A. Neuronale Netze

Autor: Markus Möller

- Bezug: Seminar bei Herrn Rammig.
- Dieser Extrakt entstand als Vorbereitung auf meine Diplomprüfung (Teil: Vertiefungsgebiet). Er faßt einige Themen einfach zusammen und mag etwas unorthodox erscheinen.
- Erstellt auf Apple Macintosh.

1. Neuronale Netze und Gehirn

Der grundlegende Unterschied zwischen Computer und Gehirn ist, daß im Gehirn viele eigenständige Neuronen mit schwacher *Rechenleistung* untereinander *verbunden* sind und im Standardcomputer ein einziger vergleichsweise starker Prozessor weitgehend allein verantwortlich ist. Dennoch ist das menschliche Gehirn dem Computer in vieler Hinsicht überlegen: Z.B. bei der Mustererkennung. Dies kann nur durch die enorme Parallelarbeit der Neuronen erklärt werden, die ihre „Daten“ untereinander austauschen, da eine Einzeloperation im Prozessor tausendmal schneller abläuft als im Gehirn.

1.1 Assoziatives Gedächtnis und Gehirntheorie

Der Aufbau des menschlichen Gehirns deutet darauf hin, daß es sich in der Funktionsweise von heutigen Computern unterscheidet. Die Gehirnrinde enthält etwa zehn Milliarden Nervenzellen mit – soweit dies experimentell geprüft worden ist – ziemlich einfachen Eingangs-Ausgangs-Beziehungen: Jedes Neuron empfängt über eine in der Regel fünfstellige Zahl von Eingängen (stark verästelte Dendriten) Signale von anderen Nervenzellen, verarbeitet sie und gibt selbst über einen Ausgang (das Axon) nur ein einziges Signal ab. Dieses wird seinerseits über mehrere zehntausend Synapsen (Schaltstellen zwischen Neuronen) an etwa ebensoviele Nervenzellen weitergeleitet.

Die über die Dendriten empfangenen Signale erregen die Nervenzelle unterschiedlich stark. Es sind sogar hemmende Signale möglich (also negatives Vorzeichen).

Übersteigt die Summe der Impulse einen bestimmten Schwellenwert, so gibt die Nervenzelle über ihr sich verzweigendes Axon (die Nervenfaser) ein Signal weiter.

Es wurde festgestellt, daß die Häufigkeit, mit der eine Synapse Nervenimpulse an die nachfolgende Zelle weiterleitet, ihre Übertragungsstärke beeinflußt. Wenn man also bei neuronalen Netzen diese Fähigkeit einbaut, d.h. daß sich die Stärke der Verbindungen zwischen den Neuronen, also der Synapsen, während der Aktivitätsausbreitung im Netzwerk verändern kann, dann lassen sich mit relativ wenigen Neuronen ziemlich leistungsfähige Lernmodelle erstellen.

Besonders oft wurde eine 1949 von Donald Hebb vorgeschlagene Regel zur Veränderung der Verbindungen simuliert. Danach wird eine Verbindung (≈Synapse) zwischen zwei Neuronen immer dann verstärkt, wenn beide Neuronen etwa gleichzeitig aktiv sind. Synapsen, die sich nach einer solchen Regel verändern, nennt man heute Hebb-Synapsen. Da sich mit ihnen in gewisser Weise Informationen speichern lassen, können sie als Modell des Gedächtnisses dienen.

In Gehirn können allerdings zwei Neuronen gleichzeitig aktiv sein, ohne daß ihre Verbindung verstärkt wird; z.B. bei unverbundenen Neuronen.

1.2 Mustererkennung mit assoziativem Speicher und neuronalen Netzen

Unter einem Muster versteht man die Gesamtaktivität aller Neurone im Gehirn. Von solchen Aktivitätsmustern sollte es möglich sein, auf die Sinneseindrücke rückzuschließen, die diese verursacht haben.

Speicherung und Abruf von Mustern in einem assoziativen Speicher lassen sich folgendermaßen realisieren: Der Speicher besteht aus einem Gitter von horizontalen und vertikalen Leitungen und ihren Verknüpfungspunkten. Zur Speicherung eines Musters gibt man die ihm entsprechende Bitfolge sowohl auf die horizontalen als auch auf die vertikalen Leitungen: Dabei werden die Leitungen aktiviert, an denen die Bitfolge den Wert 1 hat. Die Speicherung selbst geschieht über die Verstärkung der Verknüpfungen zwischen gleichzeitig aktiven Leitungen.

Man speichert also nicht das Muster selbst, welches die Aktivierungen auslöst, sondern die Korrelationen zwischen seinen Bestandteilen.

Derselbe Speicher kann mehrere Muster aufnehmen. Zum Auslesen eines Musters genügt es, einen Teil davon an den horizontalen Leitungen anzulegen. Dieses Teilmuster erzeugt über die aktivierten Verknüpfungen auch ein Signal in den vertikalen Leitungen. Durch eine Schwellendetektion, die nur maximale Erregung durchläßt, wird das richtige Muster ergänzt. Enthält die Eingabe gleich viele Teile von beiden gespeicherten Mustern, ist auch die Ausgabe eine Überlagerung dieser Muster.

Neuronale Netze lassen sich als Assoziativspeicher auffassen. Dabei entsprechen die Axone (Ausgänge) den horizontalen und die Dendriten (Eingänge) den vertikalen Leitungen im Assoziativspeicher.

Sehr ähnliche Muster werden über die eingefahrenen (verstärkten) Verbindungen zu dem selben stabilen Aktivitätsmuster ergänzt. Solche stabilisierten neuronalen Aktivitätsmuster nannte Hebb „Assemblies“ (nach dem englischen Wort für Gruppen oder Vereinigungen). Sie entstehen nach seiner Vorstellung dadurch, daß beim Lernen eines Musters die Verbindungen zwischen gleichzeitig aktivierten Neuronen gefestigt werden. Korrelationen in der Außenwelt werden so zu Verbindungen zwischen Neuronen. Solche Assemblies haben nun von sich aus die Fähigkeit, selbsttätig und automatisch die zugehörigen Muster zu vervollständigen. Wird nämlich ein genügend großer Teil einer Assembly aktiviert, so erfaßt die Erregung über das eingefahrene Netz von Verbindungen auch die anderen beteiligten Neurone: Die Assembly „zündet“.

Man realisiert dies mit einem rückgekoppelten neuronalen Netzwerk: Das über die vertikalen Axone eingegebene Teilmuster passiert zunächst das Netzwerk, läuft dann auf den horizontalen Axonen zurück und aktiviert über die beim Einspeichern verstärkten Synapsen schließlich das vollständige Muster.

Der Vorgang kann auch als Matrixoperation aufgefaßt werden. Dabei erhält das Korrelationsmatrixelement $c_{i,j}$ den Wert 1, wenn Neuron_1 mit Neuron_2 gleichzeitig aktiv ist, und den Wert 0 sonst. Durch einfache Addition der Korrelationsmatrizen für einzelne Assemblies erhält man eine Summenmatrix, in der die Korrelationen sämtlicher vorhandenen Aktivitätsmuster gespeichert sind. Umgekehrt braucht man, um ein Aktivitätsmuster zu löschen, nur dessen Korrelationsmatrix von der Summenmatrix zu subtrahieren. Auslesen geschieht durch Matrixmultiplikation. Der resultierende Faktor muß dann noch mit einem geeigneten Schwellenwert verglichen werden, der nur maximale Erregung durchläßt.

Hier würde also in einer Synapse mehr als ein Bit gespeichert werden. Der Unterschied ist, daß bei einem Bit jedes Element des Ausgabevektors anzeigt, wieviele Inputbits (Neurone) mit diesem Neuron gleichzeitig aktiviert sind, während bei dem mathematischen Modell angezeigt wird, wieviele Paare aus (Muster, Inputbit) gleichzeitig mit diesem Neuron aktiviert sind. Es wird also nicht nur darauf geachtet, wieviele der verschiedenen Inputbits diesen Output aktivieren, sondern es werden Inputbits mit der Anzahl der verschiedenen Muster gewertet, in denen sie vorkamen, beim Lernen.

1.3 Der Neuere Konnektionismus

Da viele Arbeiten zu dem Thema „Neuronale Netzwerke“ und „Konnektionismus“ schon vor etlichen Jahren oder sogar Jahrzehnten entstanden sind, wird die heutige aufstrebende Schule des Konnektionismus auch als „Neuerer Konnektionismus“ bezeichnet.

Der Inhalt des Konnektionismus ist die Erforschung und Konstruktion informationsverarbeitender Systeme, die sich aus vielen primitiven, uniformen Einheiten zusammensetzen und deren wesentliches Verarbeitungsprinzip die Kommunikation zwischen diesen Einheiten ist. Ein weiteres Charakteristikum dieser Systeme ist die parallele Verarbeitung von Information innerhalb des Systems durch eine gleichzeitige Aktivität vieler Einheiten.

Generelle Zielsetzung des Konnektionismus ist die Modellierung kognitiver Prozesse mittels derartiger Konnektionistischer Modelle. Der Konnektionismus wurde initiiert durch das Interesse daran, wie das menschliche Gehirn komplexe kognitive Leistungen vollbringen kann.

Innerhalb des Konnektionismus zeichnen sich zwei Richtungen ab, in denen konnektionistische Systeme untersucht werden. In der einen Richtung wird von den abstrakten, neuronal-orientierten Modellen ausgegangen; mit formalen Methoden werden Klassen von Modellen bzgl. ihrer Eigenschaften und Funktionsprinzipien untersucht. Die andere Richtung ist eher anwendungsorientiert; sie beschäftigt sich mit der Realisierung bestimmter Modelle kognitiver Fähigkeiten, z.B. des Spracherwerbs oder des Sprachverstehens, mittels konnektionistischer Modelle.

Man kann sagen, daß heute der Begriff „Neuronale Netzwerke“ dann verwendet wird, wenn grundlegende Eigenschaften oder Verarbeitungsprinzipien von Klassen

Neuronaler Netzwerke untersucht werden, wohingegen von „Konnektionistischen Modellen“ gesprochen wird, wenn die Netzwerke in einer bestimmten Interpretation, z.B. zur Modellierung eines kognitiven Prozesses, eingesetzt werden. Der Begriff „Konnektionismus“ wird fast ausschließlich von „Anwendern“ Neuronaler Netzwerke, z.B. im Bereich der Sprachverarbeitung, benutzt.

Die Ideen und Grundlagen konnektionistischer Modelle weisen schon eine lange Tradition auf. Daher wird die Forschung in diesem Bereich, die erst seit kurzem in einem größeren Rahmen erfolgt, oft auch als „Neuerer Konnektionismus“ bezeichnet.

Eine klassische Forschungsrichtung in diesem Gebiet, die mittlerweile gut fundiert ist, sind die sog. „Assoziativspeicher“. Das sind neuronale Netzwerke, die unter dem Gesichtspunkt der Speicherung komplexer Muster untersucht werden.

Warum Konnektionismus?

Einer der wesentlichsten Aspekte, die zur Einführung konnektionistischer Modelle führten, ist die massiv parallele Verarbeitung von Daten, die zu einer signifikanten Erhöhung der Verarbeitungsgeschwindigkeit führen soll. So wird oft angeführt, daß Neuronen eine wesentlich niedrigere Schaltgeschwindigkeit besitzen als moderne elektronische Bauteile, nämlich Zeiten im Millisekunden-Bereich gegenüber Zeiten im Nanosekunden-Bereich bei elektronischen Schaltelementen. Die Neuronen sind also um den Faktor 10^6 langsamer.

Ein weiterer bedeutender Punkt für die Entwicklung konnektionistischer Modelle ist, daß sie ein neues, die Forschung anregendes Paradigma (Beispiel, Vorbild) darstellen. So ist z.B. die Art der Wissensrepräsentation in konnektionistischen Modellen vollkommen anders als in herkömmlichen KI-Systemen. Das „Wissen“ in konnektionistischen Modellen ist im ganzen System verteilt, es ist gespeichert in der Struktur und der Gewichtung des Netzwerks. Konnektionistische Modelle bieten gerade unter den Aspekten „Wissensrepräsentation und -verarbeitung“, „Lernen und Selbst-Organisation“ und „Fehlertoleranz“ neue, erfolgversprechende Möglichkeiten.

Konnektionistische Systeme sind schwer analysierbar. Da das Wissen in einer dem Menschen schwer verständlichen Form im gesamten Netzwerk verteilt ist, kann das Verhalten des Systems mit heutigen Methoden nicht durch eine Analyse seiner

Struktur bestimmt werden. Die einzige Möglichkeit, Aussagen über seine Leistungsfähigkeit zu gewinnen, sind Tests.

Grundlagen konnektionistischer Modelle

- Die *Verarbeitungselemente* haben je nach konkreter Anwendung verschiedene Bedeutungen. Sie sind einfach und die einzigen Kontroll- und Steuermechanismen im Netzwerk. Elemente können z.B. Objekte, Merkmale oder Konzepte repräsentieren oder abstrakte Entitäten ohne Interpretation sein. Es gibt Repräsentationen, in denen konzeptuelle Objekte direkt einzelnen Elementen entsprechen, und solche, in denen sie durch ganze Teilstrukturen des Netzwerks bzw. durch bestimmte Aktivitätsmuster dargestellt werden. „Local representation“ und „distributed representation“. In einigen, insbesondere den thermodynamischen Modellen, werden die Elemente auch als Hypothesen interpretiert, und die Verbindungen drücken aus, ob Hypothesen sich gegenseitig stützen oder widersprechen bzw. stellen „Constraints“ (Beschränkungen) in der Domäne dar.
  Innerhalb eines Netzwerks kann zwischen Eingabe-, Ausgabe- und internen Elementen („hidden units“) unterschieden werden. Eingabe- bzw. Ausgabeeinheiten haben eine Schnittstelle zur Umwelt, sie erhalten externe Eingaben bzw. produzieren externe Ausgaben.
- Die *Netzwerkstruktur*, über die die Elemente verbunden sind und in der sie durch das Schicken von Signalen miteinander kommunizieren, bestimmt wesentlich das Verhalten des Systems.
  Negative Gewichte werden als inhibitorische, hemmende Verbindungen interpretiert und positive Gewichte als excitatorische, aktivitätssteigernde Verbindungen.
  Hierachische Netzwerkstrukturen, also solche, deren Elemente in „Schichten“ angeordnet sind, können eingeteilt werden in bottom-up, top-down und interaktiv arbeitende Systeme. In den meisten interaktiven Modellen findet ein Signalfluß jedoch nur zwischen benachbarten Ebenen statt. Die häufigste Systemarchitektur sind Bottom-Up-Netzwerke, die wegen des unidirektionalen, vom Systemeingang zum Systemausgang gerichteten Signalflusses aus als „feed-forward-Netzwerke“ bezeichnet werden.
  Wesentlich für die Arbeitsweise vieler Netzwerkmodelle ist die Änderung der Verbindungsgewichtung als Ergebnis eines Lernprozesses. Da Lernverfahren getrennt von der prinzipiellen Arbeitsweise des Systems untersucht werden, wird diese dynamische Komponente i.a. nicht direkt in die Modelldefinition aufgenommen.
- Die Menge der *Aktivierungszustände* ist i.a. für alle Elemente eines Netzwerks gleich.
- Die *Eingabe*- und die *Ausgabemenge* geben die Menge der möglichen Eingaben bzw. Ausgaben auf einer Verbindung an.

- Die *Ausgabefunktion* ist abhängig vom Aktivierungsgrad des Elements zum gegebenen Zeitpunkt. Es wird zunächst eine für alle Ausgangsverbindungen gleiche Ausgabe erzeugt, die erst durch die Propagierungsfunktion mit den Gewichten der einzelnen Verbindungen verknüpft wird.
- Die *Propagierungsfunktion* bestimmt anhand der Konnektionsmatrix, in der die Gewichte aller Verbindungen stehen, und des Ausgabevektors die interne Eingabe, die Netzeingabe, in die einzelnen Element. Im allgemeinen besteht die Propagierungsfunktion aus einer einfachen Summierung der gewichteten inhibitorischen und excitatorischen Einflüsse.
- Die *(externe) Eingabefunktion* ist entweder eine probabilistische Umgebungsfunktion oder sie entspricht spezifischen ausgewählten Mustern, die das System lernen oder auf die es in bestimmter Weise reagieren soll. Manchmal wird auch nur eine Initialisierung der Eingabeelemente oder aller Elemente vorgenommen, indem sie in einen bestimmten Aktivierungszustand gesetzt werden, der z.B. ein zu verarbeitendes Muster oder ein zu lösendes Problem repräsentiert. Also keine explizite Eingabefunktion dann.
- Die *Aktivierungsfunktion* bestimmt für jedes Element den Aktivierungszustand in Abhängigkeit von der aktuellen Aktivierung und den externen und internen Eingaben des Elements. Der Aktualisierungszeitpunkt kann für alle Elemente synchron oder asynchron sein.

Lernen in konnektionistischen Systemen

Die Arbeitsweise eines konnektionistischen Systems wird durch Modifizierung der Verbindungsstruktur geändert, d.h. durch die Veränderung der Gewichte der Verbindungen. Die Modifikation der Verbindungen basiert auf der Hebb'schen Hypothese, daß die Verbindung zwischen zwei Elementen verstärkt wird, wenn beide gleichzeitig aktiviert sind: $\Delta w_{i,j} = \eta \; a_i(t) \; a_j(t)$, wobei die Konstante η der Lernfaktor ist und i und j die Indizes zweier Elemente.

Als Lerneingabe wird i.a. der erwünschte Aktivierungszustand des Elements bzw. die gewünschte Ausgabe eingesetzt. Geht in die Lernregel eine solche Lerneingabe ein, ist es beaufsichtigtes Lernen im Gegensatz zum unbeaufsichtigten Lernen. Der Lernprozeß erfolgt oft in einer vom normalen Arbeitsmodus getrennten Trainings- oder Lernphase, in der die Gewichte der Verbindungen anhand der Lernfunktion justiert werden, bis das Ein-/Ausgabeverhalten des Systems wunschgemäß ist oder das System sein Verhalten nicht mehr wesentlich ändert.

Beim assoziativen Lernen wird die Lerneingabe durch das zu erzeugende Ausgabemuster bestimmt; beaufsichtigtes Lernen. Beim entdeckenden lernen soll das

Netzwerk selbstständig Regelmäßigkeiten in der Umgebung, also in einer Folge von Eingabemustern entdecken.

Die Fehlerpropagierung „error backpropagation“ ist assoziatives Lernen. Hierbei wird der Ausgabefehler rückwärts durch das Netz verfolgt und die Gewichte sukzessive, beginnend bei den Ausgabeelementen, adjustiert.

Beim Wettbewerbslernen „competitive learning“ handelt es sich um entdeckendes Lernen: Es werden immer nur die Gewichte für das Element geändert, das von der externen Eingabe am stärksten betroffen war, d.h. die größte Eingabe erhalten hat. Es wird von den Gewichten aller Eingangsverbindungen des Elements ein bestimmter Anteil abgezogen, der auf die Verbindungen, über die das Element positiv beeinflußt wurde, verteilt wird.

TABELLE 1. Klassen konnektionistischer Modelle

	Ausgabe-funktion	**Aktivie-rungszu-stände**	**Verbindun-gen**	**Aktivie-rungsfunk-tion**	**Sonstiges**
Das einfache lineare Modell	Identitäts-funktion: Ausgabe = Aktivierung	reelle Zahlen	Einheitstyp mit positi-ven, negati-ven und Null-Werten	Summe der gewichteten Eingaben	einstufig, Mustererken-nung
Lineare Schwellen-wertelemente	Identitäts-funktion	1 und 0	ganzzahlige Werte	Schwellen-wertfunktion	berechnet bel. boole-sche Funk-tionen
Anderson-Modell	Identitäts-funktion	Intervall [-1,1]	Feedback und totale Verbindung erlaubt.	Addition der gew. Einga-ben zum aktuellen Az.	Erkennen von gespr. Vokabeln, Schrift.
Grossberg-Modell		Intervall [min, max]		Langsames Abnehmen, Ruhezustand 0	automati-sches Anstre-ben eines Ruhezustan-des
Sigma-Pi-Units			Gewichtete Summe von Produkten.		Multiplika-tive Verknüp-fung von Ausgabewer-ten.
P-Units	Größer als Schwelle: Akt. A.-Konstante	[-10,10]	Konjunktive Verknüp-fung von Tei-len der Ein-gabe.		Wissensre-präsentation

TABELLE 1. Klassen konnektionistischer Modelle

	Ausgabe-funktion	**Aktivie-rungszu-stände**	**Verbindun-gen**	**Aktivie-rungsfunk-tion**	**Sonstiges**
Thermody-namische Modelle		+1 und 0	Symme-trisch, total.	Stocha-stisch, sig-moid. Abhängig von: Schwelle, akt. gew. Eingabe, Temperatur	
Hopfield-Modell		+1 und -1	Symme-trisch, Dia-gonale 0	Vorzeichen von …	Nicht immer alle Ele-mente an Berechnung beteiligt. Assoziativ-speicher, Travelling…

Problemstellungen in konnektionistischen Modellen

- Mustererkennung (pattern recognition)
  Das System soll auf ein eingegebenes Muster mit einer bestimmten „Antwort“ reagieren; Ausgabe oder stabiler Zustand.
- Merkmaldetektion (feature detection)
  Charakteristische Bestandteile oder Eigenschaften eines eingegebenen Musters erkennen.
- Mustervervollständigung (pattern completion)
  Das korrekte, vollständige Musterpendant produzieren.
- Assoziativer Zugriff (associative memory)
  Zu eingegebenem Muster zugeordnetes Ausgabemuster erzeugen. „Assoziativspeicher“. Inhaltsorientierter Zugriff.
- Selbstorganisation/Lernen
  Nicht explizite Konstruktion, sondern selbstständiger Ausbau von Verbindungen bei Trainingsphase erzielt gewünschtes Ein-/Ausgabeverhalten.
- Back-Propagation
  Aktivierungsvermittlung in Richtung der Eingabeseite bei Verarbeitungsprozeß. Lernen durch Fehlerkorrektur.
- Probabilistische Systeme
  Überwindung lokaler Minima aufgrund stochastischer Schwankungen.
- Fehlertoleranz
  Toleranz bzgl. Ausfall von Systemelementen oder unvollständigen Eingaben.

Traditionelle Anwendungsgebiete konnektionistischer Modelle sind die Mustererkennung, Modellierung von grundlegenden psychischen Prozessen, z.B. Lernen und Gedächtnis. Sprachverarbeitung.

2. Klassische und unkonventionelle Modelle

2.1 Einleitung

2.1.1 Das künstliche Neuron

Jedes Neuron besitzt eine Menge von Eingangsleitungen, die die Ausgänge von anderen Neuronen darstellen sollen. Eingaben, die über diese Leitungen zum Neuron gelangen, werden mit den jeweiligen Gewichten multipliziert und anschließend aufsummiert. Dieses Skalarprodukt stellt dann die Eingabe der Aktivierungsfunktion dar. Diese Funktion liefert das Ausgabesignal des Neurons, welches wiederum als Eingabesignal für andere Neuronen dienen kann.

Die Aktivierungsfunktion kann je nach Bedarf eine lineare Funktion, eine Schwellwertfunktion oder auch eine nicht-lineare Funktion sein.

2.1.2 Einteilung der Netzwerke

Netzwerke werden anhand ihrer Struktur klassifiziert, da Leistungsfähigkeit erst durch die Verbindung der Neuronen erreicht wird.

Einschichtiges Netzwerk

Zwei Spalten von Neuronen, bei der die erste nur zur Verteilung der Eingangssignale herhält. Gezählt wird nur die Anzahl der berechnenden Schichten. Daher „einschichtiges Netz“.

Mehrschichtiges Netzwerk

Eine Erweiterung des obigen Modells um weitere „zweite“ Schichten, die jeweils mit der vorherigen Schicht/Spalte verbunden sind.

Vollständige Vernetzung heißt, daß jedes Neuron einer Schicht mit jedem Neuron der Folgeschicht verbunden ist. Die Darstellung erfolgt mit Gewichtsmatrizen.

Die mehrschichtigen Netze stellen nur dann eine Erweiterung gegenüber den einschichtigen dar, wenn zwischen den Schichten nicht-lineare Aktivierungsfunktionen arbeiten. Ein mehrschichtiges Netz mit linearen Aktivierungsfunktionen kann in ein äquivalentes, einschichtiges Netz umgewandelt werden.
(Assoziativität der Matrixmultiplikation und der Multiplikation über die Addition).

Feedforward-Netze

haben keine Verbindungen von den Ausgängen einer Schicht zu den Eingängen derselben oder einer davorliegenden Schicht.

Recurrent-Netze

besitzen rückläufige Verbindungen. Ihr Vorteil ist, daß die Ausgabe nicht nur vom augenblicklichen Input abhängt, sondern auch von früher berechneten Ausgaben. Sie können dadurch Eigenschaften des Kurzzeitgedächtnisses simulieren.

2.1.3 Trainings-Algorithmen

Ziel des Trainings ist, daß auf eine Menge von Eingaben eine festgelegte mindestens aber konsistente Menge von Ausgaben erzeugt werden. Das Training ist die Korrektur der Gewichte. Das Netzwerk „lernt“.

Dazu werden verschiedene Input-Vektoren ins Netz eingespeist und die Gewichte ständig nach einem festgelegten Verfahren verändert. Die Gewichte konvergieren während des Trainings gegen feste Werte, so daß am Ende jeder Input-Vektor den gewünschten Output-Vektor erzeugt.

Lernen mit Lehrer

Man legt den Inputvektor an und vergleicht den Outputvektor mit dem Zielvektor. Ein gemessener Fehler wird zurück durch das Netz geschickt und die Gewichte durch einen Algorithmus so verändert, daß der Fehler minimiert wird.

Wiederholung mit verschiedenen Trainingspaaren bis das Netz zufriedenstellend arbeitet.

Lernen ohne Lehrer

Hier wird kein Zielvektor verwendet. Der Trainingsalgorithmus verändert die Gewichte so, daß zwei Anwendungen eines Trainingsvektors oder mehrere

Anwendungen ähnlicher Vektoren den gleichen Outputvektor erzeugen. Hier wird also die Menge der Eingabevektoren nach Ähnlichkeit eingeteilt.

2.2 Das Perceptron

2.2.1 Das einschichtige Perceptron

Beim Neuron des Perceptrons ist die Aktivierungsfunktion eine Schwellwertfunktion. $Out = \begin{cases} 1, \text{falls Net} > T \\ 0, \text{sonst} \end{cases}$ Wenn das Perceptron eine Funktion simulieren kann, dann heißt es, es kann sie darstellen. Um eine Funktion simulieren (darstellen) zu können, muß ein Algorithmus existieren, der die Gewichte entsprechend einstellt.

2.2.2 Das Exklusiv-Oder-Problem

Dieses einfache Problem zeigt die Beschränktheit einschichtiger Perceptrons: Exklusiv-Oder ist hiermit nicht simulierbar!

Im Koordinatensystem stellt die Gleichung $NET = W_1X_1 + W_2X_2$ eine Gerade dar, die die Fläche in zwei Halbebenen teilt: In der einen sind die NET-Werte größer, in der anderen kleiner als auf der Geraden. Da das einschichtige Perceptron seinen Output danach bestimmt, ob NET über oder unter dem Schwellwert liegt, teilt es die Eingabe(koordinaten) (X_1,X_2) in zwei Gruppen mit verschiedenen Out-Werten. Die Eingaben der zu berechnenden Funktion müssen sich also durch eine Gerade, denn $NET = W_1X_1 + W_2X_2$ ist für beliebige Gewichte eine Gerade, in zwei Gruppen teilen lassen, denen verschiedene Ergebnisse zugeordnet werden. „Lineare Separierbarkeit“ der Punktmenge. Für das Exklusiv-Oder-Problem ist keine Lösung durch ein einschichtiges Perceptron möglich, da die Eingabekoordinaten mit gleichen Funktionswerten so liegen: Man kann die Nullen und

1 0

0 1

Einsen offensichtlich nicht durch eine Gerade trennen. Für zwei Inputs erfolgt die Trennung durch eine Gerade. Für drei durch eine Ebene und für höhere Dimensio-

nen durch eine Hyperebene. Es ist kein einfacher Algorithmus bekannt, der entscheidet, ob eine Funktion linear separierbar ist.

2.2.3 Das mehrschichtige Perceptron

Die Beschränkungen, denen das einschichtige Perceptron unterliegt, können durch Hinzufügen weiterer Schichten aufgehoben werden. Es lassen sich so beliebige Polygone abfragen, indem die nachfolgenden Schichten vorausgehende Halbebenen durch „und“ oder „und nicht“ verknüpfen. Es lassen sich konvexe und konkave Bereiche erzeugen.

2.2.4 Die Delta-Regel

Die Delta-Regel ist eine wichtige, allgemeine Form des Trainings-Algorithmus für das *einschichtige* Perceptron. Lernen mit Lehrer, also Input- und Zielvektoren.

Perceptron mit nur einem Neuron:

- δ ist die Differenz aus tatsächlichem (einelementigen) Output A und dem Ziel T.
- $\Delta_i = \eta \quad x_i \quad \delta$ ist das Produkt aus Lernkoeffizient η, dem Input x_i und der Differenz δ.
- $w_i(n + 1) = w_i(n) + \Delta_i$ D.h. Das Gewicht i wird um diesen Wert verändert.

Jedes Gewicht i wird also abhängig von der Größe des (Teil-)Input x_i, den des „gewichtet“ hat, und abhängig vom entstandenen Ausgabefehler und einem Lernfaktor verändert. Die Abhängigkeit vom zugehörigen Teilinput ist sinnvoll, weil ein Gewicht mit Input 0 keinen Fehler machen kann und mit einem großen Input der Fehlerbeitrag auch wächst.

Perceptron mit einer Schicht Neuronen:

Es wird nur die Indizierung geändert. Es gibt j Inputs und i Neuronen, also auch i Outputelemente. Man betrachtet jeden Outputfehler i. Dann wird das Gewicht jeder Leitung von Input j zu Neuron i abhängig vom Produkt aus Lernfaktor, Ausgabefehler i und der Inputgröße j geändert:

Probleme beim Training des Perceptrons:

- Trainingspaare in zufälliger oder bestimmter Reihenfolge abarbeiten?

$$\delta_i = T_i - Y_i$$
$$\Delta_{ji} = \eta \; x_j \; \delta_i$$
$$w_{ji}(n+1) = w_{ji}(n) + \Delta_{ji}$$

- Größe des Lernkoeffizienten? Konstant oder variabel während des Trainings?
- Ist das Problem linear separierbar?

Die Antworten sind noch nicht erforscht.

2.3 Backpropagation

Der Backpropagation-Algorithmus ist eine mathematisch fundierte, systematische Lernmethode für *mehrschichtige* Netzwerke, die nicht den starken Beschränkungen unterliegen wie die einschichtigen.

2.3.1 Das Neuron

Backpropagation verlangt eine in jedem Punkt differenzierbare Aktivierungsfunktion. Ferner sollte sie nicht-linear seien, da sonst die Mehrschichtigkeit sinnlos ist. Im allgemeinen nimmt man die folgende leicht ab ableitbare Funktion:

$$F(NET) = \frac{1}{1 + e^{-NET}}$$
$$\frac{d}{dNET}Out = Out \; (1 - Out)$$

Dabei ist NET die gewichtete Summe der Inputs in das Neuron und Out die Ausgabe des Neurons aufgrund der Aktivierungsfunktion. D.h. F(NET)=Out.

2.3.2 Der Algorithmus

Erklärung anhand eines zweischichtigen Feedforwardnetzes. Da nur für die letzte Schicht ein Fehler direkt berechnet werden kann, korrigiert man zuerst die Gewichte dieser Schicht und pflanzt den Fehler dann Schicht für Schicht durch das Netz zurück.

Die Korrektur der Gewichte in der letzten Schicht (k) erfolgt wie beim einschichtigen Perceptron mit dem Zusatz, daß bei der Berechnung des Faktors δ aus der Differenz von Ausgabe und Ziel noch die Ableitung der Aktivierungsfunktion als Faktor hinzukommt: $\delta_{q,k} = Out_{q,k}(1 - Out_{q,k})(T_q - Out_{q,k})$ Hier ist q ein Neuron in Schicht k.

Nachdem die Gewichte der letzten Schicht korrigiert wurden, wird nun in Schicht k-1 zurückgegangen. Man führt dazu die Werte von $\delta_{i,k}$ der Neuronen i in Schicht k durch die eben berechneten neuen Gewichte zurück, so daß $\delta_{i,k-1}$ für alle Neuronen i der Schicht k-1 berechnet werden kann. Dieses $\delta_{i,k-1}$ werden dann dazu benutzt, die Gewichte der Leitungen zu korrigieren, die in Schicht k-1 führen:

$$\delta_{i,k-1} = Out_{i,k-1}(1 - Out_{i,k-1})\left(\sum_j \delta_{j,k} \quad w_{ij,k}\right)$$

Im Unterschied zur letzten Schicht nimmt man also nicht die Differenz aus tatsächlicher Ausgabe und gewünschter, sondern summiert alle Ausgabefehler der Nachfolgeneuronen über die neuen Gewichte.

Bei der Initialisierung der Gewichte ist zu beachten: Falls alle Gewichte gleicher Schicht identisch sind, und das Ausgabeziel ein einheitlicher Vektor ist, kann das Netz nicht trainiert werden, da sämtliche Korrekturen gleichmäßig passieren, und so die Ausgaben der Schichtneuronen identisch bleiben.

Backpropagation folgt bei der Fehlerminimierung dem Gefälle der Fehlerfunktion. Man kann dabei in ein lokales Fehlerminimum geraten.

2.4 Counterpropagation

Vorteile der Counterpropagation-Netze ist ihre geringere Beschränktheit gegenüber einschichtigen Netzen und ihre kürzeren Lernzeiten als bei Backpropagation-Netzen, welche allerdings weniger beschränkt sind. Das Counterpropagationnetz ist ein zweischichtiges Feedforwardnetz, das zwei Methoden kombiniert.

2.4.1 Netzwerkstruktur

Es ist ein zweischichtiges, vollständiges Feedforwardnetz:

1. Kohonen-Schicht: Die einzelnen Neurone summieren wie gehabt ihre gewichteten Eingänge. Das Neuron mit der größten Summe gibt eine 1, alle anderen eine 0 aus.
2. Grossberg-Schicht: Auch diese bildet die übliche Summe in jedem Neuron. Effekt: Alle Neuronen geben das Gewicht der Leitung aus, dessen Kohonen-Neuron die 1 sendet.

2.4.2 Kohonen-Schicht-Training

Die Kohonen-Schicht wird ohne Lehrer so trainiert, daß ähnliche Inputs dasselbe Neuron aktivieren. Es wird genau das Neuron aktiviert, dessen Gewichtsvektor dem Inputvektor am ähnlichsten ist, und der Gewichtsvektor wird sodann dem Input noch mehr angepaßt: $w_{new} = w_{old} + \alpha(x - w_{old})$. Das Skalarprodukt wird als Ähnlichkeitsmaß verwendet. Es entsteht bei der gewichteten Summierung im Neuron. Dieses Maß berücksichtigt also die einzelnen Komponenten der Vektoren nur indirekt. Es wird das Neuron mit dem größten Skalarprodukt auf 1 gesetzt und seine Gewichte angepaßt; alle anderen Neuronen erhalten 0 als Ausgangssignal.

Der Lernkoeffizient wird am Anfang groß gewählt (ca. 0,7) und während des Trainings verkleinert. Durch dieses grobe Vortraining und das exakte Feinjustieren wird die Lernzeit meistens reduziert.

Es empfiehlt sich, die Gewichtsvektoren zu normalisieren, damit sich nur die Richtung der Vektoren und nicht ihre Länge auf die Berechnung des Skalarproduktes auswirkt und somit Fehlanpassungen vermieden werden. D.h. man wählt evtl. den falschen Gewichtsvektor zum Justieren aus. Werden auch die Inputvektoren normalisiert, so liegen die Endpunkte der Gewichtsvektoren von Anfang an näher an ihrem endgültigen Platz, da die Vektoren gleiche Länge haben, was die Lernzeit auch reduziert.

2.4.3 Grossberg-Schicht-Training

Training mit Lehrer. Nach dem Input wird der Output der Kohonenschicht bestimmt. Es werden die Leitungen korrigiert, die mit einem aktivierten Neuron verbunden sind. Die Änderung ist proportional zum Unterschied von Leitungsgewicht und dem erwarteten Output des Grossberg-Neurons:

$$v_{ij}(n+1) = v_{ij}(n) + \beta(Y_j - v_{ij}(n))k_i$$

k_i = Ausgangssignal von Neuron K_i

β = Lernkoeffizient

Bei der Kohonen-Schicht kann man keinen Einfluß darauf nehmen, welches Neuron bei einer bestimmten Eingabe aktiviert wird. Die Grossberg-Schicht sorgt dafür, daß bei Aktivierung eines bestimmten Kohonen-Neurons eine bestimmte Ausgabe der Grossberg-Neuronen erfolgt.

Im Endeffekt kann man also bewerkstelligen, daß alle zu dem Input x ähnlichen Inputs den Output y erzeugen.

2.4.4 Anwendung: Datenkomprimierung bei der Bildübertragung

Wenn kleine Verzerrungen bei der Übertragung toleriert werden können, so werden weniger Bits bei der Übertragung typischer Bilder benötigt:

Man zerlegt das Bild in Teilbilder und überläßt es der Kohonen-Schicht, ähnliche Mengen von Teilbildvektoren einem Neuron zuzuordnen, dieses also zu aktivieren. Die Grossberg-Schicht wandelt die Neuronennummer in Binärdarstellung um, die dann versendet wird. Der Empfänger ist ein identisch trainiertes Netz, das aus der Binärdarstellung das Bild approximiert.

2.5 Hopfield Netze

Die bisher betrachteten Netze waren Feedforwardnetze. Ein Hopfieldnetz ist recurrent: Ausgaben werden zurückgeleitet und eine neue Ausgabe berechnet, bis die

Ausgabe konstant bleibt oder konvergiert. Ein Problem bei recurrenten Netzen ist also, ob sie stabil sind.

2.5.1 Netzstruktur

Zweischichtiges Hopfieldnetz: Schicht 0 verteilt lediglich die Eingänge bzw. zurückgeführte Ausgaben auf die Neuronen, welche die gewichtete Summe ihrer Eingänge berechnen und ihren Output mit einer nichtlinearen Aktivierungsfunktion steuern.

Binäre Systeme

Der Zustand des Netzwerks ist die Menge aller Ausgänge der Neuronen. Da die n Ausgaben binär sind, läßt sich der Netzzustand als Knotenpunkt in einem n-dimensionalen Hypercube auffassen.

Bei einer Eingabe läuft das Netz von einem Knoten zum anderen, bis es sich stabilisiert hat. Bei falscher Eingabe stabilisiert es sich in der Nähe des gewünschten Knotens.

Stabilität eines Netzes

Cohen und Grossberg haben gezeigt, daß ein Netz mit symmetrischer Gewichtsmatrix $w_{i,j} = w_{j,i}$ und einer mit Nullen gefüllten Diagonale $w_{i,i} = 0$, stabil ist. Kein Neuron erhält also seine eigene Ausgabe wieder als Teil seiner Eingabe. Und die Ausgaben zweier Neurone i und j, die das jeweilige andere Neuron als Teil seiner Eingabe bekommt, werden gleich stark gewichtet. „Teil seiner Eingabe“ bedeutet, daß jedes Neuron nicht nur einen speziellen ungewichteten Inputwert verarbeitet, sondern über die „Verteilungsschicht 0“ die Ausgaben aller Neuronen als gewichteten Input hat.

Kontinuierliche Systeme

Mit der Sigmoidfunktion $F(NET) = \frac{1}{1 + e^{-\lambda NET}}$ als nichtlineare Aktivierungsfunktion soll verhindert werden, daß große Inputs das Netz zu schnell sättigen, indem sie Werte zwischen 0 und 1 liefert für alle NET-Werte, und kleine Inputs sollen genügend aktiviert (Aktivierung nahe an 0 oder 1 liefern) werden.

Wenn $\lambda \rightarrow 0$ geht, entfernen sich die Stabilisierungspunkte immer mehr von den Knoten im Hypercube und es entstehen zusätzliche Stabilisierungspunkte/Energiefunktionsminima. Wenn $\lambda \rightarrow \infty$ geht, erhält man aus dem kontinuierlichen das diskrete System mit Schwellwert 0.

Boltzmann-Maschine

Um die Schwäche von Hopfieldnetzen, lokale Minima anzusteuern anstatt des globalen, auszubügeln, verwendet man anstelle des deterministischen Statuswechsel eines Neurons einen statistischen.

Die Neuronen haben mit einer von der Boltzmann-Verteilung abhängigen Wahrscheinlichkeit den Output 1 bzw. 0. Die Wahrscheinlichkeit wird durch eine Temperaturvariable, die langsam gesenkt wird, und von dem Netzeingang des Neurons und seinem Schwellwert gesteuert.

Die Systemenergie kommt in der Wahrscheinlichkeit auch als Variable vor. Bei niedriger Temperatur wird geringe Energie wahrscheinlicher. Das ist die Verbindung zur Energiefunktion der Hopfieldnetze.

Die Temperatur bestimmt den Grad der Wahrscheinlichkeit, mit dem ein Element bei gegebenem Eingabewert 1 oder 0 annimmt. Die Temperatur gibt die Steilheit der Wahrscheinlichkeitsfunktion an. Geht T gegen 0 wird die Kurve steiler und verhält sich wie bei Schwellenwertelementen. Das Senken der Temperatur im Verlauf der Zeit verhindert das Hängenbleiben in lokalen Minima der Energiefunktion.

2.5.2 Anwendungen

Das Problem bei den Anwendungen besteht darin, die gegebene Fragestellung auf das Netz abzubilden. Man geht dabei so vor, daß man eine Energiefunktion E' sucht, die vom Input und Output des Netzes abhängt. Dabei stellen Input und Output Problemgrößen dar. E' muß genau dann minimal sein, wenn das Problem optimal gelöst ist. Der nächste Schritt ist der, die Energiefunktion E' mit der allgemeinen Energiefunktion E durch Bestimmung der Gewichte zur Äquivalenz zu bringen. Die Idee besteht nun darin, das Netz zu Beginn in einen hohen Energiezustand zu versetzen und dann durch Starten des Netzes das globale Minimum zu finden. Daß die Energiefunktion bei jedem Berechnungsschritt abnimmt, ist sichergestellt und bewiesen worden. Was aber nicht sicher ist, ist das Finden des globalen Minimums. Hierbei sollen Techniken wie die Boltzmann-Maschine helfen.

Sehr vorteilhaft ist die Geschwindigkeit der Netze. Auch wenn die Problemstellung wächst, ändert sich die Konvergenzzeit des Netzes nur wenig.

2.6 Cognitron und Neocognitron

Beides sind Modelle zur Mustererkennung, die sich an der Funktion des Gehirns orientieren.

2.6.1 Cognitron

Das Netz ist ein Feedforward, bei dem man die jeweils vordere presynaptisch und die nachfolgende postsynaptisch nennt. Es gibt erregende und hemmende Zellen, die die jeweiligen postsynaptischen Stellen entsprechend beeinflussen. Im Gegensatz zu den bisherigen Modellen sind die postsynaptischen Neuronen nur mit einem Teil der vorherigen Schicht, ihrer sog. Verbindungszone, verbunden (analog zum Gehirn).

Der Ausgang einer Erregungszelle wird durch das Verhältnis ihrer erregenden und hemmenden Inputs bestimmt. Man verwendet das Weber-Fechner Gesetz aus der Neurophysiologie als Outfunktion, welche nicht-linear ist.

Jedem erregenden postsynaptischen Neuron i gehört eine bestimmte Verbindungszone von presynaptischen Neuronen an. Dieser Verbindungszone ist in der presynaptischen Schicht ein hemmendes Neuron zugeordnet, das mit demselben Neuron i verbunden ist. Die Gewichte dazu werden nicht trainiert, sondern so gesetzt, daß ihre Summe 1 ergibt. Das Gewicht vom hemmenden zum postsynaptischen Neuron wird jedoch trainiert.

Der Output eines Neurons wird durch die Output-Summe seiner Konkurrenzzonennachbarn gehemmt. Zur Konkurrenzzone gehören Neuronen, die mit dem betrachteten viele presynaptische Neuronen gemeinsam haben. Die Hemmung war erst rekurrent realisiert, dann jedoch durch eine Feedforwardlösung ersetzt worden, welche schneller ist (Stabilisierung).

Die Schichten sind zweidimensional aufgebaut. Jedes postsynaptische Neuron hat eine zweidimensionale Verbindungszone. Der Wirkungsbereich eines Neurons sind die Inputzellen, die das nachfolgende Neuron beeinflussen. Durch die Mehrschichtigkeit des Netzes wächst der Wirkungsbereich der Inputschicht von einer zur nächsten, bis zur Outputschicht, die den gesamten Input als Wirkungsbereich hat.

Um mit wenig Schichten auszukommen, vergrößert man die Verbindungszone mit zunehmender Schichthöhe.

Beim Training werden nur die Gewichte des Neurons, welches in seiner Konkurrenzzone am stärksten reagiert, verstärkt.

2.6.2 Neocognitron

Dieses ist eine Weiterentwicklung des Cognitrons, die sich noch mehr am Gehirn orientiert und dadurch unabhängig von Verschiebung und Deformation bei der Mustererkennung ist.

Jede Schicht besteht aus simplen und komplexen Zellen. Die simplen erkennen Muster(teile) in der vorhergehenden komplexen Schicht. Die simplen Zellen sind dazu in Bereiche aufgeteilt, von denen jeder eine bestimmte Abart desselben Musters erkennen kann. Jedes Neuron eines simplen Bereichs checkt einige „komplexe“ Neuronen, die dieselbe relative Lage in ihrem Komplexzellenbereich haben. Die Empfangsbereiche der simplen Zellen aus verschiedenen Bereichen überlappen sich dabei.

Jedem simplen Zellbereich einer Schicht ist ein komplexer Zellbereich nachgeschaltet, in dem eine komplexe Zelle jeweils einige simple Neuronen checkt. Indem die komplexe Zelle reagiert, sobald eine ihr zugeordnete simple Zelle reagiert, wird das im S-Bereich identifizierte Muster positionsunabhängiger gemacht. Das komplexe Neuron zeigt also an, ob in den Zuständigkeitsbereichen ihrer S-Zellen die Musterabart des S-Zellenbereichs entdeckt wurde.

Das Neocognitron arbeitet nach dem Prinzip der Ähnlichkeit von Zeichen. Daher führt Lernen ohne Lehrer zum Erfolg, wenn sich die Zeichen gut unterscheiden. Andernfalls ist Lernen mit Lehrer nötig, da man die Ausgabe dann mitbestimmen kann.

Ohne Lehrer: Die Gewichte zwischen K-Zelle und S-Zelle sowie zwischen hemmenden Zellen und den vorgenannten werden verstärkt, falls

- die K-Zelle auf das Muster reagiert und
- die S-Zelle die stärkste Reaktion in ihrer Konkurrenzzone hat.

Dabei werden die Gewichte der anderen Zellen im Zellbereich gleichermaßen erhöht, wie die mit der stärksten Reaktion. Dadurch erkennen alle Zellen eines Bereichs dasselbe Muster an verschiedenen Orten.

Mit Lehrer werden die Schichten ebenfalls nacheinander trainiert.

3. Schwächen & Stärken Neuronaler Netze

3.1 Schwächen und Stärken im Allgemeinen

3.1.1 Die Theorie Neuronaler Netze

Über den Fluß der Information und ihre Darstellung und Verarbeitung im Netz ist zur Zeit noch nichts bekannt. Die Gleichungen und Matrizen zum Verhalten der Gewichte und Schwellwerte sagen darüber nichts aus.

Es gibt kein einheitliches Verfahren zur Kodierung der Ein- und Ausgaben. Ferner existiert auch keine einheitliche Notation für Netzwerke.

Für den Lernfaktor α gibt es keine Theorie, die seine optimale Bestimmung erleichtert. Ist er zu klein, dann hat ein Trainingsschritt wenig Wirkung. Ist er zu groß, dann lernt das Netz nicht, weil die Gewichtsveränderungen zu groß sind.

3.1.2 Vergleich Gehirn – Neuronale Netze

Obwohl das Gehirn als Vorbild dient und die Netze auch aus Erfahrung lernen, Beispiele verallgemeinern etc., können sie doch noch nicht so komplexe Aufgaben lösen. Wie beim Gehirn ist ihr Verhalten weitgehend unvorhersagbar, da nicht immer alle Eingaben getestet werden können.

Beide arbeiten stark parallel, wobei beim Gehirn hunderte echt verschiedene Zelltypen spezielle Aufgaben erledigen, beim Netz jedoch alle Zellen einheitlich sind. Außerdem ist Zahl der Schichten, ihre Lage und Verbindung untereinander beim Gehirn anders.

3.1.3 Vergleich von Neumann-Rechner –Neuronale Netze

Da Neuronale Netze meist auf von-Neumann-Rechnern simuliert werden, wird ihre hochgradig parallele Datenverarbeitung nicht ausgenutzt.

Im Vergleich zum Gehirn und zu Expertensystemen haben Netze die Schwäche, daß sie nicht erklären können, wie sie ein Problem lösen. Ihr Entscheidungsprozeß ist nicht nachvollziehbar.

Die unterschiedlichen Aufgaben erfordern bei Netzen und von-Neumann-Rechnern meist unterschiedliche Speicherkapazitäten zur Problemlösung. Es gibt Beispiele, die die Netze begünstigen und andere, die die Anzahl der Bits zur Speicherung der Gewichte exponentiell mit der Problemgröße wachsen läßt. Man weiß in der Netzwerkforschung nicht, wieviele solcher nichtspeicherbarer Probleme es gibt.

Eine Stärke der Neuronalen Netze ist, daß ihre Trainingszeit beinahe unabhängig von der Größe der Eingabe ist.

3.1.4 Der Trainingsprozeß

Außer dem üblichen zu (un-)überwachtem Training:

Es gibt keine Garantie, daß ein Netzwerk in einer endlichen Zeit auf ein Verhalten trainiert werden kann.

Das überwachte Lernen hat den Vorteil, daß man eine Kontrolle über das Verhalten des Netzes hat. Dies ist bei unüberwachtem Lernen nicht der Fall. Allerdings muß bei überwachtem Training vorher eine Menge von Trainingspaaren entwickelt werden, was voraussetzt, daß das Einsatzgebiet theoretisch erfaßt ist und sich nicht ändert. Mit überwachtem Training läßt sich ein Netz zu sinnlosem Verhalten erziehen, was bei unüberwachtem Training nicht möglich ist, denn dort wird erreicht, daß ähnliche Eingabevektoren den gleichen Ausgabevektor erzeugen, also Konsistenz. Dies ist sinnvoll und außerdem plausibler, da in der Natur auch keine Wunsch-Ausgabevektoren vorgegeben werden. Ein unüberwachtes Lernen ist Selbstorganisation. Das Netz organisiert sich durch den Trainingsalgorithmus so, daß es wesentliche Eigenschaften der Umwelt abbildet.

3.1.5 Lernfähigkeit Neuronaler Netzewerke

Eine Stärke ist, daß Neuronale Netze sich in ihrem Verhalten verändern können als Antwort auf ihre Umwelt. Sie sind anpassungsfähig und unempfindlich gegen Störungen und Unvollkommenheit des Eingabemusters. Dadurch sind sie zur Mustererkennung gut geeignet.

Neuronale Netze sind automatisch als Resultat ihrer Struktur zu Verallgemeinerungen fähig, ohne daß ihnen Intelligenz oder anderes einprogrammiert wäre.

Eine Schwäche ist, daß das Verhalten nur konsistent, aber nicht unbedingt logisch korrekt in seinen Ausgaben ist.

3.1.6 Wirtschaftlich

Netze werden die bisherigen Computer nur ergänzen, aber nicht ersetzen, da sie in anderen Bereichen ihre Stärke haben und keine logischen Maschinen sind.

3.2 Schwächen und Stärken im Speziellen

3.2.1 Das Perzeptron

Wie gehabt.

3.2.2 Hebb'sches Lernen

Bei diesem unbeaufsichtigtem Lernen wird das Gewicht zwischen zwei Neuronen i und j erhöht, wenn beide bei einer Eingabe aktiviert waren und eine Ausgabe lieferten:

$$w(n + 1) = w(n) + \alpha \quad OUT(i) \quad OUT(j)$$

Hier ist Lernen ein rein lokales Ereignis. Es wird gut das Lernen durch Wiederholung und Gewohnheit simuliert, da oft genutzte Verbindungen gestärkt werden. Andere Formen menschlichen Lernens sind hiermit nicht erklärbar. Auch gibt es Eingabemuster, die von diesem Algorithmus überhaupt nicht gelernt werden können.

3.2.3 Die Fehlerfunktion

Bei überwachtem Training definiert man für die Abweichung der erhaltenen zu den gewünschten Ausgaben eine Fehlerfunktion. Man kann z.B. die Einzelkomponentendifferenzen quadrieren und dann summieren. Bei einem Gewicht ist die Fehlerfunktion dann eine Linie, bei zwei Gewichten eine Ebene und bei mehr eine Hyperebene. Beim Training soll die Fehlerfunktion durch Gewichtsänderungen minimiert werden.

Eine Schwäche bei deterministischen Trainingsalgorithmen ist, daß die Gewichte durch ihn auf lokale Minima der Fehlerfunktion festgesetzt werden können. Wird der Lernschrittfaktor zu groß gewählt, dann oszilliert die Gewichtseinstellung des Alg. um ein Minimum und terminiert nicht.

3.2.4 Backpropagation

Dies ist eine systematische Methode für überwachtes Lernen. Voraussetzung ist eine stetig differenzierbare Aktivierungsfunktion. Meist wird die Sigmoidfunktion genommen, die den Vorteil hat, kleine Signale zu verstärken, größere jedoch weniger und dadurch den Wertebereich auf das 0,1 Intervall zu normieren.

Der Alg. verwendet als Fehlerfunktion für vorherige Schichten die berechneten Fehler der nachfolgenden, wobei ein vorheriges Neuron als Fehlerwert alle Fehlerwerte seiner direkten Nachfolger summiert, dieses Fehler allerdings mit den durch die nachfolgende Schicht korrigierten Gewichten jeweils multipliziert.

Im Vergleich zum Perzeptron (Dort wird jeweils die Geradengleichung zur Separation der Punkte bestimmt und daraus die Gewichte. Man kombiniert mehrere Geraden bzw. Halbebenen, indem man ihre Neuronenausgänge in einer weiteren Schicht zusammenfaßt mit entsprechenden Und-, Oder- und Nicht-Funktionen d.h. deren Geradengleichung bzw. dazugehörigen Gewichten. Da die Schwellwertfunktion t;1,0 zwischen den Schichten eine nicht-lineare Aktivierungsfunktion ist, ist es echte Mehrschichtigkeit.) besteht der Vorteil, daß alle Schichten des Netzes trainiert werden, was allerdings nachteilig sehr lange dauern kann. Der Alg. ist deterministisch s.o. dazu. Weitere Schwäche ist, daß bedingt durch eine fast 0-wertige Ableitung der Sigmoidfunktion nur sehr kleine Gewichtsänderungen erfolgen evtl. und so auch die Trainingszeit verlängert wird.

3.2.5 Simulated Annealing

Dies ist ein nicht-deterministischer Algorithmus für überwachtes Lernen. Statistische Trainingsmethoden verursachen zufällige Veränderungen der Gewichte. Gewichte, die zu einer Reduzierung der Fehlerfunktion führen, werden dabei mit größerer Wahrscheinlichkeit bei ihrem Wert belassen.

Dieser Algorithmus erlaubt dem System, auch Gewichtsveränderungen vorzunehmen, die die Fehlerfunktion verschlechtern. Die sich gleichmäßig verringernde Temperatur garantiert dabei, daß nur dann negative Veränderungen zugelassen werden, wenn dies zum Erreichen eines globalen Minimums führt.

Der Nachteil dieses Boltzmann-Prozesses ist, daß sehr viel Zeit nötig ist, um das globale Minimum für alle Trainingspaare zu erreichen.

3.3 Zusammenfassung

Perzeptrons sind gut verstanden und können die wenigen separierbaren Funktionen darstellen.

Das Hebb’sche Lernen war der erste Trainingsalg. für Netze. Lernen durch Wiederholung und Gewohnheit ist damit erklärbar, andere Lernformen nicht.

Backpropagation war der erste Algorithmus zur Verringerung der Fehlerfunktion für mehrschichtige Netze. Da er deterministisch ist, bleibt er gerne in lokalen Minima hängen.

Dieses Manko haben statistische Algorithmen nicht, die das physikalische Härten von Metallen simulieren und auch „schlechte“ Gewichtsänderungen zulassen. Dafür sind sie langsam.

Eine Verbesserung ist durch die Verbindung von Backpropagation und statistischem Algorithmus zu erwarten.

4. Theorie

4.1 Das Neuron

Grundlegend für ein Neuronales Netz ist die Idee, daß jedes Neuron eine eigene Verarbeitungseinheit bildet, wobei zwischen den Eingabeneuronen, Zwischenneuronen und Ausgabeneuronen unterschieden wird.

Im Prinzip wird in jedem Neuron

1. seine Eingabe zur sog. Netzaktivität „verdichtet“,
2. die Netzaktivität durch eine Aktivierungsfunktion auf den sog. Aktivierungszustand abgebildet und schließlich
3. der Aktivierungszustand durch eine Ausgabefunktion auf den Ausgabezustand abgebildet.

Allerdings ist bei den meisten Netzen die Ausgabefunktion die Identität, so daß die Ausgabe eines Neurons dann sein Aktivierungszustand ist.

4.1.1 Die Gewichte

Zur Ermittlung der Netzaktivität wird in meisten Modellen, neben der Eingabe von Signalen anderer Neuronen, noch ein Gewicht für jede Eingabe verwendet. Außerdem kommt manchmal noch jeweils die konstante Eingabe +1 eines fiktiven Neurons hinzu, dessen zugehöriges Gewicht die Funktion eines Schwellwerts übernimmt.

4.1.2 Die Aktivierung

Die Aktivierungszustände können abhängig von der Aktivierungsfunktion diskret oder kontinuierlich (dann meistens Intervalle) sein.

Die Aktivierungsfunktion kann stochastisch (abhängig von Wahrscheinlichkeitsverteilung über die Netzaktivität) oder deterministisch (dann meistens monoton wachsend, weil es natürlicher erscheint) sein. Die deterministischen Aktivierungsfunktionen werden in differenzierbare und nicht differenzierbare eingeteilt.

4.1.3 Die Netzaktivität

Hier gibt es verschiedene Verfahren. Z.B. die gewichtete Summierung der Eingabe beim Multilayer-Perceptron oder beliebig kompliziertere.

4.2 Das Neuronale Netz

4.2.1 Die Topologie

Die Topologie läßt sich durch eine Adjazenzmatrix aller Neuronen darstellen, bei der ein Eintrag (i,j) genau dann 1 ist, wenn eine Verbindung zwischen dem Ausgang von Neuron i und dem Eingang von Neuron j besteht.

Man unterscheidet folgende Topologien:

- Ungerichtetes Netz mit (i,j)=(j,i).
- Vollständig vernetzte Struktur, wobei alle Neuronen untereinander außer mit sich selbst verbunden sind.
- Vorwärtsvermittlungsnetz
- Mehrschichtiges Netz, bei dem jede Schicht nur mit ihrer direkten Vorgänger- bzw. Nachfolgeschicht vernetzt ist.

4.2.2 Exkurs

Es ist problematisch, Neuronale Netze, die keine Vorwärtsvermittlungsnetze sind, mathematisch zu analysieren, da ja in irgendeiner Form die Ausgabe eines Neurons wieder in seine Eingabe gelangt. Die Ausgabe solcher Neurone muß nicht konvergieren.

4.2.3 Die Gewichte

Die Gewichte lassen sich in einer (N+1,N)-Matrix darstellen, in der nicht nur die Gewichte zwischen den Neuronen, sondern auch der Schwellwert (Gewicht) des fiktiven Eingangs steht.

Am Anfang wird die Matrix meistens mit zufälligen Werten initialisiert. Eine 0 bedeutet, daß keine Verbindung besteht. Im Prinzip kann also die Topologiematrix entfallen. Allerdings kann man dann nicht mehr unterscheiden, ob ein Netz sich wie z.B. ein mehrschichtiges verhält wegen den Gewichten oder wegen tatsächlich fehlender Verbindungen.

4.2.4 Die Vermittlungsregel

Die Vermittlungsregel gibt an, in welcher Reihenfolge die Aktivierungszustände der Neuronen berechnet werden. Z.B.:

- Multilayer-Perceptron: bei allen Neuronen einer Schicht gleichzeitig.

- Hopfield-Netz: Ein Neuron wird zufällig ausgewählt zur Berechnung.
- Spike-getriebenes Netz: Die Neuronen senden nur eine bestimmte Zeit (abhängig vom Erregungszustand) Signale aus.

4.2.5 Die Lernregel

Beim *unüberwachten* Lernen wird versucht, durch die Häufigkeit und die Reihenfolge der einzelnen Trainingsmuster zu lernen. Einsatz häufig zum Klassifizieren eines „verrauschten“ Musters.

Falls die Aktivierungszustände {0,1} sind, dann lassen sich die Lernverfahren folgendermaßen gliedern:

- Hebb’sches Lernen: Nur die Gewichte der Verbindungen vergrößern, bei denen beide Neuronen aktiv sind. Andere unverändert lassen.
- Hebb/Anti-Hebb Lernen: Gewichte von beidseitig aktiven Neuronen vergrößern. Beidseitig inaktive belassen und einseitig aktive verkleinern.
- Hopfield Lernen: Gewichte der Verbindungen vergrößern, bei denen beide Neuronen den gleichen Aktivierungszustand haben. Alle anderen verringern.

Beim *überwachten* Lernen soll das Netz lernen, auf bestimmte Muster die gewünschte Ausgabe zu liefern. Dazu muß meistens eine Fehlerfunktion minimiert werden, die den Abstand zwischen Sollausgabe und Istausgabe mißt. Gebräuchliche Methode dazu: Abstandmessen mir einer Norm s.u.

Das zweite Unterscheidungskriterium bei Lernregeln ist, welche Parameter durch das Lernen verändert werden. Am häufigsten werden die Gewichte geändert. Ferner kann die Netztopologie oder als drittes die Aktivierungsfunktion bzw. Ausgabefunktion modifiziert werden.

4.3 Die Gradientenmethode

Verfahren zur Anpassung der Gewichte beim Multilayerperceptron.

4.4 Die Möglichkeiten eines Neuronalen Netzes

4.4.1 Äquivalenz mit endlichen Automaten

Neuronale Netze mit einer endlichen Anzahl von Neuronen (eigentlich gar nicht anders möglich) und einer endlichen Anzahl von Aktivierungszuständen sind mit endlichen Automaten äquivalent, falls die Aktivierungsfunktion deterministisch ist.

4.4.2 Äquivalenz mit Turingmaschinen

Neuronale Netze mit abzählbar unendlich vielen Aktivierungszuständen oder mit abzählbar unendlich vielen Neuronen sind äquivalent zu Turingmaschinen.

5. Konstruktion neuronaler Netze

5.1 Lösung von Optimierungsproblemen mit neuronalen Netzen

5.1.1 Das analoge Hopfieldnetz

Das analog Hopfieldnetz eignet sich aus folgenden Gründen zur Lösung von Optimierungsproblemen:

- Die hohe Anzahl von Verbindungen zwischen den Neuronen läßt es für Aufgaben geeignet erscheinen, in denen eine kollektive Entscheidungsfindung stattfinden muß.
- Durch Übertragung eines diskreten Problems in den kontinuierlichen Raum des analogen Netzes kann eine Minimumsuche über der Energiefunktion durchgeführt werden. Diese gilt als effektiver, als die Suche im diskreten Raum.
- Kein Training des Hopfieldnetzes nötig, wodurch viel Zeit gespart wird.

Das Hopfieldnetz hat Neuronen, die jeweils einen extern gespeisten Bias besitzen, der die neutrale Position des Neurons bestimmt, indem er die Input-Output-Relation der Aktivierungs- (Gain-) Funktion entlang der Inputachse (Netzeingang) verschiebt.

Startet man das Netz aus einem beliebig gewählten Anfangszustand, so ändert es seinen Zustand so lange, bis es einen stabilen Zustand erreicht und stoppt. Dieser Zustand entspricht dann einem (nicht *dem*) Minimum in der gewählten Energiefunktion.

5.1.2 Mapping von Problemen auf Hopfieldnetze

Dazu benötigt man vier Schritte:

1. Es muß eine geeignete **Repräsentation** der Problemlösungen als Ausgangsfunktion des neuronalen Netzes gefunden werden.
2. Die **Energiefunktion** muß so gewählt werden, daß die niedrigsten Energiezustände den besten Problemlösungen entsprechen. Durch sog. *Zwangsterme* in der Energiefunktion wird eine niedrige Energie für gültige Lösungen geliefert; ungültige erhalten hohe Energie. Sog. *Kostenterme* sorgen dafür, daß von den gültigen Lösungen die besten bevorzugt werden. Im allgemeinen werden Zwangsterme höher gewichtet in der Energiefunktion als die Kostenterme, um möglichst nur gültige Lösungen zu bekommen.
3. Aus der so gewonnenen Energiefunktion werden dann die **Gewichte** (definiert durch quadratische Terme der Energiefunktion) und die **Biasterme** (…durch lineare Terme…) bestimmt.
4. Die Bestimmung der **Initialwerte** und **Parameter** erfolgt meistens experimentell. Denn bevor man die Simulation starten kann, müssen noch die Eingangsspannungen (manchmal zufällig), die Steilheit der Aktivierungsfunktion, die Gewichtung der Zwangsterme und Kostenterme festgelegt werden.

1 bis 3 sind für die Entwicklung des Netzes, 4 für die Güte der Lösung verantwortlich.

5.2 Evolutionäre Programmierung

Das Konzept der evolutionären Programmierung scheint geeignet zu sein, das Training, also die Wahl der Gewichte, eines neuronalen Netzes zu verbessern.

Betrachtet man ein Netz mit n zu bestimmenden Gewichten, so besteht die Anfangspopulation aus mehreren zufällig gewählten n-dimensionalen Vektoren, deren Komponenten den Gewichten entsprechen. Jedem Vektor wird über die Payoff-Funktion (Resultat, Entscheidung) ein Fehlerwert zugeordnet. Aus den Vektoren mit den kleinsten Fehlerwerten werden zufällig die Eltern der nächsten Generation ausgewählt. Die Nachkommen werden dabei durch Addition einer Gauß'schen Zufallsvariable mit Mittelwert Null zu jeder Komponente des Eltern-

vektors erzeugt. Die Varianz wird dabei proportional zum Fehlerwert des Elternvektors gewählt.

Die Mutation durch eine Gauß'sche Zufallsvariable garantiert, daß jede Kombination von Gewichten erzeugt werden kann. Durch die anschließende probabilistische Selektion der Eltern kann somit auch das globale Minimum der Fehlerfunktion gefunden werden.

Üblicherweise wird in jeder Generation die Hälfte der Vektoren als neue Elternvektoren ausgewählt, von denen jeder genau einen Nachkommen erzeugt. Dadurch ist immer die gleich Zahl an Vektoren in jeder Generation vorhanden. Man kann z.B. die Elternauswahl von einer Wahrscheinlichkeit für die einzelnen Vektoren abhängig machen, die wiederum von deren Bewertung abhängt.

5.3 Konstruktionsalgorithmen für Netzwerktraining

Trainingsalgorithmen für neuronale Netz arbeiten meist nach dem Prinzip, zu einem gegebenen Netz die Gewichte so zu bestimmen, daß eine gewünschte Funktionalität erreicht wird.

Es gibt auch den Ansatz, sowohl geeignete Gewichte als auch gleichzeitig ein geeignetes Netz zu konstruieren, denn:

- Topologie des Netzes unwichtig für die meisten Anwendungen.
- Einfacher, da auf kein vorgegebenes Netz angepaßt werden muß.

Das Tower-Konstruktionsverfahren

Die Erzeugung von Gewichten für ein einzelnes Perceptron, so daß eine maximale Anzahl von Trainingsmustern korrekt klassifiziert wird, ist ein klassisches Problem, für das mehrere Verfahren, z.B. der Pocket-Algorithmus, bekannt sind.

Der Pocket-Algorithmus kann dies in endlicher Zeit für separierbare Probleme. Mit steigender Iteration werden die Gewichte optimal.

Das Tower-Konstruktionsverfahren erzeugt iterativ ein Netzwerk aus Perceptrons in der Form eines Turms. Die Gewichte jedes neu eingefügten Perceptrons werden dabei durch den Pocket-Algorithmus bestimmt:

1. Konstruiere optimale Gewichte für ein einzelnes Perceptron. Wenn alle Trainingsmuster korrekt klassifiziert werden, dann fertig, sonst halte die Gewichte fest.
2. Konstruiere optimale Gewichte für ein weiteres Perceptron mit Inputs von ebenfalls allen Trainingsmustern und zusätzlich dem Output vom vorherigen Perceptron. Es kann gezeigt werden, daß dieses Perceptron dann mehr Trainingsmuster klassifizieren kann als sein Vorgänger. Halte die Gewichte auch hiervon fest.
3. Schritt 2 wiederholen, bis alle oder die maximal mögliche Anzahl von Trainingsmustern korrekt klassifiziert wird.

Da der Output jedes Levels mehr Trainingsmuster klassifizieren kann als der vorhergehende Level, erreicht man theoretisch eine Konvergenz in ein Netzwerk mit optimaler Performance.

Das gleiche Verfahren kann man sich auch mit anderer Netzstruktur vorstellen. Bei der Pyramidenkonstruktion wird im Vergleich zur Towerkonstruktion ein neues Perceptron mit den Outputs aller Vorgänger verbunden und nicht nur mit seinem direkten.

5.4 Grundlegende Konstruktionskomponenten

Was sind die wichtigsten Komponenten, die bei der Konstruktion eines neuronalen Netzes eine Rolle spielen?

Ein- und Ausgabedaten

Da neuronale Netze nur numerische Werte verarbeiten, sind Ein- und Ausgabedaten in der Regel einer Vor- bzw. Nachverarbeitung zu unterziehen, die sehr komplex sein kann.

Aktivierungsfunktion

Hier haben sich nicht-lineare Funktionen bewährt, nachdem festgestellt wurde, daß lineare Funktionen bei einigen Problemen nicht zu einer Lösung führen können. Gebräuchlich sind Sprungfunktionen, Threshold-Logik oder Sigmoidfunktionen.

Topologie

Die Anzahl der Ein- und Ausgabeneuronen ergibt sich direkt aus der Aufgabenstellung. Schwieriger ist die Bestimmung der Anzahl von verdeckten Neuronen, da

dazu kein Verfahren bekannt ist. Werden zu wenige verdeckte Elemente verwendet, so verlängert sich i.d.R. der Lernprozeß oder er scheitert ganz. Was bei zu vielen passiert, ist nicht bekannt.

Wie sollen die Verbindungen aussehen? Vollständig vernetzt, in Schichten oder matrizenmäßig angeordnet, mit oder ohne Rückkopplungen? Oder sogar ein dynamisches System, in dem sich Verbindungen auflösen oder neu aufbauen?

Es gibt z.Z. keine allgemeingültigen Konstruktionsprinzipien, man experimentiert.

5.5 Konstruktion als evolutionärer Prozeß

Jedes neuronale Netz läßt sich eindeutig durch eine endliche Menge von diskreten Parametern beschreiben. Eine Mutation erfolgt dann durch die zufällig ausgewählte Veränderung eines einzelnen Parameters. In der anschließenden Test- und Bewertungsphase wird entschieden, ob das neue Netzwerk verworfen wird oder den Vorgänger ablöst.

Zunächst wird ein initiales Netz bestimmt, das eine einfache Aufgabe löst. Durch ständige Verbesserungen soll es zu Realisierung von komplexen Funktionen geführt werden. Solange das Netz nicht die erforderliche Güte erreicht hat, erfolgen Zyklen von Veränderungen und erneutem Testen und Bewerten. Falls der Typ es erfordert auch inklusive Training.

In der Mutationsphase wird einer der diskreten Parameter verändert, durch die das Netz definiert ist. **Generelles Parameterschema** zur Definition von künstlichen Netzen:

Ein Netzwerk besteht aus orthogonal (rechtwinklig) angeordneten Spalten und Reihen von Neuronen in einem n-dimensionalen Raum. Innerhalb dieses *Neuronen-Gitters* werden *Regionen* festgelegt. Synaptische Verbindungen können zwischen verschiedenen Regionen oder innerhalb derselben Region bestehen. Sie werden als

- konvergent: einzelnes Neuron aus Quellregion ist mit mehreren aus Zielregion verbunden,
- travers: 1:1-Verbindungen,
- divergent: mehrere Quellen und ein Ziel, und
- lateral: innerhalb einer Region

klassifiziert. Weitere Parameter beschreiben die *Verbindungen selbst*. Man unterscheidet fest verdrahtete oder programmierbare und hemmende oder stimulierende synaptische Verbindungen.

Es besteht die Möglichkeit, daß lokale Minima erreicht werden, die mit diesem Algorithmus nicht mehr verlassen werden können. Hier könnte die Veränderung von mehreren Parametern gleichzeitig Abhilfe schaffen. Außerdem nimmt die Wahrscheinlichkeit für eine funktionale Verbesserung mit der Zahl erfolgreicher Mutationen exponentiell ab, da der Alg. vergleichbar mit der Bestimmung einer optimalen Kombination von Netzwerk-Parametern ist. Wenn die Bewertung automatisch für jedes Netz funktioniert ist kein Einfluß von außen nötig.

5.6 Regelbasierte Systeme und neuronale Netze

In der künstlichen Intelligenz gibt es zwei wesentliche Ansätze, um menschliches „intelligentes“ Verhalten zu simulieren. Die *deklarativen* (erklärenden) Systeme sind als Expertensysteme oder regel- bzw. wissensbasierte Systeme bekannt. *Reflexive* (rückbezügliche) Mechanismen haben ihren Ursprung nicht in kognitiver Einsicht sondern in der Intuition. Diese werden durch neuronale Netze realisiert.

Aus der Erkenntnis heraus, daß sich regelbasierte Systeme und neuronale Netze in ihrer Funktionsweise ergänzen, bietet es sich an, das beide Ansätze integriert werden, da ihre Stärken und Schwächen „spiegelverkehrt“ sind zueinander.

- Eindeutigkeit bei regelbasierten Systemen
- einfache Begründungen für Schlußfolgerungen möglich
- Netze können unpräzises und komplexes Wissen verarbeiten
- sie können durch Beispiele lernen, also ohne explizites Wissen angeben zu müssen

An vier Stellen ist der Ersatz von Expertensystemen durch Netze nicht sinnvoll:

- wenn Konsequenzen von Entscheidungen offensichtlich oder gefährlich sind
- wenn die Relation zu selten auftaucht oder deren Zusammenhang zu verborgen ist
- wenn zu viele Informationen in einer bestimmten Zeit zu lernen sind
- wenn das Netz sofortige optimale Lösungen bevorzugt, anstatt erst einen Nachteil in Kauf zu nehmen, um später den größeren Nutzen zu haben.

5.6.1 Regeln steuern ein neuronales Netz

Ein regelbasierter Monitor steuert den Ablauf des Gesamtsystems. Er entscheidet, ob für die aktuelle Steuerungsaufgabe das regelbasierte System oder das neuronale Netz benutzt wird. Solange die Steuerungsdaten des Netzwerks gut genug sind, werden sie an die ausführende Einheit gesendet. Sonst übernimmt das regelbasierte System die Steuerungsaufgabe und das Netzwerk wird mit den Daten der Regeln trainiert, bis ein erneuter Wechsel aufgrund der verbesserten Eigenschaften des neuronalen Netzes erfolgen kann.

Ähnlich wie bei Lebewesen werden die Aktionen durch Wiederholung gelernt. Die Regeln geben dabei die notwendige Hilfestellung, um das Netz zu trainieren.

5.6.2 Ein neuronales Netz lernt Regeln

Hier soll das neuronale Netz regelbasierte Schlußfolgerungen vollständig ersetzen. Es soll das interne Problemlösungsverhalten lernen, d.h. es soll nicht direkt vom Problem zur Lösung kommen, sondern die einzelnen Zwischenschritte sollen dem Netz beigebracht werden.

Die Ausgangsbasis ist die Regelmenge und der Status (Arbeitsspeicher), der den aktuellen Zustand aus binären Relationen festhält. Dieser wird solange durch die Anwendung von Regeln verändert, bis die gestellte Aufgabe gelöst ist.

Das Netz bekommt als Trainingseingabemuster den aktuellen Zustand und als Sollausgabe den durch Regeln veränderten Zustand. Dadurch lernt es die Einzelschritte. Die Entscheidung, ob der Änderungsvorschlag des Netzes oder des Regelsystem angenommen wird, trifft ein dafür offline trainiertes weiteres Netz. Wird der Vorschlag des regelbasierten Systems akzeptiert, wird das Netz nachtrainiert.

Wenn das Netz gut genug das Regelsystem ersetzen kann, wird in einer zweiten Phase versucht, zwei Schlußfolgerungen in einem Schritt durchzuführen. Dazu werden die Zustandsänderungen trainiert, die nach zwei Schlußfolgerungsschritten auftreten.

5.6.3 Vergleich der beiden Ansätze

In beiden Fällen wird so vorgegangen, daß das neuronale Netz trainiert wird, aufgestellte Regeln zu lernen, um so in Konkurrenz zu dem regelbasierten System zu

treten. Eine Überwachungsinstanz entscheidet darüber, welche Methode angewendet wird. Diese ist einmal ein Regelwerk, im zweiten Fall ein Netz.

Da in beiden Fällen zunächst Wissen in Form von Regeln vorhanden sein muß, wird der eigentliche Vorteil von neuronalen Netzen, unklares Wissen anhand von Beispielen zu lernen, nicht genutzt. Die Zielrichtung ist eher eine Verbesserung der Performance eines reinen regelbasierten Systems.

Das erste System hat den Vorteil, die vier o.g. Situationen abzufangen, die ungünstig für die Bearbeitung durch ein neuronales Netz sind.