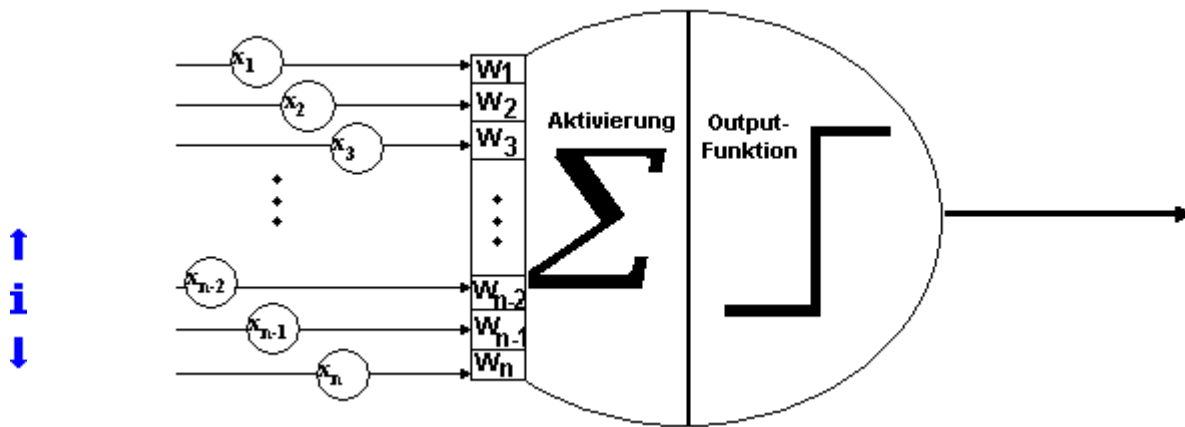


KünstlicheNeuronaleNetzwerke



Heinrich Werner
Universität Kassel

DerprinzipielleAufbauneuronalerNetze

In den folgenden Kapiteln wird der Begriff **neuronales Netz(-werk)** immer für künstliche neuronale Netzwerke stehen, andernfalls wird immer **biologisch** oder **natürlich** vorgeschrieben.

Um ein neuronales Netzwerk zu konstruieren hat man folgende Schritte durchzuführen:



1. **Eingabeschicht:** wählen Inputneuronen, n = Dimension des Eingabevektors.
2. **Ausgabeschicht:** wähle k Outputneuronen, k = Dimension des Ausgabevektors.
3. **Verarbeitungsschichten:** hier sind die Neuronen und Verbindungen zu wählen.
4. **Neuronenart:** wähle für alle Neuronen deren Aktivierungsregel und die Funktion, die aus der Aktivität den Output berechnet.
5. **Rückkopplung:** wenn es Rückkopplung gibt, ist festzulegen, wann das System stabil zu gelten hat und in welcher Reihenfolge die Neuronen aktualisiert werden müssen.
6. **Lern-Regel:** lege fest, wie das Netzwerk an die zu trainierenden Daten angepaßt wird.

Das McCulloch-Pitts Neuronenmodell

Das erste Neuronenmodell ist das **McCulloch-Pitts-Neuron** (1943), das nur zwei Outputs 0 (**inaktiv**) und 1 (**aktiv**) kennt.

Ein solches Neuron hat nur zwei verschiedene Typen von Synapsen mit den Synapsengewichten +1 (**excitatorisch**) oder -1 (**inhibitorisch**).

Die Aktivierungsfunktion: es werden excitatorische und inhibitorische Inputs (jeweils 0 oder 1) getrennt aufsummiert und dann die Resultate voneinander abgezogen, um den Aktivierungszustand zu erhalten; der (Aktivierungs-) **Zustand** ist also eine ganze Zahl.

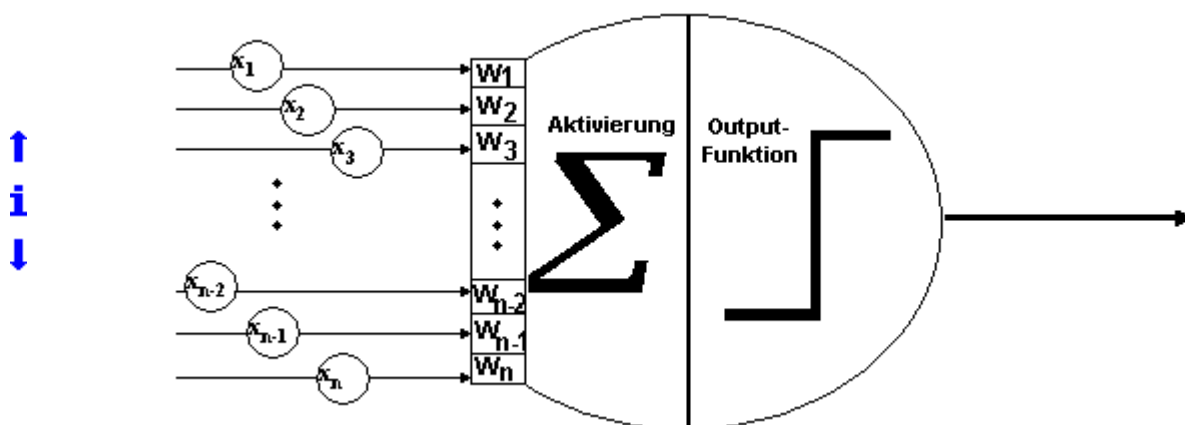
Die Outputfunktion ist eine **Schwellwertfunktion**, d.h. jedes Neuron hat einen bestimmten Schwellwert, und wenn der Aktivierungszustand diesen Schwellwert übersteigt, wird das Neuron aktiv (Output 1) und sonst bleibt das Neuron inaktiv (Output 0).

Diese starren Bedingungen haben sich in folgenden Punkten gelockert

- Outputs können beliebig im Intervall $[0, 1]$ liegen.
- Synapsengewichte und Schwellwert können beliebige reelle Zahlen sein.
- Aktivierungs- und Output-Funktionen können beliebige Funktionen sein.

Das künstliche Neuronenmodell

Der Input ist gekennzeichnet durch die **Aktivierungs-Funktion**. Diese legt fest, wie die über die Verbindungen eingehenden Reize x_i mit den Gewichten w_i zu einem Gesamtreiz zusammengefaßt werden.



Die **Output-Funktion** legt fest, wie der Output aus dem oben berechneten Gesamtreiz ermittelt wird. Der Output wird dann über die Verbindungen als Reiz an andere Neurone geleitet.

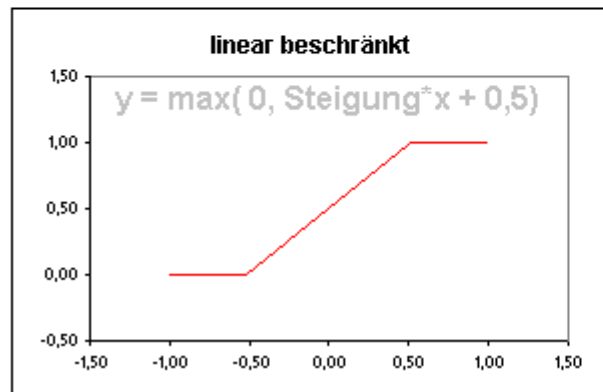
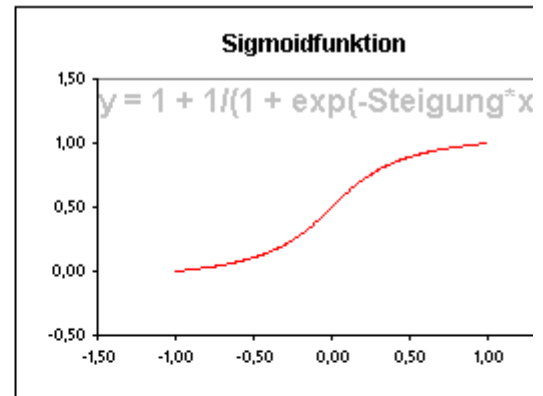
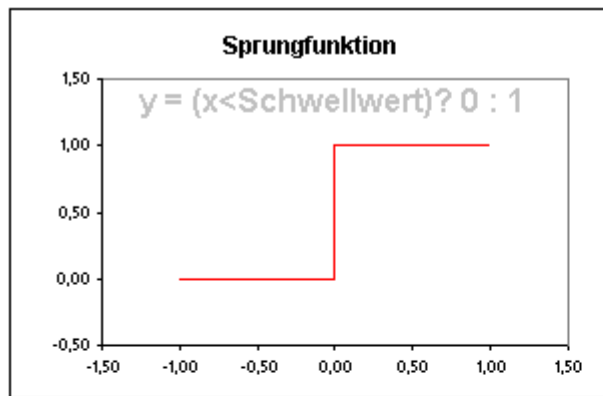
Die Aktivierungs-Funktionen der Neuronen

Es gibt i. w. vier Arten der Aktivierung von Neuronen:

Allgemein ist die **Modifikation** der Inputs. Die Inputs x_1, x_2, \dots, x_n werden stets mit ihren (Synapsen-) Gewichten w_1, w_2, \dots, w_n multipliziert: $a_1 = x_1 * w_1, a_2 = x_2 * w_2, \dots, a_n = x_n * w_n$.

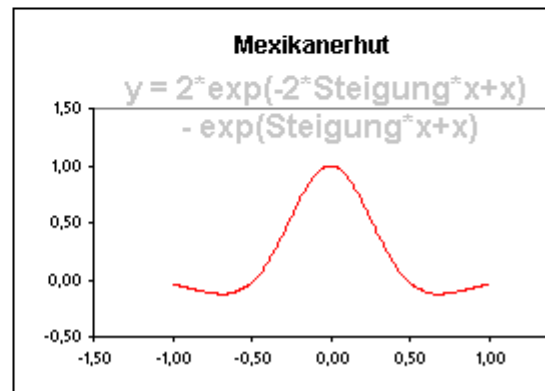
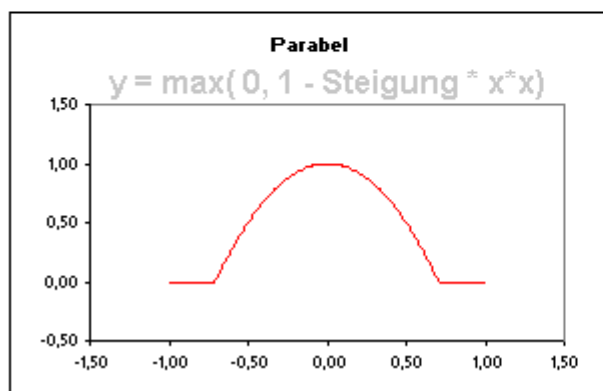
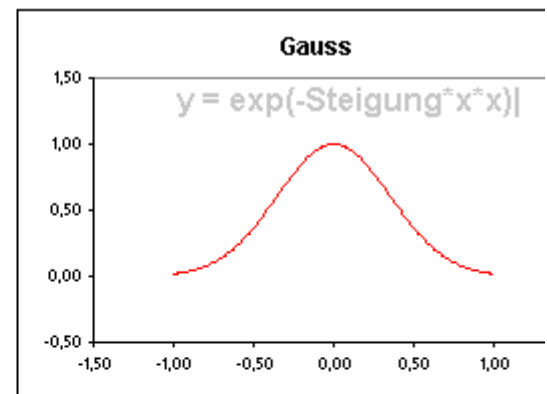
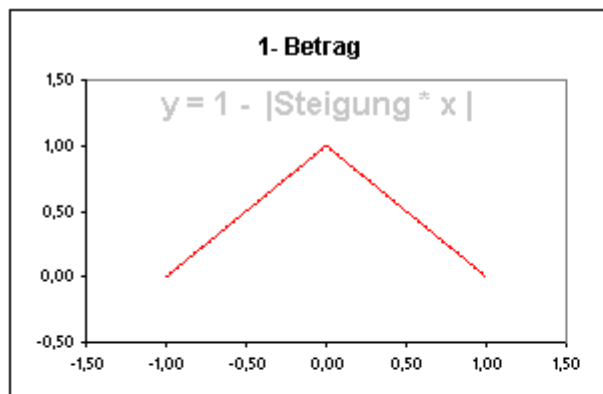
1. Die am häufigsten angewandte Regel ist die **(Skalarprodukt-Regel)**
Die modifizierten Inputs a_1, a_2, \dots, a_n werden zur Aktivität des Neurons aufaddiert: $a = a_1 + a_2 + \dots + a_n$
2. Sehr häufig ist ebenfalls die **Winner-take-all-Regel**
beide die Aktivität zunächst nach der Skalarproduktregel ermittelt wird, dann aber mit allen Aktivitäten in derselben Schicht verglichen wird und auf 0 herabgesetzt wird, wenn ein anderes Neuron höhere Aktivität hat.
3. Bisweilen kommt auch die **sigma-pi-Regel** vor
Die Inputs werden in Gruppen aufgeteilt und die modifizierten Inputs jeder Gruppe miteinander multipliziert, danach werden die Produkte p_1, p_2, \dots, p_k zur Aktivität des Neurons aufaddiert: $a = p_1 + p_2 + \dots + p_k$. Der Effekt ist, daß ein Input von fast 0 die gesamte Gruppe blockiert.
4. Selten ist die Regel, beide ebenfalls die Inputs in Gruppen eingeteilt werden (z.B. 2 Gruppen: inhibitorische und excitatorische) und Gruppenweisen nach Skalarproduktregel zusammengefasst werden. Die Ergebnisse werden dann mit einer nichtlinearen Funktion miteinander verknüpft (z.B. $a = (e^{-i})/|i|$)

Monotone Outputfunktionen



Alle Funktionen könnten wie die Sprungfunktion mit einem Schwell verschoben werden. Wie wir aber sehen werden kann auf diese Verschiebung verzichtet werden.

Radiale Basisfunktionen als Outputfunktionen



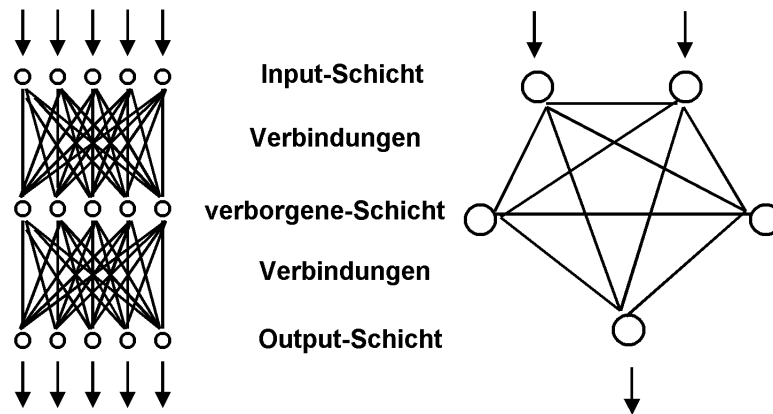


Experiment mit Outputfunktionen



Schichtenaufbau oder Rückkopplung

Für neuronale Netze wählt man entweder einen rückkopplungsfreien Schichtenaufbau oder man faßt die rückgekoppelten Neuronen zu Gruppen zusammen, die dann ihrerseits wieder Schichten bilden

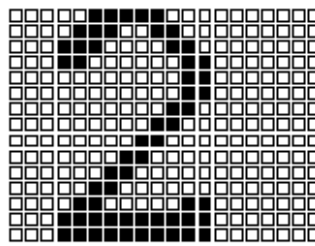
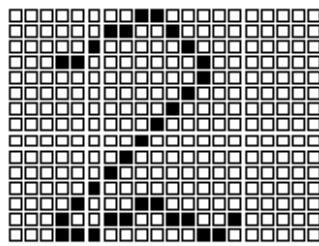


↑
i
↓

Trainingsaufgaben: Erkennung/Assoziation

WesentlicheAufgabefürNeuronaleNetzeistdie

Muster-Erkennung

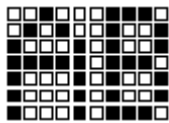


Input

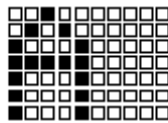
Output



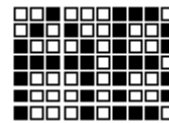
bzw. die **Muster-Assoziation**



Training



Input

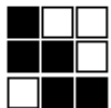
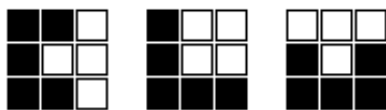
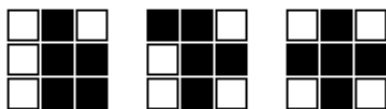
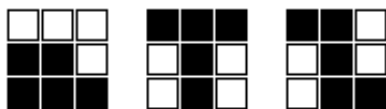


Output

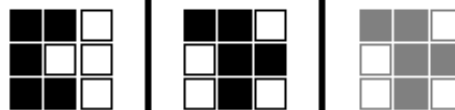
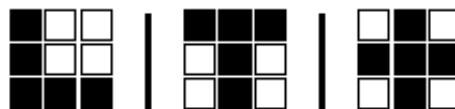
Trainingsaufgaben: Klassenbildung

EinandererAufgabenkreisistdie

Muster-Klassenbildung(Cluster)



Inputs



Outp

Clus

Trainingsaufgaben: Klassifikation

und die **Muster-Klassifikation** in existierende Klassen





Lernparadigmen

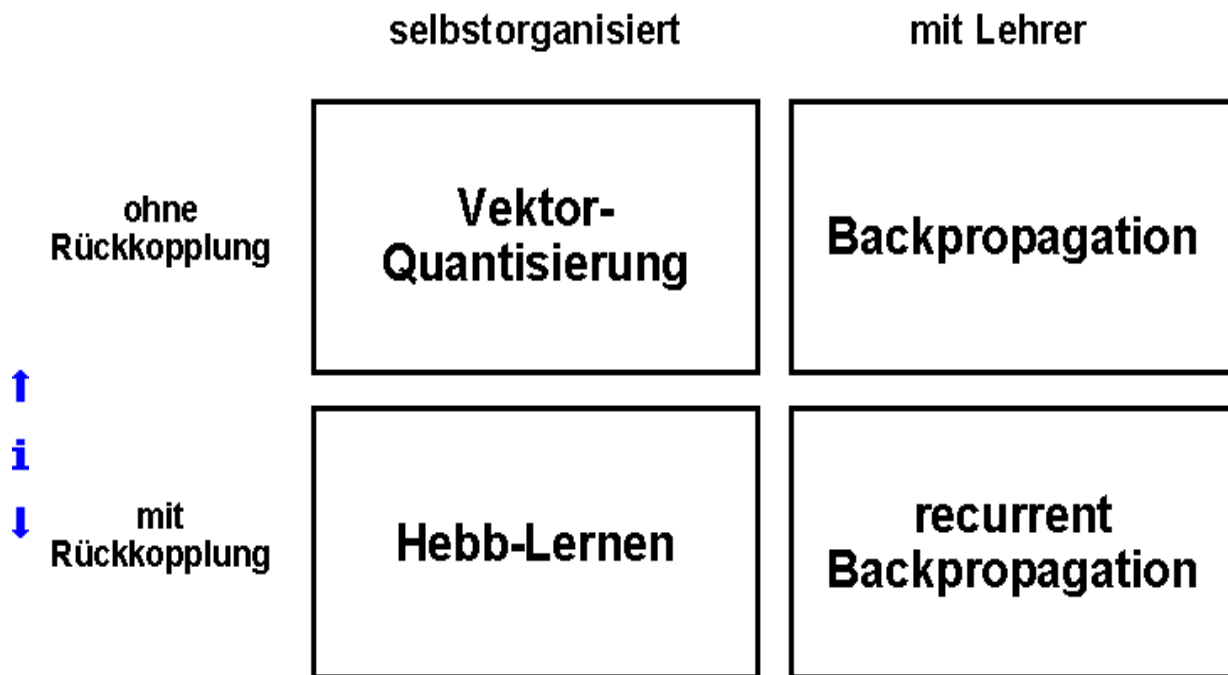
Das Lernen (Training) in neuronalen Netzwerken kann nach drei Paradigmen erfolgen:

1. **Überwachtes Lernen (supervised, mit Lehrer, output-oriented)**
Das Netzwerk verarbeitet den Trainingsinput und liefert sein Ergebnis, das der Lehrer mit dem gewünschten Output vergleicht. In Abhängigkeit vom Fehler wird das Netzwerk so verändert, daß dieser Fehler nicht mehr (in derselben Stärke) auftritt.
2. **Unüberwachtes Lernen (selforganized, durch Gewöhnung, input-oriented)**
Das Netzwerk verarbeitet die Inputs und verändert sich dabei so, daß es auf die präsentierten Inputs immer stärker reagiert.
3. **verstärkendes Lernen (reinforced, mit Belohnung, behavior-oriented)**
Das Netzwerk verarbeitet die Inputs und überprüft, ob seine Aktivitäten eine bestimmte vorgegebene Form haben. Je nach Antwort wird das Verhalten des Netzwerks verstärkt oder abgeschwächt.

Bei diesen Methoden kann man noch unterscheiden zwischen **online-(Einzelschritt)**, d.h. nach jeder Eingabe wird gelernt, und **offline(Gesamtschritt)-learning** d.h. erst nach Eingabe aller Trainingsdaten wird ein Anpassungsschritt gemacht.

Typische Beispiele für Lernmethoden

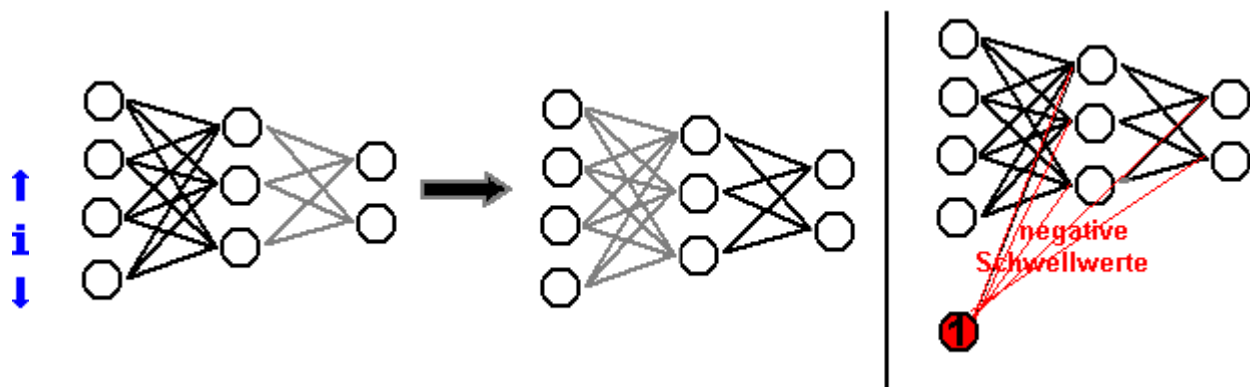
Die bekanntesten Methoden zu den einzelnen Lernparadigmen sind:



Lernen durch Gewichts-Anpassung

Die meisten Trainingsmethoden für Neuronale Netze basieren auf

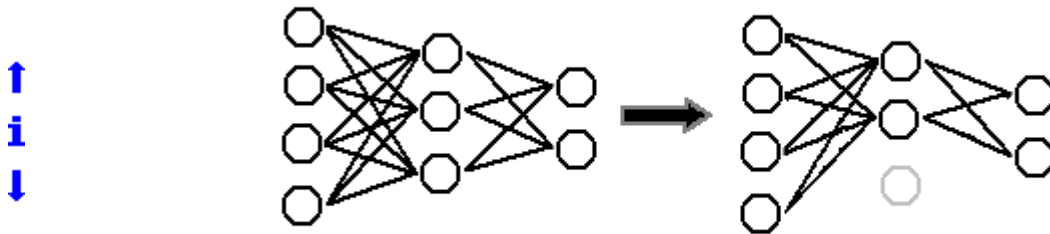
Gewichts-Anpassung



In frühen Modellen wurde auch mit der Veränderung des Schwellwerts gearbeitet, der kann aber durch ein zusätzliches Neuron, das immer 1 feuert, also ein Gewicht interpretiert werden, also realisiert die Gewichtsveränderung auch die Schwellwertanpassung

Lernen durch Topologieveränderung

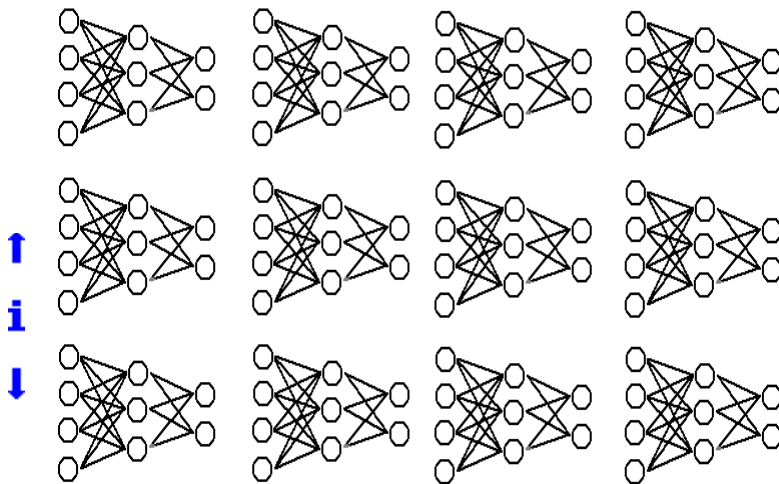
Esgibt aber auch Trainingsmethoden mit **Topologie-Änderung**, dabei werden Neuronen und/oder Verbindungen neu geschaffen bzw. gelöscht, um das Verhalten des Netzes an die Trainingsdaten anzupassen.



Dann Verbindungen vom Gewicht 0 auch als nicht vorhandene Verbindungen interpretieren kann, läßt sich die Topologieveränderung als ein Spezialfall der Gewichtsveränderung interpretieren.

Lernen mit genetischen Algorithmen

Manchmal verwendet man auch Training durch **genetische Algorithmen** dabei werden Populationen von Netzwerken konstruiert, wobei sich die "besten" stets weiter "fortpflanzen" dürfen und die schlechteren "aussterben".



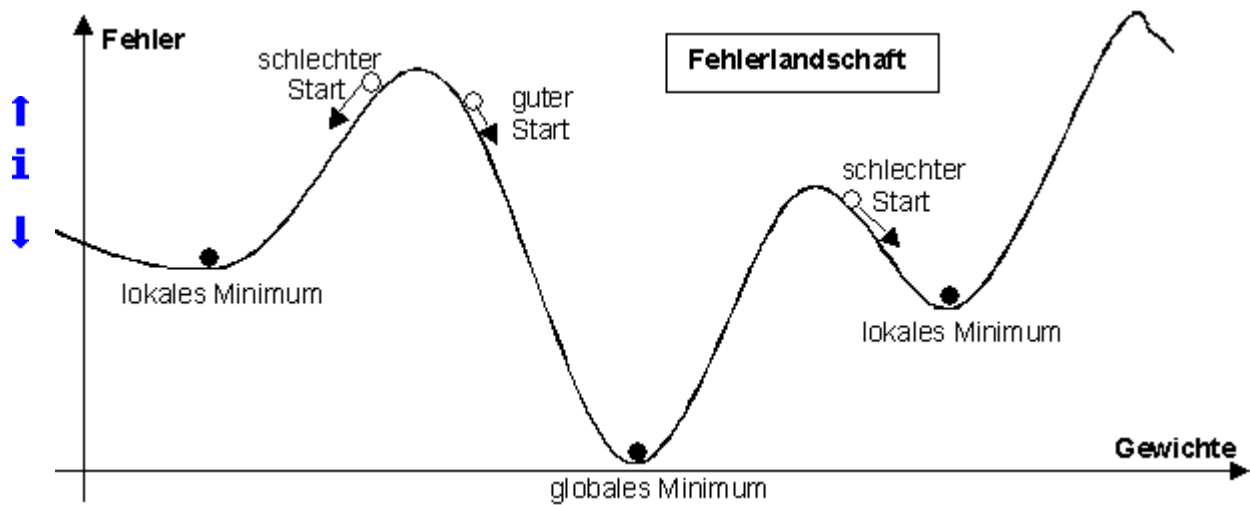
Korrektheit (Merkfähigkeit)

das Netzwerk muß auf den Daten, auf denen es trainiert ist, das gewünschte Verhalten zeigen:

1. gute Rekonstruktion trainierter Muster.
2. richtige Assoziation trainierter Musterpaare.
3. klare Trennung der trainierten Daten in unterschiedliche Cluster und korrekte Zuweisung der trainierten Daten an die vorgegebenen Klassen.

Das Problem lokaler Minima

betrachtet man die Fehlerfunktionskurve $J(\mathbf{w})$ (Kurve) über dem Raum der Gewichte, so sieht man, daß diese Funktion Unebenheiten und Vertiefungen hat. Die meisten Lern- (Optimierungs-) verfahren versuchen in dieser Fehlerfunktion abzustiegen, wodurch man sich leicht in einer der nicht optimalen Vertiefungen festfahren kann:



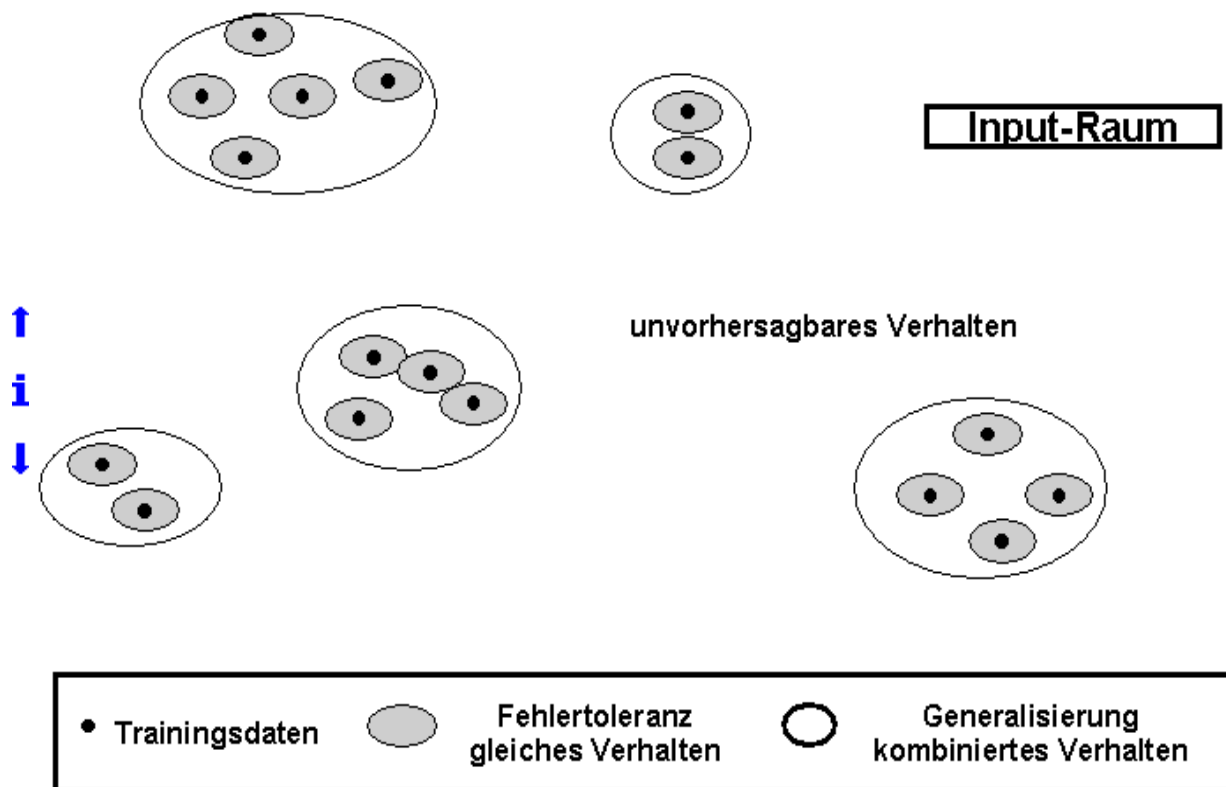
Generalisierung (Abstraktionsfähigkeit)

das Netzwerk muß auch auf die Daten, auf die es nicht trainiert ist, sinnvolle Antworten geben können:

1. nur für solche Daten vernünftig, die nah(?) an den Trainingsdaten liegen.
2. "sinnvoll" kann für Benutzer und Netz sehr verschiedene Bedeutung haben (das Netz modelliert die Wirklichkeit evtl. nicht korrekt).
3. Bei redundanten Netzausgaben kann eine Konsistenzprüfung klären, ob das Netz eine klare Entscheidung getroffen hat oder "unsicher" bzgl. seiner Ausgabe ist.

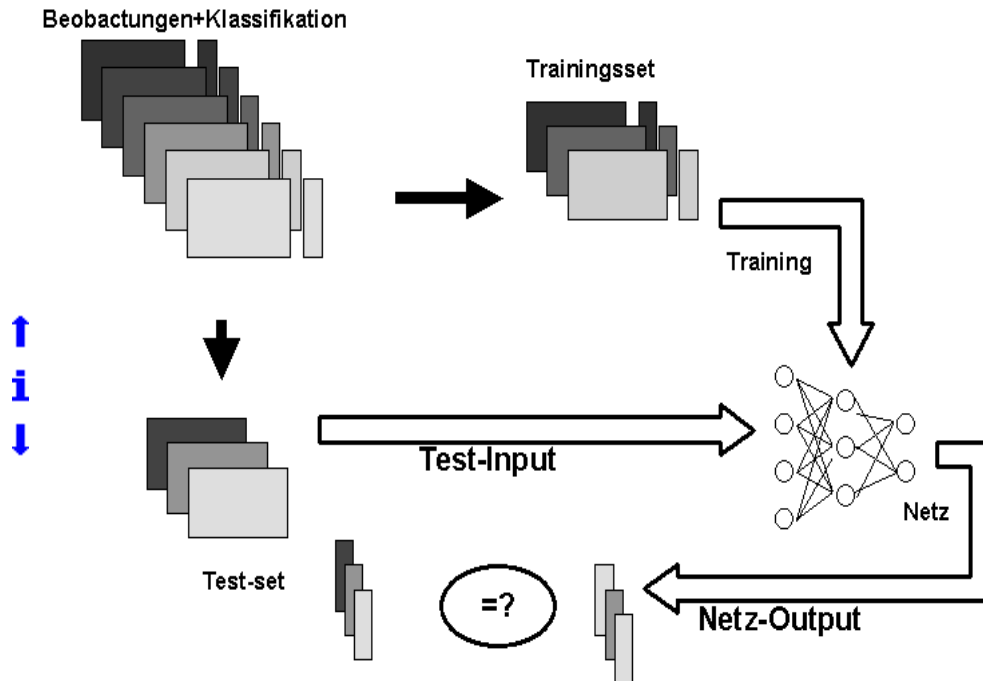
Fehlertoleranz/Generalisierung

Kleine Fehler bei der Eingabe trainierter Daten soll das neuronale Netzwerk tolerieren.



Generalisierungstest

Zur Überprüfung der **Generalisierungsfähigkeit** zerlegt man die vorhandenen Daten in zwei Gruppen, die Trainingsdaten, die für das Training verwendet werden, und die Testdaten, mit denen das System auf seine Generalisierungsfähigkeit überprüft wird.



Vorteile Neuronaler Netzwerke

NeuronaleNetzwerkehabengegenüberanderen ModellierungsmethodenfolgendeVorteile:

1. Sieverarbeitenverrauschte,unvollständigeundwidersprüchlicheInputs.
2. SiekönnenmultisensorischenInput(Zahlen,Farben,Töne,...)verarbeiten.
- ↑
i
↓ 3. NeuronaleNetzwerkeerzeugeneinimplizitesModellfürdieEingabedaten
(ohneHypothesendesAnwenders).
4. NeuronaleNetzwerkeindFehlertolerantauchgegenHardwarefehler.
DerTotalausfalleinesNeuronsverändertdasVerhaltendesNetzwerks
kaum.
5. Siesindleichtzuhandhabenundführenzuleichtinterpretierbaren
Ergebnissen.

NachteileNeuronalerNetzwerke

DenVorteilenstehenaberaucheinigegravierendeNachteile gegenüber:

1. NeuronaleNetzwerkebenötigenlangeTrainingszeiten.
- ↑
i 2. EinLernerfolgkannnichtgarantiertwerden.
- ↓ 3. Generalisierungsfähigkeitkannnichtgarantiertwerden(Overfitting).
4. BeiNeuronalenNetzenistnichtnachzuvollziehen,warumsieeine
bestimmteAntwortgeben(blackbox).

LangeTrainingszeiten

Neuronale Netzwerke benötigen lange Trainingszeiten.

- Eine große Anzahl von Trainingsdurchläufen ist notwendig.
- Dabei im Training der Auswertungsfehler minimiert wird, bleiben viele Verfahren in lokalen (Fehler-) Minima stecken, ohne den minimalen Fehler zu erreichen.
- Wegen der lokalen Minima müssen eventuell viele Serien gestartet werden.
- Bei hoher Anzahl von Verbindungen steigt die Anzahl der Trainingsdurchläufe unverhältnismäßig stark an.

keine Lerngarantie

Ein Lernerfolg kann nicht garantiert werden weil das Netzwerk entweder:

- einen zu ineffektiven Lernalgorithmus für das Problem hat.
- oder in einem lokalen Minimum eingefangen wird (schlechte Anfangsbedingung).
- oder es zu groß gewordenes Verbindungsgewicht die nachfolgenden Neuronen paralyisiert.
- oder die Netztopologie prinzipiell für das Problem ungeeignet ist (globales Minimum zu groß).
- oder die Aufgabe prinzipiell unlösbar ist (zu große Widersprüche).

keine Generalisierungsgarantie

Generalisierungsfähigkeit kann nicht garantiert werden (Overfitting):

- wenn die Testdaten zu große Unterschiede zu den Trainingsdaten haben (garbage in, garbage out).
- Bei zu langen Trainingsläufen sinkt die Generalisierungsfähigkeit.
- Zu viele verborgene Neuronen führen zu bloßem Auswendiglernen ohne Generalisierungsfähigkeit.

BlackBox Neuronaler Netzwerke

Bei Neuronalen Netzen ist nicht nachzuvollziehen, warum sie eine bestimmte Antwort geben (blackbox).

- Die Information über das Netzverhalten ist in den Synapsengewichten codiert und auf das gesamte Netzwerk verteilt.
- Das Netzwerk gibt auf jede Eingabe eine Antwort, gleichgültig ob es auf ähnliche Eingabe trainiert wurde oder nicht.
- Es gibt keine unmittelbare Möglichkeit, einem Neuronalen Netzwerk Gesetze, Regeln oder Rechenvorschriften einzugeben.

Konsequenzen für den Einsatz

- **Neuronale Netze sind ungeeignet für deterministische Aufgaben**

(Regeln, Berechnungen, absolute Fehlerfreiheit).

- **Es ist besser kleine Neuronale Netze zu verwenden**

(kurze Trainingszeiten/geringe Gefahr des Auswendiglernens/gute Generalisierungsfähigkeit/gut analysierbar).

- **Redundanz in den Outputs verwenden**

(gute Unterscheidung von klaren und unsicheren Entscheidungen des Netzwerks/gute Funktion der Reaktion des Netzwerks auf untrainierte Inputs/gute Erkennung der neuen Situationen: Novelty-Filter).

- **Vorverarbeitung gut planen**

(Entwicklung geeigneter Trainings- und Test-Dateien aus den vorliegenden Daten/Umwandlung großer Input-Vektoren in (evtl. mehrere) kleine Input-Vektoren für kleine Netze/klassische Filtermethoden bei vorliegenden Hypothesen über Verteilungen und Rauschverhalten).

- **Modularisierung des Netzwerks**

(Aufbau des Gesamtsystems aus Teilen mit verschiedenen Einzelaufgaben wie etwa: Vorverarbeitungen, Netze, Regelverarbeitungen, Berechnungen/klassischer top-down-Entwurf eines hybriden Systems, indem Netze und klassische Programmteile realisiert werden.).

Anwendungsgebiete Neuronaler Netze

Neuronale Netze finden z.B. in folgenden Gebieten Anwendung:

- **Mustererkennung**

(Handschrift-Erkennung/Spracherkennung/optische Musterklassifikation/EEG-Klassifikation).

- **Codierung**

(Bildkompression/Sprachsynthese/Rauschfilter/Mapping, z.B. in der Medizin).

- **Kontrolle**

(Qualitätskontrolle (optisch, akustisch)/Robotersteuerung/automatische Navigation/automatische (Schadens-)Diagnostik/EEG-Überwachung).

- **Optimierung**

(Parametervariation in Fertigungsprozessen/TSP-Probleme/Scheduling-Probleme).

Vergleich verschiedener Methoden

Neuronale Netze finden z.B. in folgenden Gebieten

Anwendungen:

- **Statistische Methoden**

(Numerische Verarbeitung/große Datenmengen/apriori Hypothesen/strukturierte Eingaben/nicht adaptionsfähig).

- **KI/Experten Systeme**

(Symbolische Verarbeitung/festes Regelsystem/apriori Hypothesen/strukturierte Eingaben/kaum adaptionsfähig).

- **Fuzzy Systeme**

(Numerische Verarbeitung/unscharfe Regeln/Anfangshypothesen/strukturierte Eingaben/gut adaptionsfähig).

- **Neuronale Systeme**

(Numerische Verarbeitung/Regelfrei/Hypothesenfrei/unstrukturierte Eingaben/gut adaptionsfähig).

Inhaltsverzeichnis

- | | |
|--|--|
| 1. Titelseite | 17. Lernen mit genetischen Algorithmen |
| 2. Einleitung | 18. Korrektes Lernen |
| 3. McCulloch-Pitts Neuronen | 19. Probleme lokaler Minima |
| 4. Neuronenmodell | 20. Verallgemeinerungsfähigkeit |
| 5. Aktivierungsfunktionen | 21. Fehlertoleranz/Generalisierung |
| 6. monotone Outputfunktionen | 22. Generalisierungstest |
| 7. zentrische Outputfunktionen | 23. Vorzüge neuronaler Netze |
| 8. Experimente mit Outputfunktionen | 24. Nachteile neuronaler Netze |
| 9. Netztopologie | 25. Lange Trainingszeit |
| 10. Musterklassifikation und Assoziation | 26. keine Lerngarantie |
| 11. Klassenbildung | 27. keine Generalisierungsgarantie |
| 12. Klassifikation | 28. BlackBox Verhalten |
| 13. Lernparadigmen | 29. Konsequenzen |
| 14. Typische Beispiele | 30. Anwendungsgebiete |
| 15. Lernen durch Gewichts Anpassung | 31. Vergleich zu anderen Methoden |
| 16. Lernen durch Topologieveränderung | 32. Inhaltsverzeichnis |