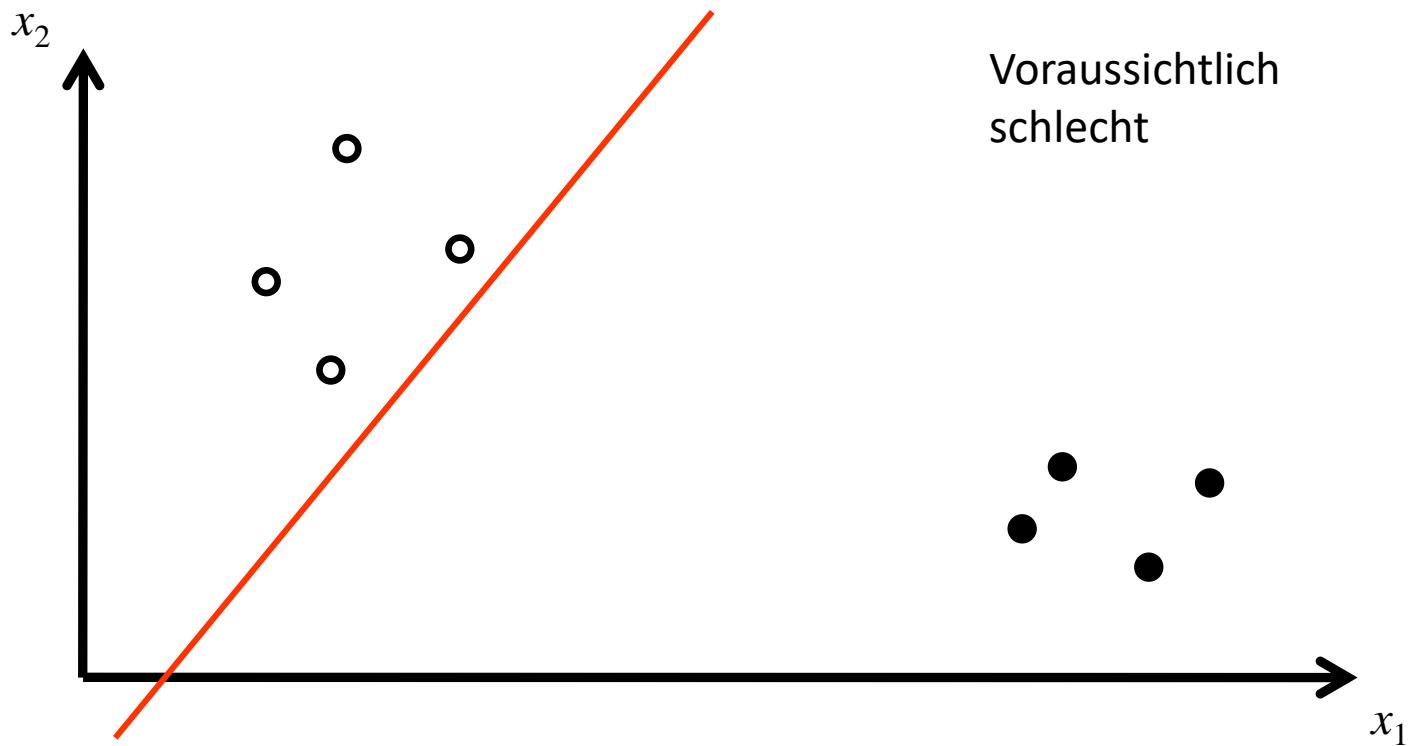
- Eingaben: Datenpunkte aus zwei Klassen (hier 2D gezeichnet in Ebene)
- Aufgabe: Trenne Beispiele der beiden Klassen (weiß und schwarz)
- Lerner (Versionsraum): Trengerade durch den Ursprung
- Lernregel:
 - Identifizierte falsch klassifiziertes Beispiel
 - Addiere dieses zum Gewichtsvektor
 - Dadurch entsteht neue Trengerade, senkrecht zum Gewichtsvektor
 - Solange bis alles richtig klassifiziert wird
- Generalisierung: Neue Punkte richtig klassifiziert?



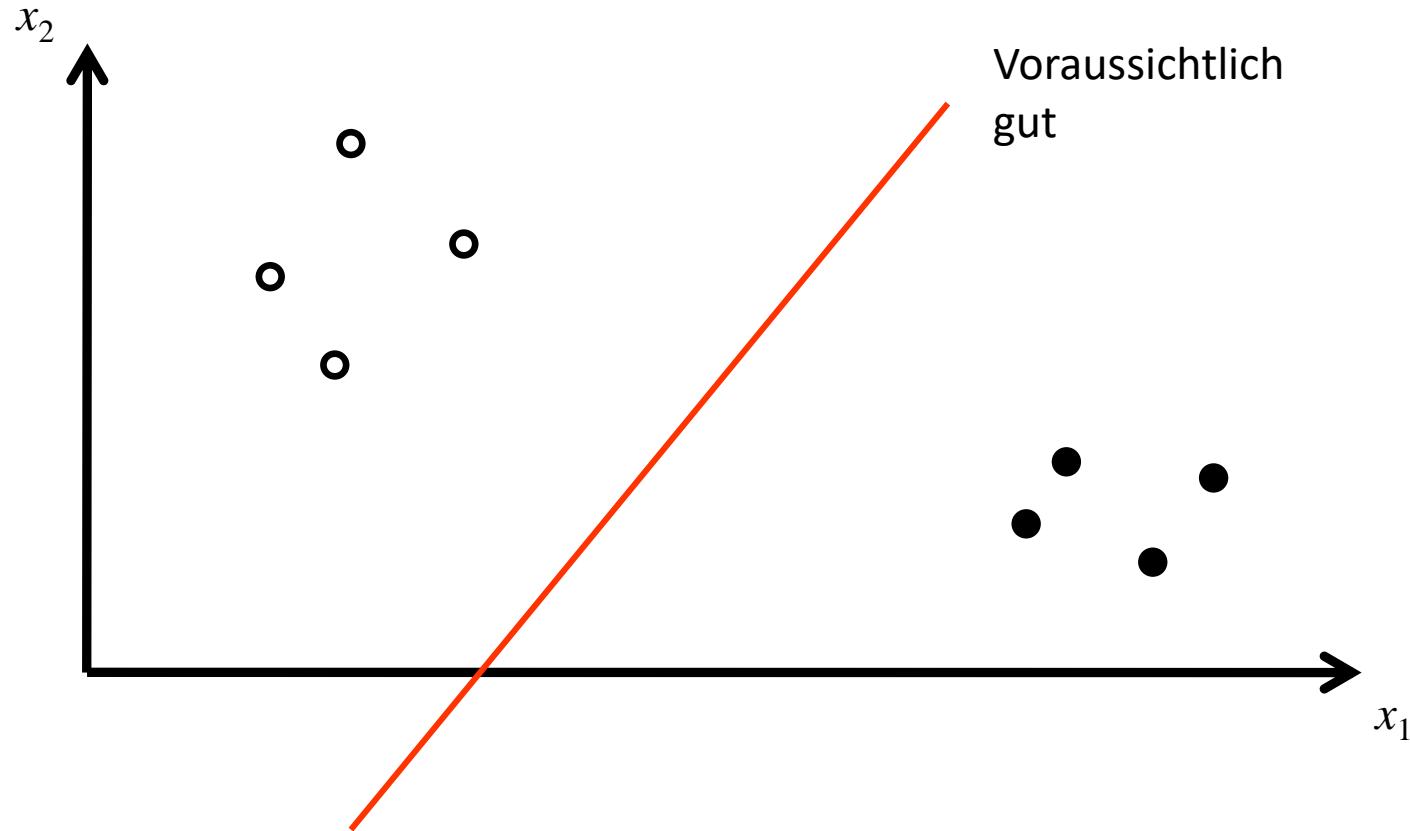
Lernen = Generalisierung

- H. Simon: “Learning denotes changes in the system that are adaptive in the sense that they enable the system to do the task or tasks drawn from the same population more efficiently and more effectively the next time.”
- Fähigkeit neue, ungewohnte Aufgaben zu lösen

Generalisierung



Generalisierung



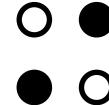
Korrektheit

- Der Perzeptron Lernalgorithmus kann nun wie folgt interpretiert werden:
 - Auf Z soll das Perzeptron mit 1 antworten, also $\mathbf{z} \mathbf{w} > 0$ für ein geeignetes \mathbf{w} gelten. Wir beginnen mit einem beliebigen \mathbf{w}_0 .
 - Ist $\mathbf{z} \in Z$ ein Element, das noch nicht korrekt interpretiert wird ($\mathbf{z} \mathbf{w} \leq 0$), dann erzeugt die Lern-Regel ein neues $\mathbf{w}_{\text{neu}} := \mathbf{w}_{\text{alt}} + \mathbf{z}$ und beim nächsten Versuch mit demselben \mathbf{z} ist $\mathbf{z} \mathbf{w}_{\text{neu}} = \mathbf{z} \mathbf{w}_{\text{alt}} + \mathbf{z} \mathbf{z} > \mathbf{z} \mathbf{w}_{\text{alt}}$.
 - Das Verfahren beginnt also mit einem beliebigen erweiterten Gewichtsvektor \mathbf{w}_0 und addiert (subtrahiert) sukzessive erweiterte Eingaben \mathbf{z} , bei denen die Ausgabe noch nicht korrekt ist.

Perzeptron-Konvergenz-Satz

■ Konvergenz garantiert, wenn Problem lösbar (Rosenblatt 1962)

■ Beispiel für nicht linear trennbare Mengen:



■ Satz:

Wenn das Perzepton eine Klasseneinteilung überhaupt lernen kann, dann lernt es diese mit der Perzeptron-Lernregel in endlich vielen Schritten.

■ Problem:

Wenn der Lernerfolg bei einem Perzepton ausbleibt, kann nicht direkt erkannt werden, ob

1. das Perzepton die Klassifikation prinzipiell nicht lernen kann oder

2. der Lernalgorithmus mit seinen endlich vielen Schritten noch nicht fertig geworden ist,

denn der Satz als reiner Existenzsatz gibt keinen Anhaltspunkt über eine obere Schranke für die Anzahl von Lernschritten.

Beweis

- Nach Voraussetzung gibt es ein \mathbf{w}^* mit $\mathbf{w}^* \mathbf{z} > 0$ für alle $\mathbf{z} \in Z$ und wir beginnen den Algorithmus mit $\mathbf{w}_0 = 0$.
- Sei a das Minimum aller $\mathbf{w}^* \mathbf{z} > 0$ und M das Maximum aller \mathbf{z}^2 .
- $\mathbf{w}_i = \mathbf{z}_1 + \dots + \mathbf{z}_i$, also ist $\mathbf{w}^* \mathbf{w}_i = \mathbf{w}^* \mathbf{z}_1 + \dots + \mathbf{w}^* \mathbf{z}_i \geq i \cdot a$.
- Wegen $(\mathbf{w}^* \mathbf{w}_i)^2 \leq \mathbf{w}^{*2} \mathbf{w}_i^2$ folgt $\underline{\mathbf{w}_i^2} \geq i^2 \cdot a^2 / \mathbf{w}^{*2}$
- Andererseits ist für alle $k \leq i$ stets $\mathbf{w}_{k-1} \mathbf{z}_k < 0$
- Also $\mathbf{w}_k^2 = (\mathbf{w}_{k-1} + \mathbf{z}_k)^2 = \mathbf{w}_{k-1}^2 + 2 * \mathbf{w}_{k-1} \mathbf{z}_k + \mathbf{z}_k^2 < \mathbf{w}_{k-1}^2 + \mathbf{z}_k^2$
- Also ist $\underline{\mathbf{w}_i^2} < \mathbf{z}_1^2 + \dots + \mathbf{z}_i^2 < i \cdot M$
- Es folgt $i \cdot M > \underline{i^2 \cdot a^2 / \mathbf{w}^{*2}}$ und damit $i < M \cdot \mathbf{w}^{*2} / a^2$

D.h. der Algorithmus terminiert in endlich vielen Schritten.

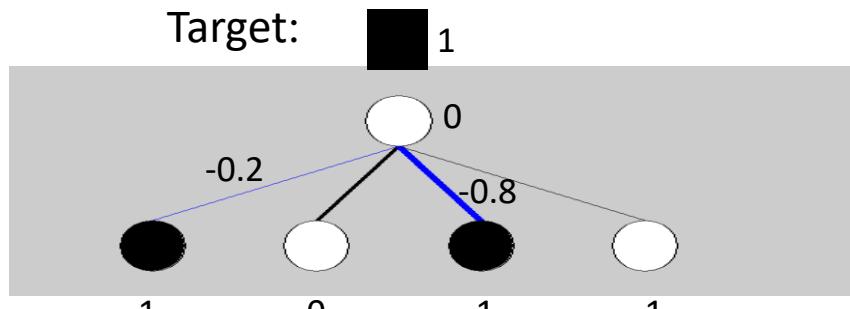
Aber: Ohne Kenntnis von \mathbf{w}^* kennen wir weder \mathbf{w}^{*2} noch a , also können wir die obere Schranke von i nicht aus dem Beweis bestimmen!

Frage: Wann kann der Algorithmus lange laufen?

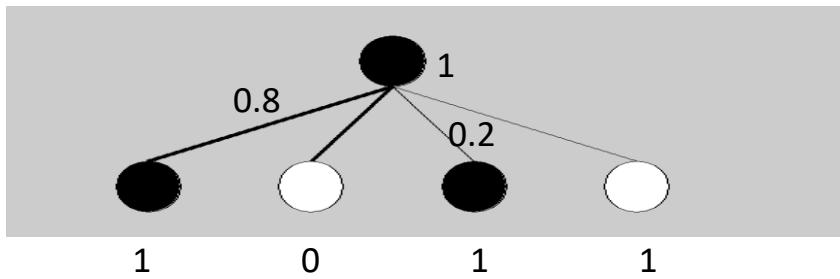
Perzeptron-Lernregel als Gewichtsadaption

- Binäre Ein- und Ausgaben
- Zielvorgabe (target)
notwendig = Lehrer
- Input wird addiert
(subtrahiert), wenn
Ausgabe falsch

$$\begin{aligned}\Delta w_{ij} &= x_i \text{ falls } y_j < t_j \\ &= -x_i \text{ falls } y_j > t_j \\ &= 0 \dots \text{sonst} \\ t_j &\dots \text{"target"}$$



Nach dem Lernschritt:



Delta-Regel

- Das Perzeptron kann auch nach der Delta-Regel trainiert werden.
- Beim Training werden dem Netzwerk Eingaben x_i aus einer Trainingsdatei präsentiert, für welche die gewünschten Ausgaben t_i bekannt sind.
- Die aktuellen Ist-Ausgaben des Netzes werden mit den Soll-Ausgaben verglichen und im Fall einer Diskrepanz die Gewichte und der Schwellwert nach folgender Formel angepasst:

Delta-Regel:

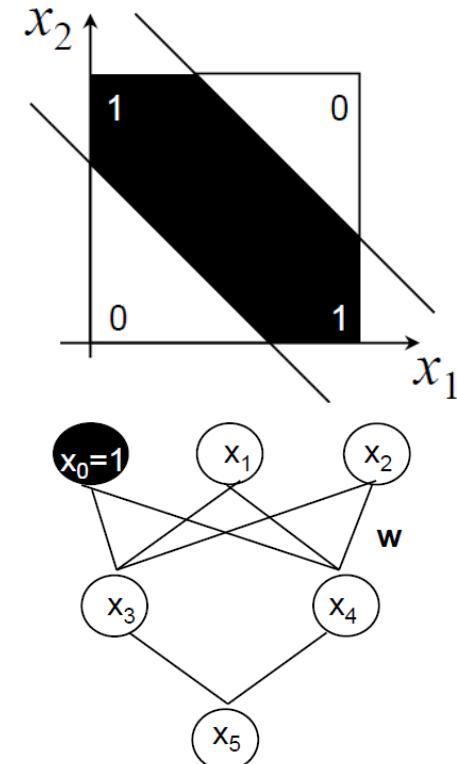
$$w_{i,\text{neu}} := w_{i,\text{alt}} + x_i(t_i - y_i)$$

Gewünschte Ausgabe Tatsächliche Ausgabe

- Der Faktor x_i sorgt dafür, dass eine Angleichung nur erfolgt, wenn die Verbindung w_i tatsächlich zu der Ausgabe beiträgt.

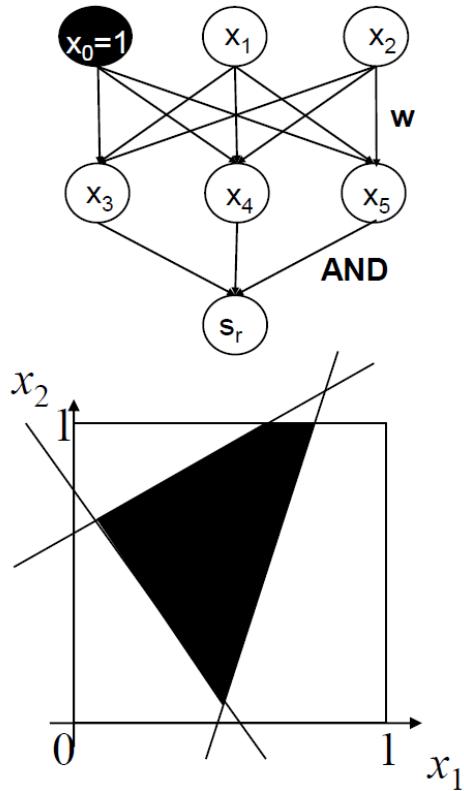
XOR-Problem

- Die boolesche Funktion XOR stellt ein nichtlinear separierbares Problem dar, weil sich die beiden Klassen auf Diagonalen gegenüberliegen.
- Mit einer Zwischenschicht lassen sich aber zwei trennende Hyperebenen definieren, zwischen denen die eine der Klassen lokalisiert werden kann.
- Komplexere Klassifikationen lassen sich durch Einfügen von Zwischenschichten realisieren.



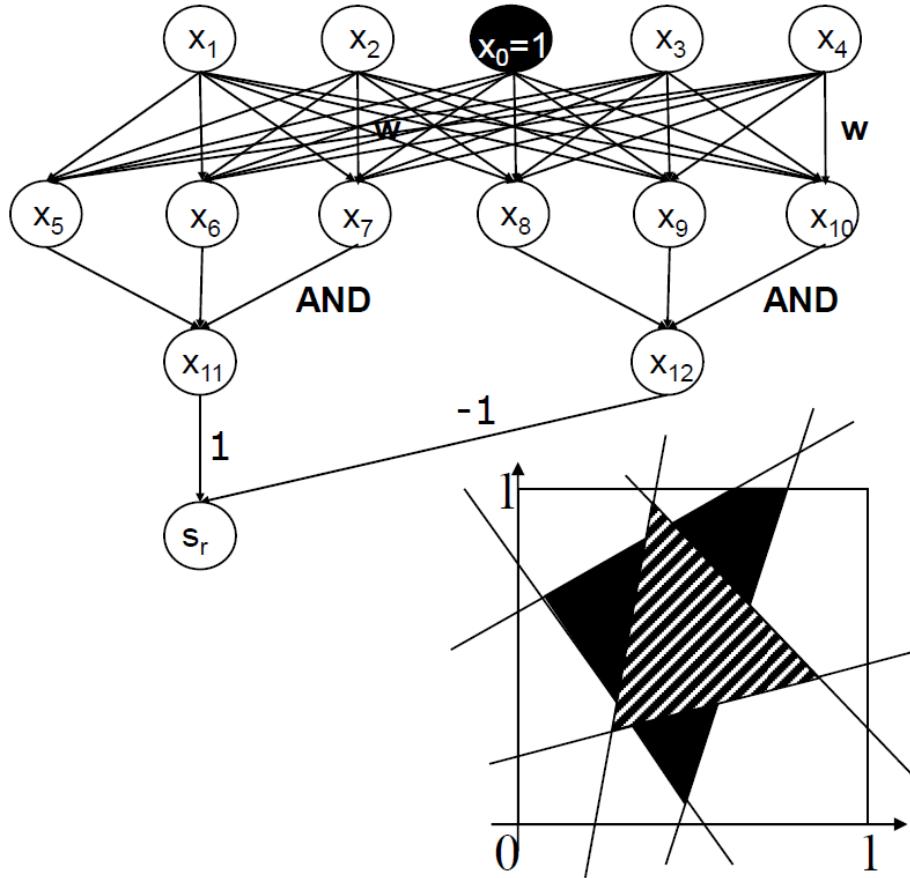
Zweistufiges Perzeptron

- Wenn man an ein Perzeptron mit M Ausgabe-Neuronen ein neues Ausgabe-Neuron mit AND-Verbindungen (Gewicht 1, Schwellwert $M-0,5$) anhängt, erhält man ein **konvexes M -flach** (den Durchschnitt von M Halbräumen) als akzeptierten (Ausgabe 1) Bereich.



Dreistufiges Perzeptron

- Mit einer weiteren Stufe durch AND NOT (Gewicht 1,-1, Schwellwert 0,5) angehängt, kann man sogar nicht zusammenhängende Bereiche als akzeptierten Bereich modellieren. (In dem Beispiel der sichtbare dunkle Bereich)



Problem: Wie trainiert man mehrere Schichten?

- Übersicht

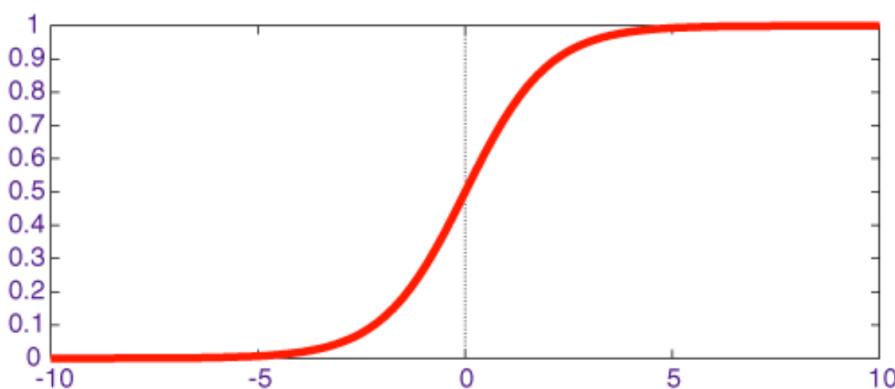
- Definieren einer Fehlerfunktion $E(w)$
- Leite Fehler nach Gewichten w ab
- Ändere w so, dass der Fehler $E(w)$ kleiner wird

- Zuvor müssen wir dafür sorgen dass wir ableiten können:

$$y = \text{sign}(\vec{w}^T \vec{x} - \theta) = \text{sign}\left(\sum_{i=1}^n w_i x_i - \theta\right)$$

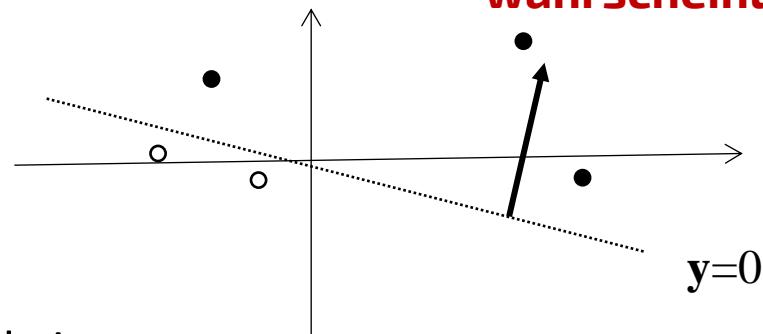
- Sign ist nicht stetig! → Approximation mit Sigmoider Funktion

Sigmoide Transferfunktion



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

Immer
wahrscheinlicher



- Ausgaben begrenzt auf [0,1]
- Quasi-linear um Nettoinput $a=0$
- Mögliche Interpretation: Wahrscheinlichkeit

Überwachtes Lernen

Ziel: finde für eine durch Beispiele gegebene Funktion $y=f(x)$ die Gewichte w_{ij} so, dass f möglichst gut durch die Netzwerkfunktion approximiert wird

Sei $\{(x_1, t_1), \dots, (x_p, t_p)\}$ eine Menge von Trainingsbeispielen

Sei w der Gewichtsvektor

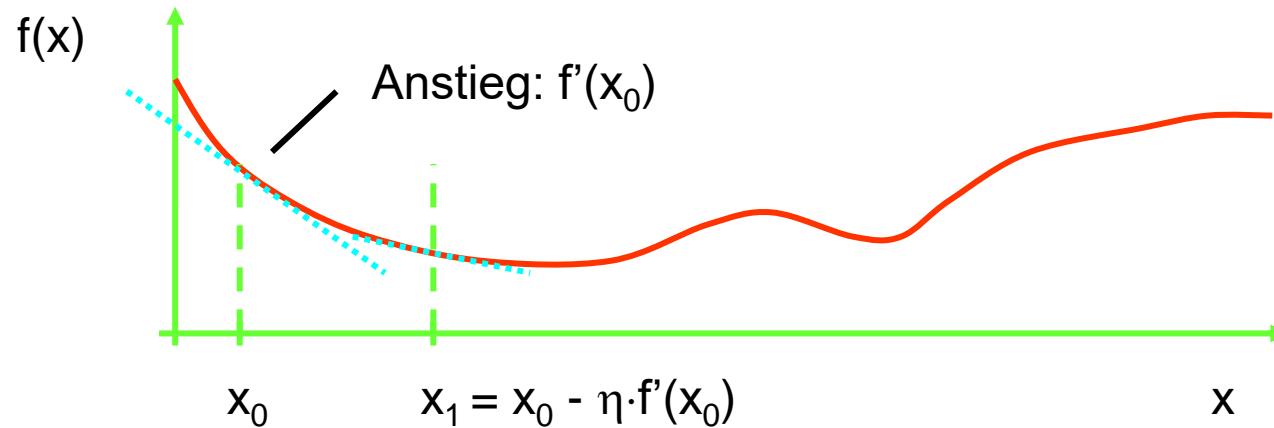
Das Netzwerk berechnet zur Eingabe x_i die Ausgabe y_i

Optimiere z.B. **quadratische Fehlerfunktion**

$$E(w) = 1/2 \sum_{i=1}^p (y_i - t_i)^2$$

Gradientenabstieg

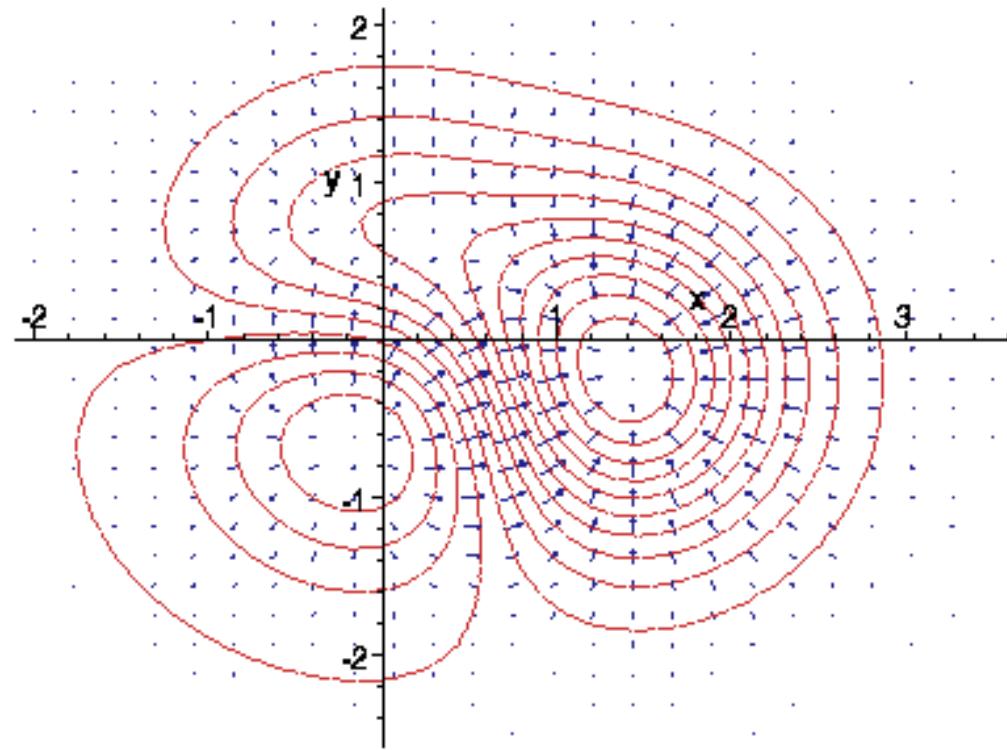
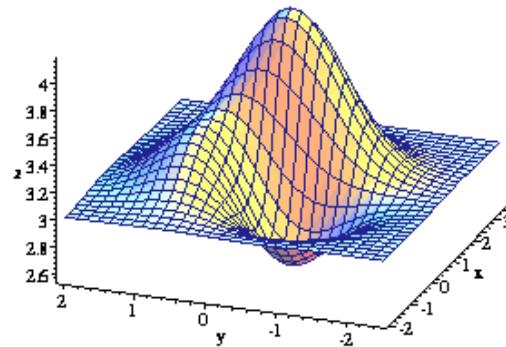
■ 1D-Beispiel: minimiere $f(x)$



Iteriere bis $f'(x_i)$ verschwindet!

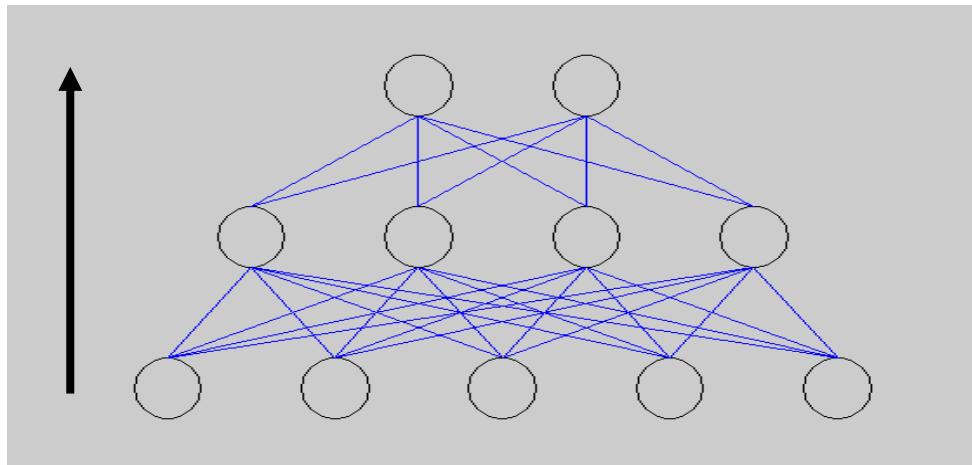
Gradientenabstieg

- 2D-Beispiel
- Folge den Pfeilen



Mehrebenen-Perceptron (MLP)

- Zwei (oder mehrere) Schichten



Ausgabe-Neuronen
(linear oder sigmoid)

Verdeckte Neuronen
(typisch sigmoid)

Eingaben