KfK 5184
Januar 1994

Modelle Maschinellen Lernens

Symbolische und konnektionistische Ansätze

Th. Weinberger, H. B. Keller, W. Jakob, B. große Osterhues
Institut für Angewandte Informatik
Projekt Schadstoff- und Abfallarme Verfahren

Kernforschungszentrum Karlsruhe

Kernforschungszentrum Karlsruhe

Institut für Angewandte Informatik
Projekt Schadstoff- und Abfallarme Verfahren

KfK 5184

Modelle Maschinellen Lernens

Symbolische und konnektionistische Ansätze

Th. Weinberger, H. B. Keller,
W. Jakob*, B. große Osterhues

(* Mitarbeit am Kapitel evolutionäre Algorithmen)

Kernforschungszentrum Karlsruhe GmbH, Karlsruhe

Kernforschungszentrum Karlsruhe GmbH
Postfach 3640, 76021 Karlsruhe

ISSN 0303-4003

Zusammenfassung:

Das Ziel einer innovativen Prozeßführung ist die Optimierung des Betriebes technischer Anlagen/Verfahren hinsichtlich einer prozeßintegrierten Vermeidung der Bildung von Schadstoffen im Sinne eines vorbeugenden Umweltschutzes.
Neben den konventionellen Methoden der Informationstechnik sind hierzu auch neuartige Verfahren wie wissensbasierte Techniken im erweiterten Sinne einzusetzen. Lernverfahren können zur automatischen Wissensakquisition und -validierung herangezogen werden.
Der vorliegende Bericht ist eine Einführung in das Gebiet des Maschinellen Lernens. Es werden sowohl die theoretischen Grundlagen als auch einige Maschinelle Lernverfahren explizit beschrieben. Vervollständigt wird diese Darstellung durch eine kurze Übersicht über evolutionäre Algorithmen und eine allgemeine Beschreibung konnektionistischer Ansätze in Form neuronaler Netze. Abgeschlossen wird diese Einführung durch eine kritische Gegenüberstellung und Bewertung dieser Ansätze, insbesondere im Hinblick auf deren Einsatz zur heuristischen Verarbeitung von Wissen im Rahmen einer innovativen Prozeßführung.

Models for machine learning - symbolic and connectionistic approaches

Abstract:

The optimization of the operation of technical systems or processes, in order to avoid the production of ecologically harmful substances, in sense of a preventive conservation of the environment, is the aim of an innovative process-control.
Besides the standard methods of computer science, new methods like knowledge-based techniques should be applied. For automatic knowledge acquisition and -validation learning techniques can be used.
This report is an introduction into the field of machine learning. The fundamental principles and some of the most popular machine learning algorithms are described explicitly. This description is completed by a short review of evolutionary algorithms and a very general introduction to connectionistic approaches like neural nets. This introduction concludes with a critical comparison of these approaches and some remarks to their practical usability with respect to heuristic processing of knowledge within the scope of an innovative process-control.

Inhaltsverzeichnis

1 Einleitung

Betrachtet man einen technischen Prozeß hinsichtlich der Bildung von Schadstoffen, so muß ein *intelligenter Ansatz zur Prozeßführung* grundsätzlich eine Minimierung der entstehenden Schadstoffe anstreben, um somit die Problematik der Beseitigung von Schadstoffen quasi "an der Wurzel" anzugehen. Daher muß das oberste Ziel bei der Führung technischer Anlagen die *prozeßintegrierte Vermeidung* der Bildung von Schadstoffen sein.

Der anhaltende Fortschritt im Bereich der Hardware, welcher weiter steigende Leistung bei sinkenden Preisen bietet, und die kontinuierliche Weiterentwicklung der Softwaretechniken erschließen heute neue Möglichkeiten, den Wirkungsgrad und die Verfügbarkeit einer Anlage mit Hilfe weiter verbesserter, d. h. intelligenterer Prozeßführungssysteme noch zu steigern.

Neben den konventionellen Methoden der Informationstechnik sind hierzu auch neuartige Verfahren wie wissensbasierte Techniken einzusetzen: Beispielsweise können Lernverfahren aus dem Bereich der Künstlichen Intelligenz zur automatischen Wissensakquisition und -validierung herangezogen werden. Das Ziel sind Systeme, die das Wissen zur optimalen Führung eines technischen Prozesses sukzessive on-line ableiten und dem Bediener zur Verfügung stellen /Keller, Weinberger 51/.

Hierzu bildet der vorliegende Bericht eine Einführung und einen Überblick in den Themenbereich des *Maschinellen Lernens*, als Teilgebiet der Künstlichen Intelligenz. Es wird nicht nur die Thematik, sondern auch die zum Verständnis notwendige Terminologie sukzessive aufgebaut.

Nach dieser Einleitung werden im zweiten Kapitel die für eine rechnerorientierte Verarbeitung von Wissen einsetzbare *Inferenzmethoden* als Grundlage für jedes intelligent agierende System vorgestellt und die Problematik der Wissensakquisition aufgezeigt. Ein Ansatz zur Lösung der Probleme der "manuellen" Wissensakquisition ist die automatische *Wissensakquisition* und damit das Maschinelle Lernen. Im dritten Kapitel wird der grundsätzliche Aufbau eines maschinell lernenden Systems beschrieben und eine mögliche *Klassifikation* für maschinell lernende Systeme vorgestellt.

Anhand der bekanntesten *maschinellen Lernverfahren,* wie ID3, die Versionenraum-Methode mit dem Kandidaten-Eliminations-Algorithmus, BACON, der AQ-Algorithmus, EPAM und DISCIPLE wird im vierten Kapitel ein Einblick in die gängigsten Techniken des maschinellen Lernens gegeben. Dieses Anliegen wird durch entsprechende Beispiele unterstützt.

Das fünfte Kapitel entstand unter Mitarbeit von Herrn W. Jakob, dem wir für seine Mitarbeit danken möchten, und stellt eine Einführung in den Bereich der *evolutionären Algorithmen*, ein insbesondere für Optimierungsaufgaben, als auch für das Lernen, gut geeignetes Paradigma, dar. Im sechsten Kapitel werden *konnektionistische Ansätze* und neuronale Netze als deren Vertreter zunächst allgemein, später anhand der bekanntesten Vertreter, in ihrem Aufbau (Architektur) und ihrer Funktionsweise beschrieben. Es werden sowohl die wichtigsten Netzwerktypen vorgestellt als auch die zugehörigen Lernalgorithmen und ihre theoretischen Fähigkeiten/Grenzen dargestellt.

Im abschließenden siebten Kapitel erfolgt eine kritische Würdigung und Gegenüberstellung der in den vorangegangenen Kapiteln beschriebenen Verfahren/Paradigmen zum Maschinellen Lernen im Hinblick auf ihren Einsatz zur Verarbeitung von *unscharfem Wissen* und der Bildung und dem Verwerfen von Hypothesen im Rahmen einer konkreten Anwendung.

Der Inhalt dieser Arbeit ist vor dem Hintergrund der Entwicklung eines maschinell-lernenden-Systems zu betrachten, das (technisches) Wissen zur Führung komplexer technischer Prozesse automatisch akquiriert und validiert (vgl. /Keller, Weinberger 49/) und dient dabei sowohl als Einstieg als auch als Überblick und Bestandsaufnahme zum aktuellen Stand der Forschung auf diesem Gebiet, aus der Sicht des Maschinellen Lernens.

Diejenigen maschinellen Lernverfahren, die zur Zeit bereits praktisch eingesetzt werden, basieren nahezu ausschließlich auf den klassischen Verfahren ID3 und AQ mit Attribut / Wert-Repräsentationen, da den Anwendern für die "moderneren" Verfahren noch keine kommerziellen Programm- und Entwicklungspakete zur Verfügung stehen /Morik 90/.

Auf dieser Grundlage existieren derzeit im wesentlichen zwei Anwendungsgebiete für Verfahren zum Maschinellen Lernen:

- Unterstützung eines Wissensingenieurs beim Aufbau von Regelbasen; bzw. eine automatische Generierung von "Teil-" Regelbasen.

  Beispielsweise wurde ein maschinell lernendes System bei BP zum Aufbau der Regelbasis eines Expertensystemes zur Modellierung des Designs eines Gas-Öl-Separators mit 3000 Regeln eingesetzt.

- Das Finden optimaler Prozeduren, z. B. im Rahmen einer Entscheidungsfindung oder bei der Arbeitsplanung.

  Als Beispiel hierfür wird vom amerikanischen Elektrokonzern Westinhouse ein maschinell lernendes System zur Optimierung des Durchsatzes in einer der Produktionsanlagen eingesetzt.

Auch die dargestellten konnektionistischen Verfahren haben im praktischen Einsatz bereits beachtliche Erfolge gezeigt. Neuronale Netze wurden z. B. in der Aluminiumerzeugung, in der Medizin, bei der Kreditvergabe im Bankwesen etc. eingesetzt (vgl. /Chaouli, Froitzheim 16/).

Die in diesem Bericht vorgenommenen Bewertungen der dargestellten Konzepte und Ansätze gründen ausschließlich auf der Problemstellung, das Verhalten und die Struktur eines dynamischen Systems zu lernen. Dieser Problemkomplex gilt als äußerst innovativ, da sich derzeit nur wenige Forschungsprojekte *ansatzweise* damit auseinandersetzen.

In ähnlicher Weise wie in diesem Bericht wird in den beiden vorangegangenen Berichten "Lernmodelle und Wissensverarbeitung" (/Keller, Weinberger 51/) und "Wissensbasierte Systeme - Darstellungs- und Verarbeitungsmodelle" (/Keller et al. 53/) einmal die Sicht der Lernpsychologie und zum anderen die Sicht der *klassischen* Künstlichen Intelligenz dargestellt. Durch diese "Trilogie" wird der interdisziplinäre Charakter, den die Forschungsaktivitäten im Bereich lernender Systeme aufweisen, umfassend dargestellt und die Entstehung eines neuen Konzeptes zur Umsetzung und Anwendung des *Prinzips der heuristischen Modellierung dynamischer Systeme* /Keller, Weinberger 50/ sowie dessen Wurzeln aufgezeigt.

2 Grundlagen intelligent agierender Systeme

Die Entwicklung intelligent agierender Systeme geschieht zumeist vor dem Hintergrund einer konkreten Anwendung. Im allgemeinen besteht die Aufgabe des intelligent agierenden Systems dann darin, einen Operateur bei der Lösung der in der Anwendung auftretenden Probleme durch Verfahren zur Klassifikation (z. B. im Rahmen einer Diagnose) und/oder durch die Bildung von Handlungssequenzen (z. B. die Planung von Bedienereingriffen, die Erstellung von Ablaufplänen etc.) zu unterstützen.

Um eine solche Unterstützung bei der Lösung von Problemen zu erreichen, muß das intelligent agierende System über Wissen um die Sachverhalte der Anwendung verfügen. Es muß die Fähigkeit besitzen, dieses Wissen in einer strukturierten Form zu repräsentieren und, im Sinne von Schlußfolgerungen, zur Wissenserweiterung (Ableitung weiteren Wissens) verarbeiten können.

Damit ein intelligent agierendes System Wissen verarbeiten kann, muß es sich dieses Wissen zunächst einmal aneignen bzw. sukzessive im Laufe der Zeit erwerben. Dieser Vorgang wird im allgemeinen als Wissensakquisition bezeichnet und man kann drei grundsätzlichen Vorgehensweisen (die manuelle, die überwachte automatische und die nicht-überwachte automatische Wissensakquisition) unterscheiden.

In dem oben verwendeten Zusammenhang bezeichnet man die Vorgehensweise bei der automatischen Wissensakquisition auch als Maschinelles Lernen. Vereinfacht ausgedrückt besteht beim Maschinellen Lernen die Aufgabe darin, durch (induktive) Verallgemeinerung statische Konzepte (z. B. Tisch, Stuhl, Kaffeetasse etc.) zu lernen. Das Lernen dynamischer Konzepte bzw. das integrierte Lernen statischer und dynamischer Konzepte war in den bisherigen Untersuchungen untergeordnet.

Für den Begriff Lernen existieren in der Literatur (insbesondere in der Lern- und Verhaltenspsychologie) zahlreiche verschiedene Definitionen. Die wohl treffendste Definition im Zusammenhang mit Maschinellem Lernen stammt von Herbert A. Simon /Simon 114/:

> Mit **Lernen** bezeichnet man adaptive Veränderungen der Fähigkeiten eines Systems, um die gleiche oder ähnliche Aufgaben, die aus der gleichen Population hervorgehen, beim nächsten Mal effizienter und effektiver behandeln zu können (Verbesserung der Performanz).

Lernen basiert dabei in der Regel immer auf kausalen, deterministischen bzw. mindestens statistisch signifikant in Erscheinung tretende und isoliert beobachtbare Sachverhalte. Das zu lernende Wissen kann dabei kategorischer, episodischer, kausaler oder funktionaler Natur sein.

Die existierenden Verfahren und Ansätze der *Künstlichen Intelligenz* insgesamt können in die Bereiche der klassischen symbolorientierten Künstlichen Intelligenz, der evolutionären Algorithmen sowie des subsymbolischen Konnektionismus eingeordnet werden. Maschinelle Lernverfahren existieren aus allen drei Bereichen.

2.1 Inferentielle Methoden des Wissenserwerbs, der Wissensverarbeitung

Die in diesem Abschnitt beschriebenen Inferenzmethoden dienen in erster Linie zum Erwerb neuen Wissens, z. B. durch eine induktive Generierung eines allgemeinen Konzeptes, dessen Beschreibung die dem lernenden System dargebotenen Beispiele subsummiert oder zur Verarbeitung des bereits enthaltenen Wissens, z. B. indem aus gegebenen Fakten und zugehörigen Relationen weitere, nicht explizit gespeicherte Fakten deduktiv abgeleitet werden. Im wesentlichen handelt es sich bei den in diesem Abschnitt dargestellten Methoden um die inferentiellen "Grundwerkzeuge" lernender Systeme. Den folgenden Betrachtungen wird ein elementares Modell eines beliebigen Schlußfolgerungsprozesses zugrundegelegt /Berns, Kreuziger 6/ und /Kodratoff, Michalski 82/:

Aus der Prämisse (P) wird mit Hilfe von Hintergrundwissen (HGW) die Aussage (K) als Konsequenz logisch gefolgert, d. h.:

$$P \wedge HGW \Rightarrow K$$

Beispiel:	Prämisse (P):	Es regnet.
	Hintergrundwissen (HGW):	Es ist sehr kalt (Winter).
	Konsequenz (K):	Es bildet sich Glatteis.

2.1.1 Deduktive Inferenz

Unter Deduktion versteht man die Ableitung von Konsequenzen aus gegebenen Voraussetzungen (vgl. auch /Heinemann, Weihrauch 39/).

Deduktive Inferenz liegt in diesem elementaren Modell dann vor, wenn aus der Beobachtung der Prämisse P - unter Berücksichtigung des Hintergrundwissens HGW - auf die Gültigkeit der Konsequenz K geschlossen wird.

Es handelt sich hierbei demnach um ein Vorwärtsverfolgen der Relation "$\Rightarrow$".

Die wesentliche Eigenschaft der Deduktion ist, daß sie **wahrheitserhaltend** ist, d. h. falls das Hintergrundwissen und die gegebenen Regeln korrekt sind, so sind es auch die abgeleiteten Konsequenzen.

Beispiel: Eines der klassischen Beispiele für einen deduktiven Schluß ist:

P:	Sokrates ist ein Mensch.
HGW:	Alle Menschen sind sterblich.
K:	Folglich ist Sokrates sterblich.

Im Falle, daß die Prämissen einer deduktiven Inferenz nicht wahr, sondern nur (in welchem Sinne auch immer) "wahrscheinlich" sind, so muß die Konklusion natürlich auch nicht stimmen. In diesem Fall spricht man von einem **approximierenden deduktiven Schluß.**

Bei der Charakterisierung maschinell lernender Programme werden deduktive Inferenzen bzgl. der Abhängigkeit vom jeweiligen Kontext subklassifziert. Man unterscheidet hierbei zwischen einer **bedingten Deduktion**, diese ist nur in einem bestimmten Kontext gültig, und der **universellen Deduktion**, einem universellen, d. h. kontext-unabhängigen deduktiven Schluß.

2.1.2 Induktive Inferenz

Unter Vernachlässigung von Hintergrundwissen versteht man unter Induktion das Gegenteil zur Deduktion, d. h. aus gegebenen (beobachteten) Konsequenzen werden diese Konsequenzen verursachenden Voraussetzungen hypothetisiert.

Induktive Schlüsse bilden ein wesentliches Instrument zur Bildung von abstrahierenden Generalisierungen, um eine aktuelle Menge von Beobachtungen zu beschreiben. Es handelt sich hierbei um das am häufigsten verwendete Paradigma für wissenschaftliche Schlußfolgerungen.

Für obiges Modell bedeutet dies: Aufgrund der beobachteten Folgerung K wird mit Hilfe des Hintergrundwissens HGW auf die Prämisse P geschlossen. P kann als Erklärung der Beobachtung K gedeutet werden, da das Faktum K mit Hilfe der oben genannten Regel aus der Prämisse P und dem Hintergrundwissen HGW logisch gefolgert werden kann. Induktion bedeutet demnach um ein Rückwärtsverfolgen der Relation "$\Rightarrow$".

Induktive Schlußfolgerungen sind dadurch gekennzeichnet, daß aus Beobachtungen, evtl. unter Verwendung von Hintergrundwissen, Hypothesen aufgestellt werden. Aus dieser Tatsache ergibt sich unmittelbar, daß Induktion **falschheitsbewahrend** ist; d. h. aus im Sinne der Relation "$\Rightarrow$" nicht gültigen oder falsch interpretierten Beobachtungen werden falsche Prämissen gefolgert. Mit anderen Worten, falls eine Aussage falsch ist, so ist auch die daraus induktiv gefolgerte Aussage falsch. Aus diesem Grund müssen induktive Schlüsse verifiziert werden. Eine solche Verifikation kann beispielsweise empirisch erfolgen.

Beispiel: Ein induktiver Schluß kann wie folgt aussehen (/Bench-Capon 4/):

K: Diese beiden Schwäne sind weiß (Beobachtung).
HGW: Jeden Schwan, der bisher angetroffen wurde, war weiß.
P: Folglich sind alle Schwäne weiß.

(Diese Aussage ist nicht richtig und müßte z. B. nach der Entdeckung Australiens zumindest in "Alle Schwäne der nördlichen Hemisphäre sind weiß" abgeschwächt werden.)

In Abhängigkeit von der Art des Hintergrundwissens (reines Domainwissen und allgemeines Basiswissen) kann man Induktion wie folgt subklassifizieren /Kodratoff, Michalski 58/:

- **Empirische Induktion,**
  die Hypothesen basieren auf gegebenen Fakten, in erster Linie unter Verwendung von Domain-unabhängigem (d. h. Basis-) Wissen (Tautologien oder gültige logische Schlüsse).

  Beispiel: HGW: Wenn eine Aussage für alle Elemente einer Menge gültig ist, so gilt sie ebenso für jedes einzelne Element (Tautologie).
  K: Alle Menschen sind sterblich, Sokrates ist ein Mensch.
  P: Folglich ist Sokrates sterblich.

- **Abduktion,**
  die Hypothesen werden generiert, indem man (Domain-abhängige) Regeln, die Beziehungen zwischen Fakten über den Domain ausdrücken zurückverfolgt; d. h. Abduktion ist die "Umkehrung" des Modus Ponens (Fallunterscheidung rückwärts).

  Beispiel: HGW: Wenn es regnet sind die Straßen naß.
  K: Die Straßen sind naß.
  P: Folglich regnet es.

  Abduktion wird hauptsächlich eingesetzt, um einen gerade gescheiterten (deduktiven) Beweis zu beenden, indem Hypothesen generiert werden, die im Falle ihrer Gültigkeit tatsächlich zu einem vollständigen Beweis führen. Selbst in stark eingeschränkten Bereichen sind in der Regel mehrere abduktive Schlüsse möglich, die alle das Problem des gescheiterten Beweises lösen. Die Auswahl der besten Hypothese hängt daher vom vorhandenen Hintergrundwissen ab. Aufgrund dieser Tatsache kann Abduktion sehr wissens-intensiv werden.

- **Konstruktive Induktion** (wissensbasierte Induktion),
  zur Generierung von Hypothesen wird sowohl Domain-abhängiges, als auch Domain-unabhängiges Wissen verwendet.

  Beispiel: HGW: Ein hoher Grad an Automatisierung in einer Firma führt zu einer hohen Zuverlässigkeit bei deren Produkten.
  P: Eine spezielle Firma verfügt über eine vollständig automatisierte Automobilproduktion. Das Fahrzeug von Herrn X stammt von dieser Firma und ist sehr zuverlässig.
  K: Folglich sind alle Fahrzeuge dieser Firma sehr zuverlässig.

Bei der Induktion geht der Informationsgehalt der Konklusion über den der Prämissen hinaus. Daher wird das Wissen in seinem Umfang echt erweitert.

2.1.3 Analogie

Analoges Schlußfolgern kann als Kombination von Induktion und Deduktion aufgefaßt werden und steht somit zwischen diesen beiden Schlußfolgerungsmechanismen.

Analoges Schlußfolgern beinhaltet die Übertragung bestimmter Eigenschaften des Hintergrundwissens auf das gegebene Problem; hierzu sind sowohl induktive als auch deduktive Schlüsse notwendig. Dies soll durch das folgende allgemeine Schema (Abbildung 1) verdeutlicht werden (/Kodratoff 56/):

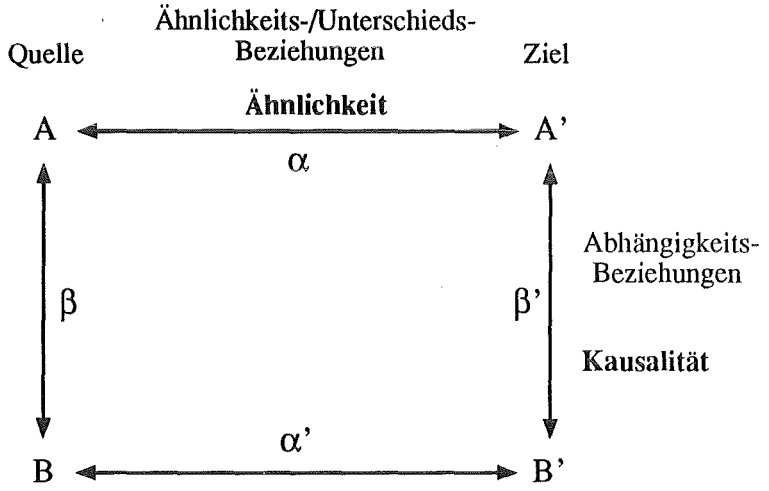

Abbildung 1

Um eine Analogie überhaupt zu erkennen, müssen Ähnlichkeiten oder gemeinsame Relationen zwischen dem Hintergrundwissen und dem gegebenen Problem erkannt und darüber hinausgehende hypothetisiert werden. Hierbei handelt es sich im wesentlichen um einen induktiven Vorgang. Wurden diese gemeinsamen Relationen jedoch erst einmal bestimmt, so erfordert die Übertragung der Eigenschaften vom Hintergrundwissen auf das momentane Problem deduktive Inferenzen.

Beispiel: Zur Beantwortung der Frage: "Können Rotkehlchen fliegen?" kann ein Analogieschluß wie folgt aussehen (/Barr, Feigenbaum 2/):

P & HGW:	Rotkehlchen sind wie Spatzen (induktive Vorgehensweise).
HGW:	Spatzen können fliegen.
K:	Folglich können Rotkehlchen fliegen (deduktive Vorgehensweise).

2.1.4 Interpolation (als Konstruktion von Lösungspfaden)

Wenn bei einem gegebenen Problem sowohl der Ausgangszustand, der Endzustand als auch einzelne Operatoren zur Transformation des Anfangs- in den Endzustand bekannt sind; unbekannt ist lediglich die Reihenfolge, in der diese Operatoren zur Anwendung gebracht werden müssen, so spricht man von einem (Planungs-) Interpolationsproblem.

Die Aufgabe besteht dann in der Auswahl und Anordnung der benötigten Operatoren, aus dem vorgegebenen Repertoire und in der Kombination derselben, so daß ihre Anwendung auf den Ausgangszustand dessen Überführung in den gewünschten Endzustand bewältigt.

Betrachtet man den nicht-notwendigerweise zusammenhängenden Graphen, der als Knoten sämtliche Zustände enthält, die durch Anwendung der gegbenen Operatoren auf den Ausgangszustand bzw. auf die hieraus resultierenden Folgezustände entstehen und dessen Kanten den jeweils angewendeten Operatoren entsprechen, so handelt es sich bei dem oben geschilderten Problem um eine andere Formulierung der Frage, ob und wie dieser Endzustand vom Ausgangszustand ausgehend erreichbar ist. Damit wurde dieses Problem auf ein aus der Graphentheorie wohlbekanntes Erreichbarkeitsproblem reduziert.

2.2 Methoden zur Wissensakquisition

Ein wesentlicher Punkt in einem intelligent agierenden System betrifft die Wissensakquisition, d. h. die Vorgehensweise zum Erwerb von Wissen. Um eine Abgrenzung zum Erlernen rein motorischer oder intuitiver Fähigkeiten, wie z. B. Fahrrad fahren, Klavier spielen etc., durch wiederholte Übung zu erreichen, versteht man unter **Wissensakquisition** das Lernen neuer symbolischer Information in Verbindung mit der Fähigkeit diese Information in effektiver Weise anzuwenden /Carbonell et al. 15/.

Bemerkenswert ist es, daß in dieser Definition der Validierungsaspekt völlig vernachlässigt wird. Geht man davon aus, daß dem System nicht nur "handverlesenes" Wissen, das keiner weiteren Überprüfung mehr bedarf, zur Verfügung gestellt wird, so würde eine solche Vorgehensweise mit der Zeit zu einer inkonsistenten Wissensbasis führen.

Unklar ist ebenfalls, ob diese Definition die Akquisition von Wissen mit neuronalen Netzen einschließt oder ob sie sich "nur" auf Verfahren zur Akquisition von Wissen in symbolischer Form bezieht und das in neuronalen Netzen enthaltene Wissen in der Regel nicht in eine symbolische Darstellung überführt werden kann.

Ein häufig nicht beachteter Aspekt betrifft die Unterscheidung von effektivem (konstruktivem) und empirisch abgesichertem Wissen. Es handelt sich beispielsweise um effektives (konstruktives) Wissen, wenn man einmal einen Tisch gesehen hat, denn man weiß dann, daß ein Tisch aus einer Platte, getragen von einer Stütze (Beine etc.), besteht. Bei empirisch abgesichertem Wissen handelt es sich beispielsweise um Wissen über Kausalbeziehungen aufgrund früherer Erfahrungen.

Grundsätzlich kann man zwei verschiedene Ansätze, sowie mögliche Kombination, zur Wissensakquisition unterscheiden /Niemann 91/:

- Den **manuellen Ansatz**, hierbei ist es Aufgabe des (System-) Designers, das notwendige Wissen zu implementieren und
- den automatischen Ansatz, wobei es Aufgabe der Maschine (des Rechners) ist, das notwendige Wissen zu erwerben. Diese Vorgehensweise wird im folgenden als **Maschinelles Lernen** bezeichnet.

2.2.1 Manuelle Wissensakquisition

Betrachtet man den manuellen Ansatz, so handelt es sich hierbei um die mühevolle Aufgabe das vorgegebene Problem zu analysieren, um es dann in einem erforderlichen und angemessenen Detailierungsgrad, zumeist in schriftlicher Form, festzuhalten.

Im wesentlichen stellt diese Vorgehensweise den Stand der Dinge bei der Entwicklung von Expertensystemen dar (vgl. z. B. /Puppe 92/). In der Regel handelt es sich jedoch bei den entsprechenden Domain-Experten und den mit der Entwicklung von Expertensystemen beauftragten (Informatik-) Experten (den sogenannten **Wissensingenieur**) nicht um ein und

dieselbe Personen. Dies wirkt sich zumeist nachteilig auf den Wissensakquisitionsvorgang aus, da der Wissensingenieur zumeist kein Experte in dem entsprechenden Fachbereich und der Domain-Experte kein Informatik-Experte ist.

Die Effektivität der vom Wissensingenieur eingesetzten Techniken zum Erwerb des (Domain-) Expertenwissens (vgl. z. B. /Puppe 92/) hängt jedoch nicht nur von der Fähigkeit und Bereitwilligkeit des Domain-Experten ab, sein Wissen weiterzugeben, sondern scheitert häufig bereits daran, daß sich gewisse Wissensformen, wie z. B. Handlungswissen, weitgehend der Verbalisierung entziehen /Laske 66/. Die Schwierigkeiten bei der Explizierung von Expertenwissen wurden beispielsweise durch empirische Untersuchungen (z. B. das Kühlschrankexperiment in /Reichert, Dörner 98/ und Lohhausen in /Dörner et al. 24/) bestätigt. Die wesentlichen Erkenntnisse derartiger Untersuchungen kann man wie folgt zusammenfassen:

- Expertenwissen ist **nicht standardisiert**,
- Expertenwissen ist häufig **unbewußt** und daher nicht explizierbar,
- Expertenwissen kann auch **falsch** sein.
  Insbesondere bei Handlungswissen kann der Experte zwar in der Lage sein einen Prozeß zu führen, obwohl sein Handlungswissen von z. B. kausal falschen Zusammenhängen im Prozeß ausgeht.

Aus diesen Gründen wird die Wissensakquisition auch als der **Flaschenhals** bei der Entwicklung von modernen wissens-intensiven Systemen der Künstlichen Intelligenz bezeichnet (/Michalski et al. 77/).

Die Probleme beim Umgang mit Expertenwissen kann man darauf zurückführen, daß es ein **heuristisches Modell** des betrachteten Systems darstellt. Dieses heuristische Modell besteht aus einem defizitären (verinnerlichten) **mentalen Modell** des Systems, das in einen allgemeinen **heuristischen Kontext** eingebettet ist. Das mentale Modell weist in der Regel die folgenden Mängel auf:

- Es ist **unvollständig**,
- es ist **instabil**, da häufig Details vergessen werden,
- es ist **mehrdeutig**, da ähnliche Abläufe oder Anordnungen häufig verwechselt werden.

Für mentale Modelle in nicht vollständig abgesicherten Domänen, ist der Kontext der Wissensentstehung relevant.

2.2.2 Automatische Wissensakquisition

Eine Lösung der dargestellten Problematik der manuellen Wissensakquisition bildet das **Maschinelle Lernen**. Es handelt sich hierbei um einen Ansatz, bei dem die Maschine (der Rechner), versehen mit einem gewissen Grundrepertoire an Wissen, mit der Fähigkeit ausgestattet wird, das zur Erfüllung ihrer Aufgaben notwendige Wissen selbständig zu erwerben, zu verarbeiten und für eine effektive Wiederverwendung zu speichern.

Die meisten existierenden lernfähigen Systeme sind jedoch nicht praxixreif, sondern Gegenstand der Grundlagenforschung der Künstlichen Intelligenz. Welche enormen Schwierigkeiten die Entwicklung selbstlernender (Experten-) Systeme bereitet, zeigen die folgenden Überlegungen /Puppe 92/:

- Ohne Vorwissen ist kein Lernen möglich ("von nichts kommt nichts"). Für die meisten (Experten-) Systeme verfügt man jedoch nur über geringes a priori Wissen in ihren Anwendungsbereich.

- Der Mensch benötigt etwa fünf bis zehn Jahre intensiver Beschäftigung mit einem Anwendungsbereich, bevor er ein Experte auf diesem Gebiet ist. Obwohl das nicht notwendigerweise auch für Programme gelten muß, gibt es doch einen Hinweis auf die möglicherweise notwendigen, komplexen Rückkopplungsschleifen zwischen einem lernenden System und seiner Umgebung.

Die meisten derzeitigen (symbolischen) Lernsysteme operieren in hochgradig vorstrukturierten Spielzeugwelten, realistische Anwendungen sind nur wenige bekannt /Puppe 92/.

Betrachtet man die Entwicklung des Menschen aus Sicht der Evolution, so stellt man fest, daß ein immenser Zeitraum vonnöten war, bis sich der Mensch aus einem Einzeller entwickeln konnte. Von diesem Punkt war es jedoch nur noch ein vergleichsweise geringer Schritt bis zu ersten menschlichen Experten. Stellt man diesem Faktum die Vorgehensweise bei der Entwicklung maschinell lernender Systeme gegenüber, so ergibt sich die Frage, ob diese Ansätze nicht auf einer zu hohen Ebene angesiedelt sind. Anders formuliert bedeutet dies: Ist es überhaupt möglich, mit den heutigen Mitteln und Methoden die Funktionsweise des Gehirns zu erklären, ohne daß man versteht, wie es sich evolutionär entwickelt hat?

Beispiel:
Wie hilfreich wäre es für die Gebrüder Wright gewesen, hätte man ihnen zur Entwicklung ihres ersten Flugzeuges die Konstruktionspläne einer Boing 747 gegeben?

3 Maschinelles Lernen

Das Gebiet des Maschinellen Lernens beschäftigt sich mit rechnerbasierten Methoden zum Erwerb neuen Wissens, neuer Fähigkeiten und neuartiger Wege der Organisation des bestehenden Wissens /Carbonell, Langley 14/. Unter dem Begriff Maschinelles Lernen werden sowohl symbolorientierte, als auch konnektionistische Verfahren verstanden.

In diesem Kapitel werden nur die klassischen, symbolorientierten Lernverfahren betrachtet, evolutionäre Algorithmen und konnektionistische Verfahren werden jeweils in einem eigenen Kapitel dargestellt.

3.1 Grundsätzlicher Aufbau eines lernenden Systems

Um die prinzipielle Vorgehensweise lernender Systeme zu verdeutlichen, wird im folgenden ein Strukturmodell eines lernenden Systems vorgestellt. Die Benennung der Komponenten des lernenden Systems erfolgt in Anlehnung an /Dillmann 22/.

Die Aufgabe dieses lernenden Systems besteht darin, daß das System die ihm gestellten Aufgaben (globale oder konkrete Zielvorgabe) schrittweise nach Wiederholungen besser ausführen kann als zuvor. Die gewünschte Verbesserung der Performanz des Systems wird dadurch erreicht, daß es neue oder modifizierte Methoden und Wissen anwendet, um seine Aufgaben mit einer verbesserten Qualität (z. B. schneller, genauer, robuster, sicherer) auszuführen.

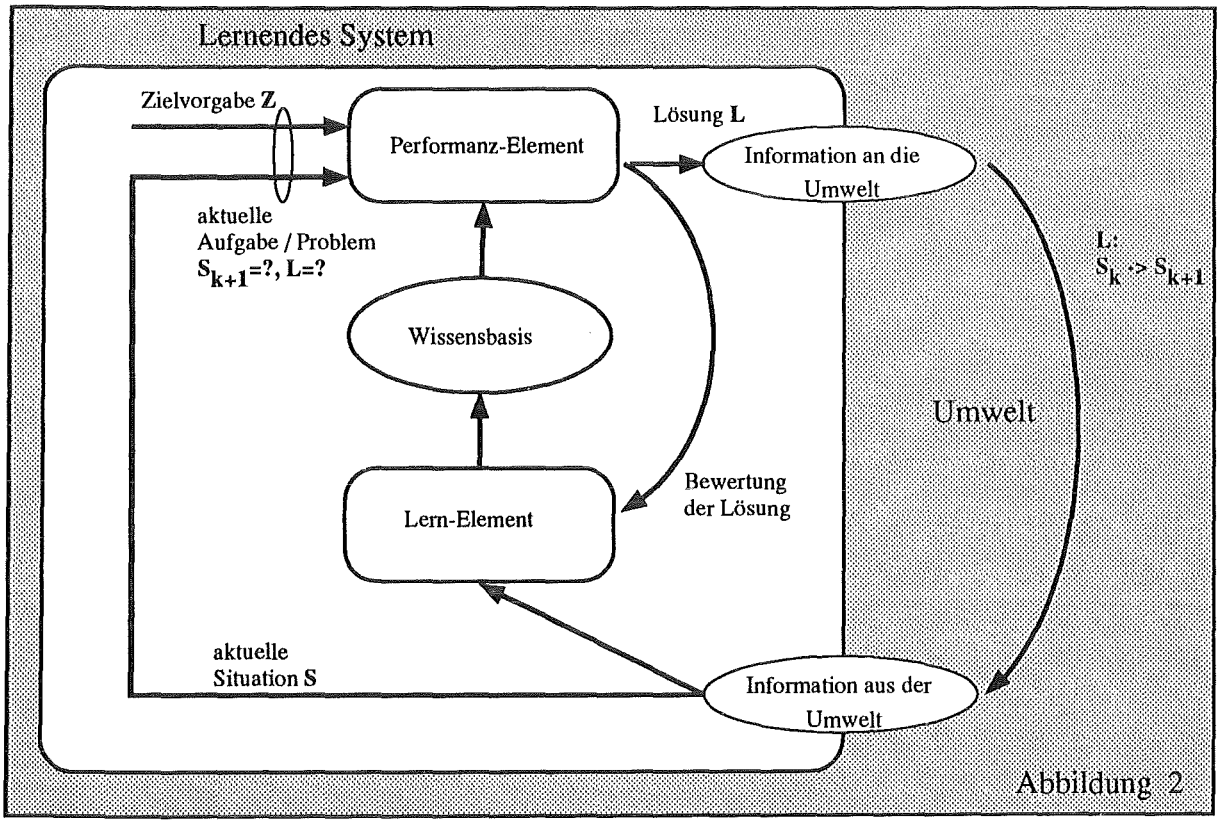

Abbildung 2

Die grundsätzliche Aufgabe des Systems besteht darin, eine ihm gestellte Aufgabe anhand vorgegebener Kriterien möglichst optimal zu lösen. In Abbildung 2 werden prozedurale Blöcke (z. B. für Verarbeitungsschritte) durch Rechtecke und deklarative Informationseinheiten (z. B. für Fakten) durch Ellipsen dargestellt. Wie üblich wird die Richtung des Informationsflusses durch Pfeile dargestellt.

Zum besseren Verständnis werden im folgenden die einzelnen Komponenten dieses allgemeinen Lernmodelles beschrieben und ihr Zusammenspiel erläutert:

- Das gesamte System ist in natürlicher Weise in die **Umwelt** eingebettet. Aus ihr erhält es (wie auch immer ausgewählte) Informationen und damit **aktuelle Situationen.** Anhand der internen **Zielvorgabe Z** ergeben sich **Aufgaben** (L=?: S_k -> S_{k+1}) bzw. **Probleme** (S_{k+1} als Funktion von Z, WB und S_k). Die interne Zielvorgabe kann eine konkrete Aufgabenstellung oder aber ein Optimierungsproblem sein. Über das **Performanz-Element** wird eine Ausgabe (die Lösung L), in Abhängigkeit von den vorgegebenen Informationen, an die Umwelt zurückgeliefert.

- Im **Performanz-Element** wird die Aufgabe, das Problem, unter Verwendung des Wissens der Wissensbasis gelöst, das Ergebnis wird an die Umwelt und an das Lern-Element zur Weiterverarbeitung weitergeleitet.

- Die Ausgabe des **Performanz-Elementes** wird im Hinblick auf die gestellte Aufgabe, das Problem, bewertet und dem **Lern-Element** zur Verfügung gestellt. Das Lern-Element verarbeitet und verwaltet dieses Wissen in Verbindung mit der aus der Umwelt gegebene Information.

- Die Lösung der **Aufgabe** bzw. des **Problems** stellt das eigentliche Lernziel dar. Aus der Sicht der Problemlösung sind die Aufgabe bzw. das Problem den Problemtypen Interpolation oder Synthese (vgl. /Dörner 23/) zuzuordnen, da bei diesen beiden Problemtypen, im Gegensatz zu dialektischen Problemen, ein konkretes Lernziel (auch Optimierung) und damit zielorientiertes Lernen stattfinden kann.

- Die **Wissensbasis** dient zur Speicherung und Verwaltung des vorgegebenen bzw. bereits gelernten Wissens. Auf ihrem Inhalt, d. h. dem Wissen und der aufgrund einer ökonomischen Verwaltung notwendigen Struktur der Archivierung, agiert das Performanz-Element im Sinne eines **problemlösenden Elementes**.

Entscheidenden Einfluß auf die Lernstrategie hat das Verhältnis der dem Lern-Element aus der Umwelt zur Verfügung gestellten Information zu den angeforderten Informationen des Performanz-Elementes. Insbesondere ist hierbei der Abstraktionsgrad der zugeführten Information von Bedeutung. Das Lern-Element muß die ihm zugeführten Informationen an die benötigten Informationen des Performanz-Elementes anpassen.

Da es dem Lern-Element a priori nicht bekannt ist, ob und wie es fehlendes Detailwissen in die Wissensbasis einfügt oder unbedeutende Details ignoriert, muß es versuchen, durch Bildung von Hypothesen die Wissenslücke zwischen dem Performanz-Element und der Wissensbasis zu schließen. Das Lern-Element erhält eine verstärkende oder abschwächende Rückmeldung bzw. eine Bewertung der Hypothese durch das Performanz-Element und kann diese, wenn nötig, modifizieren.

Die Abstraktionsebene und Qualität der Information, die dem Lern-Element zugeführt wird, legt die Art der Hypothesen fest, die das System generieren muß. Die Information kann aus Beispielen in Form einer symbolischen Beschreibung und deren Klassenzugehörigkeit, ihrer Bewertung oder ihrer Lösung bestehen. Diese Eingaben werden mit dem bereits Gelernten in Verbindung gebracht.

In ähnlicher Weise, wie oben dargestellt, wird ein allgemeines Lernmodell in /Cohen, Feigenbaum 17/ beschrieben.

Über die dargestellten Beziehungen hinausgehend, könnte ein lernendes System auch die im Performance Element, sowie die im Lernelement eingesetzten Verfahren und die interne Zielvorgabe in der Wissensbasis halten und in den Bewertungszyklus mit einbeziehen, bzw. anhand der sich ergebenden Umweltsituation modifizieren. Eine solche kognitionspsychologische Vorgehensweise ist in den existierenden Verfahren allerdings in keiner Weise erkennbar.

3.2 Lernstrategien im Überblick

Die im folgenden Abschnitt ausführlich beschiebenen Lernstrategien können zusammenfassend, ähnlich wie bei /Kodratoff 57/, wie folgt dargestellt werden:

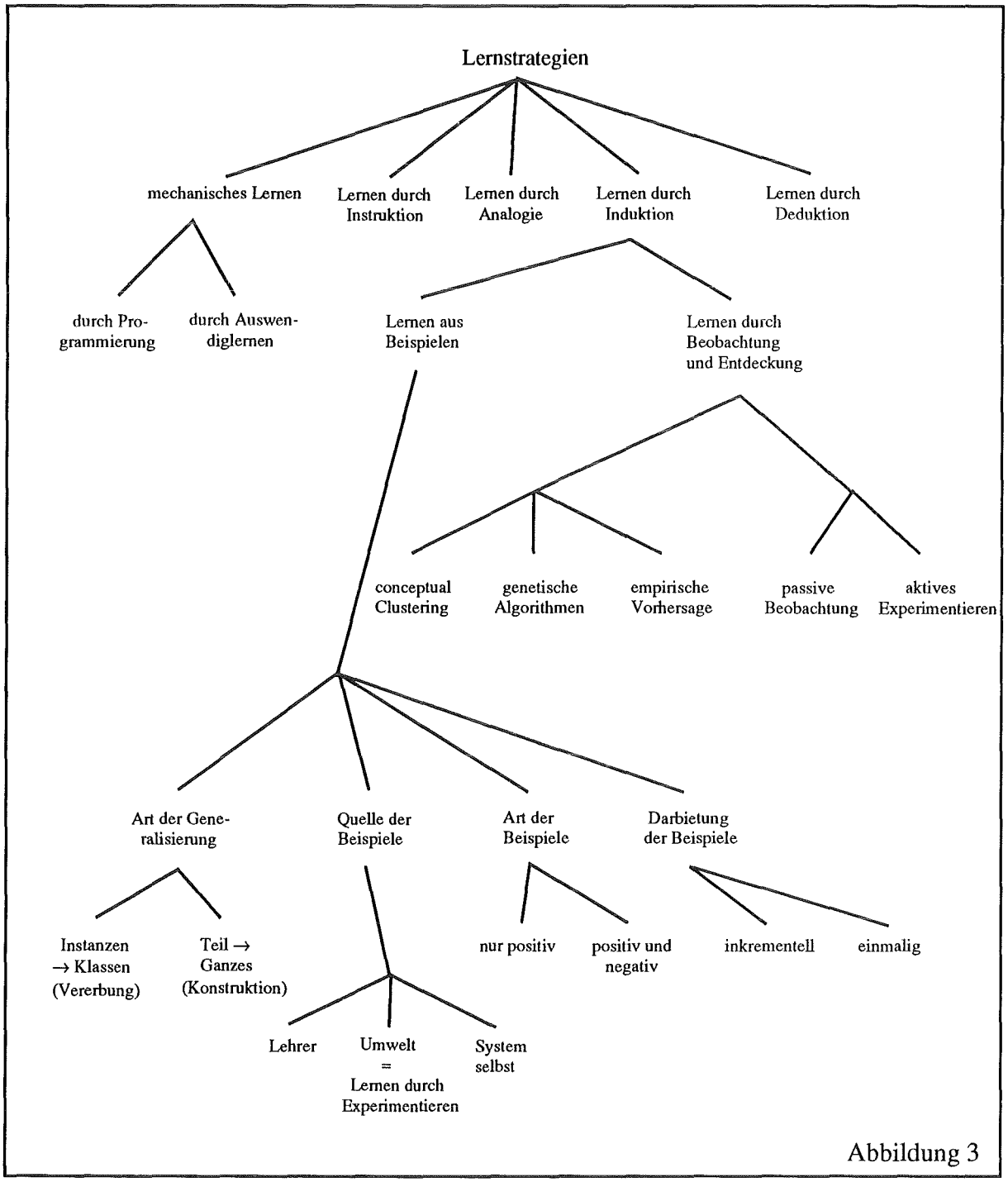

Abbildung 3

3.3 Klassifikation maschinell lernender Systeme aufgrund der Lernstrategie

Ein wesentlicher Gesichtspunkt zur Klassifikation maschinell lernender Systeme besteht in der Art der Schlußfolgerungen, die überwiegend verwendet werden. Faßt man ein solches System als ein Schlußfolgerungssystem auf, so kann man die verwendete Lernstrategie identifizieren (/Michalski et al. 80/). Eine Unterscheidung der Lernstrategien nach der Anzahl der Schlußfolgerungen, die beim Lernenden durch die gegebene Information ausgelöst werden, führt zunächst auf die beiden Extreme.

- Es werden keine Schlußfolgerungen durchgeführt, d. h. alle kognitiven Anstrengungen liegen beim Lehrer, z. B. dem Programmierer.
- Im Gegensatz hierzu steht ein System, das selbständig neue Theorien entdeckt oder neue Konzepte entwirft. Ein solches System muß eine erhebliche Anzahl von Schlußfolgerungen durchführen, um aus Experimenten und Beobachtungen strukturiertes Wissen zu erhalten.

Die im folgenden dargestellte Einordnung von Lernstrategien ist zum Vergleich verschiedener lernender Systeme im Bezug auf

- unterlagernde Mechanismen, d. h. den Grad der vom Lernenden geforderten Inferenz,
- verfügbare externe Informationsquellen,
- den Grad des bereits vorstrukturierten Wissens, auf das sie sich beziehen,

hilfreich.

Aus der Sicht des Problemlösens beschreibt die Lernstrategie die Vorgehensweise, wie der gewünschte Zielzustand, ausgehend von einem gegebenen Ausgangszustand, erreicht werden kann. Betrachtet man das Problemlösen aus der Sicht der Lernpsychologie, so ist nach Dörner ein Problem durch

- einen unerwünschten **Anfangszustand**,
- einen erwünschten **Endzustand** und
- eine **Barriere**, die die Transformation des Anfangs- in den Endzustand momentan verhindert

gekennzeichnet /Dörner 23/. Eine Klassifikation verschiedener Barrieretypen liefert die folgenden **Problemtypen** /Dörner 23/:

- Interpolationsprobleme:

  Dies sind Probleme, bei denen es darum geht, die richtige Kombination oder Folge aus einer Reihe bekannter Operationen zu bilden, um eine Interpolationsbarriere zu überwinden, die die Interpolation zwischen Anfangs- und Zielzustand verhindert.

- Syntheseprobleme:

  Der Anfangszustand ist bekannt, der Endzustand ist bekannt, unbekannt ist aber nicht nur die spezifische Kombination von Operationen, während die Einzeloperationen bekannt sind, wie dies bei Interpolationsproblemen der Fall ist; vielmehr hat man Grund zur Annahme, daß auch wichtige Einzeloperationen unbekannt sind oder nicht in Betracht gezogen wurden. In diesem Fall kommt es darauf an, die richtigen Operationen zu finden, nicht nur - wie bei Interpolationsproblemen - die bekannten zu kombinieren. Hauptaufgabe ist hier die Zusammenstellung oder Synthese eines brauchbaren Inventars von Operationen. Syntheseprobleme kann man daher auch als Probleme mit offenem Operatorinventar bezeichnen.

- Dialektische Probleme:

  Bei den bisherigen Problemtypen ist der gewünschte Zielzustand bekannt. Die größte Zahl von Problemen scheint jedoch dadurch gekennzeichnet, daß allenfalls bestimmte Kriterien für den Zielzustand bekannt sind; oft aber nicht einmal solche formuliert werden können. Man weiß lediglich, daß die gegebene Situation verändert werden muß, ohne mehr als relativ globale Kriterien dafür zu haben, wie.

  Der Grund für die Wahl des Begriffes "dialektisch" ist, daß die Lösung solcher Probleme meist in einem dialektischen Prozeß gefunden wird, in dem ein Vorschlag oder Entwurf für den Zielzustand auf äußere Widersprüche (Widersprüche des Entwurfs mit Sachverhalten außerhalb seiner selbst) oder inneren Widersprüchen (Widersprüche der Komponenten des Entwurfs zueinander) überprüft und entsprechend verändert wird.

3.3.1 Mechanisches Lernen und direkte Implantation neuen Wissens

Es werden vom (lernenden) System weder Schlußfolgerungen noch Transformationen des Wissens verlangt; das Wissen muß lediglich abgespeichert werden. Ein solches lernendes System beinhaltet die folgenden Arten der Wissensakquisition:

(1) Lernen, indem das System ohne eigene Anstrengung programmiert, konstruiert oder modifiziert wird.

(2) Lernen durch "pures" Auswendiglernen, ohne daß Schlußfolgerungen gezogen werden.

Hierbei findet keinerlei Inferenz oder irgendwie anders geartete Wissenstransformation seitens des Lernenden statt. Im Extremfall artet der Lehrer zu einer passiven Informationsquelle aus /Richter, Wendel 100/.

3.3.2 Lernen durch Instruktion (Unterweisung)

Der Wissenserwerb erfolgt mit Hilfe eines Lehrers oder einer anderen "strukturierten" Quelle. Das System transformiert das Wissen aus der Eingabedarstellung in seine intern verwendbare Darstellung und die neuen Informationen werden, um sie effektiv einsetzen zu können, in bereits vorhandenes Wissen integriert. Fehlende Detailinformationen werden entweder durch Hypothesenbildung gefunden oder von einer externen Wissensquelle, z. B. dem Lehrer, angefordert /Dillmann 22/.

Daraus erkennt man bereits, daß das System Schlußfolgerungen durchführen muß; jedoch liegt ein großer Teil der Anforderungen beim Lehrer, der das strukturierte Wissen in einer Art und Weise präsentieren muß, daß es in die Begriffswelt des Systems eingeordnet werden kann und dessen Wissen vergrößert.

Lernen durch Instruktion findet man in den meisten formalen Ausbildungsmethoden. Daher besteht die maschinelle Lernaufgabe darin, ein System zu generieren, das Instruktionen und Ratschläge aufnehmen und erlerntes Wissen effektiv anwenden kann.

3.3.3 Lernen durch Analogie

Neue Fakten oder Fähigkeiten erhält man durch Transformation und Erweiterung bestehenden Wissens. Dafür sind starke Ähnlichkeiten des bestehenden Wissens zum gewünschten neuen Konzept oder der neuen Fähigkeit erforderlich, um deren effektive Nutzung in der neuen Situation zu gewährleisten.

Lernen durch Analogie bedeutet folglich, daß für das neue Problem keine komplett neue Lösung gefunden werden muß, sondern es genügt, eine bestehende Lösung den Anforderungen entsprechend zu modifizieren.

Um entscheiden zu können, wie ähnlich sich bestehendes Wissen und gewünschte Konzepte oder Fähigkeiten sind, muß das lernende System über ein Ähnlichkeitsmaß (Metrik) auf konzeptueller Ebene verfügen.

Beispielsweise könnte ein durch Analogien lernendes System dazu verwendet werden, ein bestehendes Computerprogramm in eines zu übertragen, das eine eng verwandte, jedoch zuvor nicht enthaltene, Tätigkeit ausführt.

Lernen durch Analogien erfordert umfassendere Schlußfolgerungsfähigkeiten beim System als mechanisches Lernen oder Lernen durch Instruktion. Ein analoges Faktum oder eine analoge Fähigkeit mit relevanten Parametern muß aus dem Speicher geholt werden, dann muß das geholte Wissen transformiert, auf die neue Situation angewandt und zur weiteren Verwendung gespeichert werden.

Wird zu einem gewünschten Konzept oder einer Fähigkeit kein analoges Wissen im Speicher gefunden, so werden andere (heuristische) Lösungsmethoden, wie z. B. Trial-and-Error, angewandt.

Um die Leistungsfähigkeit der Schlußfolgerungen und des Ähnlichkeitsmaßes bei sich änderndem Wissensstand zu gewährleisten, muß eine dynamische Veränderung (Anpassung) möglich sein.

Lernen durch Analogie kann man in gewisser Hinsicht als Unterklasse des Lernens aus Beispielen auffassen, bei dem jeweils von einem Konzept (einem bereits verarbeiteten Beispiel) für ein gegebenes Konzept (einem gegebenen Beispiel) "flach", d. h. ohne sich Gedanken über einen gemeinsamen Hintergrund zu machen, gelernt wird.

3.3.4 Induktives Lernen (empirisches Lernen)

Die Verfahren zum induktiven Lernen können aufgrund der ihnen zugeordneten Form der Wissensrepräsentation in zwei grundlegende theoretische Ansätze eingeteilt werden. Bei den **symbolischen Verfahren** wird das gesamte Wissen in symbolischer Form gespeichert und verarbeitet. Im Gegensatz hierzu wird das Wissen bei den **sub-symbolischen Verfahren** (neuronale Netze) subsymbolisch, d. h. verteilt und auf einer Ebene unterhalb der symbolischen, dargestellt. Obwohl symbolische und subsymbolische Lernsysteme häufig zur Behandlung der gleichen Probleme eingesetzt werden, gibt es nur sehr wenige Untersuchungen, in denen ihre Stärken und Schwächen gegenübergestellt werden. Eine derartiger experimenteller Vergleich zwischen Perceptrons, Backpropagation (vgl. Kapitel 6) und ID3 wurde von Shavlik, Mooney und Towell in /Shavlik et al. 113/ durchgeführt. Dieser Vergleich liefert bei ausgewählten Testdaten als Ergebnis, daß Backpropagation die anderen beiden Systeme bei einem relativ geringen Umfang an Trainingsdaten übertrifft und daß es ein wenig akurater als ID3 ist, wenn die Daten gestört oder unvollständig sind. Zusammenfassend, so Shavlik et al., *übertrifft Backpropagation in bezug auf die Korrektheit der Klassifikation neuer Beispiele die anderen beiden Verfahren geringfügig, benötigt jedoch eine deutliche längere Trainingsphase.*

Charakteristisch für alle induktive Lernverfahren ist das Lernen allgemeiner Regeln oder Hypothesen anhand gegebener multipler Beispiele. Hierzu basieren induktive Methoden typischerweise auf einer großen Anzahl von Trainingsbeispielen /Dietterich 21/.

Neben der hier dargestellten Subklassifikation induktiver Lernstrategien können diese in **überwachtes Lernen** (Abschnitte 3.3.4.1 und 3.3.4.2) und **nicht-überwachtes Lernen** (Abschnitt 3.3.4.3) eingeteilt werden. Da den sub-symbolischen Lernverfahren zwei eigene Kapitel gewidmet wurden, werden in den folgenden Abschnitten dieses Kapitels ausschließlich symbolische Lernverfahren dargestellt.

3.3.4.1 Lernen aus Beispielen

Gegeben sei eine Menge von vorklassifizierten positiven Beispielen und Gegenbeispielen (negativen Beispielen) eines Konzeptes. Das System folgert, auf der Basis der Objekteigenschaften dieser Beispiele, durch Aufstellung und evtl. sogar Bewertung von Hypothesen, ein allgemeines Konzept, das alle positiven und keines der Gegenbeispiele beschreibt. Das somit gewonnene Wissen kann zur Klassifikation, Bewertung oder zur Lösung in ähnlichen Problemsituationen eingesetzt werden.

Die Methode des Lernens aus Beispielen wurde in der Künstlichen Intelligenz intensiv untersucht. Der Aufwand der Schlußfolgerungen, die vom System ausgeführt werden müssen, ist wesentlich größer als beim Lernen durch Instruktion, da kein allgemeines Konzept von einem Lehrer zur Verfügung gestellt wird, und er ist größer, als beim Lernen durch Analogien, da keine vergleichbaren Konzepte als "Samen" vorhanden sind, aus denen ein neues Konzept entstehen könnte.

Lernen aus Beispielen kann anhand des Ursprungs der Beispiele wie folgt weiter unterteilt werden:

(1) Die Beispiele stammen von einem Lehrer, der das Konzept kennt und die Beispielsequenzen für den Lernprozeß so hilfreich wie möglich generiert. Falls der Lehrer den Wissensstand des Systems kennt, so kann er die Beispiele so auswählen, daß der Lernvorgang optimiert wird.

(2) Der Ursprung ist das System selbst. Er kennt zwar seinen eigenen Wissensstand, weiß jedoch nicht, welches Konzept erworben werden soll. Deshalb kann das System Instanzen erzeugen und er verfügt über eine externe Einrichtung wie die Umwelt oder einen Lehrer, der diese als Positiv- oder Gegenbeispiele klassifiziert. Auf dieser Grundlage glaubt er zwischen konkurierenden Konzeptbeschreibungen unterscheiden zu können.

(3) Die Quelle ist die externe Umwelt. In diesem Fall verläuft der Vorgang der Beispielerzeugung ziemlich zufällig, da sich das System auf relativ unkontrollierte Beobachtungen verlassen muß.

Diese Form des Lernens aus Beispielen wird häufig auch als **Lernen durch Experimentieren** bezeichnet.

Des weiteren kann man Lernen aus Beispielen durch den dem Lernenden präsentierten Typ der Beispiele subklassifizieren:

(1) Nur positive Beispiele werden vorgegeben. In einem solchen Fall werden keine Vorkehrungen getroffen, die das System vor Übergeneralisierung bei der Hypothesenbildung bewahren. Übergeneralisierung kann jedoch dadurch vermieden werden, daß man im Rahmen des Generalisierungsprozesses nur die minimal notwendige Generalisierung betrachtet, oder daß man a priori Domainwissen einsetzt, um die möglichen Konzeptbeschreibungen einzugrenzen.

(2) Falls nur Gegenbeispiele vorgegeben werden, kann ein Lernen im eigentlichen Sinne nicht stattfinden, da keine Generalisierung erfolgt; im Gegenteil, es käme

sogar zu einer "Überdifferenzierung", die zu Wissensverlust führen würde. Daher ist diese Vorgehensweise nicht sinnvoll.

(3) Positiv- und Gegenbeispiele werden vorgegeben. In diesem Fall unterstützen die positiven Beispiele den Generalisierungsprozeß (bottom-up), während die Gegenbeispiele vor Übergeneralisierung schützen (top-down); denn das gefolgerte Konzept sollte niemals so allgemein sein, daß es einige der Gegenbeispiele umfaßt. Im wesentlichen bedeutet dies, daß durch die vorgegebenen Beispiele und Gegenbeispiele explizit Hinweise über die Zugehörigkeit zu einem zu lernenden Konzept gegeben werden. Aus diesem Grund wird diese Form des Lernens aus Beispielen auch als überwachtes Lernen bezeichnet. Dies ist die typischste Form des Lernens aus Beispielen.

- Generalisierung:
  Das Problem besteht darin, die zwischen verschiedenen Beispielen bestehenden Gemeinsamkeiten bottom-up zu induzieren, indem der Gültigkeitsbereich der Beispiele im Sinne einer Mengen-Obermengen-Relation erweitert wird. Diese übergeordnete (gemeinsame) Merkmalsmenge charakterisiert das allgemeine Konzept, dessen Instanzen sie darstellen.

- Spezialisierung:
  Das Problem der Spezialisierung besteht darin, die Merkmale top-down zu induzieren, die Unterschiede zwischen mehreren Beispielen beschreiben. Jedes Beispiel wird durch diejenigen Merkmale beschrieben, die bei den anderen nicht auftreten.

Lernen aus Beispielen besteht entweder aus einem Versuch oder ist inkrementell; dies stellt eine weitere Möglichkeit zur Subklassifikation dar. Im ersten Fall werden sämtliche Beispiele auf einmal präsentiert. Im letzteren Fall muß das System eines oder mehrere hypothetische Konzepte, die zu den gegeben Daten konsistent sind, bilden und diese, nachdem weitere Beispiele betrachtet wurden, verfeinern.

Der inkrementelle Ansatz entspricht eher dem menschlichen Lernen. Er erlaubt dem Lernenden teilweise gelernte Konzepte einzusetzen und ermöglicht es dem Lehrer, die wesentlichen Aspekte eines neuen Konzeptes vor den weniger zentralen Details herauszustellen.

Andererseits können schlecht gewählte initiale Trainingsbeispiele das System in die Irre führen, wenn sie nicht gleich von Anfang an auf die wesentlichen Aspekte des zu erlernenden Konzepts fokussieren (/Richter, Wendel 100/).

3.3.4.2 Lernen durch Operationalisieren

Beim Lernen durch Operationalisieren handelt es sich um eine weitere Strategie zum Maschinellen Lernen, wie sie in /Dillmann 22/ beschrieben wird.

Diese Lernstrategie basiert auf einem System, das bereits über Wissen aus seinem Anwendungsgebiet verfügt. Von einem "intelligenten" externen Beobachter seines Verhaltens wird ihm ein Ratschlag, Tip oder ein Hinweis zur Nachhilfe erteilt. Dieser erfolgt durch eine Eingabe in einer Weise, die das System nicht direkt befolgen kann und muß, d. h. es wird kein uneingeschränkter Gehorsam beabsichtigt.

Der Lernprozeß besteht darin, daß das System die Anweisung in eine Folge von Operationen oder in geänderte Heuristiken übersetzt, die dann bestimmte Operationsfolgen bevorzugen. Damit wird immer dann dieser Anweisung Rechnung getragen, wenn das System eine solche Operationsfolge auswählt; dadurch ergibt sich hier eine Querverbindung zum Lernen durch Instruktion.

Das Lösen von Aufgaben aus dem Anwendungsgebiet erfordert Hintergrundwissen, mit dem das System auch ohne Operationalisierung arbeitet.

Die Repräsentation des Hintergrundwissens muß so erfolgen, daß das Ergebnis der Operationalisierung leicht ergänzt werden kann.

3.3.4.3 Lernen durch Beobachtung und Entdeckung

Dies ist die allgemeinste Form des induktiven Lernens. Sie beinhaltet die Entwicklung von Systemen, theoriebildende Aufgaben, die Erzeugung von Klassifikationskriterien zur Bildung um taxonomischer, d. h. systematischer Hierarchien, etc., ohne Unterstützung eines externen Lehrers.

Die Aufgabe besteht darin, allgemeine Beschreibungen (eines Gesetzes, einer Theorie) zu bestimmen, die eine Ansammlung von Beobachtungen charakterisieren. Durch Auswertung der symbolischen Informationen der Beobachtungen werden somit Beschreibungen erzeugt, die Eigenschaften von Objekten spezifizieren, die eine bestimmte Klasse (ein Konzept) repräsentieren.

Diese Form des **nicht-überwachten Lernens** erfordert vom System umfangreichere Schlußfolgerungen als jeder der bereits behandelten Ansätze. Dem System wird weder eine Menge von Instanzen eines speziellen Konzeptes, noch eine von außen gesteuerte Möglichkeit der Klassifikation von intern generierten Beispielen zur Verfügung gestellt.

Da sich Beobachtungen und Entdeckungen gleichzeitig über mehrere Konzepte erstrecken können, tritt das Problem auf, worauf sich die Aufmerksamkeit konzentrieren soll.

Lernen durch Beobachtung kann man nach dem Grad der Wechselwirkung mit der externen Umgebung unterteilen. Als extreme Situationen treten dabei auf:

(1) Passive Beobachtung:
Das System klassifiziert und ordnet die Beobachtungen verschiedenster Aspekte seiner Umgebung taxonomisch an.

(2) Aktives Experimentieren:
Hierbei stört das System seine Umwelt und beobachtet die Ergebnisse dieser Störungen. Die Experimente können zufällig, dynamisch auf die interessantesten Kriterien oder streng durch theoretische Einschränkungen gesteuert werden.

Wenn ein System Wissen erwirbt und Hypothesen über Theorien aufstellt, dann kann es dazu gezwungen werden, seine Theorien zu bestätigen oder abzulehnen. Folglich untersucht es seine Umwelt, indem es je nach Bedarf verschiedene Beobachtungs- und Experimentierstrategien anwendet.

Diese Form des Lernens beinhaltet häufig eine Generierung von Beispielen, um hypothetische oder teilweise erworbene Konzepte zu testen.

3.3.5 Lernen durch Deduktion (analytisches Lernen)

Ein lernendes System, das diese Strategie verwendet, führt deduktive (wahrheitserhaltende) Schlüsse entweder auf bereits vorhandenem oder auf speziell zur Verfügung gestelltem Wissen aus. Dies geschieht mit dem Ziel der Restrukturierung des gegebenen Wissens in nützlichere, effektivere Formen oder um wichtige Konsequenzen aus diesem Wissen zu ziehen (/Michalski 78/).

Deduktive Lernmethoden lernen allgemeine Regeln oder Theoreme aus speziellen, vorgegebenen Beispielen. Hierzu verwenden sie eine Theorie als gültige Basis für die Generalisierungen und deduktive Mechanismen zur Herleitung von Schlußfolgerungen. Deduktive Methoden basieren typischerweise auf einzelnen Trainingsbeispielen /Dietterich 21/.

Die grundsätzliche Vorgehensweise beim deduktiven Lernen ist die folgende:

- Gegeben ist ein Trainingsbeispiel eines Konzeptes, eine Definition dieses Konzeptes und ein vollständiges Hintergrundwissen.
- Gesucht ist eine neue Definition dieses Konzeptes, die sowohl effektiver als auch operationaler ist.

4 Maschinelle Lernverfahren

Im Abschnitt 3.3 wurden die verschiedenen Lernstrategien zur Klassifikation maschinell lernender Systeme herangezogen. Im folgenden werden einige der bekanntesten Prozeduren zum Maschinellen Lernen, aufgrund des überwiegend verwendeten Inferenztyps und dem Grad des Hintergrundwissens, charakterisiert. Die sich ergebenden vier Kategorien sind eher als Tendenzen, denn als strenge Zuordnungen zu sehen; die Grenzen zwischen ihnen sind mehr oder minder fließend.

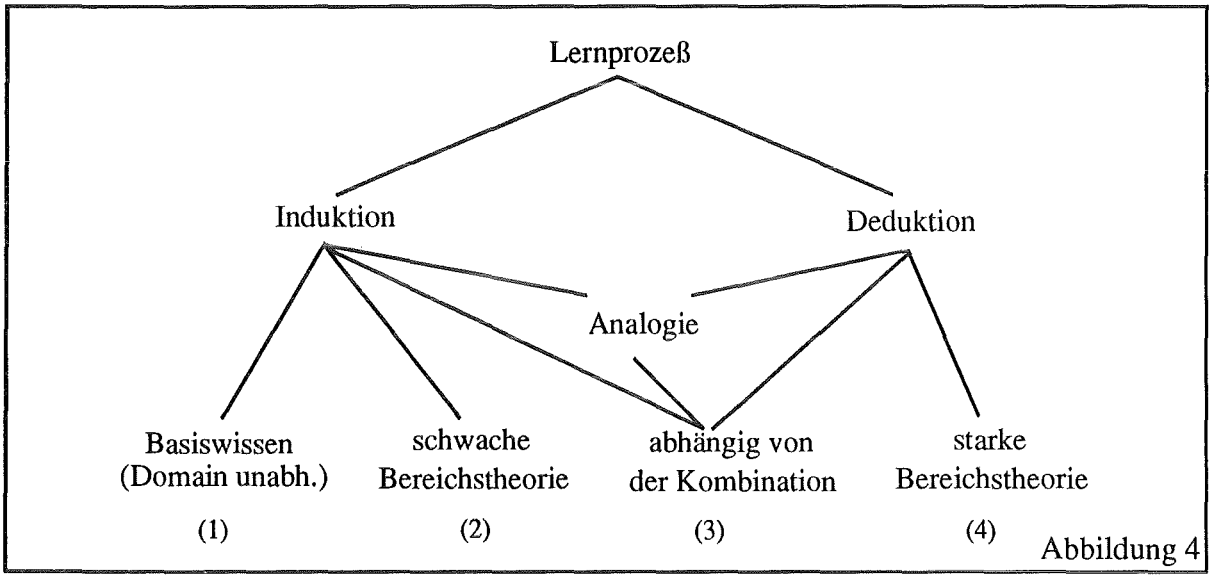

Abbildung 4

Die Wahl des überwiegend zu verwendenden Inferenztyps muß im Hinblick auf die grundsätzliche Aufgabe und der Art des dem Lernprozeß zur Verfügung gestellten Wissen gesehen werden. Soll neues Wissen, z. B. auf der Basis von Beispielen oder Beobachtungen, erzeugt werden, so müssen induktive Inferenzen (synthetisches Lernen) eingesetzt werden. Besteht hingegen die Hauptaufgabe des Lernprozesses in der Transformation oder Neuorganisation des bereits vorhandenen (gesicherten) Wissens in eine bessere Form (nach einem vorgegebenen Kriterium), so muß man auf deduktive Inferenzen (analytisches Lernen) zurückgreifen. In diesem Sinne stellen Lernprozesse, die sich auf Analogieschlüsse stützen, eine Kombination induktiver und deduktiver Methoden dar. Die somit erhaltene Klassifikation wurde durch den Grad an Hintergrundwissen, das dem Lernprozeß vorgegebenen wird, verfeinert.

Im folgenden werden die obigen vier Kategorien maschinell lernender Prozesse näher betrachtet und einige der bekanntesten, bereits realisierten Vertreter explizit dargestellt.

(1) Induktive Verfahren, die nur Domain-unabhängiges Basiswissen zur Aufstellung der Hypothesen benutzen, werden als **empirisch induktiv** bezeichnet.

Empirisch induktive Verfahren werden zur empirischen Generalisierung, zur qualitativen Entdeckung, zur begrifflichen Ballung (conceptual Clustering), zur inkrementellen Begriffsbildung, zum einfachen, fallbasierten Lernen und in konnektionistischen Methoden (vgl. Kapitel 6) eingesetzt.

Empirische Induktion kann konzeptuell als "Rückwärtsverfolgen" tautologischer Implikationen, d. h. domain-unabhängiger, allgemeingültiger Generalisierungsregeln aufgefaßt werden (vgl. Abschnitt 2.1.2).

Beispiele von Ansätzen/Systemen, die empirisch induktive Verfahren verwenden:

Symbolische Repräsentation:

- Empirische Generalisierung:
  Bei der empirischen Generalisierung werden allgemeine Beschreibungen aufgrund spezieller Beispiele erzeugt. Hierzu werden die grundsätzlichen Konzepte (Attribute), die in den Beschreibungen der Beispiele vorgegeben sind, verwendet. Beispiele hierfür sind Verfahren zum Erlernen von Regeln (für mehrere Konzepte), z. B. die AQ-Familie und Verfahren zum Erlernen von Entscheidungsbäumen (für ein Konzept), z. B. die ID3-Familie.

- Begriffliche Ballung / inkrementelle Konzeptbildung:
  Beispiele von Verfahren zum conceptual Clustering und zur inkrementellen Konzeptbildung sind CLUSTER, EPAM, UNIMEM, COBWEB und CLASSIT.

- Genetische Algorithmen:
  Die erweiterten Ansätze für genetische Algorithmen (siehe Kapitel 5) basieren auf einer symbolischen Form der Repräsentation, da Aktionsgruppen direkt kodiert werden.

  Klassische genetische Algorithmen nach Holland (siehe Kapitel 5) sind ebenfalls unter die symbolischen Verfahren einzuordnen, da letztlich auch nur Kombinationsmöglichkeiten von Symbolen (Operatoren usw.) aus einer gegebenen Grundmenge berechnet werden. Eine subsymbolische Repräsentation ist zwar vorstellbar, die berechneten Elementarterme fallen aber aus der gegebenen Grundmenge heraus und besitzen somit sematnisch keinen Bezug mehr.

Sub-symbolische Repräsentation:

- Neuronale Netze (siehe Kapitel 6).

(2) Induktive Verfahen, die auf einer schwachen Domain-Theorie aufbauen, werden auch als **konstruktiv-induktiv** bezeichnet. Es wird hierbei versucht, aus vollständigen oder gestörten Eingabedaten mit Hilfe des Hintergrundwissens möglichst viel Wissen zu gewinnen. Verfahren, die sich der konstruktiven Induktion bedienen, sind abduktive und wissensbasiert generalisierende Verfahren.

Konstruktive Induktion ist im Gegensatz zur empirischen Induktion eine wissensintensive Induktion. Hierbei wird Hintergrundwissen zur Erzeugung neuer Konzepte oder Beschreibungen sowie zur Bildung von Charakterisierungen höherer Ordnung für die gegebenen Eingabeinformationen verwendet. Im Gegensatz zur konstruktiven Deduktion basieren diese Systeme auf einer **schwachen Domain-Theorie**, die nur aus einer unvollständigen Beschreibung der Objekte des Anwendungsbereiches besteht

und keine Beschreibungen zulässiger Aktionen im Anwendungsgebiet enthält (/Tecuci, Kodratoff 117/).

Beispiele von Systemen/Ansätzen, die konstruktiv-induktive Verfahren verwenden:

PROTOS (Bareiss, Porter, Wier), XP's (Schank).

(3) Verfahren, die induktive und deduktive Inferenzen, beispielsweise in Form von Analogieschlüssen, miteinander kombinieren, werden als **Mischstrategien** bezeichnet.

Mischstrategie-Systeme sind ferner Lernsysteme, die symbolische und subsymbolische Verfahren kombinieren, Lernen durch Analogie und erweitertes fallbasiertes Lernen etc. einbeziehen.

Beispiele von Systemen/Ansätzen, die Mischstrategien verwenden:

DISCIPLE (Tecuci, Kodratoff), ARCH (Winston).

(4) Es ist bereits in der Natur deduktiver Inferenzen begründet, daß deduktive Verfahren auf einer **starken Bereichstheorie**, d. h. auf vollständigem, kohärentem und konsistentem Hintergrundwissen beruhen müssen.

Im Hinblick auf die Aufgabe des maschinell lernenden Systems können deduktive Verfahren in **konstruktiv deduktive** und **axiomatisch deduktive Verfahren** subklassifiziert werden. Der Unterschied zwischen konstruktiver Deduktion und axiomatischer Deduktion besteht darin, das erstere eine (wahrheitserhaltende) Transformation der Repräsentationsform bewirkt und dazu eher eine Anzahl von Wissenstransformationen als strikt logikbasierte Methoden einsetzt. Dies äußert sich insbesondere darin, daß in konstruktiv deduktiven Verfahren als Inferenz häufig die **bedingte Deduktion**, d. h. der deduktive Schluß ist abhängig vom jeweiligen Kontext, eingesetzt wird.

Verfahren zur konstruktiven Deduktion sind abstrahierende Verfahren, Umformulierungsverfahren und Lernen durch plausible Deduktion.

Bei axiomatisch deduktiven Verfahren spielt das vorgegebene Hintergrundwissen eine mit den Axiomen einer mathematischen Theorie vergleichbare Rolle. Axiomatisch deduktive Verfahren beruhen daher auf einem streng logischen Ansatz.

Reines erklärungsbasiertes Lernen und Verfahren zur automatischen Programmsynthese sind axiomatisch deduktive Verfahren.

Beispiele von Systemen/Ansätzen, die deduktive Verfahren verwenden:

GENESIS (DeJong und Mooney), LEX-II (Mitchell und Utgoff), SOAR (Laird, Newell und Rosenbloom), STRIPS (Fikes, Hart und Nilsson), FOO und BAR (Mostow), LEXCOP und MetaLEX (Keller), ANALOGY (Winston), Derivational Analogy (Carbonell).

4.1 Datengesteuertes Lernen eines einzigen Konzeptes

4.1.1 ID3 und der CLS-Algorithmus

Bei dem ID3-Algorithmus (Interactive Dichotomizer 3) handelt es sich um eine Weiterentwicklung auf der Basis von Hunt's CLS-Algorithmus (Concept Learning System). Der CLS-Algorithmus verwendet einen top-down, **divide-and-conquer**-Ansatz, um einen Entscheidungsbaum zur Klassifikation eines einzelnen Konzeptes, d. h. einer Klassifikation in zwei Klassen, aufgrund gegebener vorklassifizierter Beispiele zu konstruieren. Eigentlich erzeugt der ID3- bzw. CLS-Algorithmus keine Klassifikation von Elementen, sondern nur eine Klassifikationsregel; eine Klassenzugehörigkeit anhand derer der Entscheidungsbaum erzeugt werden soll, ist dem Algorithmus vielmehr vorgegeben. Zur Repräsentation der einzelnen Beispiele werden in CLS Merkmalsvektoren verwendet. CLS hat die Aufgabe für eine gegebene Menge vorklassifzierter Beispiele, die durch eine bestimmte Anzahl an Attributen mit möglichst "kleinen", diskreten Wertebereichen beschrieben werden, einen Entscheidungsbaum, der sämtliche gegebenen Beispiele korrekt klassifiziert zu erzeugen. In seiner ursprünglichen Version betrachtet ID3 maximal zwei verschiedene Konzepte.

Gegeben:

- Eine Menge bzgl. ihrer Konzeptzugehörigkeit vorklassifizierter Beispiele (d. h. positive und negative Beispiele für ein Konzept),
- eine Attributmenge zur Charakterisierung dieser Beispiele, wobei jedes Attribut einen möglichst "kleinen", diskreten Wertebereich haben sollte.

Gesucht:

- Ein Entscheidungsbaum, mit dem alle Beispiele aufgrund ihrer Attributwerte korrekt klassifiziert werden können.

Eine Klassifikationsregel in Form eines Entscheidungsbaumes kann für jede derartige Beispielmenge M gebildet werden. Eine zusätzliche Forderung an die Beispielmenge besteht darin, daß eine eindeutige Zuordnung der Beispiele zu den zugehörigen Klassen möglich sein muß. Falls die Menge M leer ist, so kann sie einer beliebigen Klasse zugeordnet werden. Falls alle Beispiele in M zur gleichen Klasse gehören, so besteht der Entscheidungsbaum aus einem Blatt, das den Klassenname beinhaltet. Im anderen Fall enthält M Repräsentanten beider Konzepte. Durch die Auswahl eines Attributes wird M in disjunkte Mengen $M_1, M_2, M_3, \ldots, M_n$ partitioniert. M_i enthält alle diejenigen Elemente von M, die den i-ten Wert auf dem ausgewählten Attribut annehmen. Jede dieser Teilmengen wird dann auf die gleiche Art und Weise weiter verfeinert, solange bis die Teilmengen nur noch Elemente eines einzigen Konzeptes enthalten. Das Ergebnis dieser Vorgehensweise ist ein Baum, dessen Blätter jeweils einen Klassennamen enthalten, jeder innere Knoten spezifiziert ein zu testendes Attribut und besitzt für jeden möglichen Wert eine abgehende Kante (/Quinlan 95/).

Beispiel: Gegeben seinen die folgenden Trainingsbeispiele (/Kodratoff 56/):

Positive Trainigsbeispiele für das Konzept ("+"):

Größe	Nationalität	Familienstand
klein	Deutsch	ledig
groß	Französisch	ledig
groß	Deutsch	ledig

Negative Trainigsbeispiele für das Konzept ("-"):

Größe	Nationalität	Familienstand
klein	Italienisch	ledig
groß	Deutsch	verheiratet
groß	Italienisch	ledig
groß	Italienisch	verheiratet
klein	Deutsch	verheiratet

Beginnt man den Aufbau des Entscheidungsbaumes mit dem Attribut Nationalität und wählt im zweiten Schritt das Attribut Familienstand, so erhält man den in Abbildung 5 dargestellten Entscheidungsbaum:

Ein Beispiel wird nun klassifiziert, indem man an der Wurzel des Entscheidungsbaumes beginnt, das entsprechende Attribut auswählt und dann der zugehörigen Kante des Entscheidungsbaumes folgt, bis man ein Blatt erreicht. Bemerkenswert ist, daß die Klassifikation eines speziellen Beispieles bereits aufgrund einer geringen Anzahl an Tests durchgeführt werden kann. In obigem Beispiel ist es für die Klassifikation bereits ausreichend, wenn man weiß, daß das Attribut Nationalität des Beispiels die Werte italienisch oder französisch besitzt. Insbesondere ist der Wert des Attributes Größe für die Klassifikation vollkommen unerheblich (/Quinlan 95/).

Aus diesem Beispiel wird ersichtlich, daß die Auswahl (Reihenfolge) der Attribute die Form (Breite und Tiefe) des Entscheidungsbaumes in entscheidender Weise beeinflußt. Die Effektivität der Klassifikationsregel hängt bei dieser divide-and-conquer-Strategie somit in entscheidender Weise von der Wahl der Partition der Beispielmengen ab. Würde man in obigem Beispiel die Reihenfolge der ausgewählten Attribute vertauschen, so wären auch im entsprechenden Entscheidungsbaum die Ebenen vertauscht. Dies würde jedoch bedeuten, daß für jede Klassifikation der Entscheidungsbaum in seiner gesamten Höhe durchlaufen werden muß.

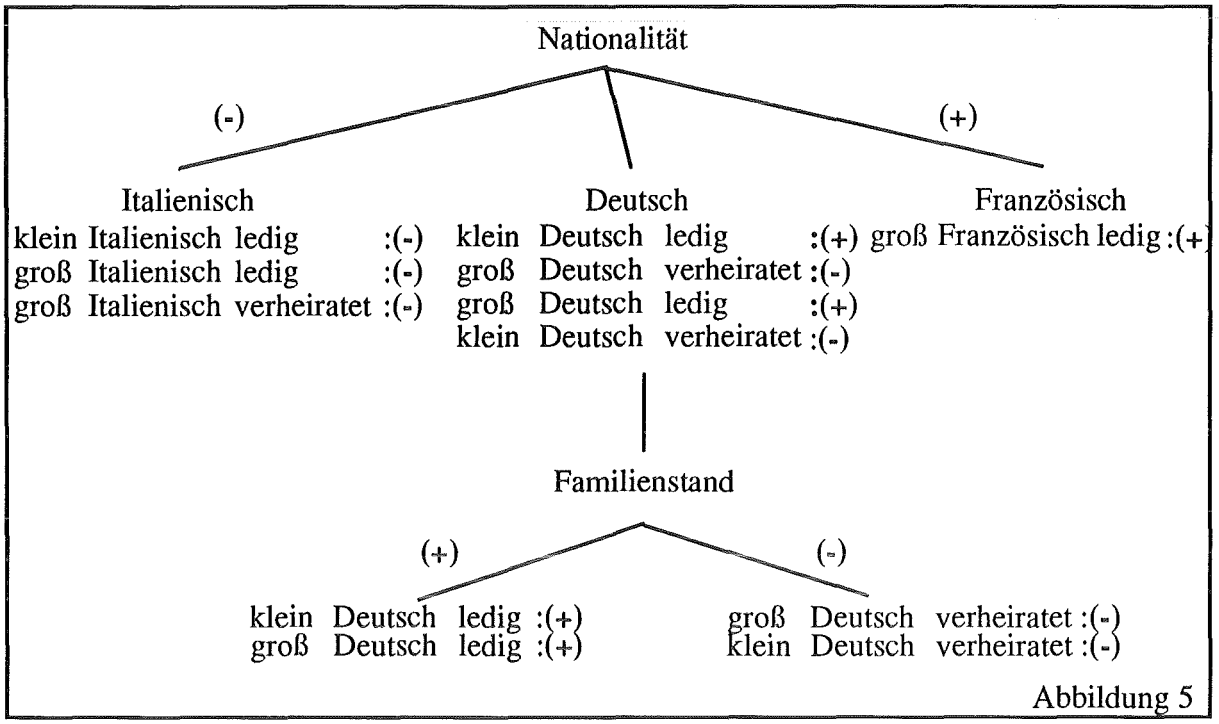

Abbildung 5

Um dies zu vermeiden, muß man beim Aufbau des Entscheidungsbaumes Wissen über den Informationsgehalt der einzelnen Attribute zu Grunde legen. Im Falle endlich vieler diskreter Attributausprägungen bedient man sich hierzu der **Shannonschen Entropie**, einem Maß aus der Informationstheorie (vgl. z. B. /Klix 55/), mit dem Ziel, die erwartete Anzahl an Tests, um ein Beispiel korrekt zu klassifizieren, zu minimieren.

Wie oben sei M die Menge der Beispiele und p sei die relative Häufigkeit derjenigen Beispiele von M, die zum Konzept (+) gehören (in bezug auf die Anzahl der Elemente von M). Angenommen es gäbe einen durch die Mengen M gebildeten Entscheidungsbaum, der alle Beispiele von M richtig klassifiziert, d. h. ein beliebiges Beispiel, auf das der Entscheidungsbaum angewendet wird, wird entweder dem Konzept zugeordnet (+) oder nicht (-). Falls M im Sinne der durch den Entscheidungsbaum zu klassifizierenden Beispiele typisch ist, ist es sinnvoll anzunehmen, daß die relative Häufigkeit dieser Beispiele (bezogen. auf die Gesamtanzahl), die durch den Entscheidungsbaum dem Konzept zugeordnet werden (+), ebenfalls p ist. Mit anderen Worten kann der Enscheidungsbaum als Nachrichtengenerator für die Nachrichten "das Konzept trifft mit Wahrscheinlichkeit p" und "das Konzept trifft mit Wahrscheinlichkeit (1-p) nicht zu" aufgefaßt werden.

Der Informationsgehalt einer solchen Nachricht ist durch

$$D(M) = -p \log_2(p) - (1 - p) \log_2(1 - p)$$

gegeben. D(M) ist ein Maß der Information, die durch den aus M abgeleiteten hypothetischen Entscheidungsbaum zur Verfügung gestellt werden muß. Alternativ kann es zur Vorhersage der Komplexität des Entscheidungsbaumes herangezogen werden (/Quinlan 96/).

Es sei A ein Attribut mit n möglichen Werten A_1, A_2, A_3, . . . , A_n. Um zum Aufbau des Entscheidungsbaumes das Attribut A als Wurzel zu verwenden, teilt man die Beispielmenge

M in Teilmengen $M_1, M_2, M_3, \ldots, M_n$ auf, wobei M_i alle Beispiele aus M enthält, deren Attribut A den Wert A_i hat. Die in diesem Fall benötigte Information kann man wie folgt ausdrücken:

$$B(M, A) = \sum_i (W_i * D(M_i)),$$

wobei das Gewicht W_i die relative Häufigkeit der Beispiele in der Teilmenge M_i ist. Der Informationsgewinn, der durch das Testen des Attributes A entstanden ist, entspricht damit der Differenz

$$D(M) - B(M, A).$$

Wie zu erwarten, ist diese Differenz immer größer oder gleich Null. Als Wurzel des Entscheidungsbaumes für die Beispielmenge M verwendet CLS daher das Attribut, das diesen Attributgewinn maximiert bzw. die verbleibende Information minimiert (/Quinlan 96/).

Ad Beispiel: Aufgrund dieser informationstheoretischen Betrachtungen kann man nun die oben bereits getroffenen (intuitiven) Aussagen über den Baumaufbau theoretisch nachvollziehen.

Die Menge M umfaßt acht Beispiele, die sich wie folgt auf die Klassen verteilen; drei Beispiele gehören zum Konzept (+) und fünf nicht (-). Damit ist $p = 3/8$ und

$$D(M) = - 3/8 \log_2(3/8) - (1 - 3/8) \log_2(1 - 3/8) = 0{,}954.$$

Wird, wie oben, als Wurzel des Entscheidungsbaumes das Attribut Nationalität ausgewählt, so ergibt sich

$$\begin{aligned} B(M, \text{Nationalität}) &= 3/8*D(I) + 4/8*D(D) + 1/8*D(F) \\ &= 3/8 * 0 + 1/2 * 1 + 1/8 * 0 \\ &= 1/2. \end{aligned}$$

Wählt man das Attribut Nationalität als Wurzel des Entscheidungsbaumes, so ergibt sich als Informationsgewinn

$$D(M) - B(M, \text{Nationalität}) = 0{,}454.$$

Würde man hingegen das Attribut Familienstand als Wurzel des Entscheidungsbaumes wählen, so ergäbe sich:

$$B(M, \text{Familienstand}) = 3/8*D(\text{verheiratet}) + 5/8*D(\text{ledig}) = 0{,}607.$$

Dies liefert einen Informationsgewinn von 0,347. Damit ist der Informationsgewinn größer, wenn man das Attribut Nationalität als Wurzel des Entscheidungsbaumes wählt, als wenn man das Attribut Familienstand wählen würde. Folglich werden die bereits in obigem Beispiel rein intuitiv getroffenen Einwände durch diese Theorie bestätigt.

Würde man das Attribut Größe als Wurzel des Entscheidungsbaumes wählen, so erhielte man nur einen Informationsgewinn von 0,003 (B(M,Größe) = 0,951). Hieraus kann man folgern, daß das Attribut Größe ziemlich unwichtig für die gewünschte Klassifikation ist.

Im Unterschied zu ID3 benötigt der CLS-Algorithmus sämtliche Trainingsinstanzen mit wahlfreiem Zugriff von Beginn an. Dadurch besteht eine praktische Einschränkung der Problemgröße, die mit dem CLS-Algorithmus gelöst werden können. Der ID3-Algorithmus wurde speziell zur Lösung extrem großer Konzeptlernprobleme konzipiert. Er verwendet einen Ansatz zur aktiven Experimentplanung, um eine "gute" Teilmenge der Trainingsbeispiele auszuwählen und benötigt lediglich einen selektiven Zugang zur Gesamtmenge der Trainingsbeispiele. Im folgenden wird der ID3-Algorithmus in seinen entscheidenden Schritten grob skizziert.

ID3-Algorithmus:

(1) Wähle zufällig eine Teilmenge der Kardinalität W aus der Gesamtmenge der Trainingsbeispiele (W heißt Window-Größe, die zugehörige Teilmenge heißt Window.)

(2) Verwende den CLS-Algorithmus um einen Entscheidungsbaum aufzubauen, der die Beispiele des Windows korrekt klassifiziert.

(3) Suche innerhalb der Gesamtmenge der Trainingsbeispiele nach Ausnahmen zum momentanen Entscheidungsbaum.

(4) Bilde ein neues Window, das aus einer Kombination einiger Trainingsbeispiele des momentanen Windows und einigen Ausnahmen, die im dritten Schritt bestimmt wurden, besteht.

Wiederhole die Schritte 2 bis 4 bis der Entscheidungsbaum alle Beispiele der Trainingsmenge korrekt klassifiziert.

Trotz der einfachen Form der Wissensrepräsentation und des direkten Lernalgorithmus sind ID3 und auch CLS sehr leistungsfähig zum Lernen eines einzelnen Konzeptes. Die Stärke von ID3 ist sicherlich durch eine ausgeklügelte Wahl der Trainingsbeispiele mitbegründet. Das wesentliche bei der dargestellten Form der Beispielauswahl besteht darin, daß auf diejenigen Trainingsbeispiele fokussiert wird, die Ausnahmen darstellen. Genau diese Beispiele werden benötigt, um die Repräsentation für das zu lernende Konzept zu verbessern.

Das Hauptproblem der CLS/ID3-Methode ist der Entscheidungsbaum. Zum einen ist das gelernte Konzept für den Menschen sehr schwer verständlich, wenn es in Form eines Entscheidungsbaumes vorliegt und zum anderen sind verschiedene Entscheidungsbäume nur schwierig auf Äquivalenz zu überprüfen (/Cohen, Feigenbaum 17/). Der Grund für die Probleme bei der Interpretation von Entscheidungsbäumen besteht darin, daß Entscheidungsbäume

zwar Entscheidungen über Konzeptzugehörigkeiten ermöglichen, jedoch keinerlei Erklärungen/Beschreibungen der Konzepte selbst anbieten /Gennari, Langley, Fisher 32/.

Aus diesem Grund zielen die meisten Weiterentwicklungen von ID3 auf ein besseres Verständnis der Ergebnisse ab. Beispielsweise erschien es in einigen Fällen sinnvoll anzunehmen, daß eine weitere Verästelung binärer Bäume, die vorzugsweise nur nach rechts erfolgt, die Verständlichkeit verbessert (/Kodratoff 56/).

Quinlan beschreibt in /Quinlan 94/, wie der ID3-Algorithmus um Techniken zur Behandlung numerischer Merkmale, gestörter Daten und fehlender Informationen (missing values) erweitert werden kann. Schlimmer und Fisher stellen in /Schlimmer, Fisher 107/ den ID4-Algorithmus vor. Hierbei handelt es sich um eine Erweiterung der traditionellen ID3-Methode, die einen inkrementellen Aufbau eines Entscheidungsbaumes, auf der Grundlage vorklassifizierter Beispiele, ermöglicht.

4.1.2 Die Versionenraum-Methode

Die Versionenraum-Methode mit dem Kandidaten-Eliminations-Algorithmus ist ein Ansatz von /Mitchell 87/ zum Lernen von Beschreibungen für ein einziges Konzept, aufgrund inkrementell vorgegebener positiver und negativer Trainingsbeispiele. Im Gegensatz zu ID3 verwendet die Versionenraum-Methode den **single-representation-trick**, d. h. die Trainingsbeispiele und die zu erzeugenden Regeln werden auf die gleiche Art und Weise repräsentiert. Etwas formaler bedeutet dies, daß die Form der Repräsentation im Beispiel- und im Regel-Raum die gleiche ist.

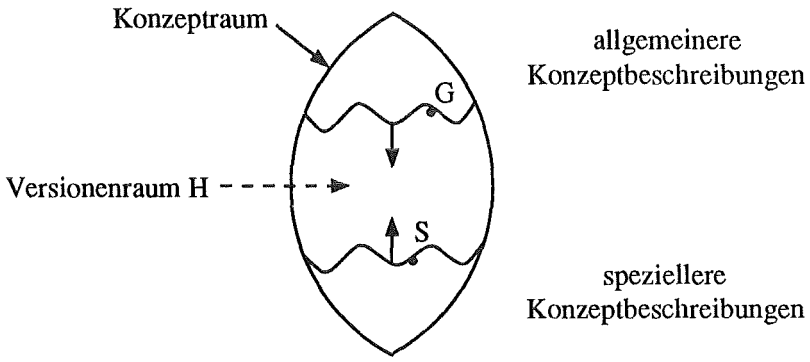

Abbildung 6

Das Ziel dieses Algorithmus besteht darin, eine vollständige, konsistente Generalisierung der Beispiele (d. h. das Konzept) zu finden. Hierbei heißt eine Generalisierung **vollständig**, wenn sie alle positiven Trainingsbeispiele umfaßt und **konsistent**, wenn sie keines der negativen Trainingsbeispiele umfaßt /Richter, Wendel 100/.

Untersucht man den Raum aller zur Konzeptbeschreibung möglichen Regeln, so umfaßt dieser alle Regeln "zwischen" der allgemeinsten Regel, der Null-Regel (alle Bedingungen wur-

den fallengelassen; sie besitzt keinerlei Einschränkungen für die Trainingsbeispiele und beschreibt daher nichts) und den speziellsten Regeln, den Trainingsbeispielen selbst.

Die Menge H der für das zu lernende Konzept plausiblen Hypothesen wird daher durch zwei Teilmengen aufgespannt werden (/Richter, Wendel 100/):

- Die Menge G der allgemeinsten Elemente aus H
  G = { g | g ist eine vollständige, konsistente Verallgemeinerung der Trainingsbeispiele, und es gibt keine Verallgemeinerung, die sowohl vollständig und konsistent als auch allgemeiner als g ist} und

- die Menge S der speziellsten Elemente aus H
  S = { s | s ist eine vollständige, konsistente Verallgemeinerung der Trainingsbeispiele, und es gibt keine Verallgemeinerung, die sowohl vollständig und konsistent als auch spezieller als s ist }.

Die Mengen S und G sind die Extrema der in bezug auf Vollständigkeit und Konsistenz zulässigen Generalisierungen. Speziellere Generalisierungen als diejenigen in S sind unvollständig; allgemeinere als diejenigen in G inkonsistent /Richter, Wendel 100/. Zur Bestimmung des gesuchten vollständigen und konsistenten Konzeptes muß man nun versuchen, den Versionenraum H möglichst klein, am besten einelementig zu machen.

Im folgenden wird, wie bei /Richter, Wendel 100/, der Kandidaten-Eliminations-Algorithmus vorgestellt, der bei sukzessiver Präsentation der Beispiel den Versionenraum verändert. Die Beispiele seinen a_1, a_2, a_3, . . . und a_1 sei positiv; weiter sei A ein künstliches Konzept, das allgemeiner als alle Aussagen ist.

Kandidaten-Eliminations-Algorithmus:

- Initialisierung: $S := \{ a_1 \}$, $G := \{ A \}$.

- Iterationsschritt: Das Trainingsbeispiel $a = a_{n+1}$ sei vorgegeben;

  - a ist positiv:

    - Entferne alle Konzepte aus G, die a nicht als Trainingsbeispiel haben.
    - Ersetze die Elemente s aus S durch die speziellsten Verallgemeinerungen von s, die a als Trainingsbeispiel haben und keines der früheren Gegenbeispiele umfassen; entferne sie, falls dies nicht möglich ist oder wenn sie dadurch allgemeiner als Elemente aus G werden.

  - a ist negativ:

    - Entferne alle Konzepte aus S, die a als Trainingsbeispiel haben.
    - Ersetze die Elemente g aus G durch die allgemeinsten Spezialisierungen von g, die nicht a aber alle früheren positiven Trainingsbeispiele als Beispiel haben; entferne sie, falls dies nicht möglich ist oder wenn sie dadurch spezieller als Elemente aus S werden.

- Abbruchkriterium: Kein Trainingsbeispiel ist mehr verfügbar oder G = S ist einelementig oder S oder G sind leer oder einer der Schritte ist nicht mehr ausführbar.

- Lernerfolg tritt ein bei G = S, und beide Mengen sind einelementig; sie enthalten dann das gesuchte Konzept.

Da in diesem Algorithmus die beiden Mengen S und G je nach Vorgabe positiver bzw. negativer Trainingsbeispiele genau dual behandelt werden, kann man sich auf positive Trainingsbeispiel beschränken. Im ersten Schritt müssen die Elemente von G, die a nicht als Beispiel enthalten, entfernt werden, denn eine weitere Verallgemeinerung würde ansonsten eine Inkonsistenz nach sich ziehen. Wird ein Element aus S durch eine nötig gewordene Verallgemeinerung noch allgemeiner als ein Element von G, so muß es aus demselben Grund entfernt werden. Falls das gesuchte Konzept K in der hier dargestellten Weise beschrieben werden kann, so sieht man für jeden Iterationsschritt:

(1) K ist in S oder eine Verallgemeinerung eines Ausdruckes von S und

(2) K ist in G oder eine Spezialisierung eines Ausdruckes von G.

Weil andererseits durch geeignete Trainingsbeispiele alle letztlich unvollständigen und inkonsistenten Konzepte ausgeschaltet werden können, ergibt sich daraus für eine hinreichend umfangreiche Darbietung an Trainingsbeispielen sowohl die Korrektheit als auch die erfolgreiche Terminierung des Algorithmus /Richter, Wendel 100/.

Ein weiteres Verfahren zum Erlernen eines einzelnen Konzeptes ist beispielsweise INDUCE (Larson, Michalski).

4.1.3 Ein Beispiel zur Versionenraum-Methode

In diesem Beispiel soll das Konzept des Kreises gelernt werden /Cohen, Feigenbaum 17/. Hierzu besteht die Menge der Trainingsbeispiele aus Kreisen, Dreiecken und Quadraten, die entweder groß oder klein sein können. Mit der Variablen y ∈ { klein, groß } kann das Konzept Kreis als (Kreis, y) dargestellt werden.

Damit ergibt sich der folgende Versionenraum, dargestellt in einer Relation vom Allgemeinen zum Speziellen.

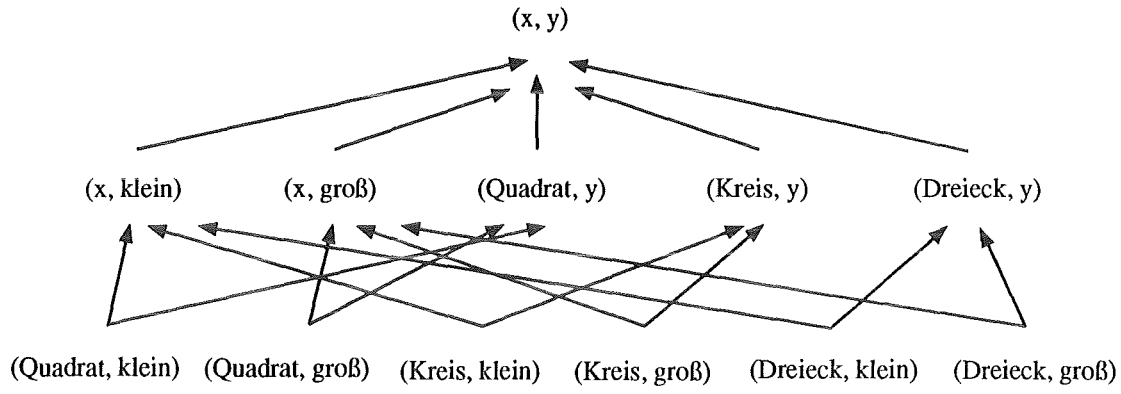

Abbildung 7

Somit ergeben sich als Initialisierung die folgenden beiden (extremen) Konzepte:

S = { (Quadrat, klein), (Quadrat, groß), (Kreis, klein), (Kreis, groß),
(Dreieck, klein), (Dreieck, groß) }

und G = { (x, y) }.

Iterationsschritte:

(1) Das erste positive Trainingsbeispiel sei (Kreis, klein). Damit wird der Versionenraum aufgespannt durch:

S = { (Kreis, klein) } und G = { (x, y) }.

(2) Das zweite Trainingsbeispiel sei negativ: (Dreieck, groß).

Es ergeben sich:

S := { (Kreis, klein) } und G := { (Kreis, y) , (x, groß) }.

G enthält jetzt zwei Elemente. Diese beiden Konzepte stellen notwendige aber keine hinreichenden Bedingungen an das zu lernende Konzept dar.

(3) Das dritte Trainingsbeispiel sei positiv: (Kreis, groß).

Dies führt zu:

$$S := \{ (x, y) \} \text{ und } G := \{ (\text{Kreis}, y) \},$$

der Algorithmus bricht ab; gelernt wurde das Konzept "Kreis".

Der dargestellte Algorithmus kann auf diese Weise nur Konzepte lernen, die im gesamten Graphen (dem Versionenraum) auch vorkommen. Damit kann das Konzept "(Kreis, klein) $\vee$ (Dreieck, groß)" nicht gelernt werden.

Der Algorithmus würde für dieses Konzept, abhängig von der Reihenfolge der Darbietung der Trainingsbeispiele entweder ein inkorrektes Ergebnis liefern oder ohne Erfolg terminieren. Diese Schwierigkeiten sind dadurch begründet, daß es sich bei dem Algorithmus um einen least-commitment-Algorithmus handelt, der nur dann generalisiert, wenn er dazu gezwungen wird. Disjunktionen hingegen stellen einen Weg dar, Generalisierungen zu vermeiden - und der Algorithmus wird niemals zum Generalisieren gezwungen.

4.1.4 BACON.x

Alle bisher beschriebenen induktiven Lernverfahren können in dem Klassifikationsschema von Abbildung 3 als Lernen aus Beispielen eingeordnet werden. Das im folgenden vorgestellte, nach dem englischen Philosophen der Naturwissenschaft, Sir Francis Bacon (1561 - 1626), benannte Lernsystem basiert auf der Strategie des Lernens durch Beobachtung und Entdeckung mit aktivem Experimentieren.

Die Programme der BACON Familie wurden zur Lösung einer Reihe von Aufgaben im Rahmen des Lernens eines einzigen Konzeptes eingesetzt. Beispielsweise wurden von Programmen der BACON Familie die klassischen Naturgesetze, wie

- das Ohmsche Gesetz,
- das Newtonsche Gesetz der universellen Gravitation und
- das dritte Kepplersche Gesetz

"wiederentdeckt". Für eine umfangreiche Darstellung der Theorie wissenschaftlicher Entdeckungen und der BACON Programme sei auf /Langley et al. 65/ verwiesen.

Da BACON in einem (Konzept-) Regel-Raum, der sich vom (Daten-) Beispielraum unterscheidet, eine Menge von (Termen) Hypothesen entwickelt, verwendet es (wie auch CLS/ID3) den single-representation-trick nicht. Zur Beschreibung der Trainingsbeispiele werden Merkmalsvektoren verwendet, wobei die Merkmale sowohl kontinuierliche reelle als auch diskrete symbolische oder numerische Werte annehmen können.

Ursprünglich (BACON.1 & 2) wurde die Welt von BACON in Daten oder Beobachtungen und Hypothesen oder Gesetze, die diese Daten erklären oder zusammenfassen eingeteilt. Erst in BACON.4 wurde diese Zweiteilung durch ein Kontinuum ersetzt, indem die Information in variierenden Beschreibungsebenen repräsentiert wird. Die niedrigste dieser Ebenen könnte man als Daten, die höchste als Hypothesen bezeichnen. Sämtliche dazwischenliegenden Ebenen beinhalten eigentlich Mischformen dieser beiden extremen Konzepte. Die Beschreibung auf einer Ebene fungiert als Hypothese bzgl. der darunterliegenden und als Datum bzgl. der darüberliegenden Beschreibung /Langley, Bradshaw, Simon 63/.

Die den Programmen der BACON Familie zugrundeliegende **Idee** ist einfach:

> Dem Programm wird eine Menge unabhängiger Variablen $v_1, v_2, v_3, \ldots, v_n$ und eine abhängige Variable v_0, sowie eine Reihe von Daten zu diesen Variablen vorgegeben. Es untersucht wiederholt die gegebenen Daten und wendet auf sie seine Verfeinerungs-Operatoren zur Erzeugung neuer Terme an. Diese Vorgehensweise wird fortgesetzt, bis BACON herausfindet, daß einer dieser Terme immer konstant ist. Ein einziges Konzept wird folglich durch die Aussage Term = konstant repräsentiert /Cohen, Feigenbaum 17/.

Die Fähigkeiten von BACON zur Entdeckung eines einzigen Konzeptes werden wesentlich von den verwendeten Heurismen zur Erzeugung neuer Terme, den Verfeinerungs-Operatoren bestimmt. Diese Heurismen sind in Form von Bedingungs-Aktions-Regeln dargestellt und werden als **Produktionen** bezeichnet. Der Bedingungsteil einer Produktion führt auf der Suche nach möglichen Mustern in den Daten umfangreiche Tests durch, der Aktionsteil erzeugt neue Terme.

Die folgenden Verfeinerungs-Operatoren waren bereits in BACON.1 implementiert /Cohen, Feigenbaum 17/:

- Feststellung der Konstanz eines Termes:
  Dieser Operator wird ausgelöst, wenn eine abhängige Variable mindestens zweimal den gleichen Wert w annimmt. Es wird eine Hypothese erzeugt, daß diese Variable immer konstant den Wert w annimmt.

- Spezialisierung:
  Dieser Operator wird ausgelöst, wenn durch die Daten ein Widerspruch zu einer bereits erzeugten Hypothese auftritt. Die Hypothese wird durch Hinzufügen einer konjunktiven Bedingung spezialisert.

- Erzeugung von Termen für die Steigung und den Achsenabschnitt:
  Dieser Operator stellt fest, daß sich zwei Variablen miteinander linear verändern und erzeugt neue Terme für die Steigung und den Achsenabschnitt dieser linearen Beziehung.

- Produktbildung:
  Dieser Operator stellt fest, wenn sich zwei Variablen (mit nicht konstanter Steigung) gegensätzlich verhalten. Er erzeugt einen neuen Term in Form des Produktes dieser beiden Variablen.

- Quotientenbildung:
  Dieser Operator stellt fest, wenn sich zwei Variablen (mit nicht konstanter Steigung) monoton (steigend oder fallend) verändern. Er erzeugt einen neuen Term in Form des Quotienten dieser beiden Variablen.

- Bildung des Termes modulo-n:
  Dieser Operator bemerkt, daß eine Variable v_1 immer dann ein und denselben Wert annimmt, wenn eine unabhängige Variable v_2 einen bestimmten Wert modulo n hat. Der neue Term v_2 modulo n wird erzeugt, wobei nur kleine Werte von n betrachtet werden.

Bei BACON.2 handelt es sich um eine Erweiterung von BACON.1, die über zwei weitere Operatoren zur Bestimmung wiederholt auftretender Sequenzen und zur Erzeugung polynomialer Terme durch wiederholte Differenzenberechnung verfügt.

BACON.3 ist eine weitere Erweiterung von BACON.1, die die Hypothesen der Konstanz-Bestimmungs-Operatoren zur Umformulierung der Trainingsinstanzen verwendet. Dadurch kann BACON.3 Information auf verschiedenen Ebenen ansteigender Komplexität repräsentieren, und es wird möglich, komplexere Gesetze zu formulieren und damit einen größeren Teil der Daten zu beschreiben. Diese erweiterte Repräsentation ermöglichte es dem System, seine eigenen Hypothesen als neue Daten aufzufassen, um somit Hypothesen rekursiv anzuwenden.

BACON.4 verfügt über einfache Möglichkeiten für den Umgang mit Daten, die in geringem Umfang verrauscht sind. Zur Untersuchung des Raumes der theoretischen Terme verwendet BACON.4 eine heuristische Suchmethode, die derjenigen in Lenat's AM Programm (/Lenat 69/) für den Raum der mathematischen Konzepte sehr ähnlich ist /Langley, Bradshaw, Simon 63/.

Für die Entwicklungsgeschichte von BACON 1 - 6 sei auf /Langley, Simon, Bradshaw 63/ verwiesen.

Das System FAHRENHEIT (/Zytkow, Zhu 122/) stellt eine Erweiterung von BACON zur Entdeckung mehrerer multidimensionaler Regularitäten und deren Wirkungsbereich dar.

Die Stärke von BACON liegt in der Fähigkeit einfache Gesetze zu entdecken, indem es reellwertige Variabeln zueinander in Beziehung setzt. Interessant sind die Heurismen, die im Rahmen dieser Vorgänge zur Bildung neuer Terme eingesetzt werden. Die Strategie von BACON.3 der Umformulierung der Trainingsinstanzen aufgrund erkannter partieller Regularitäten bietet Ansätze für weitere Untersuchungen.

Eine Schwierigkeit von BACON besteht in der Tatsache, daß die Ausführung der Operatoren nur unter sehr speziellen Bedingungen ausgelöst wird und damit das Programm sowohl durch die Reihenfolge der Variablen als auch durch die Werte der ausgewählten Trainingsinstanzen beeinflußt werden kann. Beispielsweise konnte BACON.3 aus diesen Gründen für einige Mengen von Trainingsbeispielen das Ohmsche Gesetz nicht entdecken. Um mittels BACON Konzepte effektiv zu entdecken, ist es daher notwenig, die Reihenfolge der Variablen und der speziellen Trainingsbeispiele entsprechend anzupassen.

Eine zweite Schwierigkeit von BACON besteht in dem Umgang mit verrauschten Daten. Beispielsweise ist die Ausführung des Konstanz-Bestimmungs-Operators in entscheidender Weise von der "annähernden" Gleichheit der Instanzen abhängig.

Drittens kann BACON nur relativ einfache Aufgaben zur Konzeptbildung unter Berücksichtigung nicht-numerischer Variablen behandeln. Beispielsweise kann BACON keine Konzepte entdecken, die eine interne Disjunktion (wie z. B. das Konzept "kleiner oder großer Kreis") enthalten /Cohen, Feigenbaum 17/.

4.1.5 BACON.x am Beispiel des dritten Kepplerschen Gesetzes

In diesem Beispiel soll BACON zur "Wiederentdeckung" des dritten Kepplerschen Gesetzes /Cohen, Feigenbaum 17/, nach dem die Periode p, die ein Planet für eine Umkreisung der Sonne benötigt und der Abstand d des Planeten zur Sonne in der folgenden Beziehung zueinander stehen:

$$d^3 / p^2 = k, \quad \text{für eine Konstante k.}$$

Zunächst stellt man BACON die folgenden Trainingsdaten zur Verfügung:

Instanz	Planet	Periode	Distanz
I_1	Merkur	1	1
I_2	Venus	8	4
I_3	Erde	27	9

BACON wird nun mitgeteilt, daß die Variablen p und d vom Planeten abhängen. Sobald sich ein paar Trainingsdaten angesammelt haben, werden diese von BACON dahingehend untersucht, ob einige der Verfeinerungs-Operatoren angewendet werden können. Da p und d ansteigende Werte annehmen und in keiner linearen Beziehung zueinander stehen, wird ein Operator, der den Term d / p erzeugt ausgewählt. Dieser Verfeinerungs-Operator wird ausgewählt, und die Trainingsdaten werden wie folgt umformuliert:

Instanz	Planet	Periode	Distanz	d / p
I_1	Merkur	1	1	1,0
I_2	Venus	8	4	0,5
I_3	Erde	27	9	0,33

Erneut überprüft BACON, ob einige Verfeinerungsoperatoren anwendbar sind. In diesem Fall kann der Produkt-Operator angewendet werden, da sich die Werte d und d / p gegensätzlich verändern; der Term (d / p) * d wird erzeugt. Damit können die Trainingsdaten erneut umformuliert werden:

Instanz	Planet	Periode	Distanz	d / p	d^2 / p
I_1	Merkur	1	1	1,0	1,0
I_2	Venus	8	4	0,5	2,0
I_3	Erde	27	9	0,33	3,0

In der dritten Iteration werden die umformulierten Trainigsdaten erneut im Hinblick auf die Anwendung einiger Verfeinerungs-Operatoren untersucht. Der Produkt-Operator ist anwendbar und der Term (d / p) * $(d^2 / p) = d^3 / p^2$ wird erzeugt. Damit ergibt sich als neue Menge umformulierter Trainingsdaten:

Instanz	Planet	Periode	Distanz	d / p	d^2 / p	d^3 / p^2
I_1	Merkur	1	1	1,0	1,0	1,0
I_2	Venus	8	4	0,5	2,0	1,0
I_3	Erde	27	9	0,33	3,0	1,0

Bei der Untersuchung dieser Daten wird der Konstanz-Erkennungs-Operator ausgewählt, und er erzeugt die Hypothese, daß der Term d^3 / p^2 konstant ist. BACON sammelt nun weitere Daten, um diese Hypothese zu bestätigen.

4.2 Datengesteuertes Lernen mehrerer Konzepte

In den beiden vorangegangenen Abschnitten wurden Algorithmen zum Lernen eines einzigen Konzeptes vorgestellt. Gelernt wurde eigentlich die Entscheidung, ob ein gegebenes Beispiel zu einem Konzept gehört oder nicht. Aus verschiedenen Gründen kann es jedoch nötig sein, daß mehrere Konzepte gelernt werden müssen; es kann beispielsweise vorkommen, daß von außen verschiedene Konzepte vorgegeben werden und die Klassifikation von Beispielen gemäß dieser Konzepte gelernt werden soll /Richter, Wendel 100/.

Im Gegensatz zu einem einzigen Konzept können bei mehreren Konzepten die folgenden beiden Fälle auftreten:

- entweder besitzen die Konzepte disjunkte Beispielmengen oder
- die Beispielmengen überlappen sich.

Ein wichtiges Anwendungsgebiet für das Lernen mehrerer Konzepte ist die Diagnostik, insbesondere die medizinische Diagnostik. Hier sind die Beispiele häufig in Form von Krankheits- oder Fehlersymptomen gegeben. Diesen Krankheits- oder Fehlersymptomen können mehrere Krankheiten zugrunde liegen; in der hier dargestellten Sicht entspricht dies mehreren Konzepten. Gerade bei mehreren Konzepten ist es zweckmäßig und leicht möglich, das Ergebnis des Lernprozesses in Form von Regeln darzustellen. Hierzu muß der Zusammenhang zwischen den gegebenen Beispielen und den gelernten Konzepten explizit gemacht werden /Richter, Wendel 100/.

Im folgenden wird der AQ-Algorithmus zum (induktiven) Lernen mehrerer Konzepte in einer allgemeinen Form wie bei /Richter, Wendel 100/ und /Michalski 77/ beschrieben.

4.2.1 Die Star-Methode und der AQ-Algorithmus

Diese allgemeine Methode zur Generierung struktureller Beschreibungen aus einem vom Lehrer vorklassifizierten Beispiel basiert auf dem Konzept eines "Sterns", der die Mengen alternativer, konsistenter Generalisierungen eines einzelnen Beispiels darstellt (vgl. /Michalski 77/). Der Stern für ein Beispiel entspricht im wesentlichen der Menge G aus der Versionenraummethode, dargestellt in einer symbolisierten Form eines Venndiagrammes (siehe Abblidung 8). In der Darstellung symbolisiert die schraffierte Menge den Durchschnitt der Beispiele für die Konzepte des Sterns.

Es sei N eine Menge von negativen Beispielen für ein Konzept und b ein positives Beispiel dafür, dann ist der **Stern** von b bzgl. N diejenige Menge G ("allgemeinste Konzeptbeschreibung"), die bei der Versionenraummethode bei der Präsentation von { b } und N entsteht. Diese Menge wird im folgenden mit G(b, N) bezeichnet.

Die Menge G(b, N) entspricht einer Disjunktion für das gesuchte Konzept: Die Grundidee des AQ-Algorithmus ist nun, die Mengen G(b, N) für jedes positive Beispiel b einzeln zu erzeugen. Damit transformiert der AQ-Algorithmus das Problem der Bildung von Diskrimi-

nationsregeln in eine Reihe von Problemen zum Lernen eines einzigen Konzeptes durch den Kandidaten-Eliminations-Algorithmus.

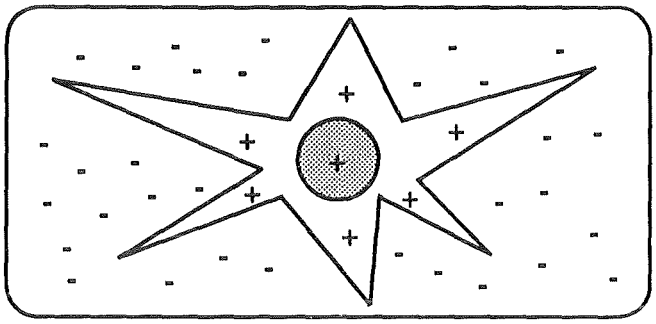

Abbildung 8

AQ-Basis-Algorithmus:

Gegeben seien zwei Mengen P, N positiver und negativer Beispiele für ein Konzept.

- Initialisiere die Menge K der Konzeptbeschreibungen mit K := ∅;
- Initialisiere die noch nicht behandelten positiven Beispiele P_n mit $P_n := P$;
- (*) Wähle ein beliebiges Beispiel $b \in P_n$;
- Bilde den Stern G(b, N);
- Wähle aus diesem Stern die "beste" Verallgemeinerung V aus und spezialisiere sie, wenn gewünscht;
- Setze $K := K \cup \{ V \}$ und $P_n := P_n \setminus$ Menge der von V erfaßten Beispiele;
- Wenn $P_n = \emptyset$, dann Stop; Sonst gehe zu (*);

Das Abbruchkriterium ist $P_n = \emptyset$. Die Lösung, d. h. die Konzeptbeschreibung, ist die Disjunktion bestehend aus den Elementen von K.

Man wählt also zu jedem positiven Beispiel b eine Verallgemeinerung V, die alle negativen Beispiele vermeidet; dabei achtet man darauf, daß ein solches V auch bereits andere positive Beispiele umfaßt, für die man das Vorgehen dann nicht noch einmal wiederholen muß. Der Begriff "beste" Verallgemeinerung ist bisher noch nicht festgelegt. Grundsätzlich kann dies auf der Grundlage verschiedener Optimalitätskriterien geschehen. In der Regel verwendet man syntaktische Kriterien wie die minimale Anzahl benutzter atomarer Relationen, die kürzeste Länge des Ausdrucks etc., wobei man sich zusätzlich von inhaltlichen (semantischen) Vorstellungen leiten lassen kann.

Häufig kann der Stern eines Beispiels sehr groß werden; es ist dann zweckmäßig, ihn in geeigneter Weise einzuschränken. Die kann durch ein Präferenzkriterium, beispielsweise durch eine Beschränkung der Anzahl der Elemente in G(b, N), geschehen.

Das Manko der ursprünglichen Version von AQ keine verrauschten Daten behandeln zu können, führte zur Entwicklung von CN2 /Clark, Niblett 18/. CN2 kombiniert die Fähigkeit und Effektivität von ID3 bei der Behandlung gestörter Daten mit der WENN-DANN-Regelform und der flexiblen Suchstrategie der AQ-Familie. Die Ausgabe von CN2 ist eine geordnete Menge von WENN-DANN Regeln, die auch als Entscheidungsliste bezeichnet wird und zur Klassifikation neuer Beispiele eingesetzt werden kann. Während der Konstruktion der Regeln verwendet CN2 eine heuristische Funktion zur Terminierung der Suche. Diese heuristische Funktion basiert auf einer Abschätzung über den Umfang der Störung in den vorgegebenen Daten /Clark, Niblett 18/.

4.3 Clusterverfahren

Bei realen Anwendungen stellt sich häufig das Problem, daß Abhängigkeiten zwischen rein numerisch verarbeitbaren Informationen (z. B. Meßwerten) erkannt und die numerisch verarbeitbaren Informationen anhand dieser Abhängigkeiten strukturiert (geclustert) werden müssen. Prinzipiell können zur Lösung derartiger Probleme die folgenden vier Verfahren herangezogen werden

- (numerische) Clustertechniken auf der Basis von Metriken (Abständen),
- statistische Clusterverfahren, d. h. Clusterung auf der Basis statistischer Verfahren in einem quasi-stationären Zustand,
- (inkrementelles) conceptual Clustering, d. h. Clusterung auf der Basis von Ähnlichkeiten und eines vorgegebenen "Konzeptrahmens" und
- neuronale Netze (vgl. Kapitel 6).

4.3.1 Clusterung auf der Basis von Abständen und statistische Clusterverfahren

Anhand der zu clusternden Objekte werden Attribute bestimmt, die für deren Charakterisierung als relevant eingeschätzt werden. Die Bestimmung der Cluster erfolgt dann auf der Basis der Werte der selektierten Attribute und eines vordefinierten Ähnlichkeitsmaßes. Ein solches Ähnlichkeitsmaß liefert jedoch nur dann sinnvolle Ergebnisse, wenn die ausgewählten Attribute für die Beschreibung der erkennbaren Objektähnlichkeiten auch relevant sind.

Das Ergebnis der Clusterung sind Cluster in Verbindung mit Informationen über numerische Ähnlichkeiten zwischen Objekten und Objektklassen. Da statistische Clusterverfahren keinerlei Beschreibung oder Erklärung der erzeugten Cluster liefern, bleibt diese Aufgabe dem Benutzer überlassen. Sie kann ihm auch durch einen sich an die eigentliche Clusterung anschließenden Analyseschritt nicht abgenommen werden. Um Cluster zu erzeugen, die einfachen Konzepten zugeordnet werden können, müssen in den Clustervorgang Konzepte eingebunden werden, die dazu dienen, Cluster als Ganzes zu charakterisieren. D. h. es müssen bereits während der Clusterung Gestaltkriterien (vgl. z. B. /Joerger 47/ oder /Keller, Weinberger 51/) berücksichtigt werden /Shapiro 112/.

Dieses Manko statistischer Clusterverfahren ist auf das zugehörige Ähnlichkeitsmaß zurückzuführen. Ein Ähnlichkeitsmaß behandelt typischerweise alle Attribute mit gleicher Wichtigkeit; d. h. es findet keine Unterscheidung zwischen relevanten und weniger relevanten Attributen statt. Jedoch selbst die Gewichtung der Attribute schafft keine Abhilfe, denn die klassischen Clusteransätze verfügen über keinerlei Mechanismen zur Auswahl und Auswertung von Attributen im Prozeß der Clustererzeugung; ebenso werden von ihnen auch keine neuen Attribute, die sich als adäquater für die Clusterung der Objekte und die Beschreibung

der Cluster, als die ursprünglich gewählten, herausstellen könnten, automatisch erzeugt /Shapiro 112/.

Beispiele für statistische Clusterverfahren sind: Regressionsanalyse, Varianzanalyse, Kovarianzanalyse, Faktorenanalyse, Clusteranalyse, Diskriminanzanalyse und die Korrelationsanalyse /Fensel, Studer 26/.

4.3.2 Begriffliche Ballung (konzeptuelle Ballungsanalyse, conceptual Clustering)

Bei der begrifflichen Ballung (conceptual clustering) handelt es sich wie auch bei der inkrementellen Konzeptbildung (Konzeptlernen) um Formen des Lernens aus Beobachtungen. Ein Unterschied zwischen Lernen aus Beispielen und der inkrementellen Konzeptbildung besteht darin, daß die resultierende Wissensstruktur im ersten Fall meistens "flach" ist, d. h. die zur Klassifizierung verwendeten Entscheidungsregeln sind unabhängig voneinander, während bei der begrifflichen Ballung und der inkrementellen Konzeptbildung eine hierarchische Beschreibung generiert wird; bei jeder Klassifizierungsregel wirken eine ganze Reihe von Entscheidungen zusammen (/Reimann, Bader, Klenner 99/).

Die Methoden der begrifflichen Ballung unterscheiden sich von den älteren Methoden der numerischen Taxonomierung dadurch, daß die Qualität der Clusterung nicht nur eine Funktion der einzelnen Objekte ist, sondern von Konzepten abhängt, die Objekt-Klassen und/oder die Abbildung zwischen Konzepten und den Klassen, die sie überdecken, beschreiben. Abgesehen von den Unterschieden in der Repräsentation und der Qualitätsbewertung bewerten alle Systeme zur begrifflichen Ballung die Klassenqualität aufgrund einer Zusammenfassung oder Konzept-Beschreibung der Klasse /Fisher 27/.

Die generelle Aufgabe besteht darin, beobachtete Instanzen oder Situationen zu klassifizieren und eine intensionale Beschreibung dieser Klassifikation zu erzeugen. Traditionelle Techniken aus dem Bereich der numerischen und statistischen Clusteranalyse (vgl. z. B. /Bortz 10/) liefern häufig inadäquate Ergebnisse, da die Instanzen den Klassen allein aufgrund eines a priori vorgegebenen numerischen Ähnlichkeitsmaßes (z. B. dem euklidischen Abstand) zugeordnet werden. Derartige Ähnlichkeitsmaße basieren einzig und allein auf dem Vergleich der Attributwerte der Instanzen und berücksichtigen keinerlei globale Eigenschaften oder Konzepte zur Charakterisierung der Instanzenklassen. Folglich können Klassen entstehen, die keine einfache Konzeptbeschreibung besitzen und daher schwierig sinnvoll zu interpretieren sind (/Michalski, Stepp 84/).

Die Aufgabe der **(konjunktiven) begrifflichen Ballung** kann wie folgt beschrieben werden (/Michalski, Stepp 84/):

Gegeben:

- Eine Menge von Instanzen (physikalisch oder abstrakt),
- eine Attributmenge zur Charakterisierung der Instanzen,
- Hintergrundwissen über Einschränkungen für den Problembereich, Eigenschaften der Attribute und ein Kriterium zur Beurteilung der Qualität der konstruierten Klassifikationen.

Gesucht:

- Cluster, die diese Instanzen in Kategorien gruppieren,
- eine hierarchische Anordnung der Instanzenklassen,
- jede Klasse soll durch ein einziges konjunktives Konzept beschrieben werden,
- Subklassen, als Nachfolger einer Elternklasse, sollen logisch disjunkte Beschreibungen besitzen und das gegebene Qualitätskriterium optimieren.

In /Michalski, Stepp 84/ wird ein Programm zur konjunktiven begrifflichen Ballung, CLUSTER/2 beschrieben. Da die gesamte Menge der Instanzen a priori vorgegeben werden muß, handelt es sich bei CLUSTER/2 um eine nicht-inkrementelle Methode. In diesem Zusammenhang stellt die Tatsache, daß die Variablen (Attribute), die die Instanzen beschreiben ebenfalls a priori vorgegeben werden müssen, ein weiteres Manko dar. Es ist daher möglich, daß andere Attribute existieren, die eine adäquatere Klassifikation zulassen würden.

4.3.3 Vergleich der dargestellten Clustertechniken - Resümée

(1) Statistische/Abhängigkeits-basierte Clusterverfahren:

- Die Bildung der Cluster wird wesentlich durch die Wahl des Ähnlichkeitsmaßes und der klassifizierenden Merkmale (Diskriminationskriterium) bestimmt.
- Die Verarbeitung der Merkmale (Attribute) erfolgt auf der Basis numerisch meßbarer Ähnlichkeit und statistischen Kriterien wie Mittel-/Erwartungswert, Varianz, Korrelationen etc..

  Bei der Einbindung statistischer Kriterien bleibt die Frage nach der Unabhän-

gigkeit der zugrundeliegenden Attribute häufig unbeantwortet; ein quasi-stationärer Zustand wird vorausgesetzt, die Größe des zugehörigen Zeitfensters wird a priori fest vorgegeben.

- Durch statistische Verfahren (wie Regressionsanalyse, Varianzanalyse, Kovarianzanalyse, etc.) können nur lineare Zusammenhänge zwischen einzelnen Meßgrößen bzw. zwischen einzelnen Meßwerten festgestellt werden.

- Es handelt sich um eine rein numerische Verarbeitung der vorgegebenen Objekte auf der Basis numerischer Attributausprägungen; Gestaltkriterien, d. h. Hintergrundwissen über die semantischen Beziehungen zwischen den Attributen der Objekte oder globale Konzepte die zur Charakterisierung von Objektkonfigurationen herangezogen werden könnten, werden nicht berücksichtigt /Michalski, Stepp 84/.

  Als Konsequenz aus dieser Tatsache besitzen die abgeleiteten Cluster häufig keine einfache konzeptuelle Beschreibung und sind daher schwierig zu interpretieren /Michalski, Stepp 84/.

- Die statistischen Clusterverfahren verfügen über keinerlei Mechanismen zur Auswahl und Auswertung von Attributen während des Prozesses der Clusterbildung. Weiterhin werden keine neuen (adäquateren) Attribute im Hinblick auf die Clusterung generiert.

- Einander widersprechende Beobachtungen werden, im Hinblick auf die Clusterbildung, miteinander verrechnet, d. h. sie beeinflussen die Größe und Lage der Cluster.

- Aufgrund des zumeist zugrundeliegenden linearen Ansatzes sind statistische Clusterverfahren nicht zur Behandlung von Datensätzen mit vielen Ausnahmefällen geeignet.

- Sie bieten nur eine geringe (keine) Transparenz der getroffenen Klassifikationen.

- Numerische Techniken berücksichtigen nicht die Art und Weise, wie der Mensch Objekte clustert /Michalski, Stepp 83/:

  (1) Untersuchungen zeigen, daß der Mensch dazu neigt, aus einer großen Anzahl von Attributen ein einziges oder wenige, sorgfältig formulierte Attribute auszuwählen und die Objekte auf der Grundlage dieser Attribute zu clustern.

  (2) Jedes Cluster enthält einander ähnliche Objekte; die auf den "wichtigen" Attributen einander ähnlich sind.

  (3) Beschreibungen derartiger Cluster können formal als logische Konjunktionen von Relationen auf diesen Attributen dargestellt werden.

(4) Von verschiedenen Clustern wird erwartet, daß sie Beschreibungen mit unterschiedlichen Werten auf den ausgewählten Attributen enthalten.

(5) Der Mensch neigt dazu, Objekte in Kategorien zu clustern, die durch disjunkte konjunktive Konzepte charakterisiert sind.

(2) Begriffliche Ballung (conceptual Clustering):

- Durch Vorgabe von Hintergrundwissen oder globalen Konzepten werden für die zu clusternden Objekte Gestaltkriterien mit einbezogen. Eine Anordnung von Objekten bildet nur dann ein Cluster, wenn diese durch ein einfaches Konzept aus einer vorgegebenen Konzeptklasse beschrieben werden kann /Michalski, Stepp 84/.

  Durch die Verlagerung des Hintergrundwissens / der globalen Konzepte in ein Ähnlichkeitsmaß wird der Übergang von conceptual Clustering zu statistischen Clusterverfahren fließend /Hanson 36/.

- Die Organisation des Wissens in einer Konzepthierarchie geht insofern über die Anordnung in is-a-Hierarchien, wie sie beispielsweise von ID3 aufgebaut werden hinaus, als daß jeder Knoten der Konzept-Hierarchie zusätzlich zum repräsentierten Konzept noch eine intensionale Beschreibung dieses Konzeptes enthält.

- Systeme zur begrifflichen Ballung müssen die vorgegebenen Instanzen ohne die Unterstützung eines Lehrers clustern, d. h. sie müssen nicht "nur" entscheiden, welche Instanzen jede Klasse enthalten soll sondern auch die Anzahl dieser Klassen festlegen. Dies ist das wesentliche Merkmal um die Arbeiten im Bereich der begrifflichen Ballung (der Konzeptbildung) und des Lernens konjunktiver Konzepte aus Beispielen zu trennen /Gennari, Langley, Fisher 32/.

- Die gegebenen Objektattribute werden auf der symbolischen Ebene verarbeitet; Metriken werden entsprechend definiert.

- Die Verfahren des conceptual Clustering berücksichtigen keinen empirischen "Verstärkungseffekt" durch wiederholtes Auftreten "gleichartiger" Beobachtungen.

- Einander widersprechende Beobachtungen werden, im Hinblick auf die Clusterbildung, gleichbehandelt.

4.4 Inkrementelle Konzeptbildung

Im folgenden wird ein Spezialfall der oben beschriebenen begrifflichen Ballung, die inkrementelle Konzeptbildung, beschrieben. Die inkrementelle Konzeptbildung unterscheidet sich von der begrifflichen Ballung dadurch, daß die Instanzen nicht auf einmal, sondern inkrementell dargeboten werden, d. h. es kann nicht nach Lern- und Ausführungsphase unterschieden werden, Lernen und Ausführung werden integriert. Interpretiert man das Problem der Konzeptbildung als ein Suchproblem im Raum der Objekt-Cluster mit einer übergeordneten Suche im Raum der Konzepte, gesteuert durch eine Suche im Raum der Konzepthierarchien, so wird bei der Konzeptbildung Lernen durch das Suchverfahren des inkrementellen Hill-Climbing realisiert.

Hill-Climbing ist eine klassische Suchmethode der Künstlichen Intelligenz /Winston 120/, bei der in jedem (Such-) Schritt alle möglichen Instantiierungen einer Ebene durchgeführt und mittels einer Evaluierungsfunktion bewertet werden. Die Pfade der Knoten der Ebene mit der besten Bewertung (evtl. mehrere) werden auf die gleiche Weise weiterverfolgt, bis keine Steiergung der Bewertung mehr erfolgt. Der wesentliche Vorteil des Hill-Climbing besteht in dessen geringen Speicheranforderungen. Dem stehen die bekannten Probleme lokaler Extremwerte und die Einschränkung auf eine (Tiefen-) Vorwärtssuche (aufgrund eines fehlenden expliziten Backtracking) gegenüber.

Die inkrementelle Natur der Aufgabe bedingt, daß sich der Teilraum des Raumes der Konzepthierarchien, auf dem die Suchprozedur des Hill-Climbing operiert, ständig verändert (Metapher: "Ameise auf einem Kieshaufen"); somit sind die speicherlimitierten Methoden des inkrementellen Lernens durch die Reihenfolge der Präsentation der Instanzen beeinfußbar. Dieser Abhängigkeit kann man begegnen, indem man die entsprechenden Hill-Climbing-Verfahren um die Möglichkeit der "Rückwärtssuche" erweitert. Durch eine solches, bidirektionales Verfahren können nicht nur neue Konzepte gebildet sondern auch bestehende Konzepte gelöscht werden. Damit können Hill-Climbing Verfahren den Effekt eines Backtracking erreichen; sie benötigen jedoch nicht den für ein explizites Backtracking notwendigen Speicherplatz.

Generell kann man die Aufgabe der **inkrementellen Konzeptbildung** wie folgt beschreiben (/Gennari, Langley, Fisher 32/):

Gegeben:

- Sequentiell vorgegebene Instanzen und deren Beschreibung.

Gesucht:

- Cluster, die diese Instanzen in Kategorien gruppieren,
- eine (intensionale) Beschreibung jeder einzelnen Kategorie, die die Instanzen zusammenfaßt,
- eine hierarchische Organisation dieser Kategorien.

Bei der inkrementellen Konzeptbildung geht es darum, eine Menge nacheinander vorgegebener Elemente (Instanzen, Ereignisse) anhand ihrer Ähnlichkeit zu Gruppen zusammenzufügen und für jede dieser Gruppen eine symbolische Beschreibung zu generieren. Eine inkrementelle Vorgehensweise ist eine wesentliche Eigenschaft von Systemen, die unter "real-world"- Bedingungen angewendet werden sollen /Fensel, Studer 26/. Zunächst werden einige Verfahren zur inkrementellen Konzeptbildung, deren Gemeinsamkeiten und wesentliche Unterschiede dargestellt. Im Abschnitt 4.4.2 wird das Programm EPAM, ein klassisches Programm zur Bildung von Kategorien, wie in /Reimann, Bader, Klenner 99/ beschrieben.

Das Ziel der Konzeptbildung besteht in dem Versuch eine Hilfe zum besseren Verständnis der Welt zu geben und Vorhersagen über ihr zukünftiges Verhalten zu ermöglichen.

4.4.1 EPAM, UNIMEM, COBWEB und CLASSIT - im Überblick

Seit der Entwicklung von EPAM wurden einige inkrementelle maschinelle Lernsysteme entwickelt, die auf dem Prinzip des Diskriminationsnetzes aufbauen, z. B. UNIMEM (/Lebowitz 68/), COBWEB (/Fisher 27/) oder CLASSIT (/Gennari, Langley, Fisher 32/). Bei allen diesen inkrementellen Verfahren zum Konzeptlernen können fünf gemeinsame Merkmale identifiziert werden /Reimann, Bader, Klenner 99/:

(1) Wissen wird in Form von **Konzept-Hierarchien** repräsentiert. Jeder Knoten in einer solchen Konzepthierarchie steht für ein Konzept und enthält dessen intensionale Definition. Die Knoten sind entsprechend ihres Allgemeinheitsgrades hierarchisch geordnet, mit spezifischeren Konzepten auf tieferen Ebenen der Hierarchie.

(2) Instanzen (Objekte, Ereignisse) werden **top-down**, d. h. beginnend mit dem generellsten Knoten der Konzepthierarchie und schrittweise zu spezifischeren Konzepten übergehend, klassifiziert. Im allgemeinen Fall muß die Auswahl des nächstspezifischen Knotens nicht vom Testen eines einzelnen Merkmals abhängen, es können auch mehrere Knoten gleichzeitig verfolgt werden. Schließlich muß die Klassifizierung nicht an einem terminalen Knoten (mit dem spezifischsten Konzept oder gar einer Instanz) enden, sondern kann auf höherer Ebene der Konzepthierarchie stoppen. Die Qualität der Klassifikation kann danach beurteilt werden, wie gut sie es erlaubt, Vorhersagen über noch nicht gesehene Merkmale der zu klassifizierenden Instanz zu machen.

(3) **Nicht-überwachtes Lernen** in dem Sinne, daß das lernende System selbst entscheiden muß, wie hereinkommende Instanzen zu gruppieren sind. Die Instanzen werden nicht vorklassifiziert (im Gegensatz zu ID3/CLS, der Versionenraum-Methode, AQ, etc.). D. h. das lernende System muß selbständig darüber entscheiden, wieviele Kategorien es bilden soll und welche Instanzen welcher Kategorie zugeordnet werden müssen.

(4) **Integration von Lernen und Performanz.** Beim inkrementellen Konzepterwerb ist die Performanz des Systems (z. B. Klassifikation) eng verknüpft mit dem Lernen (z. B. Umstrukturierung des Diskriminationsnetzes durch Vertiefung). Diese Kopp-

lung von Performanz und Lernen wird durch die Bedingung des inkrementellen Datenanfalls und des inkrementellen Wissenserwerbs quasi erzwungen. Die Wissensrepräsentationsstruktur ist von Lernschritt zu Lernschritt nur "lokalen" Veränderungen unterworfen und es findet keine extensive Neuverarbeitung bereits verarbeiteter Instanzen statt.

(5) Lernen als **inkrementelles Hill-Climbing**. Wenn man inkrementellen Konzepterwerb als Suche durch einen Raum von Konzepthierarchien versteht, dann ist Hill-Climbing die entsprechende Kontrollprozedur für die Suche. Hill-Climbing ist eine klassische Suchmethode der Künstlichen Intelligenz /Winston 120/. Die Vorgehensweise besteht darin, daß ausgehend von einem Zustand Z_t des Problemraumes alle ausführbaren Operatoren ausgeführt und die resultierenden Zustände $Z_{I,\,t+1}$ bewertet werden. Dann wird der relativ beste Zustand $Z_{i,\,t+1}$ ausgewählt. Dieser Prozeß iteriert solange, bis keine weitere Verbesserung mehr erreichbar ist.

Diese Strategie besitzt sämtliche Probleme, die mit lokalen Extremwerten verbunden sind. Sie hat jedoch auch einen großen Vorteil: Sie erfordert wenig Gedächtniskapazität zur Speicherung alter Zustände. Angewandt auf den inkrementellen Konzepterwerb wird zumeist eine modifizierte Version von Hill-Climbing verwendet, in der es keine explizite Evaluierungsfunktion zur Bewertung der Zustände, dafür aber einen "intelligenten" Zustandsgenerator gibt. Die "Intelligenz" des Zustandsgenerators besteht darin, daß es die mit einer neuen Instanz verbundenen Informationen mehr oder weniger optimal ausnutzt, um die Konzepthierarchie zu modifizieren.

Inkrementelles Hill-Climbing wird insbesondere dann problematisch, wenn wie bei Problemen der realen Welt notwendig, Veränderungen in der Umwelt, z. B. der Wechsel der Jahreszeiten, mitverfolgt werden müssen.

Die Entwicklung von CLASSIT wurde sehr stark von COBWEB beeinflußt. Der wesentliche Unterschied zwischen diesen beiden Verfahren besteht in der Repräsentation der Instanzen, der Repräsentation der Konzepte und der Evaluierungsfunktion zur Bewertung von Konzeptzuordnungen. CLASSIT verwendet die gleichen Basisoperationen und die gleiche Kontrollstrategie wie COBWEB. Ebenso wie COBWEB werden bei CLASSIT alle bekannten Attribute in jeder Konzeptbeschreibung, unabhängig von ihrer Vorhersagekraft bzw. Vorhersagbarkeit, gespeichert.

Im folgenden werden die wesentliche Unterschiede zwischen diesen vier inkrementellen Konzeptbildungsverfahren angegeben:

(1) Form der Repräsentation:

- EPAM und COBWEB können nur Attribute mit nominalen (symbolischen) Wertebereichen,
- UNIMEM kann Attribute mit nominalen (symbolischen) und numerischen Wertebereichen,
- CLASSIT kann nur Attribute mit reellen Wertebereichen

verarbeiten.

(2) Speicherung der Instanzen und Konzepte:

- EPAM:
  Jede Instanz wird als Konjunktion von Attribut-Wert-Paaren in Verbindung mit einer optional geordneten Liste von Komponenten-Objekten repräsentiert. Jede Komponente wiederum wird als Konjunktion von Attribut-Wert-Paaren mit ihren eigenen optionalen Komponenten etc. beschrieben. Damit kann EPAM Objekte mit Komponenten beschreiben, dies ist mit keinem der anderen Verfahren möglich.

- UNIMEM:
  Die Instanzen werden als Konjunktion von Merkmals- oder Attribut-Wert-Paaren repräsentiert. Jeder Konzept-Knoten wird durch Attribute, Werte und zugeordnete Gewichte beschrieben.

- COBWEB:
  Jede Instanz besteht aus einer Menge von Attribut-Wert-Paaren. COBWEB verwendet eine ähnliche Konzeptdarstellung wie UNIMEM. Allerdings speichert COBWEB die Wahrscheinlichkeit für das Auftreten eines jeden Konzeptes.

- CLASSIT:
  Für jedes in einem Konzept auftretende Attribut wird eine Normalverteilung gespeichert; diese wird durch einen Durchschnittswert und eine Standardabweichung ausgedrückt.

(3) Konzepthierarchie:

- Bei COBWEB und CLASSIT werden die Konzepte in einer echten is-a-Hierarchie angeordnet; EPAM und UNIMEM verwenden Indexierungstechniken zur Anordnung der Konzepte.

- EPAM, COBWEB und CLASSIT bilden nur disjunkte Konzepte. In UNIMEM können auch nicht-disjunkte Konzepte gebildet werden.

(4) Speicherung der Instanzen:

- Bei UNIMEM und CLASSIT müssen, im Gegensatz zu EPAM und COBWEB, Instanzen nicht unbedingt unter terminalen Konzepten gespeichert werden.

- EPAM, COBWEB und CLASSIT behalten alle jemals aufgetretenen Instanzen bei, während UNIMEM in der Lage ist, wertlose Instanzen, Konzepte und sogar Konzeptattribute wieder zu löschen.

4.4.2 EPAM (Elementary Perceiver and Memorizer)

EPAM wurde von 1956 bis 1964 von Edward Feigenbaum und Herbert Simon entwickelt und war ursprünglich als ein Gedächtnismodell konzipiert.

Beispiel: Es soll ein Diskriminationsnetz für Instanzen aufgebaut werden, die mit Hilfe der drei Attribute Farbe, Anzahl der Kerne und Anzahl der Fäden beschrieben werden.

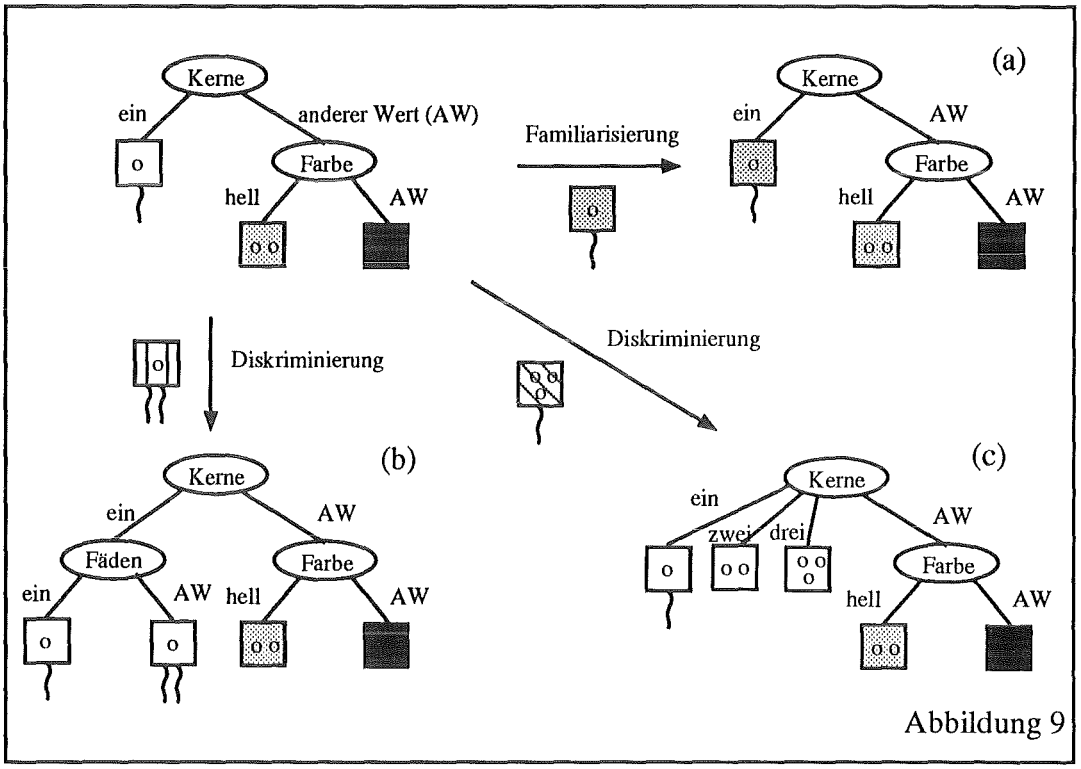

Abbildung 9

Der in der Abbildung oben links gezeigte Wurzelknoten enthält einen Test der Kernanzahl mit den zwei möglichen Resultaten KERNE=EINER oder KERNE=AW (anderer Wert). Der an der KERNE=EINER-Kante liegende Knoten ist terminal und enthält das Image KERNE=EINER & FÄDEN=EINER. Über die Farbe wird nichts ausgesagt. Der rechte Knoten ist nicht terminal und enthält einen Test auf die Farbe. Die ihm folgenden Knoten sind terminale mit den Images FARBE=HELL & KERNE=ZWEI bzw. FARBE=DUNKEL (/Reimann, Bader, Klenner 99/ und /Gennari, Langley, Fisher 32/).

Der Wissenserwerb in EPAM besteht im Aufbau eines **Diskriminationsnetzes**. Ein solches Diskriminationsnetz besteht aus Knoten und Kanten. Jeder nicht-terminale Knoten entspricht einem Test, und jede von einem solchen Knoten ausgehende Kante entspricht einem möglichen Testergebnis. Der mit einem Knoten assoziierte Test überprüft den Wert eines Attributes. Eine spezielle Kante ist die "Andere-Wert-Kante", sie erlaubt es, ein Testergebnis zu formulieren, ohne alle möglichen Testresultate aufführen (oder kennen) zu müssen. Die terminalen Knoten enthalten eine Teilmenge von Attributwerten, die für diejenigen Objekte gelten sollen, welche den jeweiligen terminalen Knoten zugeordnet werden. Der Inhalt dieser Knoten wird von Feigenbaum und Simon **Images** genannt.

EPAM kommt seiner Aufgabe dadurch nach, daß es neue Instanzen terminalen Knoten zuordnet. Dabei beginnt es am Wurzelknoten des Diskriminationsnetzes, führt den dort spezifizierten Attributtest durch und folgt der entsprechenden Ergebniskante zum nächsten Testknoten. Wenn ein terminaler Knoten erreicht ist, ist die neue Instanz klassifiziert. Die im Image stehende Beschreibung kann nun auch dazu benutzt werden, Voraussagen über nicht bereits während des Testens betrachtete Attributwerte zu machen.

Nachdem die neue Instanz einem terminalen Knoten zugeordnet wurde, wird einer von zwei möglichen Lernmechanismen angestoßen. Die beiden Lernmechanismen beruhen auf dem Vergleich zwischen Attributwerten der neuen Instanzen und Attributwerten des in Verbindung mit dem terminalen Knoten gespeicherten Images. Der eine Lernmechanismus, die **Familiarisierung** wird aktiviert, wenn alle Attributwerte des Images mit denen der neuen Instanz identisch sind, aber die neue Instanz zusätzliche Attributwerte enthält, die nicht schon im Image spezifiziert sind. Diese neuen Werte werden dem Image hinzugefügt.

Ad Beispiel: In das in Abbildung 9 dargestellte Diskriminationsnetz soll die neue Instanz FARBE=DUNKEL & KERNE=EINER & FÄDEN=EINER hinzugefügt werden. Hierzu wird der linke terminale Knoten von KERNE=EINER & FÄDEN=EINER spezialisiert zu KERNE=EINER & FÄDEN=EINER & FARBE=DUNKEL (a).

Der Lernmechanismus der Familiarisierung führt also zu einer schrittweisen Anreicherung der Image-Beschreibung.

Der zweite Lernmechanismus, die **Diskrimination** wird angestoßen, wenn Image und neue Instanz in einem oder mehreren Attributen nicht übereinstimmen. Um den ersten Knoten zu finden, hinsichtlich dessen Image und die Instanz unterschiedliche Werte haben, wird die Instanz ein zweites Mal durch das Diskriminationsnetz sortiert. Dieser Testknoten wird entsprechend modifiziert, um die festgestellten Unterschiede zwischen Image und Instanz zu erfassen. Auf diese Art und Weise wird das Diskriminationsnetz allmählich breiter.

Ad Beispiel: In das in Abbildung 9 dargestellte Diskriminationsnetz soll die neue Instanz KERNE=DREI & FARBE=HELL & FÄDEN=EINER hinzugefügt werden. Diese Instanz wird zunächst dem linken Knoten des rechten Pfades zugeordnet, der das Image KERNE=ZWEI & FARBE=HELL enthält. Der Unterschied im Attribut KERNE führt zu der Suche nach einem Test im Diskriminationsnetz, bzgl. dessen sich das Image und die Instanz zum ersten Mal unterscheiden. Das ist hier beim Wurzelknoten der Fall. Aus diesem Grund generiert EPAM nun zwei neue Kanten zu diesem Knoten, die den Attributwerten im Image und Instanz entsprechen (c).

Wird kein nicht-terminaler Knoten gefunden, hinsichtlich dessen sich Image und Instanz unterscheiden, so endet die Instanz wieder am ursprünglichen terminalen Knoten. In diesem Fall wird ein neuer Testknoten generiert, der den Unterschied zwischen Image und Instanz

erfaßt. Von diesem neuen Testknoten gehen zwei Kanten aus, eine mit dem Wert, den das Image bezüglich des Attributes hat und eine Anderer-Wert-Kante. Auf diese Art und Weise wird das Diskriminationsnetz allmählich tiefer.

Ad Beispiel: In das in Abbildung 9 dargestellte Diskriminationsnetz soll die neue Instanz KERNE=EINER & FARBE=DUNKEL & FÄDEN=ZWEI eingefügt werden. Dies führt zu dem neuen Test auf Fäden als Resultat der Diskriminierung des linken Knotens in (b) mit dem Image KERNE=EINER & FÄDEN=ZWEI.

Damit verfügt EPAM über drei Operatoren zur Gruppierung und Konzeptbeschreibung:

- Hinzufügen neuer Merkmale zu einem Image durch Familiarisierung,
- Generieren neuer disjunktiver Kanten durch Diskrimination,
- Generieren neuer Tests durch Diskrimination.

Diese Operatoren zusammen mit einem anfänglichen Diskriminationsnetz und den Instanzen spannen den Suchraum für EPAM auf. Die Anwendung der Lernoperatoren ist in EPAM nur sehr eingeschränkt kontrollierbar. Durch den Zustand des Diskriminationsnetzes und die neue Instanz ist die Operatoranwendung zum überwiegenden Teil deterministisch bestimmt; der Zustand des Diskriminationsnetzes und die neue Instanz darüber entscheiden, ob es zur Diskriminierung oder Familiarisierung kommt. Eine gewisse Freiheit (und damit die Möglichkeit für Kontrollentscheidungen) ergibt sich lediglich bei der Frage, welches Attribut während der Vertiefungsdiskrimination als neue Testgröße genommen werden soll. Die meisten Versionen von EPAM benutzen in beiden Fällen eine fixierte Sequenz von Attributen, ein ebenfalls deterministisches Entscheidungskriterium.

Schwächen von EPAM (/Reimann, Bader, Klenner 99/):

- Intensionale Beschreibungen von Konzepten tauchen erst in den terminalen Knoten auf.
- EPAM ist ungeeignet zum Erkennen unvollständig beschriebener Instanzen.
  Bei fehlenden Attributen müssen die Teilbäume vollständig abgearbeitet werden, Vorhersagen bleiben auf die Blätter beschränkt.
- EPAM geht davon aus, daß die Grenzen zwischen verschiedenen Konzepten scharf sind; d. h. jede Instanz kann nur einem Konzept zugeordnet werden. In diesem Sinne werden nur binäre Entscheidungen über die Zugehörigkeit einer Instanz zu einem Konzept getroffen.
- Ausnahmen, z. B. Pinguin als Vogel, der nicht fliegen kann, können ohne eine Änderung des Identifikationsmechanismus nicht abgespeichert werden, da die Attribute in den Tests konjunktiv verknüpft sind.
- Die Attributauswahl beim Lernen erscheint etwas ad hoc.

4.5 EBL (Explanation-Based-Learning)

In den bisher dargestellten Methoden zum Maschinellen Lernen bestand die Aufgabe darin, aufgrund einer gegebenen (oder inkrementell anfallenden) Menge von (Lern-) Beispielen allgemeine Beschreibungen (Generalisierungen) zu erzeugen, ohne dabei umfangreiches Domain-Wissen zu verwenden. Hierzu wurden die wesentlichen gemeinsamen Merkmale dieser Beispiele oder Beobachtungen identifiziert und durch Generalisierung zur Formulierung einer Konzeptdefinition verwendet. Da diese Methoden in entscheidender Weise auf erkennbare Gemeinsamkeiten innerhalb der Beispielmenge angewiesen sind, werden sie auch als **ähnlichkeits-basierte-Methoden** (similarity-based-methods) bezeichnet (/Ellman 25/). Die analytische Methode des **erklärungs-basierten-Lernens (EBL)** basiert nur auf einem einzigen Beispiel, um eine allgemeine Beschreibung zu erzeugen, verwendet jedoch umfangreiches, vorgebenenes Domain-Wissen. Im Gegensatz hierzu sind ähnlichkeitsbasierte-, empirische Lernmethoden ungeeignet, um aufgrund eines einzigen Beispieles zu lernen. Gewöhnlich benötigen sie mehrere Instanzen des zu lernenden Konzeptes.

Der Hauptunterschied zwischen ähnlichkeits- und erklärungs-basierten-Methoden besteht darin, daß erstere auf einen induktiven **Bias** zur Steuerung der Generalisierung angewiesen sind, während letztere auf ihr Domain-Wissen bauen (/Mitchell, Keller, Kedar-Cabelli 88/). Unter Bias versteht man jedes Kriterium, das von einem Algorithmus zum Konzeptlernen verwendet wird, um unter Alternativen, zu den beobachteten Trainingsinstanzen konsistenten, Generalisierungen, auszuwählen (/Ellman 25/). Ein solches Kriterium besteht beispielsweise darin, eine im Hinblick auf ihren Anwendungsbereich speziellere Formel (Regel) einer allgemeineren Formel (Regel) vorzuziehen. Problematisch ist jedoch, daß dieses Kriterium, angewandt auf alle möglichen Generalisierungen, nicht notwendigerweise zu einer eindeutigen Lösung führt /Helft 40/.

Eine Motivation für erklärungs-basiertes Lernen stammt aus dem Bereich des menschlichen Lernens. Menschen sind häufig in der Lage eine allgemeine Regel oder ein allgemeines Konzept zu erzeugen, nachdem sie nur eine einzige Instanz dieses Konzeptes beobachtet haben. In Schulbüchern ist dieser Typ des Lernens häufig zu finden. Beispielsweise könnte ein Mathematik Schulbuch ein Beispiel zur Bildung der Ableitung eines Polynoms dritten Grades enthalten, um danach dem Schüler die Aufgabe zu stellen, ein Polynom fünften Grades abzuleiten.

Die Forschungen im Bereich des erklärungs-basierten-Lernens können in die folgenden vier Richtungen unterteilt werden (/Ellman 25/):

- **Begründete Generalisierung** (justified generalization):
  Die Methode der begründeten Generalisierung wird in Verfahren zur Bildung erklärungs-basierter-Generalisierungen (**explanation-based-generalization (EBG)**) verwendet.
  Die begründete Generalisierung ist eine logisch abgesicherte Methode zur Generalisierung aufgrund von Beispielen. Zu gegebenem Hintergrundwissen HGW und einer Menge von Trainingsbeispielen M findet die begründete Generalisierung ein Konzept K das alle positiven Beispiele umfaßt und alle negativen ausschließt. Das gelernte Konzept K ist dabei eine logische Folgerung aus dem Hintergrundwissen HGW und der Menge der Trainingsbeispiele M.

Beispiele realer Systeme: - GENESIS (DeJong und Mooney),
- LEX-II (Mitchell und Utgoff).

Anmerkung: Das System GENESIS bezeichnet man auch als **Lehr-und-Lernsystem** (Learning Apprentice System, (LAS)). Ein Lehr-und-Lernsystem dient als interaktiver, wissensbasierter Berater, der zu Beginn mit einer Domain Theorie ausgestattet ist und die Fähigkeit besitzt, neues Problemlösungswissen zu assimilieren, indem er die Problemlösungsschritte, die die Benutzer im Rahmen der normalen Nutzung in das System einbringen, beobachtet und analysiert /Tecuci, Kodratoff 117/.

- **Chunking**:
  Im Rahmen des erklärungsbasierten Lernens ist Chunking ein Prozeß der Compilierung einer linearen oder baum-orientiert strukturierten Folge von Operatoren in einen einzelnen (Makro-) Operator, so daß der (Makro-) Operator die gleiche Auswirkung wie die gesamte ursprüngliche Operatorsequenz hat.

  Beispiele realer Systeme: - SOAR (Laird, Newell und Rosenbloom),
  - STRIPS (Fikes, Hart und Nilsson).

- **Operationalisierung** (operationalization):
  Operationalisierung bezeichnet den Prozeß der Transformation eines nicht-operationalen Ausdruckes in einen operationalen. Der ursprüngliche nicht-operationale Ausdruck kann dabei eine Menge von Anweisungen oder ein Konzept sein.

  Der ursprüngliche, nicht-operationale Ausdruck heißt nicht-operational bzgl. eines Agenten, wenn er nicht in Form von Daten und Aktionen dargestellt ist, auf die der Agent zugreifen kann. Ein Operationalisierungsprogramm behandelt die Aufgabe, den ursprünglichen Ausdruck, unter Verwendung von Daten und Aktionen über die der Agent verfügt, umzuformulieren.

  Beispiele realer Systeme: - FOO und BAR (Mostow),
  - LEXCOP und MetaLEX (Keller).

- **Begründete Analogie** (justified analogy):
  Eine logisch abgesichertes Verfahren zum analogen Schlußfolgern: Gegeben sei Hintergrundwissen HGW, ein analoges Beispiel X und ein Zielbeispiel Y. Gesucht ist ein Merkmal F, sodaß F(X) gilt, und man folgert daraus, daß auch F(Y) gilt. Die Schlußfolgerung F(Y) muß dabei eine logische Folgerung aus F(X) und dem Hintergrundwissen HGW sein.

  Beispiele realer Systeme: - ANALOGY (Winston),
  - Derivational Analogy (Carbonell).

Im wesentlichen liegen die Unterschiede zwischen diesen vier Kategorien in ihrer Interpretation.

4.5.1 EBG (Explanation-Based-Generalization)

Die Kernidee bei der erklärungs-basierten-Generalisierung besteht darin, daß es auf der Grundlage eines einzigen positiven Trainingsbeispiels möglich ist, eine begründete Generalisierung zu bilden, vorausgesetzt das lernende System verfügt über einige Erklärungsfähigkeiten. Der Generalisierungsprozeß besteht zumeist aus zwei Schritten (/Mitchell, Keller, Kedar-Cabelli 88/):

(1) Erklärung des Beispiels (deduktiver Beweis):
Erzeuge eine Erklärung unter Verwendung von Ausdrücken der Domain-Theorie, die beweist, daß das Trainingsbeispiel die Definition des Zielkonzeptes erfüllt. Hierbei werden die relevanten Merkmale des Beispiels von den irrelevanten getrennt.

- Bei der Konstruktion dieser Erklärung muß beachtet werden, daß jeder Ast der Erklärungsstruktur mit einem Ausdruck endet, der das Operationalisierungs-Kriterium erfüllt.

(2) Analyse der Erklärung, um das Beispiel zu generalisieren (induktive Generalisierung):

- Bestimme eine Menge hinreichender Bedingungen, die der Erklärungsstruktur genügen und stelle sie in Termen dar, die das festgelegte Operationalisierungs-Kriterium erfüllen.

Das Problem der erklärungs-basierten-Generalisierung kann wie folgt beschrieben werden (/Mitchell, Keller, Kedar-Cabelli 88/):

Erklärungs-basierten-Generalisierung:

Gegeben:

- Ziel-Konzept
  Eine Konzeptdefinition, die das zu lernende Konzept beschreibt und das Operationalisierungs-Kriterium nicht erfüllt.

- Trainingsbeispiel
  Ein Beispiel des Ziel-Konzeptes.

- Domain-Theorie
  Um zu erklären, in welcher Weise das Trainingsbeispiel eine Instanz des Ziel-Konzeptes ist, benötigt man eine Menge von Regeln und Fakten.

- Operationalisierungs-Kriterium
  Eine Bedingung und die (syntaktische) Form der zu lernenden Konzeptdefinition.

Gesucht:

- Eine weitere Konzept-Definition durch Verallgemeinerung des Trainingsbeispiels als hinreichende Bedingung für das Ziel-Konzept (d. h. das Zielkonzept wird durch sie logisch impliziert) unter Berücksichtigung des Operationalisierungs-Kriteriums.

Bemerkenswert ist die Wichtigkeit des Operationalisierungs-Kriteriums für die erklärungs-basierte-Generalisierung; würde man kein Operationalisierungs-Kriterium angeben, so wäre die ursprüngliche Konzept-Definition (Ziel-Konzept) auch immer eine korrekte Konzeptdefinition für die Ausgabe des Lernvorganges und es wäre nichts zu lernen! Durch das Operationalisierungs-Kriterium wird also nicht nur gefordert, daß die zu lernende Konzeptdefinition korrekt sondern auch in einer verwertbaren Form ist.

4.5.2 EBG an einem Beispiel

Die Vorgehensweise bei der erklärungs-basierten-Generalisierung wird im folgenden an dem häufig zitierten SAVE-TO-STACK-Beispiel (/Richter, Wendel 100/, /Mitchell, Keller, Kedar-Cabelli 88/, /Kodratoff 56/) erläutert.

Gegeben:

- Ziel-Konzept:
  Paare von Objekten < x, y >, so daß SAVE-TO-STACK (x, y) gilt, wobei
  SAVE-TO-STACK (x, y) $\Leftrightarrow$ NOT (FRAGILE (y)) $\vee$ LIGHTER (x, y).

- Trainingsbeispiel:
  ON (Object_1, Objekt_2),
  ISA (Object_1, Box),
  ISA (Object_2, Endtable),
  COLOR (Object_1, Red),
  COLOR (Object_2, Blue),
  VOLUME (Object_1, 1),
  DENSITY (Object_1, 1).

- Domain-Theorie:
  VOLUME(p1, v1) $\wedge$ DENSITY(p1, d1) $\rightarrow$ WEIGHT(p1, v1 * d1),
  WEIGHT(p1, w1) $\wedge$ WEIGHT(p2, w2) $\wedge$ LESS(w1, w2) $\rightarrow$ LIGHTER(p1, p2),
  ISA(p2, Endtable) $\rightarrow$ WEIGHT(p2, 5) (Default-Wert),
  LESS(1, 5).

- Operationalisierungs-Kriterium:
  Die Konzeptdefinition muß nur in Termen mit Prädikaten aus den vorgegebenen Beispielen (z. B. VOLUME, COLOR, DENITY) oder anderer, leicht auszuwertenden Prädikaten der Domain-Theorie (z. B. LESS) erfolgen.

Gesucht:

- Gesucht ist eine für SAVE-TO-STACK (x, y) hinreichende Konzeptdefinition unter Berücksichtigung des Operationalisierungs-Kriteriums.

Das Trainingsbeispiel besteht also aus zwei Objekten, einer roten Box (Objekt 1) und einem blauen Endtable (Objekt 2), wobei die Box auf dem Endtable steht und das Volumen und die Dichte 1 hat.

Die zugehörige Domain-Theorie sagt aus, daß sich das Gewicht eines gegebenen Objektes durch das Produkt von dessen Volumen und Dichte berechnen läßt, daß ein Objekt leichter als ein Zweites ist, wenn es ein geringeres Gewicht hat, daß ein Endtable ein (Default-) Gewicht von 5 hat und daß 1 kleiner als 5 ist.

4.5.2.1 EBG-Erklärung des Beispiels

SAVE-TO-STACK (Object_1, Object_2) muß mit Hilfe der Domain-Theorie bewiesen werden.

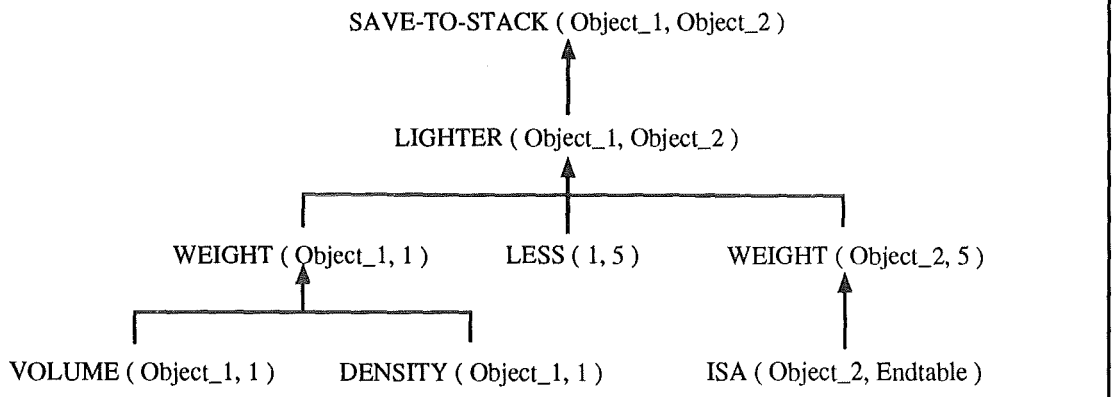

Abbildung 10

Unter Verwendung von Ausdrücken der Domain-Theorie erhält man den in Abbildung 10 dargestellten (deduktiven) Ableitungsbaum; die Pfeile stellen die logischen Implikationen im Sinne der Domain-Theorie dar.

Bereits aus diesem Beispiel wird ersichtlich, daß die Erklärung eigentlich aus einem (logischen, deduktiven) Beweis besteht. Daher hat die Erklärungsbildung im allgemeinen die Komplexität von Theorem-Beweisen.

4.5.2.2 EBG-Analyse der Erklärung

Das Zielkonzept wird nun von der Wurzel zu den Blättern durch den Beweisbaum (back-) propagiert. Dies ist eine Form der **Regression**. Die allgemeine Form der Regression einer gegebenen Formel F durch eine Regel R stellt einen Mechanismus dar, um die notwendigen und hinreichenden (schwächsten) Bedingungen zu bestimmen, unter denen die Regel R verwendet werden kann, um auf F zu schließen.

Die bei der Propagierung des Ziel-Konzeptes angewandte Generalisierungsregel ist die Ersetzung von Konstanten durch Variable. Auf diese Art und Weise erhält man an den Blättern einen generalisierten Ausdruck, der eine hinreichende Konzeptdefinition darstellt (Abbildung 11).

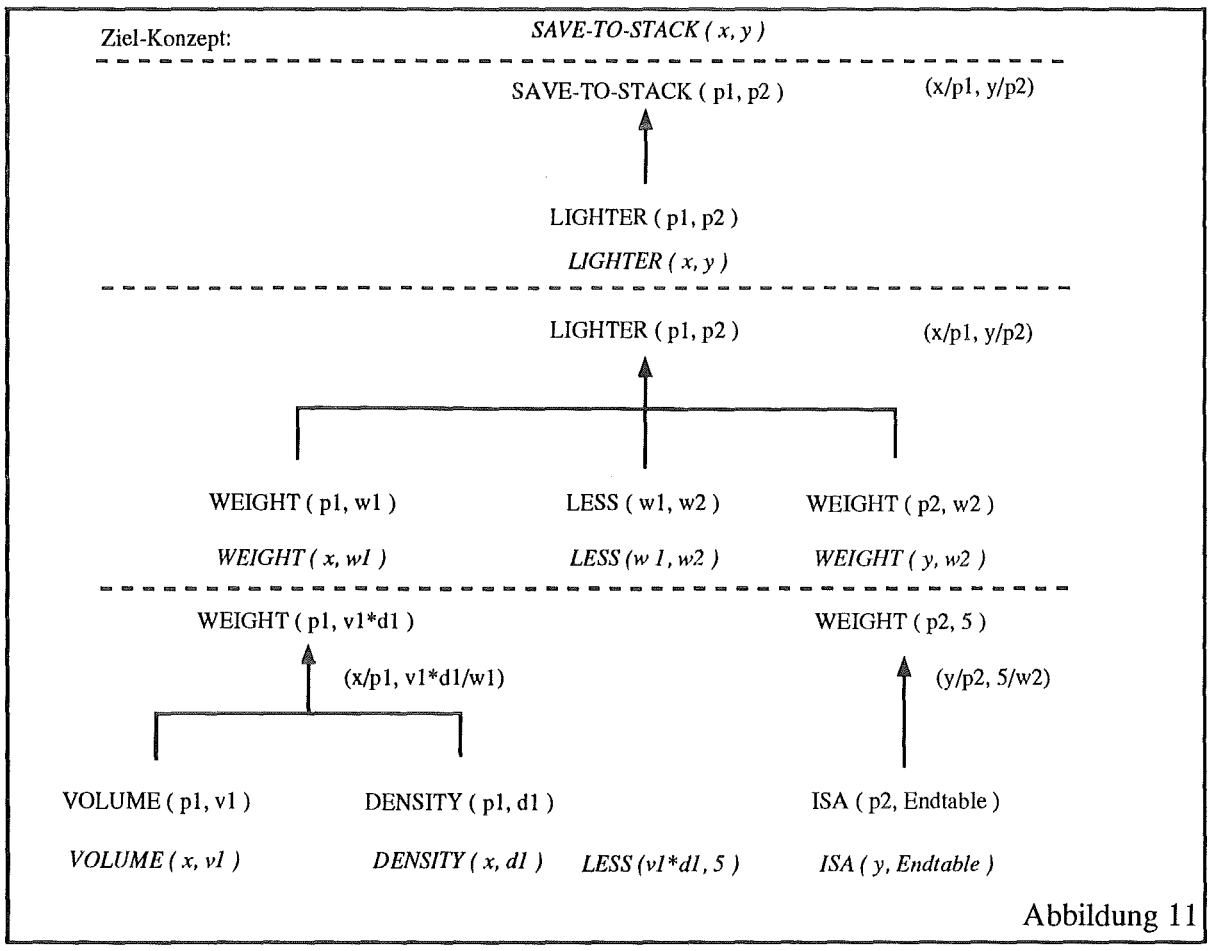

Abbildung 11

Am Beispiel des letzten in Abbildung 11 dargestellten Schrittes, bei dem eine Regression des Ausdrucks

$$\text{WEIGHT (p1, w1)} \wedge \text{WEIGHT (p2, w2)} \wedge \text{LESS (w1, w2)}$$

durch die letzten Schritte des Ableitungsbaumes (Erklärungsbaumes) durchgeführt wird, wird nun der Ziel-Regressionsvorgang etwas genauer erläutert. Für jeden Ausdruck der konjunktiven Verknüpfung findet eine eigene Regression durch die entsprechende Regel wie folgt statt: Der konjunktiv verknüpfte Ausdruck wird mit dem Konsequenzteil (rechte Seite) der Regel unifiziert und liefert somit eine Menge von Substitutionen (speziell Variablenbin-

dungen). Die mit dem Beispiel konsistente Substitution wird dann auf den Antezendenzteil (linke Seite) der Regel angewendet und liefert den sich ergebenden regressierten Ausdruck. Jeder Teil der konjunktiven Verknüpfung, der mit keinem Konsequenzteil einer anderen Regel unifiziert werden kann, wird einfach zu dem sich ergebenden regressierten Ausdruck (mit den darauf angewendeten Substitutionen) hinzugefügt.

In obigem Beispiel bedeutet dies:

Die Regression von WEIGHT (x, w1) durch die Regel

$$\text{VOLUME}(p1, v1) \wedge \text{DENSITY}(p1, d1) \rightarrow \text{WEIGHT}(p1, v1 * d1)$$

liefert

$$\text{VOLUME}(x, v1) \wedge \text{DENSITY}(x, d1).$$

Die Regression von WEIGHT (y, w2) durch die Regel

$$\text{ISA}(p2, \text{Endtable}) \rightarrow \text{WEIGHT}(p2, 5)$$

liefert

$$\text{ISA}(y, \text{Endtable}).$$

Nach Anwendung der durch die Regression erzeugten Substitutionen wird schließlich noch LESS(w1, w2) zu den regressierten Ausdrücken hinzugefügt, da mit ihm kein Konsequenzteil einer Regel unifiziert werden konnte. In diesem Fall müssen hierzu die folgenden Substitutionen berücksichtigt werden:

$$\{ x/p1, v1*d1/w1, y/p2, 5/w2 \}.$$

Damit ergibt sich LESS(v1*d1, 5) und damit als endgültige operationale Definition von SAVE-TO-STACK (x, y):

$$\begin{aligned} &\text{VOLUME}(x, v1) \\ &\wedge \text{DENSITY}(x, d1) \\ &\wedge \text{LESS}(v1*d1, 5) \\ &\wedge \text{ISA}(y, \text{Endtable}) \rightarrow \text{SAVE-TO-STACK}(x, y). \end{aligned}$$

In diesem Ausdruck werden die Merkmale des Trainigsbeispiels, die für eine allgemeine Erklärungsstruktur ausreichend sind, in Form operationaler Ausdrücke charakterisiert. In diesem Sinne stellt er eine begründete Generalisierung des Trainingsbeispiels dar; der Erklärungsbaum dient hierfür als Begründung.

Das gelernte Konzept ist jedoch schwächer als das ursprüngliche, da es für das gegebene Trainingsbeispiel und nicht für jedes mögliche Beispiel des Ziel-Konzeptes erzeugt wurde. Es ist jedoch durch die einfachen Teilaussagen leichter evaluierbar und abstrahiert von der für das Beispiel unwesentlichen Eigenschaft FRAGILE (y); andererseits kommen auch für die Begründung unerhebliche Prädikate des Beispiels wie COLOR nicht vor. In gewissem Sinne konzentriert sich die neue Konzeptdefinition also auf das "Wesentliche".

Wendet man diese Verfahren auf mehrere Beispiele an, so erhält man für jedes einzelne Beispiel ein Konzept. Diese Konzepte kann man dann disjunktiv zusammenfassen.

An diesem Beispiel wurde auch deutlich, daß erklärungs-basierte-Methoden, wie EBL, die Hauptschwierigkeit ähnlichkeits-basierter-Methoden, deren Unfähigkeit begründete Generalisierungen zu erzeugen, überwinden. Dies wird erreicht, indem angenommen wird, daß das System über Domain-Wissen, das Ziel-Konzept und ein Operationalisierungs-Kriterium verfügt. D. h. das Lernen hängt wesentlich von den bereits vorhandenen Kenntnissen des Systems ab. Eine Konsequenz hiervon besteht darin, daß der Grad der Generalisierung eines einzelnen Trainingsbeispiels wesentlich von der Allgemeinheit der Darstellung der Regeln der Domain-Theorie bestimmt wird. Eine zweite Konsequenz besteht darin, daß das lernende System seine Leistung bis zu einem Grad verbessern kann, so daß es neue Regeln für seine Domain-Theorie lernen kann.

4.6 Ein Verfahren mit hybrider Lernstrategie

4.6.1 DISCIPLE

In diesem Abschnitt wird DISCIPLE wie bei (/Kodratoff 56/) und (/Tecuci, Kodratoff 117/) beschrieben. Vorallem in (/Tecuci, Kodratoff 117/) wird der gesamte Algorithmus anhand eines Beispieles ausführlich beschrieben.

DISCIPLE ist ein multistrategie Lernsystem für den Bereich der Problemlösung, das auf einer Theorie und Methodologie zum Lernen von Expertenwissen basiert und von Tecuci und Kodratoff entwickelt wurde. Es hat die gleichen Aufgaben wie ein LAS (Learning Apprentice System, eine Erklärung erfolgte in Abschnitt 4.5), vereint jedoch mehrere verschiedene Lernstrategien anstatt rein deduktiver Methoden. Die Theorie und die Lernmethoden von DISCIPLE orientieren sich an dem Informationsgehalt der Domain-Theorie, die dem System zur Verfügung steht. Zur Auswahl der Lernmethoden werden die folgenden Formen für Domain-Theorien unterschieden (/Tecuci, Kodratoff 117/):

- **Vollständige Domain-Theorie**:
  Eine vollständige Domain-Theorie für eine gegebene Aufgabe (im Rahmen eines Problemlösungsprozesses) besteht aus vollständigen Beschreibungen der Objekte und Aktionen aus der zugehörigen Problemlösungs-Episode.
  Zur Problemlösung verwendet DISCIPLE in diesem Fall eine Methode des erklärungs-basierten Lernens (EBL) (vgl. Abschnitt 4.5) und kann somit eine allgemeine Regel aus einem einzigen Beispiel (in diesem Fall einer einzigen Aufgabe) lernen.

- **Schwache Domain-Theorie:**
  Eine schwache Domain-Theorie für eine gegebene Aufgabe besteht nur aus unvollständigen Beschreibungen der Objekte des Anwendungsbereiches. Sie unterscheidet sich in ihrer Qualität von einer vollständigen Theorie darin, daß sie keine Beschreibungen von Aktionen beinhaltet.
  Zur Problemlösung verwendet DISCIPLE in einem solchen Fall eine interaktive Lernmethode, die synergetisch EBL, Lernen durch Analogie, empirisches (induktives) Lernen und Lernen durch Benutzerbefragung (Instruktion) kombiniert.

- **Unvollständige Domain-Theorie:**
  Das gesamte Spektrum zwischen einer vollständigen und einer schwachen Domain-Theorie wird als unvollständige Domain-Theorie bezeichnet. Sie enthält nur unvollständige Beschreibungen der Objekte und Aktionen der Problemlösungsaufgabe.

  Im Falle einer unvollständigen Theorie lernt DISCIPLE eine allgemeine Regel, indem es die der schwachen Domain-Theorie zugeordenten Methode mit der zugehörigen Methode der vollständigen Domain-Theorie kombiniert.

Ein Nebeneffekt des Regellernens in Verbindung mit einer schwachen oder unvollständigen Domain-Theorie besteht in der Entwicklung der Domain-Theorie im Sinne eines Modelles der Aktionen.

Bei DISCIPLE handelt es sich um ein Werkzeug zum Aufbau von praktischen Expertensystemen. Es integriert ein "leeres" Expertensystem und ein lernendes System, indem beide die gleiche Wissensbasis benutzen. Um mit DISCIPLE ein Expertensystem aufzubauen, muß man zunächst elementares Wissen über das Anwendungsgebiet (d. h. eine inhomogene, unvollständige Domain-Theorie) in die Wissensbasis einbringen. Im nächsten Schritt wird DISCIPLE zur interaktiven Problemlösung, wie im folgenden Szenario beschrieben, eingesetzt: Der Benutzer teilt DISCIPLE das zu lösende Problem (die Aufgabe) mit,

- das Experten-Subsystem beginnt das Problem zu lösen und zeigt dem Benutzer jeden Problemlösungsschritt (d. h. jede partielle Lösung) an. In einer Art Dialog kann der Benutzer dieser partiellen Lösung zustimmen oder sie verwerfen.

  Im letzteren Fall oder wenn es für DISCIPLE nicht möglich ist eine partielle Lösung vorzuschlagen, wird der Benutzer gezwungen eine eigene Lösung einzugeben. Nachdem diese Lösung angegeben wurde, findet der folgende Lernprozeß statt. DISCIPLE versucht eine allgemeine Regel, die es ihm zukünftig, d. h. in einer ähnlichen Situation, gestattet, eine Lösung vorzuschlagen, die der vom Benutzer vorgeschlagenen Lösung ähnlich ist, zu generieren (Analogie).

Auf diese Art und Weise entwickelt sich DISCIPLE fortschreitend von einem nützlichen Helfer zu einem echten Problemlösungsexperten.

DISCIPLE kombiniert die folgenden Problemlösungsmethoden:

- Problemreduktion (divide-and-conquer),
  (d. h. ein Problem wird dadurch gelöst, daß es sukzessive auf einfache Teilprobleme reduziert wird. Dieser Prozeß wird solange fortgesetzt, bis das ursprüngliche Problem (durch Dekompositionen und Spezialisierungen) zu einer Menge elementarer Probleme, d. h. Probleme mit bereits bekannten Lösungen, reduziert wurde.

- Problemlösung durch Constraints (Einschränkung des Lösungsraumes, z. B. durch negative Beispiele) und

- Problemlösung durch Analogie (Erzeugung "ähnlicher" Lösungen für "ähnliche" Problemsituationen).

DISCIPLE's Lernproblem kann wie folgt formuliert werden:

Gegeben:

- Eine Domain-Theorie bestehend aus:
  - Einer Spezifikation der Objekttypen der realen Welt, deren Eigenschaften und Beziehungen;
  - eine Menge von Inferenzregeln zur Herleitung von Eigenschaften und Beziehungen aus anderen Eigenschaften und Beziehungen;
  - eine Menge von Aktionsmodellen, die die im Domain durchführbaren Aktionen beschreiben. Ein Aktionsmodell spezifiziert die Vorbedingungen der Aktion (d. h. die Zustände der Welt, in denen die Aktion ausgeführt werden kann), die Auswirkungen der Aktion (d. h. die Zustände, die sich nach Ausführung der Aktion ergeben) sowie die Objekte, die bestimmte Rollen in der Aktion spielen könnten (d. h. der Agent, der die Aktion ausführt, das Objekt, auf dem die Aktion ausgeführt wird, das verwendete Instrumentarium etc.).

- Eine Problemlösungs-Episode (d. h. ein vorgegebenes Beispiel) bestehend aus:
  - Einem zu lösenden Problem P,
  - einer (partiellen Lösung) S von P.

Gesucht:

- Eine allgemeine Problemlösungs-Regel, durch die ähnliche Probleme wie P eine ähnliche Lösung wie S erhalten.

4.6.1.1 Lernen in einem Domain mit vollständiger Domain-Theorie

In diesem Fall sind die Objekte und alle relevanten Eigenschaften und Beziehungen ihrer einzelnen Faktoren in der Domain-Theorie beschrieben. Einige dieser Eigenschaften und Beziehungen können explizit angegeben sein, andere werden durch Inferenzregeln, mit denen sie deduktiv hergeleitet werden können, implizit beschrieben. Die Aktionsmodelle beschreiben die Aktionen, die im Domain ausgeführt werden können. Ein vollständiges Aktionsmodell spezifiziert sämtliche notwendigen Vorbedingungen der Aktion, ihre gesamten Auswirkungen sowie alle Objekte, die bestimmte Rollen in der Aktion spielen können.

Das Lernen in einem solchen Fall basiert nun auf dem Paradigma des erklärungs-basierten Lernens, das ausführlich in den Abschnitten 4.5 und 4.5.1-2 vorgestellt wurde.

4.6.1.2 Lernen in einem Domain mit schwacher Domain-Theorie

Eine schwache Domain-Theorie über eine Probemlösungs-Episode besteht aus einer unvollständigen Beschreibung der Objekte dieser Episode und beinhaltet kein Aktionsmodell. Es ist durchaus sinnvoll derartige Situationen zu betrachten, da es für einen Experten häufig sehr schwierig ist, die Aktionen in Form von Vorbedingungen und Auswirkungen (d. h. Regeln) zu beschreiben (vgl. auch /Keller, Weinberger 50/). Auf der anderen Seite fällt es ihm wesentliche leichter, die Objekte zu beschreiben und Beispiele für Dekompositionen und Spezialisierungen anzugeben.

Aus diesem Grund wurde bei diesem Ansatz darauf verzichtet, den Experten zur Formalisierung seines Wissens zu zwingen; man versucht auf der Basis dieser Angaben, das übrige notwendige Wissen selbständig zu lernen.

In Verbindung mit einer schwachen Domain-Theorie versucht DISCIPLE den Mangel an Wissen durch die Verwendung einer integrierten Lernmethode, welche die verschiedenen Lernparadigmen erklärungs-basiertes Lernen, Lernen durch Analogie, empirisches Lernen und Lernen durch Bedienerbefragung miteinander verbindet, auszugleichen.

Diese Lernmethode wird im folgenden etwas formaler dargestellt (/Tecuci, Kodratoff 117/):

Erklärungs-basierter Modus:

(1) Finde eine Erklärung der Lösung des Benutzers und nenne sie Erklärung 1.

Analogie-basierter Modus:

(2) Übergeneralisiere die gegebene Problemlösungs-Episode (d. h. das Beispiel 1), indem alle Objekte zu Variablen werden und nenne es allgemeine Regel 1.

(3) Verwende Erklärung 1 als eine untere Schranke für die Anwendbarkeitsbedingung der allgemeinen Regel 1.

(4) Übergeneralisiere die Erklärung 1 zum allgemeinsten Ausdruck, der vom Bediener noch als Erklärung der allgemeinen Regel 1 akzeptiert wird.

(5) Verwende diese übergeneralisierte Erklärung als eine obere Schranke der Anwendbarkeitsbedingung der allgemeinen Regel 1. Die obere, die untere Schranke und die allgemeine Regel 1 definieren einen reduzierten Versionenraum (vgl. Abschnitt 4.1.2) für die zu lernende Regel.

(6) (Generierung von (positiven- und negativen) Instanzen, zur Einschränkung der Suche nach der Regel im Versionenraum.)
Suche in der Wissensbasis nach Objekt-Tupeln, die zwar der oberen, jedoch nicht der unteren Schranke genügen.

Falls es solche Objekte gibt, nenne die Eigenschaften dieser Objekte, die zum Beweis, daß die Objekte der oberen Schranke genügen, verwendet wurden, Erklärung-i und gehe zu Schritt 7.

Falls es keine solchen Objekte gibt, dann zeige dem Benutzer die obere Schranke, die untere Schranke und die allgemeine Regel 1 an und stoppe.

(7) Verwende die Objekte, die in Schritt 6 gefunden wurden zur Generierung einer Instanz der allgemeinen Regel 1. Nenne sie Instanz-1. Diese Instanz ist analog zu Beispiel 1.

(8) Zeige die Instanz-i dem Benutzer und beauftrage ihn, sie als eine gültige oder ungültige Problemreduktion zu charakterisieren. Falls die Instanz-i vom Benutzer abgelehnt wird, gehe zu Schritt 9; sonst gehe zu Schritt 14.

Erklärungs-basierter Modus:

(9) Verwende Instanz-i als ein Mißerfolg (negatives Beispiel) der zu lernenden Regel.

(10) Finde eine Erklärung, warum die Instanz-i vom Benutzer abgelehnt wurde und nenne sie Mißerfolgs-Erklärung-1.

Empirischer Lernmodus:

(11) Spezialisiere die obere Schranke soweit wie möglich, ohne dabei die Mißerfolgs-Erklärung-1 zu überdecken.

Falls die obere und die untere Schranke identisch sind, verwendet sie als eine notwendige und hinreichende Bedingung für die allgemeine Regel 1, zeige sie dem Benutzer an und stoppe; gehe zu Schritt 12 sonst.

(12) Spezialisiere (falls notwendig) die untere Schranke so wenig wie möglich, ohne dabei die Mißerfolgs-Erklärung-1 zu überdecken.

(13) Gehe zu Schritt 6.

(14) Verwende Instanz-i als eine neues positives Beispiel der zu lernenden Regel und Erklärung-i als eine wahre Erklärung der Instanz-i.

(15) Suche nach einer speziellsten gemeinsamen Generalisierung der unteren Schranke und Erklärung-i. Hierbei können zwei Fälle eintreten:

(1) Diese Generalisierung ist nicht identisch mit der oberen Schranke, dann verwende sie als neue untere Schranke und gehe zu Schritt 6;

(2) diese Generalisierung und die obere Schranke sind identisch, dann verwende sie als notwendige und hinreichende Bedingung der allgemeinen Regel 1, zeige sie dem Benutzer an und stoppe.

4.6.1.3 Lernen in einem Domain mit unvollständiger Domain-Theorie

Bei einer unvollständigen Domain-Theorie können DISCIPLE einige Objektbeschreibungen, Inferenzregeln oder Aktionenmodelle fehlen oder nur unvollständige Beschreibungen davon vorhanden sein. In einem solchen Fall kombiniert DISCIPLE die beiden in den vorangegangenen Abschnitten vorgestellten Lernmethoden. Zuerst konstruiert das System einen unvollständigen Beweis des Beispiels, generalisiert ihn wie bei einer vollständigen Domain-Theorie und bestimmt somit eine übergeneralisierte Erklärung des Beispiels. Dann verwendet das System die übergeneralisierte Erklärung als Analogiekriterium, um wie bei einer schwachen Domain-Theorie Experimente auszuführen und die allgemeine Regel zu synthetisieren.

Etwas formaler dargestellt bedeutet dies (/Tecuci, Kodratoff 117/):

(1) Beweise, daß die vom Benutzer vorgegebene Lösung tatsächlich eine Lösung des zu lösenden Problemes ist. Da die Domain-Theorie unvollständig ist kann das System dem Benutzer spezielle Fragen stellen, um mögliche Lücken im Beweis zu füllen.

(2) Falls die Lösung des Benutzers neue Aktionen enthält, dann verwende den Beweis aus Schritt 1, um anfängliche Versionenräume für die Modelle dieser Aktionen zu definieren. Als Nebeneffekt des Regellernens wird DISCIPLE dann die Modelle dieser Aktionen lernen.

(3) Übergeneralisiere den im Schritt 1 gefundenen Beweis wie in der vollständigen Domain-Theorie. Falls ein Aktionsmodell unvollständig gelernt wurde, so verwende die obere Schranke seiner Vorbedingungen und Auswirkungen. Die Blätter des übergeneralisierten Beweisbaumes stellen eine übergeneralisierte Erklärung von Beispiel 1 dar, die DISCIPLE als Analogiekriterium verwendet.

(4) Formuliere unter Verwendung der Erklärung aus Schritt 1 und der übergeneralisierten Erklärung aus Schritt 3, wie bei der schwachen Domain-Theorie, einen reduzierten Versionenraum für die zu lernende Regel.

(5) Führe, wie bei der schwachen Domain-Theorie, in dem durch Schritt 4 definierten Versionenraum Experimente zur Suche nach der Regel durch. Verwende den übergeneralisierten Beweis aus Schritt 3, um Erklärungen für die Mißerfolge zu finden.

4.6.1.4 Bemerkungen zu DISCIPLE

In Verbindung mit einer vollständigen Domain-Theorie verwendet DISCIPLE erklärungsbasiertes Lernen und kann daher anhand eines einzigen Beispieles eine begründbare Regel lernen und nicht-korrekte Beispiele zurückweisen.

Die Lernmethode in Verbindung mit einer schwachen Domain-Theorie integriert die verschiedene Lernparadigmen des erklärungs-basierten Lernens, des Lernens durch Analogie, des empirischen Lernens und des Lernens durch Benutzereingabe (Instruktion).

Die wesentlichen Merkmale dieser Lernparadigmen im Rahmen von DISCIPLE sind:

- Die Idee von "Erklärung" in der schwachen Domain-Theorie und eine heuristische Methode, um solche Erklärungen zu finden.
- Die Verwendung von Analogie, um einen reduzierten Versionenraum für die zu lernenden Regeln zu definieren.
- Die Verwendung von beidem, Erklärungen für den Erfolg und Erklärungen für den Mißerfolg, für die Suche nach der Regel im Versionenraum.
- Die Formulierung "cleverer" Fragen, um nützliches Wissen vom Experten zu erhalten.
- Die Möglichkeit, die gelernten Regeln vor dem Experten zu verbergen.
- Eine "große Zufriedenheit" des menschlichen Experten.

In Verbindung mit einer unvollständigen Domain-Theorie lernt DISCIPLE, indem es die Methode der vollständigen Domain-Theorie und die der schwachen Domain-Theorie miteinander kombiniert.

Die Integration dieser drei Lernmethoden versetzt DISCIPLE in die Lage, das in einigen aktuellen Systemen auftretende Problem des "Sturzes von der Wissensklippe" ("falling off the knowledge cliff") zu lösen. Dieses Problem besagt, daß ein System innerhalb des Bereiches seines vorgegebenen Wissens gute Dienste leistet, daß sich jedoch bei nur der geringsten Bewegung nach Außen seine Performanz rapide verschlechtert /Michalski 79/.

4.6.1.5 Schwächen von DISCIPLE

Das multistrategie Lernsystem DISCIPLE weist die folgenden Schwächen auf:

- Die Ausdrücke, mit denen DISCIPLE arbeitet, bestehen aus Prädikaten, Konstanten und Variablen - es findet keine Auswertung von Funktionen statt.
- Eine semantische Einschränkung resultiert aus der Tatsache, daß die Allgemeinheit der gelernten Regel durch die Allgemeinheit der übergeneralisierten Erklärung (das Analogiekriterium) eingeschränkt wird.
- Die Methode der Erklärungsfindung bei schwacher Domain-Theorie ist nicht mächtig genug.
- Wenn DISCIPLE Kontrollwissen, z. B. in Form von Metaregeln verwendet, so wird dieses nicht gelernt, sondern vom Benutzer vorgegeben. Falls zwei Experten verschiedene Lösungen für das gleiche Problem vorgeben, erzeugt DISCIPLE einfach zwei verschiedene Regeln.

- DISCIPLE benötigt bereits zu Beginn eine initiale Domain-Theorie und liefert keinerlei Hilfsmittel, um diese zu definieren.
  Eine Lösung hierfür wird durch das System BLIP (Morik, 1989) vorgeschlagen. BLIP ist ein interaktives Lernsystem, dessen Hauptaufgabe darin besteht, eine Domain-Theorie als ersten Schritt zur Konstruktion eines wissensbasierten Systems aufzubauen.

4.7 Kognitionstheoretisch motivierte Konzepte zum Maschinellen Lernen

In diesem Abschnitt werden zwei Verfahren vorgestellt, die sowohl dem Bereich kognitiver Lerntheorien, als auch dem Gebiet des Maschinellen Lernens zugeordnet werden können. Die folgende Darstellung basiert im wesentlichen auf /Mandl, Friedrich, Hron 72/ und /Forbus und Gentner 30/.

4.7.1 Der Ansatz von de Kleer und Brown

Der Ansatz von de Kleer und Brown zum Aufbau mentaler Modelle orientiert sich am Ansatz der **qualitativen Physik**. In diesem Ansatz wird die Funktionsweise physikalischer Systeme und Ereignisse in qualitativer Form beschrieben und erklärt.

Der Aufbau eines mentalen Modells (d. h. der Wissenserwerbsvorgang) erfolgt auf der Grundlage komplexer Schlußfolgerungsprozesse, in deren Verlauf, aufgrund einer vorhandenen mentalen Repräsentation der strukturellen Eigenschaften des Systems, auf der Grundlage einzelner Systemkomponenten dessen funktionale Eigenschaften erschlossen werden. Diesen zentralen Vorgang des Wissenserwerbs bezeichnet man als qualitative Simulation.

4.7.2 Das Ansatz von Forbus und Gentner

Das von Forbus und Gentner entwickelte Stufenmodell des Erwerbs physikalischen Wissens orientiert sich an der qualitativen Prozeßtheorie und der Struktur-Abbildungstheorie. Aus diesem Grund werden diese beiden Theorien im folgenden kurz vorgestellt.

4.7.2.1 Die qualitative Prozeßtheorie

Die **qualitative Prozeßtheorie** wird zur Modellierung von Teilen des physikalischen Wissens des Menschen verwendet. Hierzu versucht sie Vorstellungen über physikalische Prozesse zu formalisieren. Eine zentrale Annahme besagt, daß beim Denken über physikalische Sachverhalte der Prozeßaspekt im Vordergrund steht. Prozeßvorstellungen ermöglichen es, grundlegende Zusammenhänge eines physikalischen Gegenstandsbereiches zu verstehen. Die qualitative Prozeßtheorie zielt darauf ab, diese Prozesse zu beschreiben und die entsprechenden Wissensrepräsentationen des Lernenden zu untersuchen.

Ziel der qualitativen Prozeßtheorie ist die Erfassung von Denkvorgängen in formalisierter Weise, um auf dieser Grundlage Computerimplementationen von Expertensystemen vornehmen zu können.

4.7.2.2 Die Struktur-Abbildungstheorie

Die **Struktur-Abbildungstheorie** dient zur Charakterisierung der Berechnungen, die durchgeführt werden müssen, um das Wissen des Lernenden vor einer Repräsentationsform zu einer allgemeingültigeren zu transformieren. Sie betrachte den Vergleich als den zentralen Vorgang des Wissenserwerbs. Die Grundlegende Annahme besagt, daß die Grundstruktur eines Gegenstandsbereiches (Basisbereich) auf einen anderen Gegenstandsbereich (Zielbereich) übertragen werden kann. Es werden vier Arten, wie ein Vergleich zwischen Basis- und Zielbereich hergestellt werden kann, unterschieden:

- Tatsächliche Ähnlichkeit,
- Analogie,
- bloßer Anschein,
- Abstrakte Abbildung.

Die Verwendung des Vergleiches beim Wissenserwerb hängt ab von der Zugänglichkeit, d. h. der Wahrscheinlichkeit, daß eine Übereinstimmung überhaupt bemerkt wird. Notwendig dafür ist die Vertrautheit mit dem Basisbereich und der Grad der Übereinstimmung zwischen Basis- und Zielbereich. Eine weitere Voraussetzung für die Verwendung des Vergleiches ist der Nutzen, der sich aus der Übereinstimmung ziehen läßt. Außerdem spielt die Analysierbarkeit eine Rolle, d. h. die Leichtigkeit oder Schwierigkeit, mit der eine Übereinstimmung erkannt oder ausgedrückt werden kann. Befunde aus der Experten - Novizenforschung zeigen, daß Experten bei Vergleichen tiefergehende und abstraktere Kriterien anlegen, während sich Novizen dabei mehr auf oberflächliche Merkmale beziehen.

4.7.2.3 Darstellung des Ansatzes

Vor dem Hintergrund der qualitativen Prozeßtheorie und der Struktur- Abbildungstheorie beschreiben Forbus und Gentner den Erwerb physikalischer Sachverhalte unter entwicklungspsychologischer Perspektive. Ihr Stufenmodell des Wissenserwerbs umfaßt vier aufeinander aufbauende Modelle mit wachsendem Genauigkeitsgrad:

- Prototypische Erfahrung,
- Ursachen - Annahmen,
- Naive physikalische Theorien,
- Expertenwissen.

Der Übergang von einem Modell zum anderen bedingt durch die Präzisierung des Wissens auch einen Wechsel des Repräsentationsformalismus. Insgesamt wird Lernen in physikalischen Domänen als ein Durchschreiten einer Folge verschiedener mentaler Modelle (prototypische Erfahrung bis Expertenwissen) mit stetigem Informationsgewinn aufgefaßt.

4.8 Symbolische Lernverfahren - ein Resümée

Die vorgestellten Lernalgorithmen unterscheiden sich in erster Linie durch den von ihnen überwiegend verwendeten Inferenztyp (induktiv, deduktiv oder Mischformen). Bei der Entwicklung eines Lernverfahrens hat die Wahl des Inferenztyps weitreichende Konsequenzen, insbesondere für dessen Einsatzgebiet.

- Rein induktive Inferenzen kommen fast ohne Hintergrundwissen aus. Sie sind falschheitserhaltend, führen jedoch zur "Entdeckung" neuen Wissens.

- Rein deduktive Inferenzen können nur bei umfangreichem Hintergrundwissen eingesetzt werden und führen nur zu einer "Umformulierung/Aufbereitung" des bereits vorhandenen Wissens.

Das vorgestellte induktive Lernverfahren ID3/CLS zum Lernen eines einzigen Konzeptes aufgrund vorklassifizierter Beispiele basiert auf induktiven Inferenzen. Das Hintergrundwissen besteht in der Vorklassifikation der Trainingsbeispiele und der Shannonschen Entropie zur Bestimmung des Informationsgehaltes eines Attributes beim Aufbau des Entscheidungsbaumes. Beurteilt man die Fähigkeiten der Lernverfahren ID3/CLS aus der Sicht des menschlichen Lernvermögens, so handelt es sich hierbei eher um ein "cleveres" Klassifikationsverfahren, als um ein Lernverfahren. Diese Einschätzung ist damit begründet, daß durch ID3/CLS eigentlich "nur" ein informations-theoretisch optimaler Entscheidungsbaum (Klassifikationsbaum) aufgrund vorgegebener Kriterien (Attribute) aufgebaut wird.

Bei der Versionenraum-Methode in Verbindung mit dem Kandidaten-Eliminations-Algorithmus wird eine regelorientierte Beschreibung für ein einziges Konzept aufgrund inkrementell vorgegebener, vorklassifizierter Trainingsbeispiele gelernt. Das Ziel dieses induktiven Algorithmus besteht darin, eine Generalisierung der Trainingsbeispiele zu finden, die alle positiven und keines der negativen Trainingsbeispiele umfaßt.

Mit induktiven Lernverfahren, die auf dem AQ-Algorithmus basieren, können aufgrund vorklassifizierter Beispiele gleich mehrere Konzepte gelernt werden. Die Grundidee dieser Verfahren zum Lernen von Konzeptbeschreibungen besteht in der Anwendung der Versionenraum-Methode in Verbindung mit dem Kandidaten-Eliminations-Algorithmus auf jedes einzelne positive Beispiel.

Die Repräsentation der Objekte durch Attribut-Werte-Paare führt zu einer eingeschränkten Ausdrucksmächtigkeit dieser Verfahren. Die Darstellung eines Objektes durch die Werte einer fest vorgegebenen Anzahl von Attributen ist insbesondere dann unangemessen, wenn ein Objekt eine sich verändernde Anzahl an (Bestand-) Teilen von Komponenten besitzt oder wenn man an den Beziehungen eines Objektes zu einer variablen Anzahl weiterer Objekte interessiert ist /Quinlan 93/.

Während die oben beschriebenen induktiven Lernverfahren der Lernstrategie des Lernens aus Beispielen im Sinne überwachten Lernens zuzuordnen sind, verwendet die Familie der BACON.x Lernverfahren die Strategie des Lernens durch Beobachtung mit aktivem Experimentieren. BACON.x hat die Aufgabe, eine einzige konzeptuelle Abhängigkeit zwischen vorgegebenen unabhängigen und abhängigen Variablen aufgrund einer ebenfalls vorgegebenen Menge an Daten (Werte dieser Variablen) zu entdecken. Hierzu manipuliert BACON.x mit Hilfe der ihm vorgegebenen Heurismen (Verfeinerungs-Operatoren) diese Daten aktiv,

mit dem Ziel, das unterlagerte Konzept durch die Bestimmung von Abhängigkeiten zwischen den Variablen offenzulegen.

Die Lernverfahren zur begrifflichen Ballung (conceptual clustering) und zur inkrementellen Konzeptbildung sind der Lernstrategie des Lernens durch Beobachtung zuzuordnen. Die begriffliche Ballung ist eine Weiterentwicklung der klassischen statistischen und numerischen Clusterverfahren, die eine Einbindung von Gestaltkriterien bei der Clusterung vorsieht. Im Gegensatz zu den bisher dargestellten maschinellen Lernverfahren werden bei einigen Verfahren zur begrifflichen Ballung und der inkrementellen Konzeptbildung keine "flachen", sondern hierarchische Konzeptbeschreibungen generiert. Die generelle Aufgabe dieser beiden Klassen maschineller Lernverfahren besteht darin, beobachtete Situationen oder Instanzen (Objekte) zu klassifizieren. Hierzu werden nicht nur Instanzen und eine Attributmenge zu deren Charakterisierung, sondern auch Hintergrundwissen über Einschränkungen im Problembereich benötigt.

Die Verfahren zur inkrementellen Konzeptbildung (EPAM, UMIMEM, COBWEB, CLASSIT) bilden aufgrund sequentiell vorgegebener (symbolischer oder numerischer) Daten (Instanzen) hierarchisch angeordnete Kategorien zur Gruppierung dieser Instanzen. Weiterhin wird für jede Kategorie eine intensionale Beschreibung gebildet, die sämtliche ihr zugeordneten Instanzen umfaßt.

Beim erklärungs-basierten Lernen (EBL) wird, zumeist aufgrund eines einzigen Trainingsbeispiels deduktiv eine allgemeine Konzeptbeschreibung erzeugt. Die Herleitung der Konzeptbeschreibung geschieht unter Verwendung von umfangreichem Hintergrundwissen.

Bei DISCIPLE handelt es sich um ein multistrategie Lernsystem, das sowohl induktive, als auch deduktive und analoge Inferenzen durchführt und sich dabei auf Hintergrundwissen wachsenden Umfgangs und wachsender Qualität bezieht.

Die abschließend beschriebenen Verfahren von de Kleer und Brown bzw. Forbus und Gentner sind auf physikalische Domänen beschränkt und aufgrund ihrer Komplexität wohl eher von theoretischem Interesse. Sie basieren auf zum Teil umfangreichem Hintergrundwissen über die Struktur und das funktionale Verhalten des zugrundeliegenden physikalischen Systems. Aufgrund der dargestellten stufenförmigen Modellierung des Entwicklungsprozesses im Rahmen des Wissenserwerbs und der vorausgesetzten ("computeradäquate") Formalisierbarkeit des Expertenwissens stellt sich zumindest die Frage nach einer kognitiven Adäquatheit des Verfahrens von Forbus und Gentner.

Zusammenfassend können in realen Anwendungen, in Verbindung mit symbolischen Verfahren zum maschinellen Lernen, die folgenden Schwierigkeiten auftreten:

- Die Ansätze zum inkrementellen Lernen sind zur Zeit eher in der Minderzahl,
    - bei derartigen Verfahren besteht zumeist eine Abhängigkeit des Lernergebnisses von der Reihenfolge der Beispielpräsentation.
    - Keines dieser Verfahren ist in der Lage die Ergebnisse seiner induktiven Schlußfolgerungen im Laufe der Zeit, d. h. durch deren ständige Wiederholung zu bestätigen und/oder zu verfeinern.

  - Probleme treten insbesondere dann auf, wenn sich nicht nur die zu lernenden Konzepte, sondern auch der Kontext (die Umwelt) dynamisch verändert.

- Nur wenige Ansätze (z. B. UNIMEM) sind in der Lage, ihre "Wissensbasis" im Laufe der Zeit dynamisch zu verändern (bidirektionale Suche).

- Die Verarbeitung unscharfer Daten, z. B. durch Meßfehler/-ungenauigkeiten verursacht, bereitet Probleme bzw. ist überhaupt nicht vorgesehen.

- Gelernt werden nur statische Konzepte, zumeist im Sinne von besteht-aus-Relationen; Transformationsfolgen, Episoden und situative Konzepte sind mit den hier dargestellten Verfahren nicht erlernbar.

- Die gesamte Information muß entweder zu jedem Zeitpunkt vorhanden sein, oder sie wird als statistischer Wert (Mittelwert) oder Vorgabewert (Default) prognostiziert. Damit ist im Rahmen des Lernprozesses keine Fokussierung auf relevante Attribute möglich.

- Bei den Verfahren zum Lernen aus Beispielen sind die Beispiele in der Regel "handverlesen". Daher wird das zu lernende Wissen durch die Beispielauswahl bzw. -aufbereitung bereits "mundgerecht" vorverarbeitet.

5 Evolutionäre Algorithmen

Unter dem Oberbegriff **evolutionäre Algorithmen** werden fünf verschiedene Ansätze zusammengefaßt, die alle die biologische Evolution als Konzept für eine Weiterentwicklung zum Vorbild haben. Macht man sich die Mechanismen der Evolution zunutze, so kann man Lösungen gleichsam züchten, ohne dabei die Problemstruktur genau verstehen zu müssen. Zu den evolutionären Algorithmen zählen:

- die **Evolutionsstrategien** /Rechenberg 97 und Schwefel 110/, die besonders gut zur Funktionsoptimierung geeignet ist,
- die **genetischen Algorithmen** (GA) /Holland 43/, die ebenfalls häufig als Optimierungswerkzeug eingesetzt werden, aber auch zur Planung und zum adaptiven Lernen geeignet sind,
- die **Classifier-Systeme** /Holland 43/, die zur Klassifizierung und zum Erlernen von bedingtem Handeln gedacht sind,
- das **Evolutionary Programming** von /Fogel, Owens, Walsh 29/, das eine vereinfachte auf den Mutationsoperator beschränkte Form der Evolutionsstrategie darstellt und
- das **Genetic Programming** /Koza 61/, mit dessen Hilfe LISP-Programme erzeugt werden.

Betrachtet man Algorithmen und / oder Computerprogramme vor dem Hintergrund ihres praktischen Einsatzes (ihrer Anwendung), so stellen sie zumeist nichts anderes als mechanisierte / computerisierte Vorschriften zur Lösung mehr oder minder komplexer Probleme dar. Bei der klassischen imperativen Programmierung besteht der Zwang, im voraus für jede Situation, mit der das Programm konfrontiert sein könnte, festzulegen, was zu tun ist. Da dies bei realen Problemen mit hinreichender Komplexität in der Regel nicht möglich ist, sind die entstehenden Programme mehr oder minder fehlerbehaftet. Durch den Einsatz evolutionärer Algorithmen zur Problemlösung versucht man, diese Schwierigkeiten zu umgehen, indem man evolutionäre Mechanismen zur Generierung von Problemlösungen verwendet.

Beim Genetic Programming wird versucht, dies durch eine evolutionäre Züchtung der Programme selbst zu umgehen. Dieser Ansatz wird genauso wie das Evolutionary Programming hier aus Platzgründen nicht weiter dargestellt; siehe auch /Fogel und Atmar 28/.

Da die Evolutionstrategie auf denselben Paradigmen beruht wie die genetischen Algorithmen und sich beide Verfahren im wesentlichen nur in der noch zu besprechenden Repräsentationsform unterscheiden (siehe auch /Hoffmeister u. Bäck 42/), wird im folgenden, sofern nicht anders vermerkt, nur von den genetischen Algorithmen gesprochen.

Mitte der sechziger Jahre wurde von John H. Holland, basierend auf dem von ihm vorgestellten (klassischen) genetischen Algorithmus, das Konzept des Klassifizierungssystems entwickelt, mit dem die Struktur eines jeden beliebigen Computerprogrammes darstellbar

sein sollte. Da die Klassifizierungssysteme auf Mechanismen der genetischen Algorithmen beruhen, werden die genetische Algorithmen zuerst ausführlicher dargestellt.

Mit Hilfe von genetischen Algorithmen kann man zu einem gegebenen Problem einen viel größeren Bereich potentieller Lösungen schneller untersuchen als mit herkömmlichen Methoden. Umgekehrt bieten sie ein Modell der natürliche Evolution unter kontrollierten Bedingungen; daher könnten ihre Ergebnisse auch Aufschlüsse zu der Frage liefern, wie im einzelnen Leben und Intelligenz sich in der natürlichen Welt entwickelt haben. Zusammenfassend kann man sagen, daß die genetischen Algorithmen ein interessantes Werkzeug bilden, um optimale Parameter für Systeme zu finden, die

- eine nicht-stetige Zielfunktion besitzen oder
- die Berücksichtigung vieler bis extrem vieler Parameter und Randbedingungen erfordern oder
- eine gleichzeitige Optimierung nach mehreren sich eventuell teilweise widersprechenden Kriterien erfordern oder
- deren Struktur unbekannt ist oder
- deren Aufgabenstellung Adaptivität erfordert,

und die deshalb mit den üblichen Gradientenverfahren entweder gar nicht oder nur schwer bearbeitet werden können. Im Gegensatz zu vielen mathematischen Verfahren liefern genetische Algorithmen kontinuierlich Zwischenlösungen und nicht erst ein Ergebnis nach ihrer Terminierung. Da evolutionäre Algorithmen bei ihrem Lösungsfindungsprozeß den Suchraum explorieren und dabei Wissen über seine Struktur akkumulieren, das in der Population niedergelegt ist, eignen sie sich auch als Lernverfahren.

5.1 Vom natürlichen Vorbild zum genetischen Algorithmus

Die Evolution der meisten Arten steht unter dem Einfluß zweier fundamentaler Prozesse

- der **natürlichen Selektion**. Sie bestimmt, welche Individuen einer Population überleben und sich fortpflanzen können.
- Die **sexuelle Reproduktion** sorgt dafür, daß die weitergegebenen Gene gemischt und neu kombiniert werden, wenn die Kerne von Samen- und Eizelle verschmelzen. Dadurch können sich günstige Kombinationen viel schneller ergeben, als wenn jedes Individuum einfach sämtliche Gene - gelegentlich durch Mutation modifiziert - von einem Elternteil erben würde.

Selektion ist einfach: Wenn ein Organismus einen Tauglichkeitstest - etwa einen Angreifer zu erkennen und zu fliehen - nicht besteht, stirbt er. Nachzuprüfen, ob ein Lösungsvorschlag seinen Zweck erfüllt und einen untauglichen zu verwerfen, ist ebenso einfach (jedenfalls einfacher, als einen neuen zu entwickeln).

Für die Züchtung von ertragreicherem Getreide oder schöneren Rosen wird seit Jahrtausenden eine Kombination von Kreuzung (Reproduktion) und Auslese eingesetzt. Es ist jedoch nicht so leicht, dieses Verfahren auf den Computer zu übertragen. Das Hauptproblem ist, das Analogon eines genetischen Codes zu finden. Der Programmtext selbst ist hierfür nicht geeignet. Würde man beispielsweise in einem PASCAL-Programm einen Buchstaben ändern (Punktmutation) oder die erste Hälfte des Programmes mit der zweiten Hälfte eines anderen verbinden (entsprechend dem Crossing-over, dem Faktorentausch bei der Chromosomenpaarung), so ergäbe sich in den meisten Fällen nicht ein besseres oder schlechteres PASCAL-Programm, sondern überhaupt keines. Die **Repräsentation der Information** in den Genen läßt sich auf zwei grundlegende Arten bewerkstelligen: entweder **problembezogen** und **direkt** oder **neutral** und **indirekt** (vgl. auch /Bruns 13/ und /Gorges-Schleuter 34/).

Bei der **neutralen indirekten Repräsentation** operieren die genetischen Operatoren über Strings, in denen die problembezogenen Parameter in codierter Form enthalten sind. Dazu ist eine Codierung und Decodierung von der Problemebene in die Gen-Ebene und zurück notwendig. In der Sprache der Biologie erfolgt eine Abbildung zwischen den *Pheno-* und den *Genotypen*. Der Vorteil dieser Vorgehensweise liegt darin begründet, daß die genetischen Operatoren problemneutral formuliert werden können. Erkauft wird dies mit dem Aufwand für Codierung und Decodierung und dem Nachteil, daß die nun "blind" arbeitenden Operatoren auch viele ungültige Lösungen erzeugen können. Diese werden entweder durch die Bewertung herausgefiltert oder es findet ein "genetic repair" statt, das die ungültigen Teile in zulässige verwandelt. Der Vorteil des "genetic repair" liegt in der Begrenzung des Suchraums auf den gültigen Teil. Das "genetic repair" ist ein problemspezifischer Operator und kann unter Umständen sehr rechenzeit- und implementierungs-aufwendig sein. Der häufigste Vertreter der indirekten Repräsentation ist die Codierung in Bitketten (siehe auch Abschnitt 5.2). Diese Ausprägung des Verfahrens wurde von Holland unter der Bezeichnung genetische Algorithmen eingeführt. Daher wird diese Gruppe der genetischen Algorithmen auch häufig als **klassische genetische Algorithmen** bezeichnet.

Im Gegensatz dazu verzichtet die **problembezogene direkte Repräsentation** auf Codierung und Decodierung und bildet die problembezogenen Parameter direkt in den Genen ab. Die genetischen Operatoren werden zumindest teilweise anwendungsspezifisch formuliert und können daher so gestaltet werden, daß sie keine oder nur wenig unzulässige Lösungen erzeugen. Ein "genetic repair" ist meist nachwievor sinnvoll, kann aber in der Regel einfacher gestaltet werden. Verschiedene Untersuchungen, unter anderem auch von den zuvor genannten Autoren zeigen, daß die direkte Repräsentation hinsichtlich Performance und Konvergenzsicherheit der indirekten überlegen ist.

Leider gilt für beide Ansätze, daß bisher keine Theorie erarbeitet werden konnte, die das Verhalten der genetischen Algorithmen soweit erklärt, als daß sich daraus die günstigste Gestaltung eines genetische Algorithmus (Auswahl der Repräsentationsform und der genetischen Operatoren, Gestaltung der Paarungs- und der Überlebensstrategie, Festlegung von Wahrscheinlichkeiten für alle stochastischen Aktionen, usw) bei einem konkreten Anwendungsproblem ableiten ließe.

5.2 Problemlösung mit genetischen Algorithmen

In der Sprache *klassischer genetischer Algorithmen* besteht die Suche nach einer guten Lösung für ein Problem in der Suche nach bestimmten Bit-Ketten. Da klassische genetische Algorithmen, abgesehen von einer Tauglichkeitsfunktion (bzgl. deren Werten optimiert werden soll), keinerlei zusätzliche Information benötigen, werden sie auch als blind bezeichnet /Goldberg 33/. Den Raum aller möglichen Bit-Ketten (d. h. den **Lösungsraum**) kann man sich als imaginäre Landschaft vorstellen, in der jede Kette einen Punkt und ihre Qualität der Höhe dieses Punktes über einem Nullniveau entspricht. Täler sind Bereiche schlechter Lösungen; die Gipfel entsprechen den bestmöglichen Bit-Ketten.

In diesem Lösungsraum kann man durch Fixierung des Wertes der Bit-Ketten an gewissen Stellen Teilgebiete definieren, die eine gewisse Analogie zu den Längen- und Breitengraden auf einer gewöhnlichen Landkarte haben. Ein Beispiel für ein solches Teilgebiet ist die Menge aller Bit-Ketten, die mit einer Eins beginnen. Ebenso bilden alle Ketten, die mit einer Null beginnen, ein Teilgebiet, alle die an den Stellen 4 bis 6 das Muster 101 tragen, und so weiter.

Wenn man in solch einer Landschaft einen Gipfel finden will, ist es naheliegend, möglichst immer aufwärts zu gehen. Hierzu beginnt man an irgendeinem zufällig gewählten Punkt. Wenn eine kleine Modifikation die Qualität der Lösung verbessert, führt man diesen Schritt aus und probiert an der erreichten Stelle weiter; wenn nicht, geht man in eine andere Richtung.

Bei komplexeren Problemen aber hat die (imaginäre) Landschaft viele Gipfelpunkte und mit zunehmender Dimension des Lösungsraumes kann sie Tunnel, Brücken und noch kompliziertere topologische Strukturen enthalten und vor allem unstetig sein. Den höchsten Gipfel in solch zerklüftetem Gelände zu finden oder auch nur herauszufinden, in welche Richtung es aufwärts geht, wird immer schwieriger und aufwendiger. Außerdem sind die zu durchsuchenden Räume gewöhnlich riesig groß.

Genetische Algorithmen werfen das Analogon eines Gradnetzes über diese Landschaft aus. Die zahlreichen Bit-Ketten in der sich entwickelnden Population untersuchen die Landschaft in vielen Teilgebieten zugleich. Insbesondere entspricht die Intensität, mit der ein genetischer Algorithmus ein Teilgebiet durchsucht, der mittleren Höhe dieses Teilgebietes, d. h. der Wahrscheinlichkeit, dort eine gute Lösung zu finden.

Diese bemerkenswerte Fähigkeit genetischer Algorithmen, ihre Aufmerksamkeit auf die aussichtsreichsten Gebiete des Lösungsraumes zu konzentrieren, rührt von ihrer Fähigkeit her, Bit-Ketten zu reproduzieren und zu kombinieren, die Lösungen für Teilprobleme enthalten:

- Zuerst wird jede Kette aus der Population bewertet, indem die Qualität der Lösung, die sie codiert, gemessen wird.

- Zweitens werden Bit-Ketten anhand ihrer Bewertung gewichtet und reproduziert, d. h. höherbewertete Bit-Ketten werden mit einer größeren Wahrscheinlichkeit reproduziert als niedriger bewertete (nicht-deterministische Suche). Die zugrundeliegende Idee basiert auf der Darwinschen Theorie des "**survival of the fittest**" /Goldberg 33/.

- Drittens werden die reproduzierten Bit-Ketten gepaart: Das hierzu angewandte Verfahren entspricht weitgehend dem **Crossing-over** der Chromosomen von Ei- und Samenzelle.

- Für die Strategie, welche Individuen der Ausgangspopulation durch die Nachkommen ersetzt werden, gibt es viele Varianten. Geeignete Nachkommen müssen nicht unbedingt an die Stelle ihrer Eltern treten, sondern können auch beispielweise Bit-Ketten geringerer Tauglichkeit verdrängen. Die Gesamtheit der Individuen einer Population bleib so konstant.

- Viertens wird ein kleiner Teil der Bit-Ketten durch **Mutationen** modifiziert: Beispielsweise wird eines von je 10.000 Bits von einer 0 in eine 1 oder vice versa abgeändert. Mutationen allein bringen zwar die Suche nach einer Lösung im allgemeinen nicht weiter, aber sie verhindern, daß sich eine einförmige, zu keiner weiteren Evolution mehr fähigen Population entwickelt, indem sie verlorengegangene Information wieder in die Population einbringen.

Beispiel: Gegeben seien die folgenden beiden Bit-Ketten:

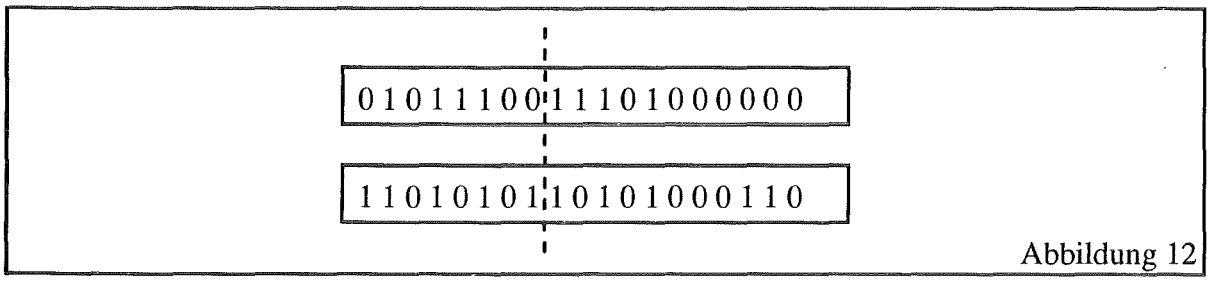

Abbildung 12

An der markierten Stelle wird ein Crossing-over durchgeführt

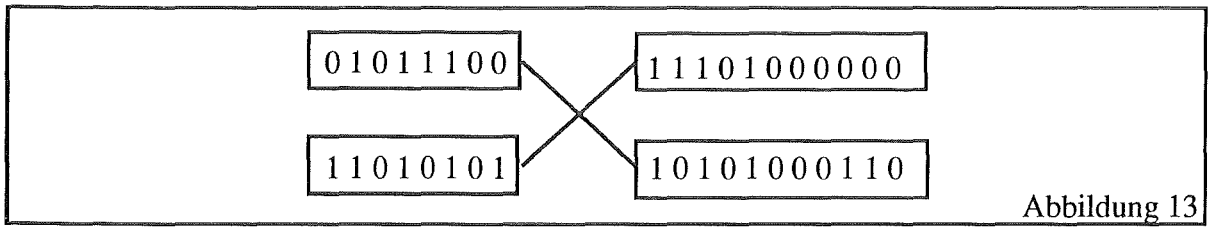

Abbildung 13

und man erhält die beiden "neuen" Bit-Ketten:

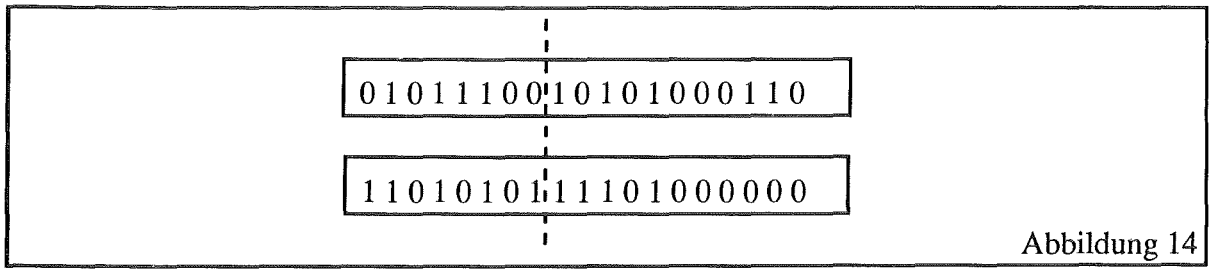

Abbildung 14

Führt man bei der ersten Bit-Kette eine Mutation beim 12-ten Bit durch, so ergibt sich die folgende Bit-Kette:

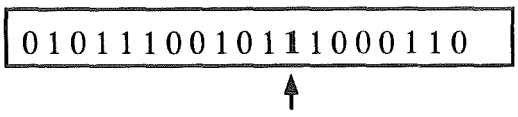

Abbildung 15

Da ein klassischer genetischer Algorithmus die tauglichsten Bit-Ketten als Eltern bevorzugt und diesen daher mehr Nachkommen verschafft, konzentriert er sich im Laufe der Zeit immer mehr auf vielversprechende Teilgebiete des Lösungsraumes, die sogenannten Zielgebiete. Auf die Dauer ist die Anzahl der Bit-Ketten, die in einem Teilgebiet liegen, proportional zur mittleren Tauglichkeit in diesem Gebiet.

Der Schlüssel zu diesem überraschenden Verhalten liegt darin, daß eine einzelne Bit-Kette sehr vielen Gebieten angehört. Beispielsweise gehört die Kette 11011001 zu den Gebieten 11******, 1******1, **0**00* etc. (der Stern bedeutet, daß an dieser Stelle der Wert der Bit-Kett nicht vorgeschrieben ist). Die größten Gebiete - also die mit den meisten Sternen - werden typischerweise von einem großen Teil einer Population untersucht. Diese **implizite Parallelität** (d. h. die gleichzeitige Suche in mehreren verschiedenen Gebieten) verschafft den genetischen Algorithmen einen imensen Vorteil gegenüber anderen Problemlösungsverfahren.

Die Wahrscheinlichkeit, daß ein Nachkomme ein Gebiet verläßt, in dem ein Elternteil liegt, hängt von der Entfernung zwischen den Nullen und Einsen ab, die das Gebiet definieren. Nahe beieinanderliegende Einsen oder Nullen, die ein Gebiet festlegen, nennt man **kompakte Bausteine** (building blocks). Sie werden mit großer Wahrscheinlichkeit das Crossing-over intakt überstehen, durch Mutation nur selten verändert werden und so an künftige Generationen mit einer Rate vererbt werden, die proportional zur mittleren Tauglichkeit der sie enthaltenden Bit-Ketten ist.

Somit gilt bei einer Reproduktion mit Crossing-over der Satz, daß die Populationsdichte in einem Gebiet zu dessen mittlerer Tauglichkeit proportional ist, nur für Gebiete, die durch kompakte Bausteine definiert sind; deren gibt es jedoch typischerweise erheblich mehr als die Population an Individuen enthält, so daß die Effekte der impliziten Parallelität sich nach wie vor auswirken.

Crossing-over erzeugt einerseits neue Stichproben aus Gebieten, die bereits in früheren Generationen überdurchschnittlich gut abgeschnitten haben und bestätigt oder widerlegt diese Einschätzungen. Andererseits entsteht möglicherweise durch Aufbrechen eines Bausteines ein gänzlich neuer Baustein und der genetische Algorithmus kann in bislang unerforschte Gebiete vordringen. Durch dieses Verhalten können genetische Algorithmen zur Entdeckung neuer Lösungen eingesetzt werden, da sie nicht an alten, "bewährten" Strategien festhalten, sondern "risikofreudig" neue erzeugen. Der Preis dafür ist darin zu sehen, daß das Crossing-over die Konvergenzgeschwindigkeit etwas verlangsamt.

5.2.1 Genetische Algorithmen zur Lösung nicht-linearer Probleme

Durch die implizite Parallelität kann ein genetischer Algorithmus nicht nur mit relativ geringem Aufwand eine große Anzahl von Gebieten im Lösungsraum durchsuchen, sondern auch mit schwierigen nicht-linearen Problemen zurechtkommen.

In diesem Zusammenhang heißt ein Problem **linear**, wenn die Tauglichkeit eines Systemes sich additiv aus den Tauglichkeiten seiner Komponenten zusammensetzt; ansonsten heißt es **nicht-linear**. Lineare Optimierungsprobleme sind, da die Komponenten ihre Tauglichkeit nicht gegenseitig beeinflußen, mit der ingenieurmäßigen Strategie des divide-and-conquer zu lösen.

Ist das Problem hingegen nicht-linear, so wächst die Schwierigkeit rapide an. Die mittlere Tauglichkeit in dem Gebiet *01*** könnte beispielsweise über dem Durchschnitt liegen, obwohl die beiden Gebiete *0**** und **1*** eine unterdurchschnittliche mittlere Tauglichkeit haben. Nicht-Linearität bedeutet nicht, daß es keine nützlichen Bausteine mehr gäbe, sondern nur, daß die genannten Bausteine, die nur aus einer einzigen Null oder Eins bestehen, dem Problem nicht angemessen sind. Das bedeutet, daß die Anzahl der potentiell sinnvollen Komponenten enorm anwächst.

Glücklicherweise läßt sich die implizite Parallelität immer noch ausnutzen. Eine Population von ein paar tausend Bit-Ketten wird viele kompakte Bausteine enthalten, die im Durchschnitt aus jeweils 100 oder mehr Individuen bestehen - ausreichend für eine gute statistische Stichprobe. Bausteine, die dank nicht-linearer Effekte überdurchschnittlich gut abschneiden, werden in künftigen Generationen automatisch länger vertreten sein.

5.3 Anwendung genetischer Algorithmen auf Klassifizierungssysteme

5.3.1 Das Prinzip der Klassifizierungssysteme

Im Sinne eines Klassifizierungssystemes besteht ein Programm aus einem System von Regeln der Form

WENN <Bedingungen> DANN <Handlungen>.

Jede Regel ist zugleich ein Stück aktionsfähiges Programm: Jedesmal, wenn ihr ein Input vorliegt, der die Bedingungen erfüllt, führt sie die vorgegebenen Handlungen aus. Bedingungen und Handlungen werden durch Bit-Ketten codiert, wobei jedes Bit der An- oder Abwesenheit einer bestimmten Eigenschaft im Bedingungsteil bzw. der Ausführung oder Nichtausführung einer bestimmten Aktion im Handlungsteil der Regel entspricht. Für jede erfüllte Eigenschaft enthält die Kette eine 1, für jede nicht erfüllte eine 0 an der entsprechenden Stelle. Die gesamte Kette ist gleichsam der genetische Code des Programmes, jedes Bit ist ein Gen.

Um mit einem derartigen Programm Aktionen aufgrund des Auftretens einer bestimmten Situation oder eines bestimmten Objektes ausführen zu können, müßte es mindestens eine Regel enthalten, deren Bedingungsteil die entsprechende Situation oder das Objekt aufgrund vorgegebener Merkmale (Eigenschaften) beschreibt. In der Praxis sollten hierzu möglichst einfache, aussagekräftige Merkmale gewählt werden. Implizit steckt in solchen Regeln eine Definition der Situation oder des Begriffes, woraus sich die Bezeichnung "Klassifizierung" erklärt.

Obwohl die Stärke dieser Regeln im Erkennen liegt, können sie auch Handlungen auslösen, indem man Bits im Ausgabeteil an entsprechendes Verhalten bindet.

5.3.2 Evolution unter Klassifizierungssystemen

Damit evolutionäre Vorgänge unter Klassifizierungssystemen stattfinden, konfrontiert man eine Population von zufällig zusammengewürfelten Ketten von Nullen und Einsen mit einer Aufgabe und bewertet jedes dieser Klassifizierungssysteme nach der Qualität der Ergebnisse, die es produziert. Um eine derartige Bewertung durchführen zu können, benötigt man lediglich minimales Wissen über die Tauglichkeitsfunktion; als Grundvoraussetzung für den Einsatz genetischer Algorithmen. Im Extremfall ist es sogar ausreichend, wenn man bei einer Lösung feststellen kann, ob sie besser oder schlechter als eine andere ist. Bit-Ketten hoher Tauglichkeit (**fitness**) werden sodann gepaart, solche niedriger Tauglichkeit werden verworfen. Den Nachkommen der erfolgreichen Bit-Ketten wird dieselbe Aufgabe wieder gestellt; dieser Prozeß wird iterativ fortgesetzt.

Im Laufe der Generationen setzen sich diejenigen Bit-Ketten durch, die bessere Lösungen liefern. Der Paarungsprozeß bringt immer wieder neue Kombinationen von ihnen hervor, die ihrerseits die Lösungen weiter verbessern.

5.3.3 Evolution durch Konkurrenz unter Klassifikationsregeln

Die biologische Evolution begünstigt häufig nicht ein besonders fähiges Einzelindividuum, sondern Gruppen, deren Angehörige miteinander in kooperativer Wechselwirkung stehen. In ähnlicher Weise kann eine Variante des Klassifizierungssystems nicht nur die Evolution individueller Regeln oder Strategien steuern, sondern Klassifizierungssysteme - entsprechend ganzen Organismen oder Gruppen - hervorbringen, die aus vielen Regeln bestehen.

Hierzu muß das Prinzip des Klassifizierungssystems ein wenig modifiziert werden. Wenn man, wie bisher beschrieben, jede Regel um so höher bewertet, je häufiger die von ihr erzeugte Aktion erfolgreich war, wird sich eine einzige Regeln auf Kosten aller anderen durchsetzen. Um die Evolution in Richtung einer Interaktion der Regeln untereinander zu lenken, läßt man Regeln darum konkurrieren, welche von ihnen in einer konkreten Situation die Aktion des Gesamtsystemes bestimmen. Alle Regeln, deren Bedingungsteil erfüllt ist, nehmen an der Konkurrenz teil (diese Situation entspricht einem Regelkonflikt in regelbasierten Systemen); unter ihnen setzt sich die stärkste durch. Dabei ist "Stärke" zunächst nichts anderes als ein zahlenmäßiges Attribut, das die jeweilige Regeln trägt. Wenn eine Aktion erfolgreich ist, wird die Stärke aller Regeln, die für diese Aktion votiert haben, erhöht, anderenfalls erniedrigt (Wettbewerbs-Lernen mit einer Gruppe von Regeln als "Sieger").

Alternativ kann man jede Regel als eine Hypothese über die Welt auffassen, mit der das System konfrontiert ist. Eine Regel tritt genau dann in den Wettbewerb ein, wenn sie sozusagen für sich in Anspruch nimmt, auf die aktuelle Situation zuzutreffen. Ihre Wettbewerbsfähigkeit hängt davon ab, wieviele gute Beiträge sie zur Lösung ähnlicher Probleme bereits geliefert hat. Der genetische Algorithmus paart starke Regeln miteinander und bringt Nachkommen hervor, in denen Bausteine ihrer Eltern kombiniert werden. Diese Nachkommen, die an die Stelle der schwächsten Regeln treten, entsprechen plausiblen, aber noch ungetesteten Hypothesen.

Der Wettbewerb unter den Regeln hilft dem System auf elegante Art, immerfort neue Anforderungen zu bewältigen. Starke Regeln, die in einer bestimmten Situation anwendbar sind, gleichen wohlbestätigten Vermutungen über die Außenwelt. Nachkommen solcher Regeln beginnen ihr Leben schwächer als ihre Eltern; sie können im Wettbewerb nur dann gewinnen und somit das Verhalten des Systemes beeinflussen, wenn es keine starken Regeln gibt, deren Vorbedingungen erfüllt sind. Sind ihre Aktionen auch noch hilfreich, so überleben sie, anderenfalls werden sie bald ersetzt. Die Nachkommen stören also das Systemverhalten in wohlbekannten Situationen nicht, sondern warten still im Hintergrund auf die Fälle, die bisher noch nicht vorgekommen sind. Dieses Verhalten ist im allgemeinen bei neuronalen Netzen (vgl. Kapitel 6) nicht zu beobachten. Hier können neugelernte

Trainingsbeispiele die Lösung für bereits gelernte durchaus beeinflussen, im schlimmsten Fall sogar auslöschen ("Übertrainieren").

Die Evolution eines Klassifizierungssystemes ändert sich drastisch, wenn man die beschriebene Konkurrenz unter Regeln einführt. Nach dem Start entwickelt das System zunächst Regeln mit einfachen Bedingungen, d. h. es behandelt sehr viele verschiedene Situationen gleich. Solche Regeln haben einen gewissen Wert als Standardrezept (default rules). Mit zunehmender Erfahrung aber gewinnt das System durch Reproduktion und Crossing-over zunehmend komplexere, speziellere Regeln, die schnell derart an Stärke zunehmen, daß sie das Systemverhalten in gewissen Fällen bestimmen.

Schließlich entwickelt sich eine Hierarchie: Schichten von Spezialregeln auf niederer Ebene behandeln die meisten Fälle, aber die Standardregeln auf der obersten Stufe kommen ins Spiel, wenn die Situation ungewohnt ist, d. h. für keine der Spezialregeln treffen die Bedingungen zu. Auf diese Weise bleiben frühere Erfahrungen auch für neue Situationen nutzbar; gleichzeitig wird vermieden, daß das System mit übermäßig spezialisierten Regeln dem neuen hilflos gegenübersteht.

Um ein Klassifizierungssystem für Aufgaben mit langfristigen Zielen (z. B. dem Gewinn einer Schachpartie) zu trainieren, bewertet ein Programmierer die Aktion des Systems erst dann, wenn es die gesamte Aufgabe erledigt hat. Im Erfolgsfall wirkt sich diese Bewertung als Stärkung, im anderen Fall als Schwächung auf alle Regeln aus, die sich an den Einzelaktionen beteiligt haben. Im Verlaufe vieler Generationen entwickelt das System Regeln, die immer früher eingreifen, um damit Voraussetzungen für späteren Erfolg zu schaffen. Das System wird so fähiger, gleichsam die Konsequenzen seiner Aktionen vorauszusehen.

Anzumerken bleibt noch, daß es sich bei genetischen Algorithmen nicht um die perfekte Nachbildung der Evolution bzw. deren Mechanismen handelt. Vernachläßigt werden beispielsweise Aspekte wie Gruppendynamik und eine zeitliche Verzögerung der Nachfolge, d. h. daß sich genetische Veränderungen erst einige Generationen später auswirken können.

5.4 Praktische Anwendungen genetischer Algorithmen

Obwohl bereits Anfang der 80er Jahre entwickelt, werden genetische Algorithmen erst in den letzten Jahren verstärkt untersucht und angewandt. Dafür gibt es drei wichtige Gründe: Genetische Algorithmen sind vergleichsweise sehr rechenzeitintensiv; sie gehören zur Klasse der "*number crunching*"-Verfahren. Mit den früher verfügbaren Rechnern war an einen praktischen Einsatz unter Kostengesichtspunkten in der Regel nicht zu denken. Andererseits ist das Verfahren von seiner Struktur her parallel und muß im Grunde für die klassischen Rechner erst "künstlich" sequentialisiert werden. Bis vor kurzem war an vergleichsweise preiswerte Rechner mit MIMD-Parallelität (Multiple Instruction, Multiple Data) jedoch nicht zu denken. Daher wird auch erst in letzter Zeit von dem Verfahren adäquaten Parallelimplementierungen berichtet, vgl. /Gorges-Schleuter 34/, /Lohmann 71/, /Starkweather 116/ oder /Jakob und Blume 45/.

Der zweite Grund liegt in dem Mißtrauen der Ingenieurwissenschaften gegenüber heuristischen Verfahren im Allgemeinen. Wie alle nicht-deterministischen Verfahren erzeugen genetische Algorithmen weder exakte Lösungen, noch erzielen sie bei mehrmaliger Anwendung gleiche Ergebnisse (wohl aber solche von vergleichbarer Qualität). Das bedeutet, daß konkrete Lösungen nicht rekonstruierbar sind. Es fehlen also Eigenschaften, die ein Ingenieur von "seinen" Lösungsverfahren gewohnt ist und die in ihm Vertrauen in die erzeugte Lösung erwecken. Wenn jedoch die Problemstellung einer exakten Berechnung mit den heutigen Mitteln und Methoden nicht zugänglich ist, stellen heuristische Verfahren wie die genetischen Algorithmen die einzige Möglichkeit dar, um überhaupt zu Lösungen zu kommen.

Zum Dritten ist die Informatik eine vergleichsweise junge Disziplin und es gab zunächst genügend "einfachere" Aufgaben, die mit der konventionellen imperativen Programmierung lösbar sind. Entsprechend wurden auch die Forschungsressourcen auf diese Gebiete konzentriert. Erst in letzter Zeit stößt man die Grenzen konventioneller Programmierparadigmen und sucht nach neuen Lösungen. Als Beispiele hierfür mögen die Expertensysteme oder die Fuzzy-Logik genügen. Erschwerend kommt für die genetischen Algorithmen noch hinzu, daß bei vielen potentiellen Einsatzgebieten das Vorurteil besteht, wonach das konkrete Problem zu kompliziert und undurchschaubar sei, um mit einem Rechner überhaupt gelöst werden zu können.

5.4.1 Anwendung im Bereich des maschinellen Lernens

Gerade wenn es um Anwendungen im Bereich des maschinellen Lernens geht, stellt sich natürlich die Frage, wie man die Stärken genetischer Algorithmen im Rahmen eines lernenden (adaptiven) Systemes ausnutzen kann.

Im Sinne adaptiven Lernens werden die Möglichkeiten betrachtet, bei denen das Lern-Element Veränderungen im Verhalten des Performanz-Elementes auslösen kann, um im Laufe der Zeit zu einer qualitativ besseren Lösung zu gelangen (/De Jong 20/).

Auf welche Art und Weise können nun genetische Algorithmen eingesetzt werden, um diese Aufgabe zu erfüllen?

- Eine Möglichkeit besteht darin, genetische Algorithmen zur Veränderung der Parametermenge, die ein zuvor entwickeltes Programm zur Lösung der Aufgabe kontrolliert, zu verändern.

- Eine zweite und auch interessantere Möglichkeit besteht darin, Veränderungen in komplexeren Datenstrukturen, wie z. B. Programmen, die das Verhalten des Performanz-Elementes kontrollieren, durchzuführen.

- Eine dritte Möglichkeit besteht darin, das Programm des Performanz-Elementes selbst zu verändern.

Beispiele von zumeist theoretischem Interesse zu diesen Ansätzen befinden sich in (/De Jong 20/).

Ein anderer Ansatz besteht darin, einen genetischen Algorithmus mit direkter Repräsentation selbst als Lern- und Performanz-Element zu benutzen. Blume /Blume 9/ berichtet von einem auf genetischen Algorithmen basierenden System zur Erlernung zielgerichteter Roboterbewegungen. Dabei erfolgt eine Aktionsplanung für einen freidefinierten Roboter mit bis zu 16 rotatorischen Achsen. Das System ist in der Lage, einmal erlerntes und in der Population gespeichertes Wissen auf ähnliche oder gänzlich andersartige Aufgabenstellungen gleichermaßen zu übertragen. Eigene Untersuchungen haben gezeigt, daß der Lösungsfindungsprozeß basierend auf den Ergebnissen einer mehr oder minder andersartigen Aufgabe um das 4- bis 6-fache schneller abläuft verglichen mit einem Start ohne jegliches Vorwissen, siehe /Jakob, Gorges-Schleuter, Blume 46/.

5.4.2 Allgemeine Anwendungen

Abgesehen vom Anwendungsgebiet "Maschinelles Lernen" gibt es bereits eine Reihe konkreter Beispiele für den Einsatz genetischer Algorithmen in der Praxis /Holland 43/, /Schaffer 105/, /Schwefel und Männer 111/, /Belew und Boker 3/ und /Männer und Manderick 73/:

- Steuerung eines Pipeline-Systemes,

- Entwurf von Kommunikationsnetzwerken,

- Entwurf eines Mantelstrom-Düsentriebwerkes,

- Layoutentwurf und -optimierung für integrierte Schaltungen,

- Synthese und Optimierung (adaptiver) Regler,

- Synthese und Optimierung (adaptiver) Fuzzy-Regler,

- Produktionsplanung, Maschinenbelegung,

- Verschnittminimierung durch optimierte Anordnungsplanung,
- Stunden- und Fahrplanoptimierung,
- Optimierung von Designprozessen,
- Adaptive Aktionsplanung,
- Optimierung der Beladung von Druckwasserreaktoren,
- Selbst-optimierende Image-Segmentierung zur Mustererkennung bei Farbbildern,
- Phantombildkonstruktion für Fahndungszwecke,
- Routenplanung und Scheduling im Eisenbahnnetz

Auch zu den Themen Entwurf und Optimierung der Struktur neuronaler Netzwerke und Verbesserung des Lernverhaltens neuronaler Netwerke gibt es zahlreiche Arbeiten, siehe (Abschnitt 6.6).

5.5 Bewertung des Ansatzes evolutionärer Algorithmen

Bei dem im Verlauf dieses Kapitels vorgestellten Ansatz genetischer Algorithmen handelt es sich um einen der bekanntesten Vertreter evolutionärer Algorithmen. Der Unterschied zwischen genetischen Algorithmen und den in den 70er Jahren entwickelten Evolutionsstrategien besteht im wesentlichen in der Kodierung der genetischen Information und in der Ausprägung der genetischen Operationen.

Zusammenfassend kann man die Stärken und Schwächen genetischer Algorithmen wie folgt darstellen:

Stärken:

- Genetische Algorithmen sind gut geeignet für Optimierungs- und Planungsprobleme. Hinsichtlich der Anzahl der Eingangsparameter und der Optimierungskriterien sind sie konventionellen Verfahren weit überlegen. Dies gilt insbesondere für alle Fälle, bei denen eine Verbesserung vorhandener Lösungen ausreicht und das absolute Optimum eher von theoretischem Interesse ist.

- Braun zeigte in /Braun 11/, daß genetische Algorithmen einen sehr effizienten Ansatz zur Lösung von kombinatorischen Optimierungsproblemen (z. B. Traveling Salesman Problem) bieten können. Mit den dort vorgestellten insulären genetischen Algorithmen gelang es ihm beispielsweise, die optimale Lösung für das Traveling Salesman Problem mit über 400 Städten zu finden.

- Sie sind insbesondere dann gut geeignet, falls die Zielfunktion nicht oder nur schwer mathematisch formulierbar ist.

- Sie stellen kaum Anforderungen an den Suchraum; Eigenschaften wie Stetigkeit oder ähnliches werden nicht verlangt.

Schwächen:

- Wie alle heuristischen Verfahren liefern sie bei mehrmaliger Anwendung auf das gleiche Problem nicht die gleiche Lösung. Einmal gefundene Lösungen sind nicht reproduzierbar.

- Genetische Algorithmen sind rechenzeitintensiv und eigenen sich nur im Zusammenhang mit konventionellen Komponenten für Realtime-Aufgaben. Dieser Nachteil kann jedoch durch eine Parallelimplementierung z. B. auf Rechnernetzen ausgeglichen werden.

- Die zur Lösungsfindung benötigte Zeit ist nicht konstant; es können aber Erwartungswerte bestimmt werden.

- Aufgrund der unzureichenden Theoriebildung kann bei einem gegebenen Problem nicht angegeben werden, welche Ausprägung eines genetischen Algorithmus die adäquate ist. Die erfolgreiche Anwendung genetischer Algorithmen erfordert daher viel Erfahrungswissen. Vollkommen analoge Probleme treten auch bei der Anwendung neuronaler Netze auf (vgl. Abschnitt 6.8).

Diese Beurteilung ist im Kontext des adaptiven Lernens wie folgt zu ergänzen:

Vorteile:

- Allein mit Hilfe erweiterter genetischer Algorithmen kann bereits ein lernendes System aufgebaut werden.
- Genetische Algorithmen unterstützen ein kontextabhängiges Agieren (Interaktion) indem z. B. Benutzereingaben in Form von Umbewertungen die weitere Richtung der Evolution beeinflussen.
- Eine Fokussierung auf relevante Eingangsgrößen ist möglich, wenn die Codierung dies gestattet und die Genlänge dynamisch ist.

Nachteile:

- Genetische Algorithmen sind nur bedingt geeignet in Domänen, in denen ein Optimieren und Testen, z. B. aufgrund von Sicherheitsaspekten, nicht durchführbar ist.
- Genetische Algorithmen verwenden eine unstrukturierte Form der Wissensrepräsentation.
- Sie bieten keinerlei Erklärungsfähigkeit; d. h. sie können (im Gegensatz zu regelbasierten Systemen) dem Benutzer keine strukturierten Lösungswege als Begründung ihrer Ergebnisse zur Verfügung stellen.

6 Konnektionismus - neuronale Netze

Die Bezeichnung neuronales Netz oder künstliches neuronales Netzwerk steht für eine rechnerbasierte Struktur zur Nachbildung der Eigenschaften und des Verhaltens von Gehirnstrukturen, wie Selbst-Adaption, Lernen und parallele Verarbeitung /Niemann 91/. Die Funktionalität neuronaler Netze ergibt sich aus dem Zusammenwirken (Verbindungen) einer großen Anzahl gleichartiger, relativ einfacher Grundeinheiten, den Neuronen. Durch eine Verbindung dieser zahlreichen eigenständigen Neuronen (elementare Ausführungseinheiten, Prozessoren) entstehen neuronale (vernetzte) Strukturen. Aufgrund der, für die neuronale Struktur, zentralen Bedeutung der Verbindungen zwischen den einzelnen Neuronen hat sich für diese Forschungsrichtung der Begriff **Konnektionismus** eingebürgert.

In diesem Kapitel werden nur die grundlegenden Aspekte der Theorie neuronaler Netze dargestellt. Für eine umfangreichere Darstellung, insbesondere auch bzgl. dem neurobiologischen Hintergrund, sei z. B. auf /Gehirn und Kognition 31/, /Rumelhart, McClelland et al. 103/, /Brause 12/, /Rojas 102/ oder /Birbaumer, Schmidt 7/ verwiesen.

6.1 Neurobiologische Grundlagen

Die künstlichen neuronalen Netze bilden nur sehr elementare Eigenschaften natürlicher neuronaler Strukturen nach. Basis aller Modelle ist das Neuron des menschlichen Nervensystems (Zentralnervensystem), von denen der Mensch im Gehirn etwa 25 Milliarden Einheiten besitzt. Ein **Neuron** bzw. eine Nervenzelle ist die funktionelle Grundeinheit mit allen zugehörigen Prozessen. Neuronen sind eines der Hauptmerkmale für Tiere; Pflanzen besitzen keine Nervenzellen. Bisher wurden beim Menschen zwischen 7 und 100 verschiedene Neuronen-Klassen nachgewiesen.

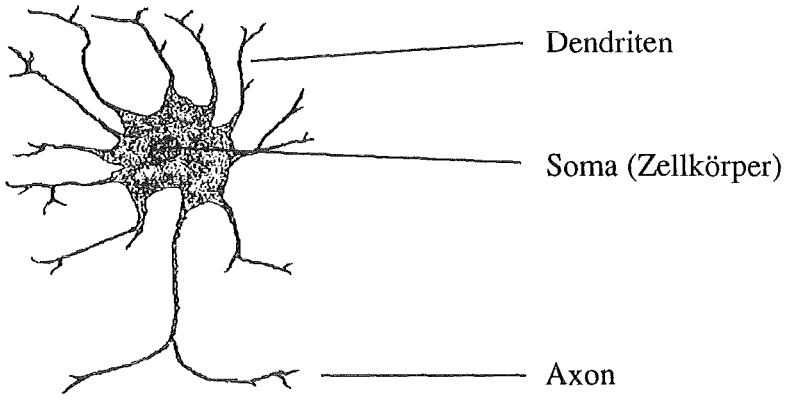

Abbildung 16

Der Grundplan aller Neuronen ist einander sehr ähnlich. Sie besitzen einen **Zellkörper** (Soma) mit dem **Zellkern** (**Nukleus**) sowie mehrere Fortsätze, einem **Axon** und meist mehreren **Dendriten**. Nur das Axon leitet Signale vom Neuron zu anderen Nervenzellen,

wobei dies auch Muskel- oder Drüsenzellen sein können. Bündel von Neuronen oder Nervenfasern organisieren sich zu höheren **Nervenstrukturen**. Die Leistung des Nervensystems werden über die Wechselwirkung zwischen den einzelnen Neuronen oder Neuronenverbänden erbracht.

Bestimmte Nervenzellen haben sich darauf spezialisiert, sensorische Signale an andere Nervenzellen zu übertragen. Diese Signale definieren den Zustand im oder auch außerhalb des Körpers. Diese Neuronen werden als Sensoren oder **Rezeptoren** (z. B. Augen und Ohren) bezeichnet. Über die **Effektoren** (z. B. Muskeln und Drüsen) als ausführende Organe kann der Körper selbst oder aber auch die Umwelt beeinflußt werden.

Ein Axon kann mit einem anderen Axon, mit Dendriten oder aber auch mit dem Soma direkt verbunden sein. Diese Verbindung in einem neuronalen Pfad heißt **Synapse**. Die einzige Beziehung zwischen den beiden Neuronen an diesem Punkt besteht in deren Kontakt. Der Impuls, der sich durch das eine Neuron bewegt, löst an dieser Stelle einen Impuls im zweiten Neuron aus.

Impulse breiten sich nur in eine Richtung, je nach Neuronen-Klasse aber mit unterschiedlicher Geschwindigkeit aus. Das Ende einer Nervenfaser (die Dendriten) wird bis zu einem Schwellenwert oder sogar darüber hinaus von anderen Nervenzellen (z. B. eines Rezeptor Organes) stimuliert. Chemische und elektrische Veränderungen setzen ein und bilden einen Impuls. Wenn der Impuls das andere Ende der Nervenfaser (das Axon) erreicht hat, kann er einen Impuls in einer anderen Nervenzelle, was dort evtl. zu einer Verstärkung oder aber zu einer Hemmung führt, induzieren. Nach der Übertragung des Impulses kehrt die Nervenzelle wieder in ihren ursprünglichen (Ruhe-) Zustand zurück und ist somit für einen neuen Impuls bereit. Die Reaktion eines Neurons auf eingehende Signale hängt aber auch von der Vergangenheit ab. Folgesignale führen innerhalb bestimmter Zeitspannen nicht zum gleichen Maß an Verstärkung (oder Hemmung). Es tritt ein Sättigungseffekt ein.

Die Neuronen des zentralen Nervensystems des Menschen sind in eine Umgebung, die Gliazellen, eingebettet, welche neben der Stützfunktion möglicherweise auch eine funktionale Eigenschaft besitzen.

In Verbindung mit künstlichen neuronalen Netzen sind die folgenden Prozesse in den Neuronen wesentlich:

- Die **Eingangsfunktion** (I):
  Die in die Synapsen einlaufenden Signale sind gewichtet, d. h. einige Signale sind stärker als andere. Einige Signale wirken anregend, andere hemmend. Die Auswirkungen aller gewichteten Eingaben werden intern summiert.

- Die **Ausgabefunktion** (O):
  Falls die Summe größer oder gleich dem Schwellenwert des Neurons ist, so liefert das Neuron eine Ausgabe, d. h. es "feuert". Es handelt sich somit um eine "Alles oder Nichts Situation", entweder "feuert" das Neuron oder nicht.

  - Die Schwellenwertfunktionen integrieren die Energie einlaufender Signale im Nervensystem über Raum und Zeit.

- Die **Aktivitätsfunktion** (A):
  Durch die Aktivität des Nervensystems (z. B. erhöhte Aufmerksamkeit, Müdigkeit) kann die Übertragungsrate der Signale (positiv und negativ) beeinflußt werden. Diese Fähigkeit, Signale zu regulieren, ist ein Mechanismus zum Lernen.

Graphisch kann man ein künstliches Neuron in etwa wie folgt darstellen:

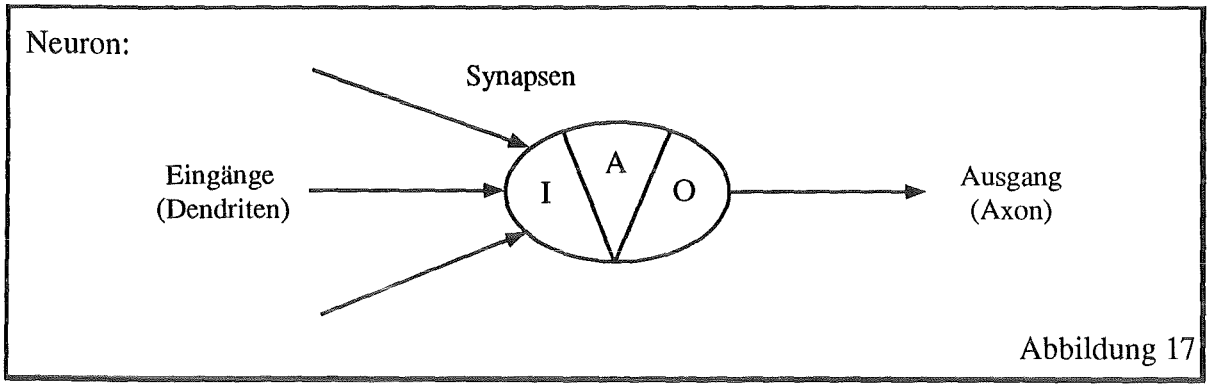

Abbildung 17

6.2 Allgemeine Funktionsprinzipien neuronaler Netze

Das konventionelle Lösungsparadigma zur Realisierung computergestützter Systeme (d. h. die klassische imperative Programmierung) basiert auf der formalen Beschreibung des Ausgangsproblems und der Angabe eines Lösungsverfahrens durch Spezifikation der einzelnen Schritte sowie der Art und Weise ihrer Abarbeitung. Dieser Ansatz zeigt dann Schwächen, wenn das Ausgangsproblem nicht bzw. nur mit unverhältnismäßig hohem Aufwand exakt **im voraus** beschrieben werden kann.

Aus diesem Grund weichen sowohl die wissensbasierten Methoden (z. B. in Expertensystemen) als auch neuronale Netze von diesem konventionellen Lösungsansatz ab. Bei den wissensbasierten Methoden wird nicht ein bestimmtes Lösungsverfahren beschrieben; vielmehr werden nur Fakten und Beziehungen zwischen diesen explizit deklariert und dann unter Anwendung allgemeingültiger Verfahren der Wissensverarbeitung (Inferenz, vgl. Abschnitt 2.1) sowie heuristischer Elemente des Problemlösungswissens (Auswahlstrategien) auf die Lösung konkreter Probleme angewendet.

6.2.1 Subsymbolische Repräsentation und parallele Verarbeitung

Neuronale Netze / konnektionistische Modelle zielen auf die (Re-) Konstruktion von Problemlösungsfähigkeiten ab. Sie bedienen sich eines Netzes von Verarbeitungselementen (Neuronen), die durch Signalaustausch (Netzstruktur) miteinander verbunden sind.

Das Wissen wird in neuronalen Netzen durch die Gewichte der Verbindungen und die spezifischen Funktionen der einzelnen Neuronen gespeichert. Im Gegensatz zu symbolischen Verfahren, bei denen einzelne Objekte über Symbole direkt bezeichenbar und benennbar sind, werden in einem neuronalen Netz die Objekte über ihre Eigenschaften repräsentiert. Die Eingangsinformation des Netzes stellt die wertemäßige Ausprägung der Eigenschaften (Merkmalsmenge / Attribute) der einzelnen Objekte dar.

Über einen Lernvorgang kann diese Information in eine interne Repräsentation überführt werden. Da die Beziehung zwischen interner Repräsentation eines Objektes und seiner externen symbolischen Bezeichnung in aller Regel nicht eineindeutig ist, spricht man von subsymbolischer Repräsentation (Ebene unterhalb der symbolischen Form). Nur im Falle einer extern eindeutigen diskreten Codierung bei gleicher Kardinalität der internen Merkmalsausprägungen und der Objektmenge wäre eine eineindeutige Zuordnung und damit eine symbolische Referenz möglich. Eine solche eindeutige Zuordnung ist aber in aller Regel nicht das Ziel des Lernvorgangs, es wird vielmehr versucht eine abstrakte Fassung der angebotenen Informationen zu lernen (Klassen der Eingangsmuster). Damit kann das Netz auch auf Eingaben reagieren, die sich von den gelernten unterscheiden und eine korrekte Klassifikation durchführen oder eine approximative Antwort berechnen (ohne die Kenntnis der vollständigen Objektmenge). Diese Fähigkeit wird auch unter dem Begriff Plastizität eingeordnet. Je nach Struktur des neuronalen Netzes und der dargebotenen Eingabewerte als Eigenschaftsausprägungen der zu lernenden Objekten erfolgt eine mehr oder weniger verteilte Repräsentation dieser Objekte über die am Lernvorgang beteiligten Neuronen.

Es kann eine vollständig verteilte oder aber auch eine "kleinräumige" Zuständigkeit bzgl. den das Objekt charakterisierenden Eingangswerten existieren. Im Grenzfall einer "kleinräumigen" Zuständigkeit bei klar definierbarer Menge an vorkommenden Objekteigenschaften kann z. B. bei einem neuronalen Netz mit einer versteckten Schicht jedem dieser versteckten Neuronen eigenschaftsgemäß ein externes Objekt zugeordnet werden. Eine ausschließliche Zuordnung bei einer erweiterten Objektmenge kann davon aber nicht abgeleitet werden.

Die Problemlösungseigenschaften des neuronalen Netzes - im Hinblick auf das spezielle Ausgangsproblem, für welches das neuronale Netz konstruiert wurde, entstehen durch das Zusammenspiel der Netzbestandteile, d. h. die Verhaltensmuster sind im neuronalen Netzwerk gespeichert. Dies bedeutet, daß die Zustände einzelner Neuronen bzw. ihre Verbindungen nicht mehr oder nicht mit vertretbarem Aufwand in Symbole überführt werden können, die für den Menschen verständlich sind. Der Lösungsweg kann deshalb nicht wie bei den Expertensystemen unter Bezugnahme auf angewendete Symbole und Operatoren a posteriori erklärt werden. Dies ist eine Folge der **subsymbolischen** Form der Verarbeitung.

Der Verarbeitungsfluß in einem neuronalen Netz muß nicht sequentiell wie bei konventionellen Softwaresystemen sein. Vielmehr können die einzelnen Verarbeitungselemente des neuronalen Netzes, wie die natürlichen Neuronen vom Ablauf her, nebenläufig ausgeführt werden. Auf einem Mehrprozessorsystem werden die Neuronen echt **parallel** tätig (in Abhängigkeit des Informationsflusses). In der Regel dienen nur einige ausgewählte Neuronen der Eingabe in das Netz und der Ausgabe; ihnen werden Eigenschaften der Objekte oder Konzepte der realen Welt zugeordnet. Voraussetzung für eine effiziente Parallelisierung ist die Lokalität bei der Verarbeitung in den einzelnen Neuronen. Diese wird durch entsprechende Funktionen gewährleistet /Rojas 102/.

Wegen der parallelen Verarbeitung und der verteilten Repräsentation zeichnen sich neuronale Netze durch ein gutes Näherungsverhalten aus (Musterklassifikation bei verrauschten Daten, Mustervervollständigung). Bei entsprechender Unterstützung durch Hard- bzw. Software können auch komplexere Problemstellungen mit zahlreichen und unsicheren Einflußfaktoren, wie sie vom Menschen zu Teil problemlos gelöst werden, sich jedoch bisher mit algorithmischen oder wissensbasierten Ansätzen nicht zufriedenstellend behandeln ließen, relativ schnell bearbeitet werden. Das herausragende Beispiel stellt hierbei die Bilderkennung dar.

6.2.2 Automatische Lernfähigkeit der internen Repräsentation

Eine wichtige Eigenschaft, die neuronale Netze von klassischen oder auch wissensbasierten Systemen unterscheidet, ist ihre Lernfähigkeit auf der Ebene von Mustern. Die Effektivität konventioneller Softwaresysteme hängt von der Validität und Korrektheit des Lösungsverfahrens ab. Wenn sich die Ausgangsbedingungen ändern, ist eine Analyse der Veränderungen notwendig, und das Lösungsverfahren muß explizit angepaßt, d. h. reimplementiert werden. Ähnliches gilt für Expertensysteme, da ihre Leistungsfähigkeit von der Qualität des akquirierten Wissens und dessen angemessener Repräsentation abhängig ist. Wenn die zu verarbeitende Situation unvorhersehbar war oder wenn sich der Wissensstand des Experten verändert, muß auch die Wissensbasis des Systems explizit aktualisiert werden.

Anders ist es hingegen bei neuronalen Netzen. Da ihre Problemlösungsfähigkeit induktiv aufgebaut wird, kann bei einer Veränderung der Ausgangsbedingungen das Netz durch Einspeisung der aktuellen Falldaten automatisch angepaßt oder gänzlich neu **trainiert** werden /Kurbel, Pietsch 62/. Durch das plastische Verhalten kann auch auf zuvor nicht direkt gelernte Eingabewerte eine Ausgabe erzeugt werden. Diese Fähigkeit neuronaler Netze, notwendige Beziehungen zu lernen, gleicht ihren Nachteil, daß sie in aller Regel nicht formal spezifizierbar und funktional synthetisierbar sind, aus; allerdings ist der Lernvorgang in großen Teilen experimenteller Natur (siehe hierzu auch Kapitel 5).

6.2.3 Phasen bei neuronalen Netzen

Im allgemeinen unterscheidet man zwei Phasen innerhalb der Entwicklung eines neuronalen Netzes:

(1) Die **Lernphase**, innerhalb derer sich das neuronale Netz, angestoßen durch vorgegebene Trainingsbeispiele gemäß einer definierten Lernregel (vgl. Abschnitt 6.4) verändert, um ein bestimmtes Lernziel, zum Beispiel gegebene Eingaben bestimmten Klassen zuzuordnen (automatische Klassifikation z. B. von Werkzeugteilen) zu erreichen.

(2) Die **Anwendungs- oder Funktionsphase**, innerhalb derer das Netzwerk unter Verwendung des Wissens aus den "gelernten" Trainingsbeispielen zu einer vorgegebenen Eingabe eine Ausgabe, d. h. das Ergebnis erzeugt.

Um einer Überschätzung / Fehleinschätzung der Möglichkeiten neuronaler Netze vorzubeugen, wird ausdrücklich darauf hingewiesen, daß neuronale Netze kein universelles Lösungsparadigma für eine vollständige und komplexe Anwendung bilden. Konnektionistische Lösungsansätze können eher als Bausteine gesehen werden, die für spezielle Aufgaben wie z. B. Datenvorverarbeitung, Bildverarbeitung, Musterverarbeitung etc. sehr gut geeignet sind.

Um in komplexen Anwendungsgebieten gute Ergebnisse zu erzielen, müssen neuronale Netze in Verbindung mit eher "klassischen" Formen, d. h. Algorithmen oder Methoden der Künstlichen Intelligenz eingesetzt werden. Ein wesentliches Problem besteht hierbei darin, wie neuronale Netze und klassische Methoden am effektivsten kombiniert werden können.

6.2.4 Klassifikationsmerkmale neuronaler Netze

Eine erste grobe Einteilung neuronaler Netze kann anhand zweier Eigenschaften erfolgen:

- der Architektur / dem internen Aufbau und
- dem operationalen Verhalten.

Unter dem Begriff der **Architektur** sind

- die Struktur des Netzes (Art der Verbindungen) und
- die Art der Bausteine, der Neuronen mit ihren internen Eigenschaften,

zu verstehen.

Das **operationale Verhalten** wird durch

- den Berechnungszeitpunkt für jedes Neuron (synchron / asynchron)
- die Informationsflußrichtung (vorwärts / rückgekoppelt (rekurrent))
- das Vorhandensein eines internen Gedächtnises im Neuron (Abhängigkeit der Reaktion auf Eingangssignale von der Vergangenheit)
- das Lernprinzip (überwacht / nicht-überwacht) und
- das Lernproblem
    - diskrete Klassifikation
    - approximative Berechnung (inklusive Mustervervollständigung)

charakterisiert. Die Architekturmerkmale werden im Abschnitt 6.3, das Lernverhalten im Abschnitt 6.4 behandelt.

6.3 Architektur neuronaler Netze

Unter der Architektur eines neuronalen Netzes versteht man seinen strukturellen Aufbau, d. h. die Art und die Verbindung der Elementarelemente (Neuronen) untereinander. In diesem Sinne kann man sich ein neuronales Netz als einen gewichteten gerichteten Graphen vorstellen; die Knoten stellen die Neuronen, die gewichteten Kanten stellen die Verbindungen zwischen den einzelnen Neuronen und die Richtung des Informationsflusses dar. Im folgenden werden die einzelnen Elemente neuronaler Netze beschrieben und ihre Funktion dargestellt.

6.3.1 Die Neuronen

Im Unterschied zur Biologie bzw. Neurologie benutzt die Theorie neuronaler Netze kein Modell, das alle Aspekte eines Neurons exakt beschreibt, sondern nur ein Modell, das eine sehr grobe Verallgemeinerung darstellt.

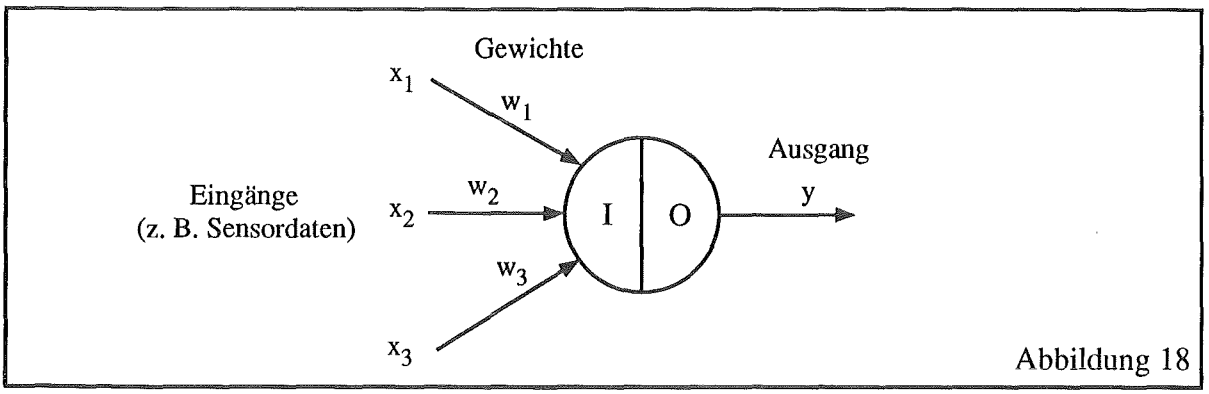

Abbildung 18

Das Grundmodell eines Neurons stützt sich im Wesentlichen auf die Vereinfachung von McCulloch und Pitts aus dem Jahre 1943, die ein Neuron als eine Art **Addierer mit Schwellenwert** betrachten (mehrere Eingänge, aber nur einen Ausgang).

In der Literatur wird diese Modellvorstellung für Neuronen häufig auch als Neuronen-Elemente, Prozeßelemente oder auch einfach Units bezeichnet. Die Verbindungen (Synapsen) eines Neurons nehmen eine Aktivierung x_i mit einer bestimmten Stärke w_i von anderen Neuronen auf, summieren diese und lassen dann am Ausgang y (Axon) des Neurons eine Aktivität entstehen, sofern die Summe vorher einen Schwellenwert T überschritten hat (siehe Abbildung 20) /Brause 12/.

Durch die Verwendung unterschiedlicher Funktionen zur Summation der Eingänge (z. B. euklidisches Skalarprodukt, gewichtete Summe über das Produkt der Eingaben, Hamming-Distanz etc.) zur Aktivierung eines Neurons und zur Berechnung der Ausgabe des Neurons (z. B. binär, linear, linear-begrenzt oder sigmoid), können verschiedene Formen von Neuronen gebildet werden.

Es sind in erster Linie die im folgenden beschriebenen Prozesse der Summationsfunktion, der Aktivierungsfunktion und der Übertragungsfunktion, die die Dynamik eines neuronalen Netzes bestimmen.

Mathematisch kann man die Input-Signale und die Gewichte als Vektoren (i_1, i_2, ..., i_n) und (w_1, w_2, ..., w_n) auffassen. Die Berechnung des gesamten Input-Signals erfolgt mit Hilfe einer **Summationsfunktion**, z. B. die gewichtete Summe, d. h. dem euklidischen Skalarprodukt der beiden Vektoren, was geometrisch als ein Maß für die Ähnlichkeit dieser beiden Vektoren aufgefaßt werden kann.

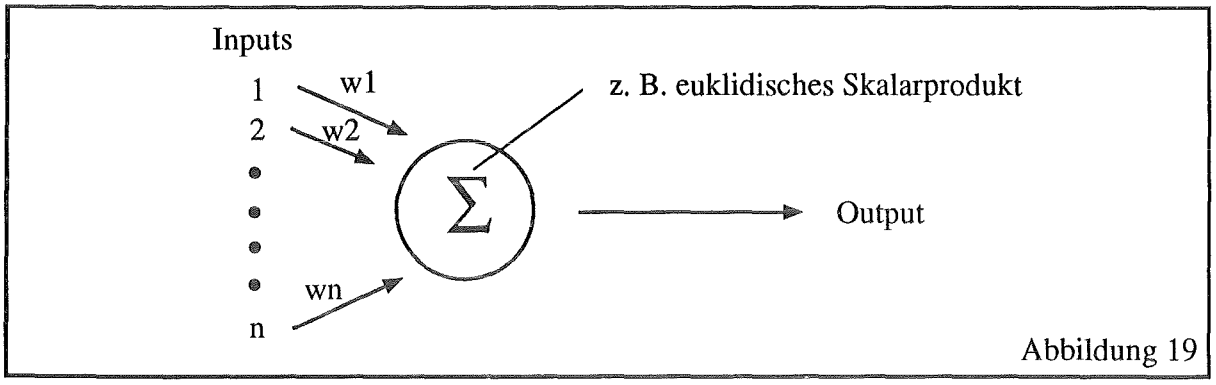

Abbildung 19

Wenn die gewichtete Summe der Input-Signale größer als der Schwellenwert ist, dann erzeugt das Prozeßelement ein Signal. Falls das Ergebnis der Summationsfunktion kleiner als der Schwellenwert ist, so wird kein Signal (oder ein hemmendes Signal) erzeugt; beide Antworten sind signifikant.

Eine wichtige Funktion ist die **Übertragungsfunktion** T (**Schwellenwert-**, **Output-Funktion**). Diese Funktion ist im allgemeinen nichtlinear, da lineare Funktionen aufgrund der Tatsache, daß ihr Resultat proportional zur Eingabe ist, nur beschränkte Möglichkeiten bieten und sich viele Probleme mit linearen Funktionen nicht lösen lassen (vgl. Abschnitt 6.4.2.1).

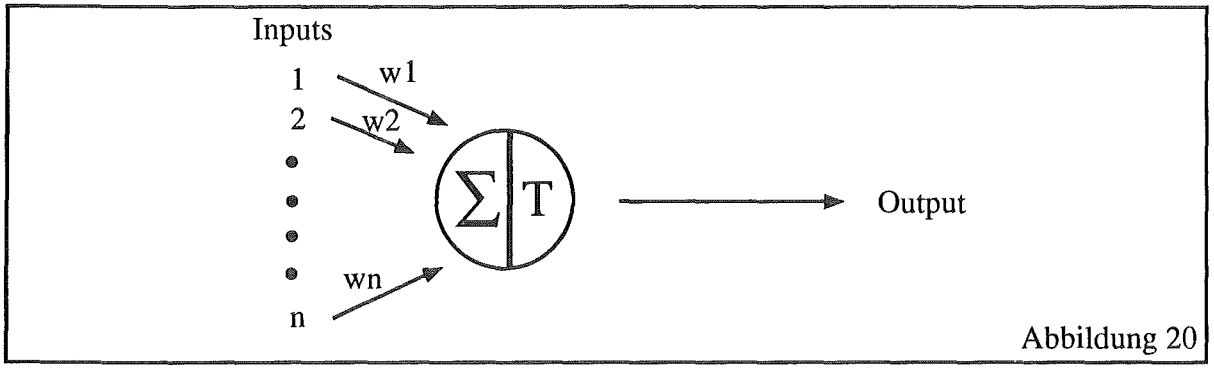

Abbildung 20

Die normalerweise üblichen Übertragungsfunktionen sind Treppenfunktionen oder sigmoide Funktionen, wobei in den Anwendungen sigmoide, d. h. S-förmige, Funktionen bevorzugt werden, da sie außer der Nichtlinearität noch über gewisse Differenzierbarkeitseigenschaften verfügen.

Die Anforderungen an eine Übertragungsfunktion bestehen darin, daß diese für negative oder zu kleine positive Werte ein definiertes Minimum (entsprechend der Ruhefrequenz beim Neuron) oder 0 liefert und ab einem gewissen Schwellenwert allmählich oder abrupt einen Maximalwert, häufig 1 annimmt (/Köhle 59/).

Ein weiterer Prozeß im Prozeßelement betrifft die **Aktivierungsfunktion**. Das Ergebnis der Summationsfunktion kann als Argument für eine Aktivierungsfunktion dienen, bevor es an die Übertragungsfunktion (T in Abbildung 20) übergeben wird. Der Sinn der Aktivierungsfunktion besteht darin, daß sie es ermöglicht, die Ausgabe in Abhängigkeit von der Zeit zu variieren. Die Aktivierungen früherer Zeitpunkte geben dem Prozeßelement ein Gedächtnis, mit dem beispielsweise Adaption modelliert werden kann. Allgemein ist die Aktivierungsfunktion von vorhergehenden Aktivierungen und von einem Satz von Parametern abhängig (/Köhle 59/). In den meisten Fällen wird als Aktivierungsfunktion die Identität verwendet, da die Untersuchungen auf diesem Gebiet noch nicht sehr weit fortgeschritten sind. Den Wert der Aktivierungsfunktion bezeichnet man als **Aktivierungswert**.

6.3.2 Die Anordnung von Neuronen zu mehreren Schichten (Layers)

Da ein neuronales Netz, das aus nur einem einzigen Neuron besteht, höchstens von theoretischem Interesse ist und sich wohl kaum für den Einsatz in der Praxis eignet, umfaßt ein neuronales Netz üblicherweise eine größere Anzahl an Neuronen.

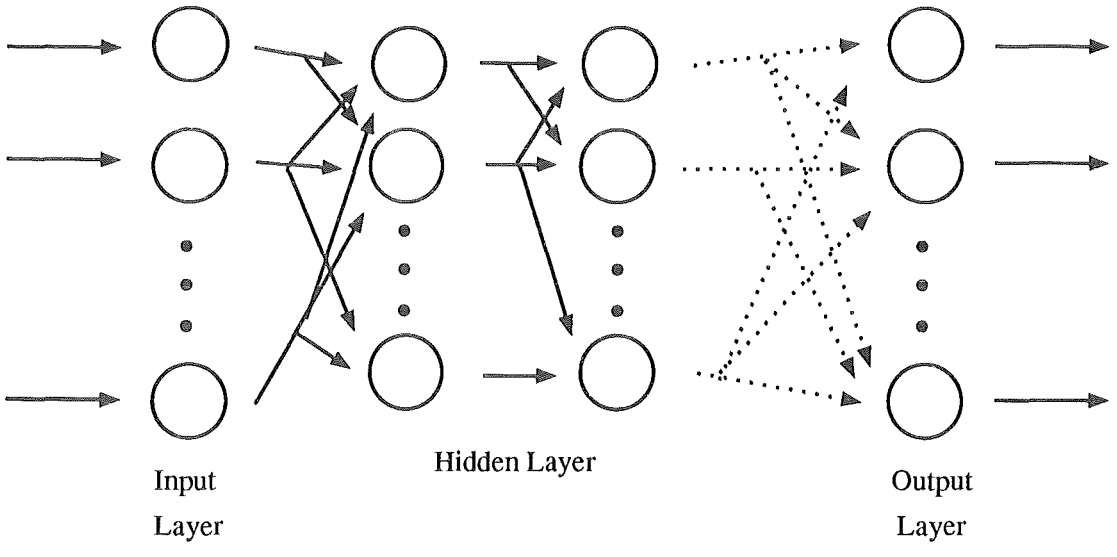

Abbildung 21

Zur Strukturierung werden Neuronen mit gleicher (paralleler) Funktion zu einem Funktionsblock (einer Schicht, Layer) zusammengefaßt. Die einzelnen Schichten werden in einer Folge "nacheinander" angeordnet und in geeigneter Weise miteinander verbunden. Jede Schicht ist wie ein Programm-Modul, das mit Hilfe der genau festgelegten Schnittstelle (Eingänge,

Ausgänge, etc.) eine festgelegte Spezifikation erfüllt, wobei die eigentliche Implementierung nicht entscheidend ist, solange die gewünschte Funktion tatsächlich erbracht wird.

Schichten, die weder aus Eingabe- noch aus Ausgabeeinheiten bestehen, werden als verdeckte Schichten (hidden Layer) bezeichnet; das Netz in Abbildung 21 besteht beispielsweise aus drei Schichten (zwei mit vollständig vernetzten Eingängen, eine davon als verdeckte Schicht). Die erste Schicht enthält die Eingabeeinheiten und dient ausschließlich der Verteilung der Eingabesignale; sie stellt eigentlich keine "echte" Schicht dar. Aufgrund der fehlenden Verbindungen zu den Ein- bzw. Ausgängen kann der Aktivitätszustand der Neuronen der verdeckten Schicht nicht direkt durch die "Außenwelt" festgelegt werden, sondern ist nur mittelbar über die internen Verbindungen des neuronalen Netzes beeinflußbar /Brause 12/. Die Neuronen einer verdeckten Schicht stellen die Grundlage zur Bildung neuer Merkmale und interner Repräsentationen dar /McClelland, Rumelhart 75/.

6.3.3 Strategien zur Verbindung einzelner Schichten (Layer)

Für die Anzahl und die Richtungen der Verbindungen innerhalb eines neuronalen Netzes sind theoretisch nur kombinatorische Grenzen gesetzt. Für die Richtung der Verbindungen und damit des Informationsflusses kann man grundsätzlich die folgenden Formen unterscheiden:

- **Feed-forward (vorwärts gekoppelte) Netze:**
  Die Eingänge einer Schicht werden ausschließlich von den Ausgängen der Schicht davor gespeist, d. h. die Neuronen einer Schicht haben keinerlei Einfluß auf einander und auf davorliegende Schichten. Die Grundidee dieser Charakterisierung besteht in der Existenz von Funktionseinheiten (Neuronen oder Neuronen-Schichten), die nicht rückgekoppelt sind.

  Graphentheoretisch bedeutet dies, daß der zugehörige gewichtete, gerichtete Graph zyklenfrei ist.

- **Rekurrente Netze:**
  Rekurrente Netze werden auch **feed-back (rückgekoppelte) Netze** genannt. In ihnen kann jedes Neuron mit sich oder mit jedem anderen Neuron des Netzes verbunden sein. Es handelt sich somit um Netzwerke, die Verbindungen und damit eine Informationsverarbeitung in alle Richtungen (d. h. auch Querverbindungen und Rückkopplungen) zulassen /Spies 115/. Da in einem feed-back Netz ein Neuron mit jedem anderen Neuron verbunden sein kann, haben feed-back Netze in der Regel nur eine Schicht. Um ein Ergebnis zu erhalten/produzieren, muß ein feed-back Netz dieses wiederholt berechnen, bis die Neuronen einen stabilen Zustand erreichen.

  Graphentheoretisch bedeutet dies, daß der zugehörige gewichtete, gerichtete Graph auch Zyklen enthalten kann.

6.4 Netzwerkmodelle und ihre Lernfunktionen

6.4.1 Lernen in neuronalen Netzen

Für das Lernen in neuronalen Netzen ist entscheidend, wie das Ausgabe- oder Reaktionsverhalten des neuronalen Netzes gebildet wird bzw. geändert werden kann und demnach auch, wie und wo das neuronale Netz etwas speichert, das sein Verhalten bestimmt. Das statische (gespeicherte) Wissen des neuronalen Netzes liegt in seinen Verbindungen (deren Gewichten) und in seinem Aufbau, das dynamische (aktuelle) ist in den Aktivierungswerten enthalten, die, wenn es sich um Neuronen der Ausgabeschicht handelt, die Reaktion des neuronalen Netzes auf ein präsentiertes Eingabemuster darstellen.

Das zu Erlernende (allgemein ein Muster) wird in neuronalen Netzen nicht explizit gespeichert. Vielmehr sind die Gewichte zwischen den einzelnen Prozeßelementen, die es erlauben, daß ein bestimmtes Muster immer wieder erzeugt werden kann, gespeichert. Lernen in einem neuronalen Netz bedeutet zumeist, die Gewichte richtig zu trainieren. Man geht dabei meist von einer Initialisierung mit Zufallszahlen aus und wendet dann eine Lernstrategie zur Veränderung der Gewichte des Netzes an. Dazu werden dem Netz wiederholt Muster präsentiert, die das Problem möglichst vollständig abdecken; es werden keine expliziten Regeln gelernt, sondern die Regeln werden anhand der Daten implizit mitgelernt (generalisiert).

Prinzipiell kann man drei Arten der Veränderung von Gewichten unterscheiden:

(1) Entwicklung neuer Verbindungen,

(2) Abbruch vorhandener Verbindungen,

(3) Veränderung der Gewichte bereits existierender Verbindungen.

Die Varianten (1) und (2) können als Spezialfälle von (3) angesehen werden, wenn dem Verbindungsabbau ein Setzen des entsprechenden Gewichtes auf 0 entspricht und dem Verbindungsaufbau einem Ändern des Gewichtes von 0 auf einen Wert ungleich 0.

Da noch keine Klarheit herrscht, nach welchen Kriterien der Auf- respektive Abbau der Verbindungen erfolgen soll, sind die Varianten (1) und (2) Gegenstand aktueller Forschungsvorhaben.

Eine auffallende Gemeinsamkeit der bekannten Lern-Algorithmen für neuronale Netze ist deren ausschließliches Verarbeiten lokaler Informationen. Außerdem ist interessant, daß sie zumeist ohne übergeordnete Instanz zu implementieren sind, d. h. ohne globale Ablaufsteuerung. Dadurch kann das Lernen in fast allen Fällen parallel und ohne globale Zugriffe vor sich gehen, was sich bei entsprechender Hardware sehr günstig auf den zeitlichen Aufwand auswirkt und sicherlich auch eher den Vorgängen beim Lernen des Menschen entspricht.

6.4.1.1 Verschiedene Formen des Lernens in neuronalen Netzen

Grundsätzlich kann man die verschiedenen Lernansätze nach dem Ursprung der zu erlernenden Muster, ähnlich dem Lernen aus Beispielen vgl. Abschnitt 3.3.4.1, einteilen. Hierzu unterscheidet man:

(1) **Überwachtes Lernen oder Klassifizieren:**
(supervised learning, learning by reinforcement)

Beim überwachten Lernen muß das neuronale Netz vor seinem praktischen Einsatz "trainiert" werden. Das Training besteht darin, daß man dem neuronalen Netz Eingabe- und Ausgabedaten, eine Trainingsmenge präsentiert; d. h. zu jedem gegebenen Eingabemuster präsentiert man dem neuronalen Netz das gewünschte vorklassifizierte Ausgabemuster.

Während der Lernphase wird die tatsächliche Ausgabe des neuronalen Netzes mit der gewünschten Ausgabe verglichen. Die Gewichte werden im gesamten neuronalen Netz so verändert, daß im nächsten Lernschritt eine bessere Übereinstimmung der Netzausgabe mit dem Muster erreicht wird. Das Ziel sämtlicher Lernalgorithmen besteht in einer nachhaltigen Minimierung des Fehlers zwischen dem gewünschten und dem tatsächlichen Ausgabemuster durch in der Regel kontinuierliche Modifizierung der Gewichte.

Wenn der Lernvorgang abgeschlossen ist, dann werden die Gewichte "eingefroren", d. h. es kann dann nicht weiter gelernt werden. Damit findet eine deutliche Unterscheidung zwischen Trainingsphase und Einsatzphase des neuronalen Netzes statt.

Beispiele überwachter Lernansätze sind:

(1) Selbstassoziation (Auto Association),

(2) Musterassoziation (Pattern Association),

(3) Klasseneinteilung (Classification Paradigm) und

(4) Fehlerpropagierung (Error (Back-) Propagation).

(2) **Nicht-überwachtes Lernen oder Clustering:**
(unsupervised learning, learning by doing)

Im Gegensatz zum überwachten Lernen benötigen die neuronalen Netze beim nicht-überwachten Lernen keine externe Beeinflussung zum Auslösen einer Anpassung ihrer Gewichte. Dafür verfügen sie über eine interne Möglichkeit (ein Optimierungskriterium, globales Gütekriterium) ihre Performanz zu überwachen.

Bei einem Algorithmus zum nicht-überwachten Lernen mit Hilfe neuronaler Netze kann der Wettbewerb zwischen den Neuronen die Basis für das Lernen bilden (sogenanntes Wettbewerbslernen). Das Training miteinander im Wettbewerb stehender Neuronen-Cluster verstärkt die Antwort bestimmter Neuronen-Gruppen auf spezielle

Reize, diese Gruppen werden miteinander und mit einer angemessenen Antwort verbunden. Während des Trainingsprozesses ordnen sich die einzelnen Neuronen selbständig bestimmten Musterklassen zu und werden damit zu Musterdetektoren.

6.4.2 Feed-forward-Netze

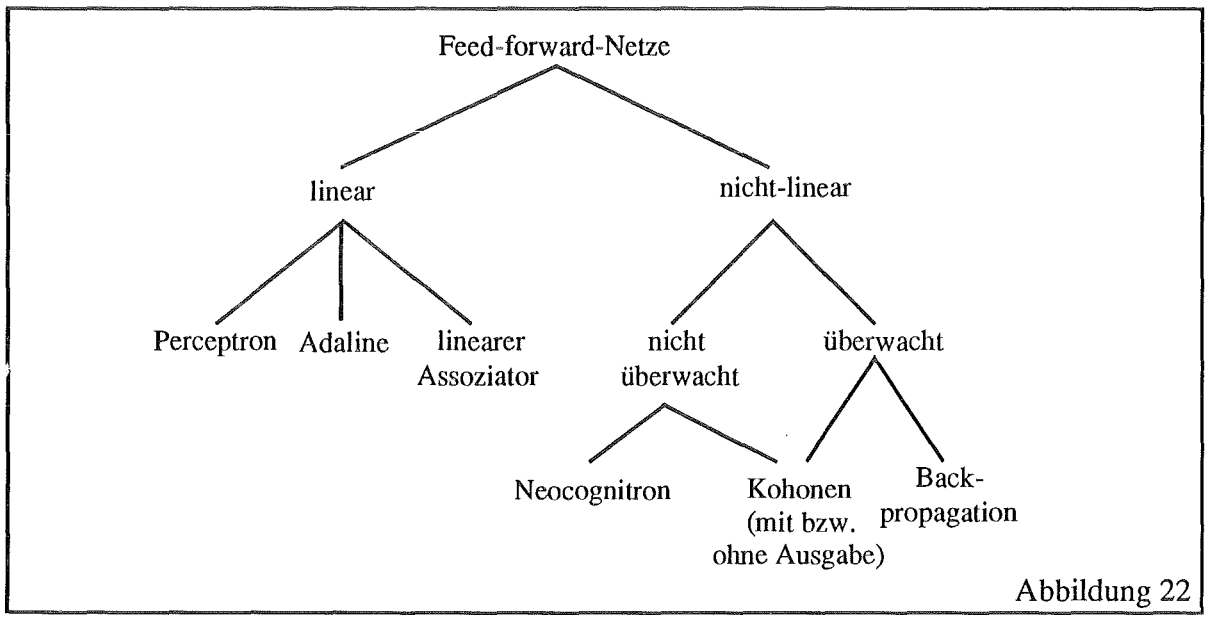

Abbildung 22

6.4.2.1 Perceptron

Das erste genauer beschriebene, algorithmisch orientierte Modell kognitiver Fähigkeiten wurde von dem Psychologen F. Rosenblatt 1958 vorgestellt. Das Grundschema der verschiedenen existierenden Perzeptronvarianten besteht aus einer Eingabe S (z. B. künstliche Retina zur Aufnahme physikalischer Energie wie Licht), einer festen Kodierung A (Assoziationsschicht typischerweise ein logisches Entscheidungselement), in der jede Einheit (Neuron) dieser Schicht mehrere Bildpunkte der Retina beobachten kann, und einer Verarbeitungsschicht R (Response-Schicht), in der jede Einheit lernen soll, nur genau auf eine Klasse von Eingabemustern der Retina anzusprechen. Die Verbindungen zwischen den Schichten S und A wurden als speziell ausgebildete, mehr oder weniger zufällig vorhandene, feste Zuordnungen verstanden; die Verbindungen zwischen A und R dagegen als modifizierbar (trainierbar) angenommen.

Die binäre logische Ausgabe y_i der Response-Schicht wird genau dann wahr, wenn die Aktivität des Neurons einen Schwellenwert T überschreitet. Eine Einheit der Response-Schicht kann somit als ein Neuron mit binärer Ausgabefunktion angesehen werden. In seiner ein-

fachsten Form kann ein Perceptron entscheiden, ob ein Eingabemuster einer von zwei vorgegebenen Klassen angehört (lineare Separierbarkeit) /Weis, Kulikowski 118/.

Die faszinierende, neue Idee dieses Ansatzes bestand nun darin, durch Eingabe verschiedener Trainingsmuster und ihrer Bewertung durch einen externen Lehrer (überwachtes Lernen) die Maschine dazu zu bringen, selbst die Gewichte für eine korrekte Klassentrennung richtig einzustellen.

Zum Lernen verwendete Rosenblatt in dem Perzeptron-Algorithmus eine "rückgekoppelte" Lernregel, die er back-coupled error correction nannte und als Vorläufer des Back-Propagation-Algorithmus (aus Abschnitt 6.4.2.2) angesehen werden kann. Bei dieser Lernregel wird die Differenz zwischen der gewünschten Ausgabe und der tatsächlichen Ausgabe des Netzwerkes dazu benutzt, die Gewichte der Verbindungen und den Schwellenwert so zu modifizieren, daß das Netz die ihm gestellte Klassifikationsaufgabe mit einer verbesserten Qualität löst.

Das berühmte Perceptron-Konvergenztheorem (vgl. z. B. /Minsky, Pappert 86/) besagt nun, daß der Perceptron-Algorithmus tatsächlich nach einer endlichen Anzahl von Iterationen bei konstantem Inkrement (der Lernrate) erfolgreich terminiert, falls die beiden Klassen linear separierbar (d. h. die Klassengrenze stellt im Merkmalsraum eine Hyperebene dar) sind. Anderenfalls, so zeigen Simulationen, wird die Klassengrenze bei der Iteration nur periodisch hin und her geschoben /Brause 12/, d. h. der Algorithmus oszilliert (Perceptron Cycling Theorem /Minsky, Pappert 86/. Der Fall linearer Separierbarkeit tritt in der Praxis jedoch recht selten auf. Damit sind Perceptrons beispielsweise nicht in der Lage, das XOR-Problem zu lösen. Allerdings wußten Minsky und Pappert bereits, daß jedes derartige für Perceptrons unlösbare Problem in ein für multilayer Perceptron lösbares Problem konvertierbar ist. Das Manko bestand jedoch darin, daß der Lernalgorithmus für das Perceptron auf ein Netzwerk mit mehr als einer Schicht in seiner ursprünglichen Form nicht anwendbar ist. Minsky und Pappert waren sogar davon überzeugt, daß für multilayer Perceptrons kein Lernverfahren gefunden werden kann /McClelland, Rumelhart 75/.

Damit zeigten die Untersuchungen von Minsky und Pappert, daß einfache Perceptrons prinzipiell nicht die "höheren" Leistungen zeigen können, die man von ihnen erwartete. Dies führte zu einer Ernüchterung der Forscher auf diesem Gebiet, die sich dann mehr dem Gebiet der Künstlichen Intelligenz und damit formalen, logischen (symbolischen) Methoden der Informationsverarbeitung zuwandten /Brause 12/.

6.4.2.2 Back-Propagation Netzwerke

Durch die Entwicklung des Back-Propagation Algorithmus als Lernverfahren für multilayer Netzwerke wurde nicht nur die Überzeugung von Minsky und Pappert widerlegt, sondern auch das gesamte Gebiet der konnektionistischen Verfahren erhielt einen neuen Aufschwung.

Die grundlegende Idee des Backpropagation-Algorithmus liegt in der Verbindung eines neuronalen Netzes auf der Basis von Perceptrons mit einer (nicht-linearen) Schwellenwertfunktion und einem Gradientenverfahren (vgl. auch Abschnitt 6.4.4).

Der Back-Propagation-Algorithmus ist immer dann einsetzbar, wenn ein neuronales Netz mit Beispielen trainiert werden soll, um eine unbekannte stetige Funktion $y = f(x)$ möglichst gut zu approximieren (vgl. Abschnitt 6.5).

Back-Propagation ist eine Verallgemeinerung der Delta-Regel (vgl. Abschnitt 6.4.4) auf Netzwerke mit beliebig vielen Schichten und einer feed-forward-Verarbeitung. Im Gegensatz zum Perceptron enthält jedes Neuron nun eine kontinuierliche sigmoide Ausgabefunktion mit Werten aus dem Intervall [0, 1]. Eine solche Funktion ist beispielsweise die Fermifunktion /Ritter, Martinetz, Schulten 101/:

$$\sigma(x) = (1 + \exp(x))^{-1}.$$

Die prinzipielle Idee ist, daß die verdeckten Neuronen eine interne Repräsentation der Musterassoziationen durch Rückwärtspropagieren eines Fehlers von der Ausgabeschicht in Richtung Eingabeschicht (error-back-propagation) erlernen.

Das Lernverfahren gliedert sich in zwei Schritte: Zuerst wird das angelegte Muster in Richtung Ausgabeschicht propagiert (durchlaufen), um dort die Reaktion des Netzes auf das präsentierte Muster zu generieren. In der zweiten Phase erfolgt die Gewichtsveränderung, abhängig vom Grad der Falschheit der Netzausgaben. Wie bereits im vorangegangenen Abschnitt erwähnt, wurde die grundlegende Idee des Fehlerrücksendens von Rosenblatt bereits 1962 formuliert.

Die Ausbreitung durch das Netz erfolgt schichtweise, d. h. die Aktivierungen der Neuronen werden zuerst in jener verdeckten Schicht berechnet, die der Eingabeschicht am nächsten liegt. Dann erfolgt die Berechnung der Aktivierung der nächsten, näher bei der Ausgabeschicht gelegenen Schicht, bis schließlich die Ausgabeschicht selbst erreicht wird.

In der zweiten Phase erfolgt die Fehlerbestimmung und die entsprechende Gewichtsveränderung wiederum schichtweise. Dabei wird bei der Ausgabeschicht begonnen, da hier das gewünschte Muster zur Verfügung steht (überwachtes Lernen) und mit dem tatsächlichen vom Netz produzierten Muster verglichen werden kann. Aus der Differenz dieser beiden Muster wird ein sogenanntes Fehlersignal gebildet, von dem einerseits die Änderung der Gewichte zwischen der Ausgabeschicht und dessen benachbarter verdeckter Schicht und andererseits auch die Berechnung des neuen Fehlersignals für die nächste verdeckte Schicht abhängt.

Wurde der Fehler bis zur letzten verdeckten Schicht zurückgesendet und wurden dabei alle Gewichtsänderungen vorgenommen, dann kann ein (neues) Muster angelegt und vorwärts propagiert werden. Wendet man diese Lernprozedur wiederholt an, so wird der Fehler

schrittweise verringert. Wichtig ist, daß man andere, bereits erlernte Muster mit diesem Vorgehen wieder zerstören kann (Übertrainieren). Der Lernvorgang wird, wenn der Fehler entsprechend klein geworden ist, als beendet angesehen und abgebrochen.

Da es sich bei dem Back-Propagation Algorithmus um ein Gradientenverfahren handelt, kann sich das neuronale Netz in einem lokalen Minimum verfangen und die gestellte Aufgabe nicht fehlerfrei erlernen (d. h. der "Algorithmus konvergiert nicht gegen die richtige Netzwerkantwort"). Bei binären Aufgabenstellungen ist das nicht störend, da die Entscheidbarkeit (und nicht die Fehlerfreiheit) ausschlaggebend ist.

Da es keinerlei Anhaltspunkte dafür gibt, daß Synapsen auch in umgekehrter Richtung verwendet werden können oder daß Neuronen Fehler (Abweichungen) rückwärts propagieren, ist Backpropagation als Modell des biologischen Vorbildes wenig plausibel /Hinton 41/.

6.4.2.3 Topologie-erhaltende Abbildungen (Kohonen)

Eine direkte Folgerung aus der Verbindung jedes Neurons einer Schicht mit jedem Neuron der Folgeschicht besteht darin, daß die Lage eines Neurons innerhalb einer Schicht keine Rolle für die ausgesandten bzw. empfangenen Verbindungen spielt. Bei dem im folgenden vorgestellten Modell hingegen spielt die Anordnung der Neuronen innerhalb einer Schichtstruktur jedoch eine wichtige Rolle /Ritter, Martinetz, Schulten 101/.

Die wichtigste Wechselwirkung zwischen Neuronen ist die gegenseitige Hemmung der Neuronen innerhalb einer neuronalen Funktionsgruppe (Schicht). Indem man allen Neuronen regelmäßig angeordnete Punkte eines Raumes (Koordinaten) zuordnet, werden zwischen den Neuronen Abstände und Nachbarschaften definiert. Die Wechselwirkungen innerhalb einer Schicht werden zu lokalen Wechselwirkungen, die bei verschiedenen Mitgliedern dieser Schicht verschieden wirken können. Diese Anordnung bietet besondere Möglichkeiten: Sie kann nicht nur eine Abbildung von hochdimensionalen Musterpunkten auf niederdimensionale (z. B. zweidimensionale Gitter-) Punkte bewirken, sondern dabei auch eine wichtige Eigenschaft besitzen, daß benachbarte Muster auch auf benachbarte Neuronen abgebildet werden.

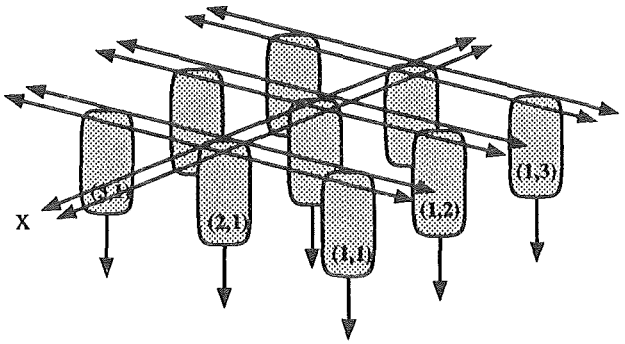

Abbildung 23

Eine Abbildung, die den Musterraum reduziert, verliert dabei zwar Information; um jedoch die Probleme bei der Trennung hochdimensionaler Musterklassen weiterhin zu verdeutlichen, sollte eine solche Abbildung die wichtige Eigenschaft besitzen, daß benachbarte Punkte im Musterraum auch im Bildraum benachbart bleiben. Abbildungen mit der Eigenschaft der Nachbarschaftserhaltung werden in der Mathematik auch topologie-erhaltend genannt /Brause 12/.

Es seien N Punkte $w_1, \ldots, w_n$ gegeben, die beispielsweise in einer zwei-dimensionalen Nachbarschaft zusammenhängen sollen. Assoziiert man mit jedem Punkt (Gewichtsvektor) eine Prozessoreinheit (Neuron), so hat bei einer einfachen rechteckigen Netzmasche jedes Neuron vier direkte Nachbarn. Als Eingabe erhält jedes Neuron parallel die gesamte Eingabeinformation des n-dimensionalen Mustervektors x (vgl. Abbildung 23, aus /Brause 12/).

Als Mittelpunkt einer Aktivierung kann nun ein einziges Neuron c mit den Koordinaten (i,j), als "selektiertes Neuron" mit der stärksten Aktivierung identifziert werden (winner-takes-all Prinzip). Das Gewicht des Neurons c und die Gewichte der "umliegenden" Neuronen werden durch eine Funktion modifiziert, die dem Querschnitt eines Sombrero gleicht und daher **Mexikanerhutfunktion** genannt wird. Durch die Mexikanerhutfunktion werden die Gewichte der Neuronen des Neuronengitters in Abhängigkeit von der Entfernung zu c verändert; d. h. die Gewichte der Neuronen die sehr nahe bei c liegen, werden erhöht, bei entfernteren Neuronen werden sie abgeschwächt. Daher ist nicht nur das Neuron c, sondern parallel dazu alle Neuronen k innerhalb der Nachbarschaft am Lernprozeß beteiligt. Den Konkurrenzkampf um die höchste Aktivierung gewinnt aufgrund des Prinzips der Nachbarschaftserhaltung nur dasjenige Neuron c, dessen Gewichtsvektor den kleinsten Abstand zum Eingabemuster x hat. Diese Form der Aktivierung bedeutet eine Quantisierung des Eingabemusters (**Vektorquantisierung**): Alle geringfügigen Variationen dieses Musters werden auf ein und denselben Gewichtsvektor abgebildet. Aufgrund dieser Vorgehensweise wird eine Aufteilung des Musterraumes in einzelne Klassen vorgenommen; der Gewichtsvektor ist der Klassenprototyp.

Im Zuge der Lernphase werden zum einen die Gewichtsvektoren benachbarter Neuronen in die gleiche Richtung korrigiert, so daß sie die Tendenz haben, ähnlich zu werden (**Selbstordnung**); zum anderen erhalten die Neuronen der Gitterrandpunkte als Eingabe mindestens die Punkte x am Rande der Verteilung, so daß ihre Gewichtsvektoren "an den Rand streben" und das Netz aus den Gewichtsvektoren im Laufe der Zeit auseinanderziehen (**Entfaltung**) /Brause 12/. Diese von Teuvo Kohonen entwickelte Lernregel wurde durch das Lernen in biologischen Systemen inspiriert und wird nur in nicht-überwachten-Lernsituationen angewandt.

6.4.2.4 Weitere Beispiele für feed-forward-Netze

Im folgenden werden drei weitere Beispiele für feed-forward-Netze skizziert /Brause 12/. Für weitere Beispiele sei auf die bereits mehrfach zitierte Literatur verwiesen.

(1) Assoziative Speicher:

Die in den heutigen Rechnern üblichen Hauptspeicher sind physikalisch als Listenspeicher organisiert, d. h. auf m Adressen $x^1, \ldots x^m$ werden Inhalte $y^1, \ldots, y^m$ gespeichert. Zum Abruf von y^i präsentiert man die physikalische Adresse x^i und erhält wieder den Inhalt (oder einen Zeiger darauf). Will man einen Inhalt finden, ohne dessen physikalische Adresse zu kennen (**inhaltsorientierte Adressierung**), so muß bei einer vollkommen zufällig zusammengestellten Liste im ungünstigsten Fall die gesamte Liste durchsucht werden. Dies wirkt sich bei verschiedenen (z. B. zeitkritischen) Anwendungen inhaltsorientierter Adressierung sehr ungünstig aus.

Ein assoziativer Speicher ist aus diesem Grund so organisiert, daß anstelle m physikalischer Adressen m **inhaltsorientierte Schlüsselworte** $x^1, \ldots x^m$ benutzt werden, zu denen die dazu assoziierten m Inhalte $y^1, \ldots, y^m$ als Tupel (x^i , y^i) möglichst effektiv abgespeichert werden. Für die interne Organisation werden in den verschiedenen Ansätzen unterschiedliche Möglichkeiten verwendet.

(2) Wettbewerbs-Lernen (Competitive Learning):

Die Wechselwirkung zwischen den einzelnen Neuronen besteht in der gegenseitigen Hemmung. Bei einer vorgegebenen Eingabe wirkt sich diese so aus, daß von allen Neuronen dasjenige mit der größten Aktivierung ausgewählt wird und die maximale Ausgabe $y_i = 1$ annimmt (winner-takes-all); die Ausgabe aller anderen Neuronen bleibt Null (d. h. maximal gehemmt).

Beim Wettbewerbs-Lernen, wie es von Rumelhart und Zipser in /Rumelhart, Zipser 104/ untersucht wird, geht man von einem hierarchischen Netzwerk aus, dessen Schichten - außer der Eingangsschicht - aus disjunkten Gruppen, sogenannte **Cluster**, sich gegenseitig hemmender binären Neuronen (mögliche Zustände aktiv oder inaktiv) bestehen. Jedes Neuron einer Schicht kann eine Eingabe von jedem Neuron der direkt darunterliegenden Schicht erhalten und selbst Signale zu jedem Neuron der direkt darüberliegenden Schicht aussenden. Die Verbindungen innerhalb der einzelnen Schichten wirken hemmend, diejenigen zwischen den Schichten erregend. Die einzelnen Neuronen eines Clusters hemmen sich gegenseitig, so daß immer nur ein einziges aktiv sein kann. Die aktiven Neuronen einer Schicht repräsentieren daher die Eingabemuster der nächst höheren Schicht. Ein gegebenes Cluster besitzt eine feste Anzahl an Neuronen, die Zahl der Neuronen kann jedoch von Cluster zu Cluster variieren.

Der Lernprozeß geht nun vonstatten, indem das aufgrund eines vorgegebenen Eingabemusters aktivierte Neuron eines jeden Clusters einer Schicht die Gewichte der Verbindungen zu aktivierten Clustern der darunterliegenden Schicht auf Kosten seiner restlichen Eingabe-Verbindungen erhöht. Alle anderen Verbindungen bleiben unbeeinflußt. Verringert man bei jedem Verarbeitungsschritt die Zahl der Neuronen pro Cluster und die Zahl der Cluster pro Ebene bis auf eins, so erhält man - aufgrund der

Aktivierung in der höchsten Ebene - schließlich eine diskrete Entscheidung (Klassifikation) über das präsentierte Muster.

In der Regel wird beim Wettbewerbs-Lernen eine konstante Lernrate gewählt; dadurch konvergiert der Algorithmus nicht zwangsläufig, sondern stark musterabhängig. Wählt man als **Lernrate** eine zeitabhängig gegen Null konvergierenden Funktion, so kann das Konvergenzverhalten während der Lernphase gesteuert (erzwungen) werden.

(3) Aufmerksamkeits-gesteuerte Systeme:

Um die Idee des Wettbewerbs-Lernens zur Mustererkennung einsetzen zu können, muß das oben erwähnte Problem musterabhängiger Konvergenz beseitigt werden. Eine Möglichkeit besteht darin, eine Kontrolle der Klassifizierung einzuführen. Hängt diese Kontrolle von einem Parameter ab, so läßt sich ein solcher Parameter als **Aufmerksamkeit** bezeichnen. Es ist allerdings anzumerken, daß es sich bei der ART-Architektur um kein reines feed-forward-Netz handelt, da auch Rückkopplungen im Sinne einer Hemmung (Gegenkopplung) verwendet werden. Die bekanntesten Arbeiten auf diesem Gebiet ist die ART (Adaptive Resonance Theory) von Gail Carpenter und Stephen Grossberg (z. B. /Brause 12/).

6.4.3 Rückgekoppelte Netze

Da bisher keine einheitliche Klassifikation von feed-back-Netzen bekannt ist, werden in der folgenden Abbildung nur einige Netzwerkkonzepte dargestellt:

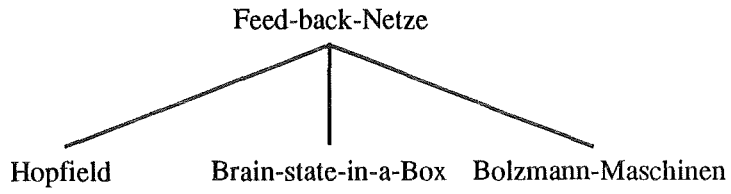

Abbildung 24

6.4.3.1 Hopfield-Netze

Das Hopfield-Modell (Hopfield, 1982) beschreibt eine Verbindung zwischen den magnetischen Anomalien in seltenen Erden (Spingläsern) und Netzen aus rückgekoppelten, binären Neuronen und interpretiert die für das Verhalten des Systems ausschlaggebende Zielfunktion als Energie. Viele physikalische Modelle für Spingläser gehen von Atomen aus, die sich gegenseitig beeinflussen, so daß das Energiepotential des Gesamtsystems einem Energieminimum zustrebt. Die Atome wirken dabei wie kleine Magnete (Spins), die in einem äußeren, konstanten Magnetfeld entweder parallel oder antiparallel orientiert sein können. Den Zustand der Dipolmagnete kann man auch mit "Spin auf" oder "Spin ab" oder den Zahlen +1 und -1 bezeichnen. Bezeichnet man die Kopplungskoeffizienten zwischen dem Atom i und dem Atom j mit w_{ij}, so ist das lokale Magnetfeld z_i bei Atom i eine Überlagerung aus den Zuständen y_i aller anderen Atome und der Wirkung eines äußeren Feldes T. Sind alle $w_{ij} > 0$, so ist das Material ferromagnetisch; wechselt das Vorzeichen von w_{ij} periodisch im Kristallgitter, so ist das Material antiferromagnetisch; bei einer zufälligen Verteilung des Vorzeichens spricht man von **Spingläsern** /Brause 12/.

Das Hopfield-Modell betrachtet die Wechselwirkung vieler, miteinander gekoppelter Einheiten und versucht, durch die Definition einer Zielfunktion ("Energie") die Konvergenz der Zustandsfolge vom initialen Zustand in einen stabilen Zustand zu zeigen. Dieser stabile Zustand entspricht einem lokalen Minimum der Zielfunktion. Durch die binären Ausgabefunktionen und die Nebenbedingungen eines Problems, die sich in der Definition der Gewichte widerspiegeln, existieren meist mehrere lokale Minima, die nicht mit dem globalen Minimum übereinstimmen müssen /Brause 12/.

Dieses Modell hat das gleiche Verhalten wie ein System aus n Neuronen mit binärer Ausgabefunktion (d. h. binären Neuronen) mit einer (um den Nullpunkt) zentrierten Ein- und Ausgabe { -1, +1 }. Jedes Neuron ist mit jedem anderen verbunden, das Hopfield-Netz zeigt keinerlei hierarchischen Aufbau, alle Neuronen sind sowohl Eingabe- als auch Ausgabeeinheiten /Köhle 59/. Der Wert y_i der Ausgabe eines Neurons wird als sein Zustand betrachtet; der Vektor $y := (y_1, \ldots, y_n)$ als Systemzustand. Bei symmetrischen Kopplungen der Neuronen sind die Gewichte der Verbindungen zwischen den Neuronen ebenfalls symmetrisch,

d. h. $w_{ij} = w_{ji}$, $i,j \in \{ 1, \ldots, n \}$, $i \neq j$. Da Rückkopplungen eines Neurons auf sich selbst nicht zugelassen sind, gilt $w_{ii} = 0$, $i \in \{ 1, \ldots, n \}$. Analog zum zugrundeliegenden physikalischen Modell strebt das Hopfield-Netz einem energieminimalen Zustand zu; d. h. ein Aktivierungszuwachs eines Neurons i und damit eine Zustandsveränderung darf nur mit einer Energieminderung verbunden sein (deterministisches Verfahren). Der Zustand der Neuronen wird mit dem Ziel der Minimierung des Energieniveaus so lange manipuliert, bis eine für die aktuelle Formulierung von Zielzustand und Restriktionen (durch die Energiefunktion) optimale Lösung gefunden wird /Kurbel, Pietsch 62/.

Das Hopfield-Modell läßt sich besonders gut bei Problemen anwenden, bei denen viele einzelne, unabhängige Größen existieren, die sich miteinander koordinieren müssen, um gemeinsam eine optimale Lösung zu bilden. Sie streben "unaufhaltsam" in Richtung minimaler Energie und somit unter Umständen ebenso "unaufhaltsam" in eine lokales Minimum. Damit besteht das wesentliche Problem bei diesem Ansatz darin, das globale Minimum bzw. ein möglichst gutes, suboptimales, lokales Minimum zu finden /Brause 12/.

Hierzu können unterschiedliche Techniken bzw. Weiterentwicklungen des Hopfield-Modells eingesetzt werden. Eine solche Technik ist das Simulated Annealing. Es handelt sich hierbei eigentlich um eine Weiterentwicklung des bekannten Monte Carlo Algorithmus von Metropolis et al. /Metropolis, Rosenbluth, Teller 76/, welcher ursprünglich dazu benutzt wurde, Mittelwerte über große Systeme der statistischen Mechanik numerisch zu berechnen.

Die Idee läßt sich folgendermaßen skizzieren. Zur approximativen Lösung eines Optimierungsproblems, dem Finden einer optimalen Konfiguration, beginnt man mit einer zufälligen Startkonfiguration und verändert diese durch stochastische Mutationen. Eine Mutation ist hierbei eine kleine Veränderung der Konfiguration: Etwa bei dem als Travelling-Salesman-Problem bekannten Optimierungsproblem werden in einer Rundtour entweder zwei Kanten oder zwei Städte vertauscht. Dabei werden Mutationen, die eine Verbesserung (im Sinne einer Verkürzung der Rundtour) erzielen, stets akzeptiert. Insoweit entspricht es einem (konventionellen) lokalen Optimierungsverfahren. Um jedoch auch lokale Optima prinzipiell wieder verlassen zu können, werden Verschlechterungen um ΔE mit der Wahrscheinlichkeit $e^{\Delta E / T}$ in Abhängigkeit von einem sogenannten Temperaturparameter T akzeptiert. Das sogenannte Annealing Schedule $\lambda n * T(n)$ bestimmt nun, wie die Temperatur Schritt für Schritt herabgesetzt wird /Braun 11/.

6.4.3.2 Weitere Beispiele für rückgekoppelte Netze

Im folgenden werden zwei weitere Beispiele für rückgekoppelte Netze skizziert /Brause 12/. Für weitere Beispiele sei auf die bereits mehrfach zitierte Literatur verwiesen.

(1) Brain-state-in-a-box:

Dieses klassische Modell von Anderson, Silverstein, Ritz und Jones, 1977 besteht aus einer Erweiterung des korrelativen Assoziativspeichers (einem speziellen Assoziativspeicher). Die Erweiterung besteht darin, daß die Ausgänge auf die Eingänge zurückwirken und sich dadurch eine Rückkopplung ergibt.

(2) Bolzmann-Maschinen:

Bei einer Bolzmann-Maschine handelt es sich um ein modifiziertes Hopfield-Netzwerk, das um zusätzliche Einheiten (hidden-units) erweitert wurde. Durch diese Einführung einer geschichteten Struktur ist im Unterschied zum Hopfield-Modell keine vollständige Vernetzung mehr gegeben.

Die grundlegende Funktion einer Bolzmann-Maschine besteht in einer **Relaxation**, dem Einstellen des Gleichgewichtes bei Vorgabe bestimmter Nebenbedingungen. Beim Übergang von einem Zustand zu einem anderen wird durch Angabe von Übergangswahrscheinlichkeiten versucht, das "Festkleben" in lokalen Minima zu vermeiden. Mit einer nicht-deterministischen Ausgabefunktion ergibt sich ein Gleichgewicht im Netz, das durch eine Bolzmann-Verteilung für einen optimalen Zustand mit minimaler Energie charakterisiert wird (die Wahrscheinlichkeitsverteilung für beliebige Zustände geht im Grenzwert in eine Bolzmann-Verteilung für einen optimalen Zustand mit minimaler Energie über). Als Gütekriterium zur Bewertung der einzelnen Zustände wird in Bolzmann-Maschinen die Information oder **Entropie** eines Zustandes (bzw. die Minimierung der fehlenden Information) verwendet (vgl. auch Abschnitt 4.1.1).

6.4.4 Allgemeine Lernregeln für Modelle neuronaler Netze

Im folgenden werden einige allgemeine Lernregeln für neuronale Netze kurz vorgestellt, die sich entweder keinem der bereits behandelten Netztypen direkt zuordnen lassen oder die Grundlage für eines der dargestellten Verfahren bilden /McCord Nelson, Illingworth 74/, /Brause 12/, /Köhle 59/:

(1) Hebb-Regel:

Dies ist die älteste und am weitesten verbreitete Lernregel. Sie besagt, wenn die Neuronen a und b zugleich (wiederholt) stark aktiviert sind, so erhöht sich die Stärke ihrer Verbindung.

Die Hebbsche Lernregel kann jedoch nur dann erfolgreich angewendet werden, wenn die Eingabemuster zueinander orthogonal sind /Spies 115/.

(2) Gradienten-Abstiegs-Regel (Gradientenverfahren):

Dies ist ein mathematischer Ansatz, um den durch eine "Gütefunktion" ausgedrückten Fehler zwischen den aktuellen und den gewünschten Ausgaben zu minimieren. Die Gewichte werden um einen Betrag modifiziert, der proportional zum Wert der ersten Ableitung der Fehlerfunktion bzgl. der Gewichte ist.

Obwohl das Konvergenzverhalten sehr langsam ist, wird diese Methode sehr häufig verwendet. Die Delta-Regel ist ein Spezialfall dieser Methode.

Im wesentlichen bestehen die folgenden (allgemeinen) Probleme bei Gradienten-Abstiegs-Methoden:

- Sie sind auf die Existenz von Ableitungen angewiesen,
- sie können zwischen zwei "Lösungen" oszilieren, ohne zu konvergieren,
- aufgrund der lokalen Strategie (das von ihnen gesuchte Minimum ist das "beste" in einer Umgebung des Ausgangspunktes) können sie gegen eine falsche Lösung (lokales Minimum) konvergieren; dieses Problem tritt jedoch bei allen Hill-Climbing-Methoden auf.

(3) Die Delta-Regel:

Diese häufig verwendete Regel basiert auf der einfachen Idee, die Stärke der Verbindungen kontinuierlich zu modifizieren, um die Differenz (das Delta) zwischen dem gewünschten Ausgabewert und dem momentanen Ausgabewert eines Neurons zu verringern. Im Unterschied zur Hebb-Regel wird nicht das Produkt aus Ein- und Ausgabe zur Modifikation (Verbesserung) des Verknüpfungsgewichtes herangezogen, sondern nur das Produkt aus Eingabe und Abweichung der tatsächlichen Ausgabe vom vorgegebenen "Sollwert" des Trainingssignals /Spies 115/.

Unter der Bedingung, daß alle Eingänge des neuronalen Netzes linear unabhängig sind (dies ist eine schwächere Bedingung als die bei der Hebbschen Lernregel geforderte Orthogonalität), wird durch diese Lernregel das Lernen beliebiger Assoziationen möglich /Lawrence 67/.

Diese Regel wird auch **Widrow-Hoff-Lernregel** oder Methode der kleinsten Fehlerquadrate (least mean square learning rule) genannt und wurde von Widrow und Hoff in deren ADALINE (ADAptive LINear Elements) Netzwerkmodell erfolgreich eingesetzt.

(4) Grossbergs Lernregel:

Grossbergs Lernregel kombiniert die Hebb-Regel mit dem biologischen Vorgang des Vergessens.

In Grossbergs Modell wird jedes neuronale Netz aus Instars und Outstars gebildet; ein Instar ist ein Neuron, das viele Eingaben empfängt, ein Outstar dementsprechend eines, das seine Ausgaben zu vielen anderen Neuronen sendet. Hier erlauben die Verbindungen den Rückruf eines konzentrierten Bildes von einem einzigen Outstar-Knoten; das Muster wird verteilt gespeichert.

Falls ein Knoten sowohl eine hohe Input- als auch eine hohe Outputaktivität aufweist, so werden die zugehörigen Gewichte signifikant verändert. Falls entweder der gesamte Input oder die Ausgaben kleine Werte annehmen, so werden die Veränderungen der Gewichte ebenfalls gering ausfallen, und die Gewichte können auf unwichtigen Verbindungen sogar gegen 0 gehen.

Die Zeit spielt bei Grossbergs Lernmodell eine wesentliche Rolle. Falls ein Eingabereiz weggelassen wird, so wird im Laufe der Zeit, wenn das Vergessen einsetzt, auch die Antwort (in der Ausgabe) auf diesen Reiz nachlassen.

Falls die Wahrscheinlichkeiten der Zustände im Modell denjenigen der Umwelt entsprechen, dann verfügt das neuronale Netz über ein abstraktes Modell der Umwelt.

6.5 Theoretische Aussagen über die Fähigkeiten neuronaler Netze

6.5.1 Approximation stetiger Funktionen

Die Grundproblematik neuronaler Netze besteht darin, eine gewünschte Funktion mit Hilfe vieler Einzelfunktionen zu erreichen. Ähnlich dem Stone-Weierstraß Theorem, das in einem kompakten topologischen Raum die beliebig dichte Approximation von stetigen Funktionen durch Polynome garantiert, zeigten Hornik, Stinchcombe und White in /Hornik, Stinchcombe, White 44/, daß sich mit Hilfe der neuronalen Funktionen eines Netzwerkes aus drei Schichten und Neuronen mit nicht-linearen Ausgabefunktionen, ähnlich dem in Abbildung 21 dargestellten, eine vorgegebene Menge von diskreten Funktionswerten direkt darstellen oder in einem Intervall eine kontinuierliche Funktion beliebig gut approximieren läßt.

Diese Aussage wurde von den gleichen Autoren sogar noch auf die folgende Form erweitert: Neuronale Netze mit verborgenen Knoten und einer kontinuierlichen Aktivierungsfunktion sind sogar in der Lage, jede meßbare Funktion zu simulieren /Levelt 70/.

Ungeklärt sind jedoch die Fragen nach der Anzahl der benötigten Neuronen, bei vorgegebener Approximationsgüte und welche Eigenschaften der zu approximierenden Funktion bei der Bestimmung der Anzahl der benötigten Neuronen eine Rolle spielen. Cybenko geht in /Cybenko 19/ jedoch davon aus, daß für die überwiegende Mehrheit an Approximationsproblemen eine astronomische Anzahl von Neuronen benötigt wird. Aus diesem Grund sind die theoretischen Ergebnisse zumindest von streitbarer praktischer Relevanz.

Von Hartley und Szu wurden in /Hartley, Szu 38/ verschiedenen neuronale Netzwerkmodelle im Hinblick auf ihre Mächtigkeit (im Sinne von möglichen Berechnungen) untersucht. Sie kamen dabei zu den folgenden Ergebnissen:

- Falls das Netzwerkmodell aus nur endlich vielen Neuronen besteht und jedes Neuron nur endlich viele Zustände annehmen kann, so ist das neuronale Netzwerkmodell und ein endlicher Zustandsautomat äquivalent.

  Bei einem **endlichen Zustandsautomaten** handelt es sich um das mächtigste Berechnungsmodell mit einer endlichen Anzahl von Zuständen. Man kann sich einen endlichen Zustandsautomaten als ein Tableau vorstellen, das einen neuen Zustand als Funktion von einem alten Zustand und einer Eingabe enthält.

- Falls das neuronale Netzwerkmodell aus unendlich vielen Neuronen (mit endlich oder unendlich vielen Zuständen) oder aus endlich vielen Neuronen mit unendlich vielen Zuständen besteht, so ist es äquivalent zu einer Turing-Maschine.

  Mit realen Komponenten kann aufgrund bestehender Hardwareeinschränkungen kein Neuron mit unendlich vielen unterscheidbaren Zuständen realisiert werden.

  Eine **Turing-Maschine** besteht aus einem endlichen Zustandsautomaten, einem unendlichen Band (zur Speicherung) und einem bewegbaren Schreib- und Lesekopf zur Verbindung der beiden.

6.5.2 Tabellarische Darstellung der wichtigsten neuronalen Modelle

Netzwerk	Erfinder/Entwickler	Jahr	Primäre Anwendungen	Einschränkungen	Bemerkungen
Perceptron (mit und ohne Hidden-Layer)	Frank Rosenblatt, Cornell University.	1957	Erkennung gedruckter Buchstaben	Komplexe Buchstaben (z. B. chinesische Schriftzeichen) können nicht erkannt werden; überempfindlich gegenüber Veränderungen in Größe, Translation und Drehung.	Das älteste bekannte Netzwerk; hardwareimplementiert; heute selten verwendet; li-. neare Schwellenwert-Prozeßelemente.
Adaline/ Madaline (ADAptive Linear Elements und Multiple (parallele) ADALINEs)	Bernard Widrow, Stanford University.	1960-62	Adaptives Auslöschen von Radar-Störungen; adaptive Modems; adaptive Gleichrichter (Echo-Unterdrükkung) in Telefonleitungen.	Eine lineare Beziehung zwischen dem Input und Output wird vorausgesetzt.	Mächtige Lernregel; seit über 20 Jahren in kommerziellem Einsatz (adaptive Rauschunterdrückung in der Telekommunikation); linere Schwellenwert-Prozeßelemente.
Avalanche	Stephen Grossberg, Boston University.	1967	Kontinuierliche Spracherkennung; Lehren von motorischen Kommandos für Roboter-Arme.	Keine einfache Möglichkeit, die Geschwindigkeit zu verändern oder Bewegungen zu interpolieren.	Man betrachtet nur Klassen von Netzwerken, kein Einzelnetzwerk kann alle diese Aufgaben übernehmen; Grossberg Schwellenwert-Prozeßelemente.
Cerebellatron	David Mar, MIT; James Albus, NBS; Andres Pellionez, NYU.	1969-82	Kontrolle motorischer Aktionen von Robotor-Armen.	Es wird eine komplizierte Kontrolle der Eingabe benötigt.	Ähnlich wie das Avalanche Netzwerk kann es verschiedene Kommandosequenzen mit verschiedenen Gewichten zur Interpolation von Bewegungen (so fein wie nötig) verbinden.
Back Propagation	Paul Werbos, Harvard University; David Parker, Stanford Uni-University; David Rumelhart, Stanford University.	1974-85	Sprach-Synthese aus Text; adaptive Kontrolle von Roboter-Armen; Bewertung von Kreditvergaben im Bankwesen.	Nur überwachtes Lernen - korrekte Eingabe- Ausgabebeispiele müssen in großem Umfang dargeboten werden.	Heute das populärste Netzwerk - es arbeitet sehr gut und ist einfach zu erlernen; sigmoide. Prozeßelemente.
Brain-State in a Box	James Anderson, Brown University.	1977	Extraktion von Wissen aus Datenbanken	Eine Entscheidung wird in einem Schritt getroffen - keine iterativen Schlußfolgerungen.	Ähnlich zu bidirektionalen assoziativen Speichern zur Vervollständigung fragmentaler Eingaben; prinzipiell sind alle Elemente miteinander verbunden; Feedbacks sind zugelassen.

Netzwerk	Erfinder/Entwickler	Jahr	Primäre Anwendungen	Einschränkungen	Bemerkungen
Neocognitron	Kunihiko Fukushima, NHK Laboratories.	1978-84	Erkennung von handgeschriebenen Buchstaben	Es benötigt eine ungewöhnlich hohe Anzahl von Prozeßelementen und Verbindungen.	Das komplizierteste, je entwickelte Netzwerk; unempfindlich gegenüber Unterschieden in Größe, Translationen, Rotation; fähig selbst komplizierte Buchstaben zu erkennen.
Adaptive Resonance Theorie	Gail Carpenter, Northeastern University; Stephen Grossberg, Boston University.	1978-86	Muster-Erkennung, speziell wenn das Muster kompliziert oder für Menschen ungewohnt (z. B. Radar oder Sonar) ist.	Überempfindlich gegenüber Translationen, Drehungen, Veränderungen der Größe.	Sehr hochentwikkelt; es wurde noch nicht auf viele Probleme angewandt.
Self-Organizing Map	Teuvo Kohonen, Helsinki University of Technology.	1980	Bildet eine geometrische Region (z. B. ein rechtwinkliges Gitter) auf ein anderes (z. B. ein Flugzeug) ab.	Extensives Training ist erforderlich.	Effektiver als viele algorithmische Techniken zur Berechnung aerodynamischer Strömungen; sehr schnell (real-time); kann kontinuierlich Lernen.
Hopfield	John Hopfield, California Institute of Technology and AT&T Bell Laboraties.	1982	Erkennen von kompletten Daten oder Bildern aus Fragmenten.	Nicht lernfähig, die Gewichte müssen im voraus festgelegt werden.	Kann in einem großen Anwendungsbereich implementiert werden; thermodynamische Prozeßelemente.
Bidirectional associative-Memory (BAM)	Bart Kosko, University of Southern California.	1985	Inhaltsadressierbarer assoziativer Speicher.	Geringe Speicherdichte; die Daten müssen einwandfrei kodiert werden.	Am leichtesten zu lernendes Netzwerk - ein gutes Werkzeug in der Ausbildung; Fragmentpaare von Objekten werden mit kompletten Paaren assoziiert.
Boltzmann und Cauchy Maschinen.	Jeffrey Hinton, University of Toronto; Terry Seynowsky, John Hopkins University; Harold Szu, Naval Research Laboratories.	1985-86	Muster-Erkennung für Bilder, Sonar und Radar.	Boltzman Machine: Lange Trainingszeiten; Cauchy Maschine: Erzeugt Störungen in statistisch einwand--freien Situationen.	Einfache Netzwerke in denen eine Störungsfunktion zum Finden eines globalen Minimum verwendet wird; thermodynamische Prozeßelemente.
Counterpropation	Robert Hecht-Nielsen, Hecht-Nielsen Neurocomputing Corporation.	1986	Kompression von Bildern; Statistische Analysis; Bewertung von Kreditvergaben im Bankwesen.	Bei jeder Problemgröße benötigt man eine große Anzahl an Prozeßelementen und Verbindungen, um einen hohen Genauigkeitsgrad zu erreichen.	Funktionen als ein selbst-programmierender look-up-table; ähnlich zur Back Propagation, nur einfacher und daher auch nicht so mächtig.

6.5.2.1 Prozeßelemente und Lernparadigmen

In der folgenden Tabelle werden wie bei /Köhle 59/, ergänzt durch /McCord Nelson, Illingworth 74/, die bereits in Abschnitt 6.4.4 behandelten klassischen Modelle für Prozeßelemente und die zugehörigen Lernparadigmen dargestellt:

	Summationsfunktion	Aktivierungsfunktion	Übertragungsfunktion	Lernmodus	Lernparadigma
Einfache lineare Prozeßelemente	Euklidisches Skalarprodukt.	keine	linear	überwacht	Hebb-Regel
Lineare Schwellenwertelemente	Euklidisches Skalarprodukt.	keine	Schwellwert	überwacht	Delta-Regel
Brain State in a Box	Euklidisches Skalarprodukt.	eigene	keine	überwacht	Hebb-Regel
Self Organizing map	Euklidisches Skalarprodukt.	eigene	sigmoid	nicht überwacht	eigene
Grossberg-Prozeßelemente	Euklidisches Skalarprodukt.	eigene	sigmoid	überwacht	eigene
Sigma-Pi-Prozeßelemente	Gewichtete Summe über das Produkt der Eingaben.	beliebig	beliebig	überwacht	beliebig
P-Prozeßelemente	Euklidisches Skalarprodukt.	eigene	Schwellwert	überwacht	eigene
Thermodynamische Prozeßelemente	Euklidisches Skalarprodukt.	keine	stochastisch sigmoid	überwacht	Cachy-Boltzmann Maschinen

6.6 Optimierung der Netzwerk-Architektur mit genetischen Algorithmen

Will man neuronale Netze zur Lösung eines konkreten Problems einsetzen, so stellt sich zunächst die Frage nach der Auswahl eines geeigneten Netzwerktyps, einer Lernregel (bzw. geeigneter Parameter) und dann nach der konkreten Architektur (Anzahl der Layer, Anzahl der Neuronen pro Layer). Zur Beantwortung dieser Fragen gibt es derzeit keine allgemeinen Aussagen (siehe auch Abschnitt 6.8).

Ein Ansatz besteht darin, genetische Algorithmen zur Bestimmung / Optimierung von Netzwerkstrukturen einzusetzen. Die allgemeine Vorgehensweise kann im wesentlichen wie folgt dargestellt werden:

- Die Architektur, d. h. die Anzahl der Neuronen und deren Verbindungen wird in einen genetischen Code übersetzt (geordnete Menge von (binär) Ziffern).
    - Für die Initialisierung wird eine bestimmte Anzahl derartiger Codierungen als "Anfangspopulation" (in der Regel zufällig) erzeugt.
- Die Mitglieder dieser Anfangspopulation (dies sind neuronale Netze) werden anhand der geforderten Aufgabe getestet und aufgrund ihrer Performanz bewertet (Tauglichkeitsfunktion).
- Diejenigen Mitglieder mit dem höchsten Tauglichkeitsgrad werden zur Bildung der nächsten Generation nach den a priori festgelegten Regeln des genetischen Algorithmus herangezogen.
- Diese Vorgehensweise wird solange fortgesetzt, bis keine Verbesserung der Performanz mehr feststellbar ist; Mutation kann eingesetzt werden, um eine Konvergenz gegen ein lokales Minimum zu vermeiden.

Eine vollständige Übersicht über die Literatur, die diesen Themenbereich betrifft, würde den hier vorgegebenen Rahmen sprengen. Aus diesem Grund werden im folgenden nur einige wenige, jedoch charakteristische Arbeiten zitiert. Montana und Davis /Montana, Davis 89/, Whitley et al. /Whitley et al. 119/ und Scholz /Scholz 109/ haben genetische Algorithmen benutzt, um Backpropagation Netzwerke mit vorgegebener Struktur zu trainieren. Das Ziel ist es, durch den Einsatz genetischer Algorithmen robustere Lernverfahren zu erhalten. Miller et al. /Miller et al. 85/ haben Nebenbedingungen für die Verbindungsmatrix von Backpropagation Netzen untersucht. Schaffer, et al. /Schaffer et al. 106/ und Harp und Samad /Harp, Samad 37/ verwendeten einen genetischen Algorithmus, um die Größe, die Struktur und die Lernparameter eines neuronalen Netzes zu bestimmen.

6.7 Praktische Anwendungen neuronaler Netze

6.7.1 Anwendungen des Backpropagation-Algorithmus

6.7.1.1 Steuerung einer Folgefahrt

Berns und Hofstetter /Berns, Hofstetter 5/ verwenden einen Backpropagation-Algorithmus zur Steuerung des Verhaltens (entkoppeltes) "Folgen" eines mobilen Roboters. Der mobile Roboter soll dabei einem zweiten baugleichen Fahrzeug unter Zuhilfenahme von Abstandsinformationen folgen, die aus drei an der Stirnseite des Fahrzeuges plazierten Ultraschallsensoren stammen. Das Verhalten und die Lernfähigkeit des Folgealgorithmus wurde in einer Computersimulation getestet.

Das hierzu eingesetzte neuronale Netz, das die besten Ergebnisse lieferte, besteht aus drei Schichten, einer Eingabe-, einer Zwischen- und einer Ausgabeschicht. Als Eingabe dienen die Werte der drei Ultraschallsensoren, ausgegeben wird ein Radius (von 11 möglichen Radienklassen) und dessen Richtung (links oder rechts) als Steuerparameter. Bei den getesteten Netzen mit mehr als einer Zwischenschicht war ein deutlich langsameres Lernverhalten festzustellen. Da das Lernen erheblich mehr Zeit als ein Propagierungsschritt benötigt und somit den Echtzeitanforderungen des Systemes nicht genüge getragen werden kann, wird in eine Lern- und eine Steuerungsphase unterschieden.

Als Ergebnis des Lernprozesses konnte man beobachten, daß das Backpropagation-Netz nur dann die Folgefahrt optimal lernen konnte, wenn die Lerndaten repräsentativ ausgewählt wurden. Hierbei kam es vor allem darauf an, daß die Extremsituationen und weniger alle Ausgangsklassen gleichmäßig repräsentiert waren.

In puncto Fehlertoleranz wurde festgestellt, daß bei einem Ausfall von bis zu 50% der Neuronen der Zwischenschicht kaum eine Beeinträchtigung des Folgeverhaltens zu beobachten war. Als weiteres Ergebnis im Rahmen der Fehlertoleranz wurde festgestellt, daß die Folgefahrt durch einen Ausfall des mittleren Sensors nicht entscheidend beeinflußt wurde und daß selbst ein Ausfall des rechten oder linken Sensors noch eine einfache Folgefahrt gestattete.

6.7.1.2 Prognose von Aktienkursen

In /Schöneburg, Nieß, Sautter 108/ wird eine Untersuchung über die Einsatzmöglichkeiten von neuronalen Netzen für die Prognose von Zeitreihen beschrieben. Komplizierte Zeitreihen, wie Aktienkurse, stellen einen guten Prüfstein für solche Systeme dar. Um die Leistungsfähigkeit eines Netzwerkmodells für die Kursprognose zu testen, wurde versucht, die Tageskurse zufällig ausgewählter Aktien großer deutscher Unternehmen mit verschiedenen Typen neuronaler Netze vorherzusagen.

Als Test- und Trainingsdaten wurden die Kurse von drei ausgewählten deutschen Aktien verwendet. Der Trainingszeitraum für die Netzwerke betrug 42 Tage; prognostiziert wurden nach den 42 trainierten Tagen maximal 56 Tage. Die Testergebnisse mit einem Backpropagation-Netz werden im folgenden kurz erläutert.

In einer direkten Gegenüberstellung des tatsächlichen mit dem prognostizierten Kursverlauf kann man feststellen, daß der prognostizierte Kursverlauf dem tatsächlichen mit einer gewissen Zeitverschiebung "hinterherhinkt". In einem Hinweis auf die Originalliteratur von Schöneburg wird dieses Verhalten bei /Brause 12/ dadurch erklärt, daß das System wohl "die alte Börsenregel entdeckt" habe, für die Vorhersage den aktuellen Tageswert plus eine kleine Veränderung in die richtige Richtung anzusetzen.

Es ist zweifelsohne eine mögliche Strategie, durch eine lineare Interpolation bzw. abgebrochene Taylorentwicklung (Tageswert plus erste Ableitung) eine Prognose für eine unbekannte Funktion zu erreichen. Die Prognose müßte aber besser werden, wenn der tatsächliche, innere Mechanismus eines Aktienkurses (sofern es einen solchen gibt) vom neuronalen Netz gelernt wird /Brause 12/.

Das eigentliche Problem der Aktienkursanalyse ist jedoch ein anderes: Wenn man nur den Kursverlauf aus der Vergangenheit eingibt und daraus erwartet, daß das neuronale Netz eine fehlerfreie Prognose abgibt, so setzt man doch voraus, daß alle für eine fehlerfreie Prognose nötigen Informationen bereits vorliegen. Dies ist aber, wie die tägliche Praxis zeigt, nicht der Fall: Der Aktienkurs kommt als Resultierende vieler menschlicher Aktionen zustande, die von einer Menge anderer Informationen, wie beispielsweise Kursverlauf anderer Aktien, Goldpreis, allgemeine Wirtschaftslage etc. abhängen. Ein neuronales Netz kann - wenn überhaupt - nur mit Hilfe der Informationen eine zuverlässige Prognose abgeben, die den "Machern" der Börsenkurse ebenfalls bekannt sind /Brause 12/.

6.7.1.3 Mustererkennung

Bei der Mustererkennung handelt es sich um dasjenige Gebiet, indem bisher die meisten praktischen Anwendungen neuronaler Netze anzusiedeln sind. Die Aufgabe besteht darin, daß das neuronale Netz auf ein eingegebenes Muster mit einer bestimmten "Antwort", z. B. durch eine entsprechende externe Ausgabe oder durch Ausbildung eines bestimmten (stabilen) Aktivierungszustandes, reagieren soll. Im Rahmen der Mustererkennung kann man die folgenden Spezialfälle unterscheiden /Kemke 54/:

- Merkmaldetektion:
  Das neuronale Netz soll bestimmte charakteristische Bestandteile oder Eigenschaften eines eingegebenen Musters erkennen, z. B. Kanten oder Ecken eines visuellen Musters. Diese Forderung kann beispielsweise durch Wettbewerbs-Lernen (vgl. Abschnitt 6.4.2.4) erreicht werden.

  In /Rumelhart, Zipser 104/ werden einige Beispiele für den Einsatz von Wettbewerbs-Lernen, unter anderem auch für das Lernen von Worten und Buchstaben, angegeben.

- Mustervervollständigung:
  Das neuronale Netz soll zu einem unvollständigen, verrauschten oder abgeänderten Muster das korrekte, vollständige Pendant produzieren. Dies kann man beispielsweise durch Backpropagation-Netze oder auch mit Hilfe von Kohonen-Netzen erreichen.

Ein System zur Verarbeitung abgeänderter Muster wurde beispielsweise von der Ruhr-Universität Bochum entwickelt, das menschliche Gesichter unabhängig von Mienenspiel, Blickwinkel oder Dreitagebart wiedererkennt /Würtz 121/.

Kaiser beschreibt in /Kaiser 48/ den Einsatz eines Backpropagation-Netzes zur Erkennung von Ziffern, die durch ein 5 x 7 Pixel großes Punktraster dargestellt werden.

- Assoziativer Zugriff:
  Das neuronale Netz soll zu einem eingegebenen Muster das zugeordnete Ausgabemuster erzeugen. Für diese Aufgabe bieten sich assoziative Speicher (vgl. Abschnitt 6.4.2.4) geradezu an.

  Von Kohonen wird in /Kohonen 60/ ein ausführliches Beispiel für den Einsatz assoziativer Speicher zur Bilderkennung (menschliche Gesichter) gegeben.

6.7.2 Anwendungen von Kohonen-Netzen

6.7.2.1 Robotersteuerung

In /Ritter, Martinetz, Schulten 101/ wird die folgende Anwendung einer um Ausgabewerte erweiterten Version eines Kohonen-Netzes zum Aufbau eines selbständig lernenden Robotersystems beschrieben. Die vom Roboter zu bewältigende Aufgabe ist die Positionierung seines Manipulators (z. B. eines Greifers oder einer Schweißelektrode) hin zu in seinem Arbeitsbereich vorgegebenen Zielorten.

Die Fähigkeit der "Endpositionierung" ist Voraussetzung für die Bewältigung einer ganzen Reihe von Aufgaben, mit denen ein Robotersystem in der Praxis konfrontiert wird. Zu diesen Aufgaben gehört beispielsweise das Greifen von Objekten, das Anbringen von Schweißpunkten, das Einsetzen von Bauelementen etc.

In diesem Anwendungsbeispiel betrachten Ritter et al. einen Roboter, der aus einem dreigelenkigen Arm besteht, der in seinem Arbeitsbereich montiert ist. Der Sockel ist um seine eigene Achse drehbar und zwei weitere Gelenke bewegen den Arm in einer vertikalen Ebene. Um in seiner Umwelt erfolgreich agieren zu können, liefern ihm zwei Kameras Informationen über die Raumposition anvisierter Zielobjekte.

Entsprechend der dreidimensionalen Topologie des durch das neuronale Netz zu repräsentierenden Arbeitsbereichs wird ein dreidimensionales Kohonen-Netz, d. h. ein Kohonengitter eingesetzt. Während der Trainingsphase wird die Position der anzufahrenden Zielpunkte im Arbeitsbereich zufällig ausgewählt. Vor jeder Bewegung wird das Ziel von den Kameras erfaßt, deren Signale dann dem Kohonen-Netz zugeführt werden. Jedes Neuron ist für ein ganz bestimmtes, von den Kameras erfaßtes Raumgebiet zuständig (Topologie-Erhaltung). Dasjenige Neuron, welches gerade für die momentane Lage des Zielobjektes zuständig ist, wird aktiviert und gibt seine Ausgabegrößen an die Motorsteuerung des Roboterarms wei-

ter. Der Lernalgorithmus für die korrekte Positionierung des Manipulators beruht dann auf einem Gradientenabstiegsverfahren mit einer quadratischen Fehlerfunktion.

Da die beiden Kameras nicht etwa die dreidimensionalen Koordinaten des Zielobjektes, sondern nur die beiden Orte, an denen das Zielobjekt in ihren beiden Gesichtsfeldern erscheint, zur Verfügung stellen, muß das Kohonen-Netz die geeigneten Ausgabegrößen für die Gelenkmotoren zur richtigen Positionierung des Manipulators selbst gewinnen. Weiterhin erhalten die Neuronen keinerlei Informationen über die Abmessungen der Roboterarmsegmente. Aus diesem Grund müssen diese geometrischen Daten für die richtige Umsetzung der Kamerainformationen in Motorsignale selbständig gelernt werden. D. h. um seinen Manipulator korrekt positionieren zu können, muß das Robotersystem die Transformation von den Bildkoordinaten der Zielposition zu den erforderlichen Winkelstellungen des Roboterarms ausführen können. Diese Transformation ist von der Geometrie des Roboterarms sowie der Stellung und der Abbildungscharakteristik der Kameras abhängig und soll durch das Lernverfahren selbständig adaptiert werden.

Da keine a priori Informationen vorgegeben werden, wird der Roboterarm anfangs eine fehlerhafte Endposition ansteuern; die Abweichung wird von den Kameras erfaßt und jedesmal zu einer Verbesserung des benutzten Ausgabewertes des entsprechenden Neurons verwendet. Anschließend wird dem Roboter ein anderer Zielpunkt vorgegeben, wodurch er Gelegenheit für einen neuen Lernschritt erhält.

6.8 Stärken und Schwächen neuronaler Netze

Im folgenden werden die Stärken und Schwächen neuronaler Netze kurz gegenübergestellt. Das Ziel dieser Gegenüberstellung ist eine möglichst "wert-neutrale" globale Beurteilung der zuvor dargestellten konnektionistischen Ansätze. Da es für die praktische Einsetzbarkeit neuronaler Netze zahlreiche Beispiele gibt (vgl. z. B. /Chaouli, Froitzheim 16/), kann an ihrem praktischen Wert kein Zweifel bestehen. Es soll hier lediglich der Blick dafür geschärft werden, daß neuronale Netze weder ein universell einsetzbares Lösungsparadigma darstellen, noch der Weisheit (im Bereich des Maschinellen Lernens) letzter Schluß sind.

(1) Stärken neuronaler Netze:

- Gutes Näherungsverhalten, Ausfallsicherheit, Fehlertoleranz:
  Aufgrund der verteilten Repräsentation und parallelen Verarbeitung des Wissens besitzen neuronale Netze ein gutes Näherungsverhalten, eine hohe Fehlertoleranz und sind relativ robust gegenüber Ausfällen einzelner Verarbeitungselemente (Neuronen).

  Da neuronale Netze auch mit unvollständigem und teilweise widersprüchlichem Wissen arbeiten können, entfällt die bei komplexen praktischen Problemen zeitaufwendige und häufig unmögliche Sicherstellung der Konsistenz der Wissensbasis /Hafner, Geiger, Kreßel 35/.

- Lernfähigkeit auf Musterebene:
  Neuronale Netze können durch Anpassung der Gewichte ihrer Verbindungen aufgrund von Trainingsdaten ein Problemlösungsverhalten lernen.

- Einsparungen von Kosten bei Sensoren:
  Neuronale Netze können sich auf beliebige Sensorkennlinien einstellen und können oft sogar zeitliche Schwankungen von Sensoreigenschaften kompensieren, falls diese langsam genug stattfinden. Dadurch wird die Notwendigkeit zur Vorverarbeitung von Sensordaten stark eingeschränkt und es ist häufig sogar möglich, billigere Sensoren zu verwenden /Hafner, Geiger, Kreßel 35/.

(2) Schwächen neuronaler Netze:

- Für den Einsatz neuronaler Netze zur Lösung eines konkreten Problems stellt sich zunächst die Frage nach der Auswahl eines geeigneten Netzwerktyps und dann nach der konkreten Architektur (Anzahl der Layer, Anzahl der Neuronen pro Layer). Zur Beantwortung dieser Fragen gibt es derzeit keine allgemeinen Aussagen. Somit ist man beim konkreten Einsatz eines neuronalen Netzes zumeist auf die Erfahrung, das Fingerspitzengefühl des Entwicklers oder auf ein wenig Glück angewiesen.

  Um mit Hilfe eines neuronalen Netzes ein (implizites) Modell aus gegebenen Daten ableiten zu können, versucht man zu erreichen, daß die Anzahl der Freiheitsgrade des Modells mit derjenigen des Systems, das die Daten erzeugt, übereinstimmt. Zur Bestimmung einer konkreten Netzwerkarchitektur existieren zwei komplementäre Ansätze, den konstruktiven und den pruning Algorith-

mus. Der konstruktive Algorithmus zielt darauf ab, ein approximatives Modell aufzubauen, um dann neue Neuronen hinzuzufügen und damit mehr Details zu lernen. Auf diese Art und Weise wird versucht, die optimale Netzgröße von "unten" anzunähern. Im Gegensatz hierzu beginnt der pruning Algorithmus mit einem Netzwerk, das bekanntermaßen zu groß ist und eliminiert dann Neuronen und Kanten, die nichts zum Modell beitragen.

In ähnlicher Weise erfahrungsabhängig gestaltet sich die Auswahl der Trainingssequenzen nach Art und Anzahl. Durch "gute" Trainingssequenzen kann sowohl die Lernphase des neuronalen Netzes verkürzt als auch seine Performanz verbessert werden. Da es sehr oft unmöglich ist, das Verhalten eines neuronalen Netzes deterministisch vorherzusagen, kann keine Garantie für das Verhalten neuronaler Netze in Situationen übernommen werden, die nicht Bestandteil der Trainingssequenzen waren /Hafner, Geiger, Kreßel 35/.

- Neuronale Netzwerke sind im allgemeinen intolerant gegenüber Wissenserweiterung (inkrementelles Lernen):
  Wenn ein neuronales Netzwerk die Menge X gelernt hat, und man lehrt es dann die Menge Y, dann hat es in der Regel die Menge X wieder "vergessen". Dies ist eine direkte Folge davon, daß das Turing-Prinzip - die Trennung von Programm und Daten - aufgegeben wurde. Die einzige Art, wie ein neuronales Netzwerk etwas dazulernen kann, ist alles Alte wieder von neuem mitzulernen (/Levelt 70/).

- Aus der Erlernbarkeit eines bestimmten Wissensbestandes kann nicht gefolgert werden, daß ein größerer Wissensbestand derselben Art ebenfalls erlernbar ist.

  Angenommen, ein neuronales Netz kann eine Sprache mit Sätzen vom Typ "Wenn Hans sagt, daß es regnet, schwindelt er" lernen. Dann gibt es noch keine Garantie dafür, daß das Modell mit Sätzen wie "Wenn Hans sagt, daß Peter sagt, daß es regnet, schwindelt er" zurechtkommt. Aber vielleicht kann ein größeres neuronales Netz auch das noch. Leider gibt es dann wieder keine Garantie dafür, daß das Netzwerk mit Sätzen wie "Wenn Hans sagt, daß Peter sagt, daß Klaus sagt, daß es regnet, schwindelt er", etc. umgehen kann. Kurzum, auf diese Art kann man niemals erfahren, ob ein Netzwerk eine Sprache erlernen kann, die unbegrenzte Rekursion dieser Art zuläßt (/Levelt 70/).

- Kein Fokus auf relevante Eingangsgrößen / Zeitspannen:
  Im Rahmen des Lernprozesses (d. h. der Trainingsphase) müssen dem neuronalen Netz immer Werte für alle Eingangsgrößen (bzw. die vollständige Zeitspanne) vorgegeben werden, unabhängig davon, ob einzelne Eingangsgrößen (bzw. die gesamte Zeitspanne) relevant für die Problemlösung sind oder nicht.

- In neuronalen Netzen enthaltenes Wissen ist im allgemeinen nicht explizierbar:
  Die Zustände einzelner Neuronen bzw. ihre Verbindungen können im allgemeinen nicht mehr oder nicht mit vertretbarem Aufwand in Symbole überführt werden, die für den Menschen verständlich sind /Kurbel, Pietsch 62/.

- Neuronale Netze besitzen keine Erklärungsfähigkeit:
  Die Problemlösungsfähigkeit des neuronalen Netzes entsteht durch das Zusammenspiel vieler einzelner Neuronen. Der Lösungsweg kann daher nicht, wie bei Expertensystemen, unter Bezugnahme auf die angewendeten Symbole erklärt werden. Für den Anwender stellt sich daher ein neuronales Netz als ein Black-Box-System dar /Kurbel, Pietsch 62/.

- In /Braun 11/ wird untersucht, für welche kombinatorischen Optimierungsprobleme sich neuronale Netze eignen. Am Beispiel des Assignment Problems und des Travelling Salesman Problems wird gezeigt, daß sich neuronale Netze für lineare Optimierungsprobleme mit einer Lösung aus dem Bereich der natürlichen Zahlen eignen. Hingegen scheitern neuronale Netze, wenn keine Lösung aus dem Bereich der natürlichen Zahlen vorhanden ist.

7 Zusammenfassung und Ausblick

Bei dem Versuch, die in den vorangegangenen Kapiteln vorgestellten maschinellen Lernverfahren in einer allgemeinen Form kritisch zu bewerten bzw. einander unter Abwägung von Vor- und Nachteilen direkt gegenüberzustellen, erreicht man sehr schnell einen Punkt, bei dem man eingestehen muß, daß eine derartige Gegenüberstellung ohne weiteres, d. h. losgelöst von einer konkreten Aufgabe nicht sinnvoll ist. Es wäre beispielsweise unsinnig, bei vorhandenem, umfangreichen Hintergrundwissen ein rein induktives Lernverfahren einzusetzen und somit diesen "Wissensvorsprung" nicht zu nutzen. Im Gegensatz hierzu ist es ohne umfangreiches Hintergrundwissen sogar unmöglich, deduktive- oder analogie-basierte Lernverfahren einzusetzen. Ähnlich verhält es sich mit einem externen Lehrer (z. B. zur Vorklassifizierung und Auswahl der Beispiele) im Hinblick auf eine Unterstützung des Lernprozesses und mit der Tatsache, ob ein inkrementelles oder nicht-inkrementelles Lernverfahren eingesetzt werden muß. In gewisser Weise wirken einige der realen Randbedingungen als "k. o.-Kriterien" für die Auswahl eines Lernverfahrens.

Aus diesen Gründen erfolgt die im weiteren dargestellte Bewertung der vorgestellten Lernverfahren auf der Grundlage der im folgenden skizzierten konkreten Aufgabe.

Gegeben sei ein komplexer dynamischer technischer Prozeß (z. B. der Verbrennungsprozeß einer Müllverbrennungsanlage) von dem einzig und allein die Meßwertkurven der relevanten Systemgrößen als bekannt vorausgesetzt werden können. Im Hinblick auf die Anwendung eines maschinellen Lernverfahrens zur Ableitung der Struktur und des Verhaltens des unterlagerten technischen Systems, ergeben sich damit die folgenden **Probleme:**

- Bis auf die Kenntnis der relevanten Systemgrößen existiert keinerlei Hintergrundwissen über den zugehörigen dynamischen technischen Prozeß,
- die gemessenen Prozeßdaten fallen sequentiell an; anzustreben ist eine on-line Verarbeitung der Prozeßdaten,
- die Meßwerte sind mit Meßungenauigkeiten und Meßfehlern behaftet,
- es existiert kein externer Lehrer, der erfolgreich Bedienereingriffe bzw. Fehlschläge auch als solche klassifiziert.

Aufgrund dieser Probleme können für ein maschinelles Lernverfahren die folgenden **Anforderungen** abgeleitet werden:

- Das Lernen muß nach der Strategie des Lernens durch Beobachtung und Entdeckung erfolgen. Es steht demnach kein Lehrer zur Auswahl und Vorklassifikation der Trainingsbeispiele zur Verfügung.
- Das maschinelle Lernverfahren muß eine inkrementelle Verarbeitung der anfallenden Daten unterstützen.

- Die dem maschinellen Lernverfahren zugrundeliegende Form der Wissensrepräsentation muß für eine dynamische Veränderung des Wissens (z. B. Bildung, Verstärkung und/oder Verwerfen von Hypothesen) geeignet sein.

- Für die Ableitung von Schlußfolgerungen aus den gemessenen Werten müssen überwiegend induktive Inferenzen durchgeführt werden. Statt einer deduktiven Validierung, wie bei analogie-basierten Verfahren, besteht hier nur die Möglichkeit einer empirischen Validierung.

    - Durch die empirische Validierung ist es notwendig, daß (induktiv) Hypothesen gebildet werden können, die dann (d. h. empirisch, im Laufe der Zeit) bestätigt oder verworfen werden müssen.

- Aufgrund der Meßfehler/-ungenauigkeiten müssen unscharfe Daten verarbeitet werden können.

- Durch das maschinelle Lernverfahren müssen sowohl lineare als auch nicht-lineare Zusammenhänge im zugrundeliegenden Systemverhalten erfaßt werden.

- Um eine effektive und effiziente Weiterverarbeitung des gelernten Wissens zu garantieren, muß es in einer strukturierten Form im System repräsentiert werden.

- Das gelernte Wissen muß zur Erklärung des Systemverhaltens herangezogen werden können.

Vor diesem Hintergrund können für die oben vorgestellten maschinellen Lernverfahren die folgenden k. o. - Kriterien angeführt werden:

- ID3 / CLS - Algorithmus:

    - Benötigt vorklassifizierte Beispiele,

    - in der ursprünglichen Form nicht-inkrementell und nicht zur Verarbeitung unscharfer Daten geeignet,

    - die Form der Wissensrepräsentation (hier Entscheidungsbaum) kann nur um Komponenten erweitert werden (monoton), eine Zurücknahme von Entscheidungen im Sinne einer nicht-monotonen, dynamischen Repräsentations- und Verarbeitungsstruktur ist nicht möglich,

    - in seiner ursprünglichen Version kann nur ein einziges Konzept gelernt werden,

    - durch die Form der Wissensrepräsentation (Attribut-Werte-Paare) können keine strukturierten Objekte, Relationen variabler Anzahl zwischen Objekten repräsentiert und damit gelernt werden.

- Versionenraum-Methode mit Kandidaten-Eliminations-Algorithmus:
  - Benötigt vorklassifizierte Beispiele,
  - es kann nur ein einziges Konzept gelernt werden,
  - es können keine unscharfen Daten verarbeitet werden,
  - die zur Beschreibung des Konzeptes gelernte Regel kann nur um Bedingungen erweitert werden; bereits bestehende Bedingungen können weder modifiziert noch gelöscht werden,
  - durch die Form der Wissensrepräsentation (Attribut-Werte-Paare) können keine strukturierten Objekte, Relationen variabler Anzahl zwischen Objekten repräsentiert und damit gelernt werden.

- BACON.x:
  - Keine inkrementelle Verarbeitung,
  - es können keine unscharfen Daten verarbeitet werden,
  - es werden nur zusätzliche Terme erzeugt, "sinnlos" erzeugte Terme werden beibehalten.

- AQ-Algorithmus (Star-Methode):
  - Vorklassifizierte Beispiele,
  - für die Monotonie und die eingeschränkte Ausdrucksmächtigkeit der Wissensrepräsentation gelten die gleichen Aussagen wie bei der Versionenraum-Methode.
  - keine Strategien/Heurismen zu Verarbeitung unscharfer Daten.

- EPAM / UNIMEM / COBWEB / CLASSIT:
  - keine Strategien/Heurismen zu Verarbeitung unscharfer Daten.
  - einmal gebildete Konzepte/Konzepthierarchien können nicht empirisch (höchstens statistisch, aufgrund von Wahrscheinlichkeiten COBWEB und CLASSIT) validiert werden,
  - bis auf UNIMEM werden alle je aufgetretenen Instanzen gespeichert, wertlose Instanzen können nicht gelöscht werden; d. h. monotone Form der Wissensrepräsentation.

- EBL/EBG:
  - Das notwendige, umfangreiche Hintergrundwissen ist nicht vorhanden,
  - es können keine unscharfen Daten verarbeitet werden,
  - die Wissensbasis kann nur monoton erweitert werden.

- DISCIPLE:
  - Das notwendige, umfangreiche Hintergrundwissen ist nicht vorhanden,
  - der für Rückfragen benötigte Experte ist nicht vorhanden,
  - es können keine unscharfen Daten verarbeitet werden,
  - die Wissensbasis kann nur monoton erweitert werden.

- Genetische Algorithmen:
  - Keine strukturierte Repräsentation des Wissens,
  - keine Erklärungsfähigkeit.

- Konnektionistische Ansätze / Neuronale Netze:
  - In der Regel ist mit neuronalen Netzen kein inkrementelles Lernen möglich (Übertrainieren),
  - keine Fokussierung auf relevante Größen im Eingangsbereich möglich,
  - das Wissen in neuronalen Netzen ist nicht explizierbar; damit besitzen sie auch keine Erklärungsfähigkeit.

Bereits durch diesen doch recht oberflächlichen Vergleich und die Bemerkungen in Abschnitt 4.3 wird deutlich, daß keines der oben beschriebenen (symbolischen, subsymbolischen und numerisch/statistischen) maschinellen Lernverfahren zur Lösung der skizzierten Aufgabe eingesetzt werden kann.

Zu den hier dargestellten Problemen mit den "klassischen" Ansätzen des maschinellen Lernens in Verbindung mit der skizzierten realen Anwendung kommen noch die damit verbundenen Probleme der "klassische" Formalismen zur Wissensrepräsentation (vgl. /Keller, Weinberger 53/).

Vor dem Hintergrund einer konkreten Anwendung des bereits oben skizzierten Typs und der bereits aufgeführten Probleme maschineller Lernverfahren sowie den Grenzen der analytisch/experimentellen Modellierung komplexer dynamischer technischer Systeme /Keller,

Weinberger 49/ wurde von den Autoren in /Keller, Weinberger 52/ ein Konzept zur Modellierung und Wissensverarbeitung derartiger Systeme vorgestellt. Das kognitive **Prinzip der heuristischen Modellierung** bildet den Kern dieses Konzeptes und beinhaltet gleichzeitig Wissensableitungs, -repräsentations und Verarbeitungselemente.

Komplexe dynamische technische Systeme sind beispielsweise aufgrund der vielfältigen internen Wechselbeziehungen weder analytisch noch experimentell modellierbar. Hingegen ist ein menschlicher Anlagenfahrer (Bediener, Operateur) auch ohne die Kenntnis eines mathematischen Systemmodells durchaus in der Lage, ein solches komplexes dynamisches technisches System sicher und stabil in einem vorgegebenen Toleranzbereich zu führen. Diese Fähigkeit des menschlichen Anlagenfahrers beruht daher eher auf seiner empirischen Erkenntnis über den Aufbau und das Verhalten des technischen Systems in Verbindung mit seinem allgemeinen heuristischen Wissen (Heurismen) als auf einem das technische System beschreibende Differentialgleichungssystem.

Genau diesen Punkt berücksichtigt das Prinzip der heuristischen Modellierung, indem versucht wird, aufgrund des beobachteten Systemverhaltens inkrementell (Kausalitäts-) Hypothesen über mögliche Ursache-Wirkungs-Beziehungen zwischen einzelnen (vorgegebenen) Systemgrößen abzuleiten. Diese (Kausalitäts-) Hypothesen sind anfänglich rein kombinatorisch begründbar, verdichten sich jedoch im Laufe der Zeit zu empirisch abgesicherten Erkenntnissen (Erklärungen) über das zugrundeliegende Systemverhalten.

Aus diesen empirisch bestätigten Hypothesen über das Systemverhalten können (Fuzzy-) Regeln zur Beschreibung des Systemverhaltens dynamisch generiert werden. In einem zweiten Schritt werden die empirisch bestätigten Hypothesen zur Ableitung einer möglichen Systemstruktur herangezogen.

Auf der Grundlage dieses Konzeptes und unter Einbeziehung von Problemlösungsfähigkeiten wurde eine Struktur für ein lernfähiges hierarchisch aufgebautes System, das **C^3R-System** (children's cognitive learning behavior for causal reasoning about dynamic systems) /Keller, Weinberger 52/ entwickelt.

Referenzen

/1/ Allman, W. F.,
Menschliches Denken - Künstliche Intelligenz,
Droemer Knaur, München, 1990.

/2/ Barr, Avron; Feigenbaum, Edward A.,
The Handbook of Artificial Intelligence,
William Kaufmann, Inc., Los Altos, Volume 1, 1981.

/3/ Belew, R. K.; Booker, L. B. (Eds.),
Proceedings of the Fourth Conference on Genetic Algorithms,
Morgan Kaufmann, San Mateo, 1992.

/4/ Bench-Capon, T. J. M.,
Knowledge Representation - An Approach to Artificial Intelligence,
The APIC Series 32
Academic Press, London, 1990.

/5/ Berns, Karsten; Hofstetter, R.,
Die Anwendung eines Backpropagation-Algorithmus
zur Steuerung einer Folgefahrt,
Robotersysteme, Heft 7, S. 53 - 58, 1991.

/6/ Berns, Karsten; Kreuziger, Jürgen,
Symbolische und subsymbolische Lernverfahren und deren Integration,
FZI Publikation 28, Forschungszentrum Informatik Karlsruhe und Universität Karlsruhe, Fakultät für Informatik, Institut für Prozeßrechentechnik und Robotik, Februar, 1992.

/7/ Birbaumer Niels; Schmidt Robert F.,
Biologische Psychologie,
Springer-Verlag, Berlin, 1987.

/8/ Bloc, L.,
Computational Models of Learning,
Springer-Verlag, New York, 1987.

/9/ Blume, C.,
GLEAM - a System for simulated "intuitive learning",
In: /Schwefel, Männer 111/.

/10/ Bortz, Jürgen,
Statistik für Sozialwissenschaftler,
3. Auflage, Springer-Verlag, Heidelberg, 1989.

/11/ Braun, Heinrich,
Massiv parallele Algorithmen für kombinatorische Optimierungsprobleme und ihre Implementierung auf Parallelrechnern,
Dissertation, Universität Karlsruhe (Technische Hochschule),
Fakultät für Informatik, 1990.

/12/ Brause, Rüdiger,
Neuronale Netze,
B. G. Teubner, Stuttgart, 1991.

/13/ Bruns, R.,
Incorporation of a knowledge-based scheduling system into a genetic algorithm,
In: Görke, W., Rininsland, H., Syrbe, M. (Eds.),
Information als Produktionsfaktor, Informatik aktuell, 22. GI-Jahrestagung,
Springer-Verlag, Heidelberg, S. 547 - 553, 1992.

/14/ Carbonell, J.; Langley, P.,
Machine Learning,
In: Shapiro; Eckroth (Eds.),
Encyclopedia of AI, Band I,
John Wiley & Sons, 1987.

/15/ Carbonell, J. G.; Michalski, R. S.; Mitchell, T. M.,
An Overview of Machine Learning,
In: /Michalski, Carbonell, Mitchell 80/, S. 3 - 23.

/16/ Chaouli, Michel; Froitzheim, Ulf, J.,
Strickmuster der elektronischen Intelligenz,
highTech, München, S. 64 - 75, November 1991.

/17/ Cohen, P. R.; Feigenbaum E. A.,
The Handbook of AI,
William Kaufmann Inc., Los Altos, Vol. III, 1982.

/18/ Clark, Peter; Niblett, Tim,
The CN2 Induction Algorithm,
Kluwer Academic Publishers, Boston, Mass., Journal of Machine Learning,
Vol. 3, S. 261 - 283, 1987.

/19/ Cybenko, G.,
Approximation by Superpositions of a sigmoidal Function,
Journal of Mathematics of Control Signals and Systems, Springer-Verlag,
New York, Vol. 2, S. 303 - 314, 1989.

/20/ De Jong, Kenneth,
Genetic-algorithm-based learning,
In: /Kodratoff, Michalski 58/, S. 611 - 638.

/21/ Dietterich, T. G.,
5. AAAI-86 Learning Papers: Developments and Summaries,
Journal of Machine Learning, Kluwer Academic Publishers, Boston, Mass.,
Vol. 2, No. 2, 1987.

/22/ Dillmann, Rüdiger,
Lernende Roboter- Aspekte maschinellen Lernens,
Fachberichte Messen - Steuern - Regeln 15,
Springer-Verlag, Berlin, 1988.

/23/ Dörner, Dietrich,
Problemlösen als Informationsverarbeitung,
Kohlhammer Standards Psychologie, dritte Auflage, 1987.

/24/ Dörner, D.; Kreuzig, F.; Reither, F. & Stäudel, T. (Hrsg.),
Lohhausen. Vom Umgang mit dynamischen Systemen,
Sprache & Kognition, Springer Verlag, Band 9, Heft 3, S. 143 - 154, 1990.

/25/ Ellman Thomas,
Explanation-based-learning - A survey of programs and perspectives,
ACM Computing Surveys, Vol. 21, No. 2, June 1989.

/26/ Fensel, Dieter; Studer, Rudi (Eds.),
Künstliche Intelligenz und Statistik,
Universität Karlsruhe (TH), Institut für Angewandte Informatik
und Formale Beschreibungsverfahren, Bericht 220, März 1991.

/27/ Fisher, Douglas, H.,
Knowledge acquisition via incremental conceptual clustering,
Journal of Machine Learning, Kluwer Academic Publishers, Boston, Mass.,
No. 2, S. 139 - 172, 1987.

/28/ Fogel, D. B.; Atmar, W. (Eds.),
Proceedings of the 1st Annual Conference on Evolutionary Programming,
La Jolla, CA., 21 - 22 Feb. 1992,
Evolutionary Programming Society, San Diego, Ca., 1992

/29/ Fogel, D. B.; Owens, A. J.; Walsh, M. J.,
Artificial Intelligence through Simulated Evolution,
John Wiley & Sons, 1966.

/30/ Forbus, Kenneth D.; Gentner, Dedre,
Learning physical Domains: Toward a theoretical Framework,
In: /Michalski, Carbonel, Michell, 81/, S. 311 - 348.

/31/ Gehirn und Kognition,
Spektrum der Wissenschaft: Verständliche Forschung,
Spektrum der Wissenschaft Verlagsgesellschaft, Heidelberg, 1990.

/32/ Gennari, John H.; Langley, Pat; Fisher Douglas,
Models of incremental concept formation,
Elsevier Science Publishers, Amsterdam, Journal of
Artificial Intelligence, 40, S. 11 - 61, 1989.

/33/ Goldberg, David E.,
Genetic algorithms in search, optimization and machine learning,
Addisson-Wesley Publ. Comp., New York, 1989.

/34/ Gorges-Schleuter, M.,
ASPARAGOS an asynchronous parallel genetic optimization strategy,
In: /Schaffer 105/, S. 422 - 427.

/35/ Hafner, Sigrid; Geiger, Hans; Kreßel, Ulrich,
Anwendungsstand künstlicher neuronaler Netze in der Automatisierungstechnik,
Teil 1: Einführung,
Automatisierungstechnische Praxis (atp), Oldenbourg Verlag, München, 34,
Heft 10, S. 592 - 599, 1992.

/36/ Hanson, Stephen José,
Conceptual Clustering and Categorization: Bridging the Gap between
Induction and causal Models,
In: /Kodratoff, Michalski 58/, S. 235 - 268.

/37/ Harp, S. A.; Samad T.; Guha, A.,
Towards the genetic synthesis of neural networks,
In: /Schaffer 105/.

/38/ Hartley, Ralph; Szu, Harold,
A comparison of the computational power of neural network models,
In: Maureen Candill; Charles Buttler (Eds.)
Proceedings IEEE 1st International Conference on Neural Networks, Vol. III,
San Diego, Ca., S. III-15 - III-22, 21. - 24. Juni 1987.

/39/ Heinemann, B.; Weihrauch, K.,
Logik für Informatiker,
B. G. Teubner, Stuttgart, 1991.

/40/ Helft, Nicolas,
Induction as nonmonotonic Inference,
In: Brachman et al. (Eds.),
Proceedings of the 1st international Conference on Principles of Knowledge
Representation and Reasoning,
Morgan Kaufmann, Los Altos, Ca., 1989.

/41/ Hinton, Geoffrey E.,
Connectionist Learning Procedures,
In: Buchanan, Bruce G.; Wilkins, David C. (Eds.),
Readings in Knowledge Acquisition and Learning,
Morgan Kaufmann Publishers, San Mateo, Ca., S. 362 - 386, 1993.

/42/ Hoffmeister, F., Bäck, T.,
Genetic Algorithms and Evolutionary Strategies: Similarities and Differences,
In: Schwefel, H.-P.; Männer, R. (Eds.),
Parallel Problem Solving from Nature,
First International Workshop, Dortmund,
Springer-Verlag, Heidelberg, LNCS, 496, 1991.

/43/ Holland, John H.,
Genetische Algorithmen,
Spektrum der Wissenschaft, Spektrum der Wissenschaft Verlagsgesellschaft mbH,
Heidelberg, S. 44 - 51, September 1992.

/44/ Hornik, K.; Stinchcombe, M.; White, H.,
Multilayer feed-forward networks are universal approximators,
Pergamon Press, Neural Networks, Vol. 2, S. 359 - 366, 1989.

/45/ Jakob, W.; Blume, C.,
Verbesserte Planung und Optimierung mit Hilfe eines erweiterten genetischen Algorithmus,
Tagungsband des Transputeranwendertreffens, Aachen, September 1993.

/46/ Jakob, W.; Gorges-Schleuter, M.; Blume, C.,
Application of genetic algorithm to task planning and learning,
In: /Männer, Manderick 73/.

/47/ Jörger, K.,
Einführung in die Lernpsychologie,
Herderbücherei, Band 9043, Freiburg, 13. Auflage, 1989.

/48/ Kaiser, Michael,
Künstliche Neuronale Netze - Grundlagen und Anwendungsbeispiele,
Automatisierungstechnische Praxis (atp), 34, S. 539 - 545, 1992.

/49/ Keller, Hubert B.; Weinberger, Thomas,
Ein kognitiver Ansatz zur Bestimmung von Ursache-Wirkungs-Beziehungen und Schlußfolgerungen in komplexen technischen Prozessen,
Vortrag 5. Fachtagung Maschinelles Lernen, Osnabrück, 15. & 16. Juli 1992.

/50/ Keller, Hubert B.; Weinberger, Thomas,
Heuristische Modelle - ein Arbeiten mit Hypothesen?,
Proceedings des Workshops "Modellierung und Simulation im Umweltbereich",
Rostock, 25. & 26. Juni 1992.

/51/ Keller, Hubert B.; Weinberger, Thomas,
Lernmodelle und Wissensverarbeitung,
KfK-Bericht Nr. 5002, Kernforschungszentrum Karlsruhe GmbH, 1992.

/52/ Keller, Hubert B.; Weinberger, Thomas,
Simulation of children's learning behavior,
In: Maceri, Franco; Iazeolla, Giuseppe (Eds.),
EUROSIM '92 Simulation Congress,
North-Holland, Amsterdam, S. 85 - 90, 1993.

/53/ Keller, H. B.; Weinberger, Th.; Kugele, E.; große Osterhues, B.,
Wissensbasierte Systeme - Darstellungs- und Verarbeitungsmodelle,
KfK-Bericht Nr. 5066, Kernforschungszentrum Karlsruhe GmbH, 1992.

/54/ Kemke, Christel,
Der neuere Konnektionismus,
Informatik Spektrum, Springer Verlag, Band 11, S. 143 - 162, 1988.

/55/ Klix, Friedhart,
Information und Verhalten,
Verlag Hans Huber, Bern - Stuttgart - Wien, 1971.

/56/ Kodratoff, Yves,
Characterising Machine Learning Programs: A European Compilation,
In: Sleeman, D.; Bernsen, N. O.,
Artificial Intelligence - Research Directions in Cognitive Science:
European Perspectives,
Vol. 5, Lawrence Erlbaum Associates Ltd., U. K., 1992.

/57/ Kodratoff, Yves,
Introduction to Machine Learning,
Pittman, London, 1988.

/58/ Kodratoff, Yves; Michalski, Ryszard (Eds.),
Machine Learning - an Artificial Intelligence Approach,
Volume III, Morgan Kaufmann, Los Altos, Ca., 1990.

/59/ Köhle, Monika,
Neuronale Netze,
Springer-Verlag, Wien, 1990.

/60/ Kohonen, Teuvo,
Self-organization and associative memory,
Springer-Verlag, Heidelberg, 3. Auflage, 1989.

/61/ Koza, J. R.,
The Genetic Programming Paradigm: Genetically Breeding Populations of
Computer Programs to Solve Problems,
In: Soucek, Branko (Ed.),
Dynamic, Genetic, and Chaotic Programming,
John Wiley & Sons, S. 203 - 321, 1992.

/62/ Kurbel, Karl; Pietsch, Wolfram,
Eine Beurteilung konnektionistischer Modelle auf der Grundlage ausgewählter Anwendungsprobleme und Vorschläge zur Erweiterung,
Wirtschaftsinformatik, Vieweg Verlag, Stuttgart, Heft 5, S. 355 - 364, Okt. 1991.

/63/ Langley, Pat; Bradshaw, Gary L.; Simon, Herbert A.,
Rediscovering chemistry with the BACON system,
In: /Michalski, Carbonell, Mitchell 80/, S. 307 - 329.

/64/ Langley, Pat; Simon, Herbert A.; Bradshaw, Gary L.,
Heuristics for empirical Discovery,
In: /Bolc 8/.

/65/ Langley, Pat; Simon, Herbert A.; Bradshaw, Gary L.; Zytkow, Jan M.,
Scientific Discovery - computational Explorations of the creative Prosesses,
MIT-Press, Cambridge, Massachusetts, 1987.

/66/ Laske, Otto E.,
Ungelöste Probleme bei der Wissensakquisition für wissensbasierte Systeme,
Künstliche Intelligenz (KI), FBO, Baden-Baden, Heft 4, S. 4 - 12, 1989.

/67/ Lawrence, Jeannette,
Neuronale Netze,
SYSTHEMA Verlag GmbH, München, 1992.

/68/ Lebowitz, M.,
Experiments with incremental concept formation: UNIMEM,
Journal of Machine Learning, Kluwer Academic Publishers, Boston, Mass.,
No. 2, S. 103 - 138, 1987.

/69/ Lenat, D. B.,
Automated theory formation in mathematics,
Proceedings of the 5th International Joint Conference on Artificial Intelligence,
IJCAI, Cambridge, Ma., S. 833 - 842, 1977.

/70/ Levelt, W. J. M.,
Die konnektionistische Mode,
Sprache & Kognition, 10, Heft 2, S. 61 - 72, 1991.

/71/ Lohmann, R.,
Application of Evolution Strategy in Parallel Populations,
In: /Schwefel, Männer 111/.

/72/ Mandl, Heinz; Friedrich, Helmut Felix; Hron, Aemilian,
Theoretische Ansätze zum Wissenserwerb,
In: Mandl, Heinz; Spada, Hans (Hrsg.).
Wissenspsychologie
Psychologie Verlags Union, München, S. 123 - 160, 1988.

/73/ Männer, R.; Manderick, B.,
Parallel Problem Solving from Nature 2,
Proceedings Second Conference, Brussels, Belgium,
North-Holland, Amsterdam, 1992.

/74/ McCord Nelson, Marilyn; Illingworth, W. T.,
A practical Guide to neural Nets,
Addison-Wesley Publishing Company, Inc. N. Y., 1990.

/75/ McClelland, James L.; Rumelhart, David E.,
Explorations in parallel distributed Processing,
MIT-Press, Cambridge, Mass., 5th Printing, 1991.

/76/ Metropolis, N.; Rosenbluth, M.; Teller A und E.,
Equation of state calculation by fast computing machines,
Journal Chem. Physics, Vol. 21, S. 1087 - 1092, 1953.

/77/ Michalski, Ryszard S.,
A Theory and Methodology of Inductive Learning,
In: /Michalski, Carbonell, Mitchell 80/, S. 83 - 134.

/78/ Michalski, Ryszard S.,
Learning Strategies and Automated Knowledge Acquisition - An Overview,
In: In: /Bolc 8/.

/79/ Michalski, Ryszard S.,
Understanding the nature of learning: Issues and research Directions,
In: /Michalski, Carbonell, Mitchell 81/, S. 3 - 25.

/80/ Michalski, Ryszard S.; Carbonell, Jaime G.; Mitchell, Tom M. (Eds.),
Machine Learning - an Artificial Intelligence Approach,
Volume I, Springer-Verlag, New York, 1984.

/81/ Michalski, Ryszard S.; Carbonell, Jaime G.; Mitchell, Tom M. (Eds.),
Machine Learning - an Artificial Intelligence Approach,
Volume II, Morgan Kaufmann, Los Altos, Ca., 1986.

/82/ Michalski, Ryszard S.; Kodratoff, Yves,
Research in Machine Learning; Recent Progress, Classification of Methods,
and Future Directions,
In: /Kodratoff, Michalski 58/, S. 3 - 30.

/83/ Michalski, Ryszard S.; Stepp, Robert E.,
Automated construction of Classifications: Conceptual Clustering versus
numerical Taxonomy,
IEEE Transactions on Pattern Analysis and Machine Intelligence,
Vol. 5, No. 4, S. 396 - 410, Juli 1983.

/84/ Michalski, Ryszard S.; Stepp, Robert E.,
Learning from Observation: Conceptual Clustering,
In: /Michalski, Carbonell, Mitchell 80/, S. 331 - 363.

/85/ Miller, G. F.; Todd, P. M.; Hedge, S. U.,
Designing neural networks using genetic algorithms,
In: /Schaffer 105 /.

/86/ Minsky, Marvin L.; Pappert, Seymour A.,
Perceptrons,
MIT-Press, Cambridge, Mass., Expanded Edition, 1988.

/87/ Mitchell, Tom M.,
Version spaces: A candidate elimination approach to rule learning,
IJCAI 5, S. 305 - 310, 1977.

/88/ Mitchell, Tom M.; Keller, Richard M.; Kedar-Cabelli Smadar T.,
Explanation-Based Generalization: A Unifying View,
Journal of Machine Learning, Kluwer Publishers, Boston, Mass.,
Vol. 1, S. 47, 80, 1986.

/89/ Montana, D. J.; Davis, L.,
Training feedforward neural networks using genetic algorithms,
In: Proceedings of the 11th International Joint Conference on Artificial
Intelligence, S. 762 - 767, 1989.

/90/ Morik, Katharina,
Applications of Machine Learning,
In: Wetter, Th. et al.,
Current Developments in Knowledge Acquisition - EKAW '92,
Lecture Notes in Artificial Intelligence 599,
Springer Verlag, Heidelberg, S. 9 - 13, 1992.

/91/ Niemann, H.,
Pattern Analysis and Understanding,
Springer Series in Information Sciences,
Springer-Verlag, Berlin, zweite Auflage, 1989.

/92/ Puppe, Frank,
Einführung in Expertensysteme,
Studienreihe Informatik,
Springer-Verlag, Berlin, 1988.

/93/ Quinlan, J. Ross,
Inductive Knowledge Acquisition from structured Data,
In: Motoda, H., Mizoguchi, R., Boose, J., Gaines, B. (Eds.),
Knowledge Acquisition for knowledge-based Systems,
ISO-Press, Amsterdam, Washington, 1991.

/94/ Quinlan, J. Ross,
Induction of Decision Trees,
Journal of Machine Learning, Kluwer Academic Publishers, Boston,
Mass., No. 1, S. 81 - 106, 1986.

/95/ Quinlan, J. Ross,
Learning efficient classification procedures and their
application to chess end games,
In: /Michalski, Carbonell, Mitchell 80/, S. 463 - 482.

/96/ Quinlan, J. Ross,
The effect of noise on concept learning,
In: /Kodratoff, Michalski 58/, S. 149 - 166.

/97/ Rechenberg, I.,
Evolutionsstrategien: Optimierung technischer Systeme nach Prinzipien
der biologischen Evolution,
Frommann-Holzboog Verlag, 1973.

/98/ Reichert, Ute; Dörner, Dietrich,
Heurismen beim Umgang mit einfachen dynamischen Systemen,
Projekt Mikroanalyse, Bericht Nr. 55, Lehrstuhl Psychologie II,
Universität Bamberg.

/99/ Reimann, Peter; Bader, Markus; Klenner, Manfred,
Computermodelle inkrementellen Konzepterwerbs,
Forschungsbericht des Psychologischen Instituts der
Albert-Ludwigs-Universität Freiburg i. Br., Nr. 64, 1990.

/100/ Richter, Michael M.; Wendel Oliver,
Lernende Systeme,
Vorlesungsskript Universität Kaiserslautern, WS 1990/91.

/101/ Ritter, Helge; Martinetz, Thomas; Schulten, Klaus,
Neuronale Netze,
Addison-Wesley, New York, 2. Auflage, 1991.

/102/ Rojas, Raúl,
Theorie der neuronalen Netze - Eine systematische Einführung,
Springer-Verlag, Berlin, 1993

/103/ Rumelhart, David E.; McClelland, James L. and the PDP Research Group,
Parallel Distributed Processing,
Volume I, MIT Press, Cambridge, Mass., 9. Auflage, 1989.

/104/ Rumelhart, David E.; Zipser, D.,
Feature discovery by competitive learning,
In: /Rumelhart, McClelland 103/, S. 151 - 193.

/105/ Schaffer, J. D. (Ed.),
Proceedings of the Third International Conference on Genetic Algorithms,
Morgan Kaufmann, San Mateo, Ca., 1989.

/106/ Schaffer, J. D.; Caruana, R.; Eshelman, L.,
Using genetic search to exploit the emergent behavior of neural networks,
Philips Laboratories, Briar Cliff Manor, N. Y., 10510, 1989.

/107/ Schlimmer, Jeffrey C.; Fisher, Douglas,
A Case Study of incremental Concept Induction,
In: Proceedings of the national conference on AI,
Philadelphia, PA., S. 496 - 501, 1986.

/108/ Schöneburg, Eberhard; Nieß, Jürgen; Sautter, Uwe,
Aktienkurs-Prognose mit neuronalen Netzwerken,
CHIP PLUS 7, S. XIII - XVI, 1990.

/109/ Scholz, A.,
A learning strategy for neural networks based on a modified evolutionary strategy,
In: /Schwefel, Männer 111/.

/110/ Schwefel, H.-P.,
Numerical Optimization for Computer Models,
John Wiley & Sons, 1981.

/111/ Schwefel, H.-P.; Männer, R. (Eds.),
Parallel Problem Solving from Nature,
Springer-Verlag, Heidelberg, LNCS 496, 1991.

/112/ Shapiro, Stuart C. (Ed.),
Encyclopedia of Artificial Intelligence,
Volume 1 & 2, Second Edition, John Wiley & Sons, Inc., New York, 1992.

/113/ Shavlik, J. W.; Mooney, R. J.; Towell, G. G.,
Symbolic and Neural Learning Algorithms: An Experimental Comparison,
Journal of Machine Learning, Kluwer Academic Publishers, Boston, Mass.,
No. 6, S. 111 - 143, 1991.

/114/ Simon, Herbert A.,
Why should Machines Learn?,
In: /Michalski, Carbonell, Mitchell 80/, S. 25 - 37.

/115/ Spies, Marcus,
Unsicheres Wissen,
Spektrum Akademischer Verlag GmbH, Heidelberg, 1993.

/116/ Starkwaether, T.; Whitley, L. D.; Mathias, K.,
Optimization using distributed genetic algorithms,
In: /Schwefel, Männer 111/.

/117/ Teccuci, Gheorghe; Kodratoff, Yves,
Apprenticeship learning in imperfect domain theories,
In: /Kodratoff, Michalski 58/, S. 514 - 551.

/118/ Weis, Sholom M.; Kulikowski, Casimir A.,
Computer Systems that learn,
Morgan Kaufmann Publishers, San Mateo, Ca., 1991.

/119/ Whitley, D.; Starkweather, T.; Bogart, C.,
Genetic algorithms and neural networks: optimizing connections
and connectivity,
Parallel Computing, 14, 1990.

/120/ Winston, Patrick Henry,
Künstliche Intelligenz,
Addison-Wesley Publishing Company, Bonn, 1987.

/121/ Würtz, Rolf P.,
Gesichtserkennung mit dynamischen neuronalen Netzen,
Spektrum der Wissenschaft, Spektrum der Wissenschaft Verlagsgesellschaft mbH,
Heidelberg, S. 18 - 22, April 1992.

/122/ Zytkow, Jan M.; Zhu, Jieming,
Application of empirical discovery in knowledge acquisition,
In: Kodratoff, Yves (Ed.),
Machine Learning - EWSL - 91, Proceedings,
Springer-Verlag, Berlin, 1991.