Modularisierung von neuronalen Netzen

Dissertation zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften
an der Technischen Fakultät
der Universität Bielefeld

von Hendrik Wagner

vorgelegt im Januar 1999

Inhaltsverzeichnis

Teil I

Kapitel 1

Inhaltsübersicht

Diese Arbeit ist im Rahmen des Graduiertenkollegs "Strukturbildungs-Prozesse" der Deutschen Forschungsgemeinschaft (DFG) entstanden und beschäftigt sich mit einer speziellen Form der Strukturiertheit von Systemen, nämlich der Modularität. Konkret geht es dabei um die Modularität von neuronalen Netzen.
In Kapitel 2 untersuche ich zunächst zwecks einer genaueren Begriffsbestimmung die Verwendung des Begriffs Modularität in verschiedenen Disziplinen, insbesondere der Elektronik/EDV, der Biologie/Zoologie und den Neurowissenschaften, arbeite charakteristische Merkmale des Begriffs heraus und versuche eine abstrakte Definition.
In Kapitel 3 richtet sich der Fokus dann auf neuronale Netze. Es werden zunächst einige Netzmodelle mit modularem Charakter vorgestellt, die man in der Fachliteratur beschrieben findet. Dabei gehe ich aus von Modellen, die in der neurowissenschaftlichen Debatte um die Modularität von kognitiven Prozessen argumentative Verwendung finden und beschreibe anschließend Netzmodelle, die motorisches Verhalten, insbesondere Laufverhalten, simulieren und durch ihren modularen Aufbau geprägt sind. Anschließend wird konkreter auf die spezielle Klasse der Multilagen-Perzeptrone (MLP) eingegangen, wobei zwei verschiedene Arten, wie ein MLP in Module zerfallen kann, vorgestellt werden. Ich schildere jeweils einige Beispiele für modulare Modelle aus der Literatur, die ich danach anordne, wie manuell, bzw. automatisch die Aufteilung der Aufgabe erfolgt.
In Kapitel 4, dem experimentellen Teil der Arbeit, wird die Modularisierung von Multilagen-Perzeptronen näher untersucht. Dazu verwende ich Test-Probleme unterschiedlicher Art, die aufgrund ihrer Konstruktion erwarten lassen, daß modulare Netze zu ihrer Lösung geeignet sein könnten. Zunächst interessiert mich, ob eine bereits vor Beginn des Trainings aufgrund der Kenntnis der Aufgaben-Konstruktion festgelegte modulare Netzstruktur bei diesen Aufgaben vorteilhaft ist. Eine weitergehende Frage betrifft dann die Entstehung modularer Netzstrukturen im Verlaufe des Lernens in nicht derart vorstrukturierten Netzen. Dazu werden Verfahren entwickelt und getestet, die eine solche Modularisierung unterstützen. Schließlich wird ein evolutionäres Verfahren entwickelt und getestet, das die optimale modulare Vorstrukturierung ohne Kenntnis der Aufgaben-Konstruktion allein aus den Daten herleitet.
In den Experimenten ist es wichtig, ein Maß für die Modularität der Netze zur Verfügung zu haben. Dazu werden ein auf einer optimalen Partitionierung der Neuronenmenge beruhendes Maß und alternativ ein leichter zu berechnendes aber weniger aussagekräftiges modul-unabhängiges Strukturiertheits-Maß entwickelt.
In Kapitel 5 werden die Ergebnisse der Arbeit schließlich zusammengefaßt und diskutiert.

Kapitel 2

Modularität in Technik und Natur

2.1 Modularität als technischer Begriff: Das Baukastenprinzip

Modularität ist weder formal mathematisch noch in der natürlichen Sprache ein klar definierter Begriff. Wenn man über Modularität abstrakt ohne Bezug zu einem bestimmten engen Verwendungsgebiet sprechen möchte, etwa um zu prüfen, inwieweit der Begriff dienlich zur Beschreibung von Sachverhalten sein kann, für die er bisher noch nicht verwendet wird, so muß man versuchen, sich den allgemeinen Begriff induktiv aus seiner Verwendung in den verschiedenen Sachgebieten herzuleiten. In den drei ersten Abschnitten dieses Kapitels werde ich zu diesem Zweck auf die Sachgebiete Technik, Biologie und Neurowissenschaft eingehen und jeweils am Ende zusammengefaßt, was Modularität im jeweiligen Gebiet bedeutet, wozu sie dient und wie sie entsteht.

Modularität ist ein Begriff, der uns vor allem aus der Welt der Technik vertraut ist. Hier hat er seinen Ursprung, auch wenn er gelegentlich in anderen Bereichen Verwendung findet. Im Lexikon wird ein Modul 1972 als eine austauschbare Funktionseinheit in einem elektrischen Gerät beschrieben oder noch konkreter als eine Baugruppe aus auf einer Keramikplatte eng zusammengebauten und mit Kunstharz vergossenen elektronischen Bauelementen (Brockhaus, 1971; Knaur's Lexikon der Technik, 1972). Solche Module waren in den 60er Jahren in der Elektronikbranche modern, bevor sie von sogenannten "integrierten Schaltungen" abgelöst wurden. Diese "Chips" vereinen im Gegensatz zu den Modulen eine Vielzahl unterschiedlicher Funktionen auf engstem Raum und können flexibel als Gesamtschaltung für bestimmte Aufgaben hergestellt werden. Sie stellen als Bauteil eines Gerätes in gewissem Sinne immer noch ein Modul dar, können aber nur als Ganzes ausgetauscht werden, während bei Geräten in Modulbauweise einzelne Teilfunktionen austauschbar und auch unabhängig herstellbar sind.
Die Modulbauweise stellt also historisch ein Übergangsstadium dar, das für eine Weile eine Möglichkeit bot, komplexere elektrische Geräte als zuvor herzustellen und zu reparieren, indem statt unübersichtlichen und nur als Gesamtschaltung funktionstüchtigen Verschaltungen von elektronischen "Atomen" wie Kondensator, Spule, Widerstand und Transistor in ihrer Funktion genau spezifizierte, und damit austauschbare, wiederverwendbare und getrennt, d.h. auch arbeitsteilig und u.U. kostengünstiger herstellbare Teilschaltungen verwendet wurden, deren Verbindungsstruktur einfacher und klarer und damit auch weniger fehleranfällig, bzw. leichter auf Fehler hin zu untersuchen war.
Der Begriff Modul findet heute auch in anderen Bereichen der Technik Verwendung, und

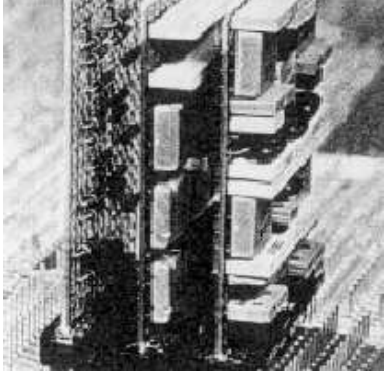

Abbildung 2.1: Elektronisches Modul. Aus: (Knaur's Lexikon der Technik, 1972)

zwar neben dem Maschinen- und Fahrzeugbau ganz besonders in der Datenverarbeitung und Software-Entwicklung. Hier wurden im Zuge der Professionalisierung der Programmierbranche Prinzipien des "Software-Engineering" entwickelt, die eine strukturierte und robuste Entwicklung und Pflege von großen Programmen erst möglich machten. Die Aufteilung von Programmen in Module ist dabei von herausragender Bedeutung. Sie ermöglicht die unabhängige Entwicklung und Änderung von Programmteilen durch genaue Festlegung der Eingabe- und Ausgabedaten der Module, einer sogenannten Schnittstellen-Spezifikation. Auch wird die Verständlichkeit des Gesamtprogramms erhöht, wenn man sich bei der Analyse zunächst auf die einfachen Beziehungen zwischen Modulen konzentrieren kann, und die u.U. komplexen Abläufe im Inneren des Moduls verborgen bleiben ("information hiding"). In jüngerer Zeit hat sich die Art und Weise der Modularisierung von Programmen dahingehend weiterentwickelt, daß die Gesamtdatenmenge, auf der ein Programm operiert, in Untermengen geteilt wird, die jeweils mit allen Funktionen, die auf ihnen operieren, zu "abstrakten Datentypen" oder "Objekten" zusammengefaßt werden. Zu einer Aufteilung des Datenflußes kommt damit eine Aufteilung der Zuständigkeiten von Funktionen für nur bestimmte "Arten" von Daten hinzu. Dies kann die Verständlichkeit von Programmen dadurch noch weiter erhöhen, daß diese Programmkonstrukte auf einer äußeren Ebene semantischen Konzepten der Lebenswelt angenähert werden. Schließlich können große Programme auf unterschiedlichen Hierarchieebenen modular sein, von der groben Einteilung in "Kernel" und (graphische) Oberfläche bis hinunter zum einzelnen Algorithmus für eine Teilfunktion, der wie z.B. der Quicksort-Algorithmus den von ihm zu bearbeitenden Datensatz rekursiv bis hin zu nur noch Elementaroperationen erfordernden Kleinstdatensätzen aufteilt. Eine wichtige Motivation für eine modulare Aufteilung ist zuletzt auch die Möglichkeit, verschiedene Programmteile, sofern sie unabhängig voneinander sind, parallel auf verschiedenen Prozessoren ablaufen zu lassen und somit die Laufzeit des Programms zu verringern.

Was bedeutet Modularität also in der Technik? Zusamenfassend läßt sich sagen, daß in der Technik von Modulen gesprochen wird, wenn elektrische Geräte, Maschinen oder Programme nicht direkt aus elementaren Teilen, seien es Widerstände, Schrauben oder Programmzeilen, aufgebaut sind, sondern sich aus komplexeren Komponenten zusammensetzen. Durch die Verbindung dieser Module entsteht dabei insgesamt eine einfachere Struktur als durch einen direkten Aufbau, weil die Teile, aus denen ein Modul selbst besteht, untereinander stärker als nach außen hin verbunden sind. In diesem Sinne sind die Module voneinander relativ unabhängig. Die Module, aus denen ein technisches System

besteht, können unterschiedliche Funktionen haben und sie können selbst wieder statt aus elementaren Teilen aus Modulen aufgebaut sein, wodurch dann eine geschachtelte und in diesem Sinn hierarchische Gesamtstruktur entsteht.
Wozu dient Modularität? Der Vorteil modularer technischer Systeme besteht darin, daß die Module als Folge ihrer relativen Unabhängigkeit, also der einfachen und klar definierbaren Verbindungsstruktur, getrennt voneinander entwickelt, repariert, ausgetauscht und an anderer Stelle wiederverwendet werden können. Außerdem ist der Aufbau der modularen Systeme besser verständlich.
Und wie kommt Modularität zustande? Die Kunst des Ingenieurs besteht oft gerade darin, eine Aufteilung eines Problems zu finden, die eine Lösung durch ein modulares System erlaubt. Zwei Design-Prinzipien stehen ihm dabei zur Verfügung. Er kann zum einen "bottom-up" vorgehen, d.h. mit kleinen Lösungen für Teilprobleme beginnen, und aus diesen immer größere Systeme zusammensetzen. Bei diesem Vorgehen kann er auf evtl. bereits vorhandene Module zurückgreifen. Problematisch ist dann aber, ob sich diese letztlich zu einer Lösung zusammenfügen lassen. Zum anderen kann er vom Gesamtproblem ausgehen, das er zunächst in wenige etwas kleinere Probleme zerlegt, die dann evtl. noch weiter zerlegt werden ("top-down"). Problematisch ist hier, daß nicht gesichert ist, ob man auf diese Weise zu bereits gelösten Teilproblemen gelangt.

2.2 Modularität: ein Begriff für die Biologie?

Die Beschreibung biologischer Systeme durch den technischen Begriff Modularität erscheint zunächst seltsam unangebracht. Und tatsächlich finden einige Merkmale, die den Begriff im technischen Bereich wesentlich kennzeichnen, in der Natur keine Entsprechung, und zwar jene, die einen kreativen, planenden, intelligenten Ingenieur voraussetzen. In der Natur gibt es keinen Schöpfer, der einen großen Entwurf in handliche, an ihrer Schnittstelle klar definierte Teile zerlegt, oder der einzelne Module unter Einbringung seines Wissens systematisch kombiniert. Es gibt keinen Ingenieur, dessen Arbeit dadurch erleichtert wird, daß ihm die Struktur verständlich ist, daß er einzelne Module unabhängig von den anderen ändern oder austauschen oder einmal entwickelte Module an anderer Stelle wiederverwenden kann.
Andere Merkmale hingegen finden sich auch in der Natur. Auch lebende Systeme sind durch eine hierarchische Schachtelung funktionaler Einheiten gekennzeichnet, die in sich abgeschlossen sind und trotz eines komplexen Inneren nur über schmale Schnittstellen mit anderen Einheiten kommunizieren. Auch in der Natur können anscheinend große Organismen nur existieren, wenn sie aus solchen getrennten Teilsystemen aufgebaut sind. Die biologischen Systeme entstehen nicht durch rationale Schöpfungsakte, sondern entwickeln sich in steter Anpassung an ihre Umwelt durch Prozesse wie Evolution, Wachstum oder Lernen. Die Art der Organisation, die diese Prozesse hervorbringen, zeichnet sich durch zwei Merkmale aus, nämlich funktionelle Differenzierung und Integration in eine hierarchisch geschachtelte Ordnung. Unter funktioneller Differenzierung kann man auch das Prinzip der Arbeitsteilung verstehen (Hesse, 1935). Je verschiedenartige Teile widmen sich speziellen Funktionen an verschiedenen Orten und sind für ihr Überleben darauf angewiesen, daß andere Spezialisten ebenfalls ihre Arbeit tun. Diese Spezialisierung ist bestimmt durch die kleinste lebende Einheit des Organismus, die Zelle, samt ihren Unterstrukturen. Ihre Differenzierung in verschiedenartige Zellarten ermöglicht den Aufbau der nächsthöheren Hierarchiestufe, nämlich den Aufbau von Gewebe verschiedenen Typs, aus denen sich in der nächsten Stufe die unterschiedlichen Organe und Organsysteme

zusammensetzen, aus denen schließlich der Gesamtorganismus aufgebaut ist. Diese Differenzierung und Integration ist sowohl ein Merkmal der Entwicklung der Individuen als auch der Evolution der Arten.
Hesse beschreibt die Vorteile dieser komplexen Organisation im Vergleich zu einfacheren Lebensformen. Sie führt insgesamt zu einer höheren "Energie der Lebensäußerungen", ermöglicht aber auch eine feinere Abstufung der Einzelleistungen in Anpasssung an die jeweiligen Bedürfnisse. Die einzelnen Spezialisten sind nicht durch Nebenfunktionen beeinträchtigt und können sich ganz ihrer Funktion anpassen. Es entsteht eine Vielzahl von Variationsmöglichkeiten, je nachdem wieviel ein Spezialist in einem Individuum oder einer Art von seiner speziellen Eigenart beiträgt. Die evolutive Anpassung ist dadurch erleichtert, daß Änderungen nur an einer relativ geringen Zahl von Körperzellen notwendig sind, um sich veränderten Gegebenheiten anzupassen. Schließlich ermöglicht die Differenzierung auch erst das Größenwachstum, indem sie kompaktere Körperformen, als sie ein Verband gleichberechtigter Zellen gestattet, zuläßt, und indem sie die Bildung von Stützgewebe und schließlich des mittleren Keimblatts und der Skelettknochen ermöglicht.
Hesse beschreibt aber auch den großen Nachteil der komplexen Organisation, nämlich ihre mangelnde Robustheit: das Ganze wird nämlich insofern zum Sklaven seiner Teile, als jede Störung eines Einzelorgans den ganzen Körper betrifft. Außerdem werden durch die gesteigerte "Intensität der Lebensäußerungen" auch die Ansprüche gesteigert, die die Teile an die Versorgung mit Nahrung und Sauerstoff und an die Gleichmäßigkeit der äußeren Bedingungen stellen. Eine besondere Stellung zwischen sehr einfachen und sehr komplexen Organisationsformen nimmt die Wiederholung gleichartiger Elemente ein, so z.B. wenn statt wenigen großen gleichartigen Zellen viele kleine für die gleiche Funktion verwendet werden. Hesse nennt das intensive Arbeitsteilung (statt extensiver Arbeitsteilung durch funktionale Differenzierung). Ein anderes Beispiel ist die Entstehung von Organismen, die in gleichartige Segmente gegliedert sind, von denen jedes noch alle lebensnotwendigen Organe enthält. Diese Organisation ist zum einen immer noch sehr robust, zum anderen scheint sie oft die Vorstufe zu weiter ausdifferenzierten Formen zu bilden (Wagner, 1995).
In der Tatsache, daß Teile, die sich mehr voneinander unterscheiden auch stärker voneinander abhängig sind, scheint zunächst ein Widerspruch zu liegen: wie sollen die verschiedenen Teile in ihrer Unterschiedlichkeit noch miteinander kommunizieren können und was sollte z.B. ein Wadenmuskel der Leber mitzuteilen haben? Die Kommunikation zwischen entfernt liegenden Teilen wird in höheren Organismen durch zwei zentrale Mechanismen ermöglicht, nämlich chemisch durch den Blutkreislauf und elektrisch durch das Nervensystem. Aber über diese Träger lassen sich nur Signale übermitteln, die zur Funktion anregen können: funktionieren müssen die einzelnen Organe dann selber. Indem die Abläufe in den einzelnen Organen also nicht im Détail von den Zuständen anderer Organe abhängen, sind sie unabhängig, indem sie aber sowohl auf Reize als auch auf das Überleben des Gesamtorganismus angewiesen sind, sind sie unselbständig. In diesem Sinne kann man die Organe als halb-autonome Einheiten bezeichnen.
Wagner findet in der biologischen Literatur die Tatsache, daß Organismen aus halbautonomen Einheiten aufgebaut sind, an vielen Stellen gewürdigt und faßt die verschiedenen Konzepte unter dem Stichwort der Modularität zusammen. Er zitiert das Konzept der Gen-Netze (Bonner, 1988)), wonach sich Gene und Genprodukte im Laufe der Entwicklung in getrennte Gruppen teilen und weist auf das alte Konzept der Dissoziabilität (Needham, 1933; Gould, 1977) der Entwicklungsprozesse hin, wonach sich diese, obwohl sie in der Natur integriert verlaufen, experimentell trennen lassen: so kann Wachstum ohne Differenzierung erfolgen, Kernteilung ohne Zellteilung etc.. Er selber schlägt vor, die Homologen der vergleichenden Anatomie, also die Grundeinheiten des morphologischen

Phänotyps, als entwicklungsmäßig und genetisch individualisierte Teile und als "Bausteine" (building blocks) des Anpassungsprozesses anzusehen.
Wagners besonderes Interesse gilt der Frage nach der Evolution des Strukturmerkmals Modularität: Worin liegt der Überlebenswert dieser Eigenschaft und wie wird sie hergestellt? Nach Wagner liegt der Vorteil einer modularen Struktur darin, daß der Evolutionsprozeß als solcher durch eine modulare Organisation erleichtert wird: wenn unabhängige biologische Funktionen existieren und im Genom unabhängig voneinander codiert sind, so erhöht sich dadurch die genetische Variierbarkeit einzelner Merkmale, da die Wahrscheinlichkeit von - möglicherweise tödlichen - "Nebenwirkungen" (pleiotropischen Effekten) verringert ist. Auf diese Weise kann die Evolution schrittweise verlaufen, eine Eigenschaft, die, wie Computersimulationen mit evolutionären Algorithmen zeigen, für die Effizienz der Evolution entscheidend ist. Eine ähnliche Vorstellung hatte bereits Hesse geäußert (siehe oben), ohne über die genaue Idee des genetischen Codes verfügen zu können.
Über die Art und Weise, wie die Evolution der Modularität vonstatten gehen könnte, sagt Wagner, daß der Weg der Parzellierung eines Ganzen als allgemeiner Mechanismus weiter verbreitet ist als die Integration von Einzelnem. Als Modell für den Mechanismus der Parzellierung durch Selektion schlägt er vor, daß jeweils immer nur einzelne Eigenschaftskomplexe zu einem gegebenen Zeitpunkt einem Druck, sich zu verändern, ausgesetzt sind, während alle anderen Komplexe dem Druck unterliegen, sich gerade nicht zu verändern.

Was bedeutet Modularität also in der Biologie? Auch in der Biologie, sind wie in der Technik Module Teile eines Gesamtsystems (meistens eines Organismus, aber z.B. auch einer Population von Individuen), welche voneinander relativ unabhängig ("halb-autonom") sind, weil sie über ein komplexes Eigenleben und relativ wenige Verbindungen zur Umwelt verfügen. Die Spezialisierung von Modulen auf unterschiedliche Funktionen und die Bildung von geschachtelten Strukturen sind herausragende Strukturmerkmale vieler biologischer Systeme. Aber auch das ungeschachtelte Nebeneinander gleichartiger Module tritt auf ("extensive/intensive Arbeitsteilung").
Wozu dient die Modularität von biologischen Systemen? Erst die Aufteilung in relativ unabhängige Teilsysteme ermöglicht funktionale Spezialisierung und Schachtelung. Aber Modularität hat einen zusätzlichen Vorteil, nämlich daß sie der Evolution die Möglichkeit bietet, den Phänotyp veränderten Umweltbedingungen schrittweise anzupassen, indem jeweils nur einige wenige Module verändert werden brauchen. Die unabhängige Veränderbarkeit von Modulen ist also ein Vorteil, den technische und biologische modulare Systeme teilen.
Wie entsteht die Modularität von biologischen Systemen? Auch in der Biologie gibt es einen "bottom-up" und einen "top-down" Weg zur Modularität, also die Integration von einzelnem und die Parzellierung eines Ganzen. Wie diese Umordnungen auf der Ebene des Genotyps, bzw. der Genotyp-Phänotyp-Funktion verursacht werden, ist eine aktuelle Frage der Forschung.

2.3 "The Modularity of Mind"

Es könnte sein, daß die Neurowissenschaft, insbesondere die kognitive Neurowissenschaft und die Neuroanatomie, die erste nicht-technische Disziplin gewesen ist, die explizit den Begriff Modularität gebraucht hat, ja wo dieser Begriff sogar einen zentralen Diskussionspunkt bezeichnet. Dies mag damit zu tun haben, daß schon seit vielen Jahren die Diskussion um eine Analogie von Gehirnen und Computern diese Wissenschaften prägt

und somit die Einführung eines zunächst so technisch anmutenden Begriffs hier naheliegender gewesen ist.

Neuroanatomische Module

In der Neuroanatomie werden die Begriffe modular und hierarchisch zur Beschreibung der neuronalen Feinstruktur der sensorischen Bereiche des Cortex, z.B. des visuellen Cortex', verwendet (VanEssen et al., 1990). Modular heißt in diesem Zusammenhang, daß nicht alles mit allem verbunden ist, sondern in bestimmten Arealen z.B. Streifenmuster der okularen Dominanz gefunden werden oder daß es parallele Informationsströme gibt, die unterschiedliche Merkmale des sensorischen Signals verarbeiten, wie die P- und M-Ströme, die die visuelle Detektion des "Was" bzw. des "Wo" eines Objekts leisten.

Fodors Vorstellung einer modularen Kognition

Was die kognitive von der rein anatomischen Neurowissenschaft unterscheidet, ist die Frage nach dem Zusammenhang von Gehirn und Geist. Es kann wohl kein Denken im eigentlichen Sinne ohne eine Art Gehirn geben, aber inwieweit Denken, Wahrnehmung und Bewußtsein durch die neurophysiologische Grundlage eingeschränkt oder determiniert werden ist zusammen mit der Frage, wie sich die Funktionsweise dieser Prozesse in psychologischen Kategorien beschreiben läßt, ein zentrales Problem, das man hier zu klären versucht.
Die Frage nach der Modularität des Gehirns entstammt dieser Diskussion und ist somit auch eine Frage nach der Modularität des Geistes. Aufgebracht wurde sie durch das einflußreiche Buch "The modularity of mind" von Jerry Fodor (Fodor, 1983; Fodor, 1985). Dieser hat darin Thesen über die Organisation von Gehirn und Geist aufgestellt, die ich im folgenden zusammenfassen möchte, bevor im anschließenden Abschnitt Rezeption und Kritik dieser Thesen geschildert werden.
Fodor richtet sich mit seinen Thesen vom Aufbau der Kognition gegen eine Richtung in der Neurowissenschaft, die in der Kognition alle Prozesse als vollständig untereinander verknüpft ansieht, den Interaktionismus. In Anschluß an die Ideen der Fakultätspsychologie von Gall aus den zwanziger Jahren, sieht Fodor die Kognition als prinzipiell zusammengesetzt an. Er teilt dabei die kognitiven Prozesse in zwei Arten, nämlich Wahrnehmung auf der einen und höhere Prozesse wie Denken und Problemlösen auf der anderen Seite. Von den Wahrnehmungsprozessen behauptet er, sie seien modular, während alle höheren Prozesse nicht-modular seien.
Modulare (Wahrnehmungs-) Prozesse sind nach seiner Auffassung durch folgende Eigenschaften gekennzeichnet:

- Sie sind informationell gekapselt (informationally encapsulated) (oder auch: kognitiv undurchdringlich, cognitively impenetrable). Sie laufen ab, ohne auf Informationen aus höheren Zentren oder anderen Modulen angewiesen zu sein. Insbesondere lassen sie sich nicht willentlich beeinflussen.

- Sie sind bereichsspezifisch (domain-specific). Sie verarbeiten nur jeweils bestimmte Arten von Daten.

- Sie liefern seichte Ausgaben (shallow outputs). Die Ausgaben müßen weiterverarbeitet werden, um semantischen Gehalt zu erlangen.

- Sie sind obligatorisch (mandatory). Ihre Ausführung kann nicht verhindert werden.

- Sie sind schnell, weil sie auf "festverdrahtete" (hard-wired) neuronale Verbindungen zurückgreifen.
- Sie sind deshalb auch angeboren (innate).
- Sie sind lokal.
- Sie haben eine bevorzugte Verarbeitungsrichtung (bottom-up).

Beispiele sind für Fodor die visuelle Wahrnehmung, aber auch das Verstehen gesprochener Sprache. Die Prozesse der höheren Kognition haben für Fodor dagegen all jene Eigenschaften, die die modularen Prozesse nicht haben; sie sind langsam, global, willentlich beeinflußt, haben keine bevorzugte Richtung und integrieren viele oberflächlich verschiedene Bereiche. Als ein analoges Beispiel für solche Prozesse nennt er den historischen Prozeß wissenschaftlicher Entdeckungen.
Aus Fodors früheren Arbeiten stammt seine Vorstellung von einer angeborenen einheitlichen "Sprache des Denkens" (language of thought, LOT), in der Repräsentationen der Außenwelt wie in einer formalen Grammatik manipuliert werden (Fodor, 1975). Diese Manipulationen finden nun nach seinen Architekturvorstellungen in den höheren Prozessen statt, während die Module die Übersetzungsarbeit leisten.

Rezeption von Fodor

Die Thesen von Fodor haben seit ihrem Erscheinen ein Leitbild dargestellt, das zu widerlegen oder zu bestätigen viele Wissenschaftler sich herausgefordert gefühlt haben und das so dazu beigetragen hat, die Aussagen der Neurowissenschaft wesentlich zu schärfen. Eines der Gebiete, in denen Fodors Thesen Anlaß zu fruchtbaren Diskussionen gegeben haben, ist die klinische und experimentelle Neuropsychologie, wo durch Moscovitch und Umiltà dem Merkmalskatalog von Fodor entsprechende Kriterien für die Modul-Eigenschaft definiert worden sind, die eine direkte Anwendung auf klinische Befunde zulassen (Moscovitch, 1995; Moscovitch and Umiltà, 1990). Drei Eigenschaften werden dabei als essentiell herausgestellt: Das Kriterium der informationellen Kapselung gilt danach als erfüllt, wenn die Funktion eines Moduls trotz allgemeinen intellektuellen Zusammenbruchs, der durch Degeneration oder Schäden in anderen Bereichen hervorgerufen ist, intakt bleibt. Als Beispiel nennt Moscovitch die Erhaltung von 3-D Objektrepräentationen in Alzheimer-Patienten, obwohl diese die Natur der Objekte nicht mehr erkennen. Bereichsspezifität ist erfüllt, wenn sich Belege für eine funktionale Spezialisierung finden, d.h. wenn lokale Schäden nur die Verarbeitung von bestimmten Arten von Information beeinträchtigen. Als Nachweis für solche nur lokalen Beeinträchtigungen dienen "Doppelte Dissoziationen", worunter man die Beobachtung versteht, daß ein Schaden an einem Ort eine Fähigkeit stŕker beeinträchtigt als eine andere, eine Verletzung an einem anderen Ort aber gerade das Gegenteil bewirkt. Schließlich ist das Kriterium der seichten Ausgabe erfüllt, wenn sich nachweisen läßt, daß in bestimmten Bereichen die normale Informationsverarbeitung zwar funktioniert, die Ausgaben aber nicht mehr semantisch interpretiert werden können. So gibt es Patienten, die auf einer präsemantischen Ebene noch mit visuellen Eingaben umgehen können, indem sie z.B. verschiedene Ansichten des gleichen Objekts einander zuordnen können, die diese Objekte aber nicht mehr identifizieren oder einer Klasse zuordnen können.
Mit Hilfe dieser Kriterien ist versucht worden, eine Reihe kognitiver Prozesse als modular zu klassifizieren. Als prototypisches Beispiel eines modularen Prozesses gilt dabei

die Wiedererkennung von Gesichtern, Review in (Nachson, 1995). Andere Prozesse sind dagegen hinsichtlich ihrer Einordnung als modular/nicht-modular noch umstritten: für räumliche Aufmerksamkeit etwa, die in den sogenannten "Neglect Syndromen gestört ist, ist unklar, ob es jeweils bereichsspezifische Mechanismen für visuelle, taktile, auditorische und motorische Aufmerksamkeit gibt (Umiltà, 1995) oder ob zumindest in einigen der Bereiche die Aufmerksamkeit durch gemeinsame Mechanismen gesteuert wird (Behrmann et al., 1995; Soroker et al., 1995).
Auch bei der Sprachwahrnehmung, einem klassischen Kandidaten für ein Modul, ist seit langem umstritten, inwieweit top-down Einflüsse (die für ein echtes Modul nach Fodor verboten sind) die Wahrnehmung beeinflussen. Faust et al haben dazu das interessante Ergebnis gefunden, daß sich die Gehirnhälften in diesem Punkt unterscheiden (Faust et al., 1995): Während in der linken Hemisphäre top-down Einflüsse in Form von Erwartungen bei der Worterkennung eine Rolle spielen (nicht-modular), ist dies in der rechten Hirnhälfte nicht der Fall (modular).
Ferner wird durch Moscovitch vorgeschlagen, daß auch das Gedächtnis, das nach Fodor eher den höherer Prozessen zugeordnet ist und damit als nicht-modular gilt, aus bereichsspezifischen Modulen aufgebaut sein kann(Moscovitch, 1995). Er schlägt vor das episodische Gedächtnis modular zu nennen, da Schäden an der relevanten Region (MTL/H) nur das Erinnern von kürzlich erworbenen, bewußt erlebten Informationen beeinträchtigen (Bereichsspezifität), das Gedächtnis im Falle von Demenz weitgehend erhalten bleibt, andererseits der willentliche Abruf von Erinnerungen auch im Normalfall extrem schwierig sein kann (informationelle Kapselung) und die vom MTL/H bereitgestellten Erinnerungen einer Interpretation durch höhere Prozesse erfordern, wie sich an den inkohärenten Erinnerungen von Patienten mit Schäden im Frontalbereich, aber nicht im MTL/H zeigt (seichte Ausgabe).
Andere Neuropsychologen hingegen betonen entgegen dem allgemeinen Trend, daß auch graduelle Modelle der Hirnfunktionen mit den klinischen Befunden konsistent sind (Goldberg, 1995) und verweisen auf die anatomisch im Vergleich zum Thalamus sehr homogene Struktur der Großhirnrinde.

Ein anderer Bereich neben der klinischen Neuropsychologie, dessen Theorieentwicklung durch das Leitbild der Modularität beeinflußt worden ist, ist die Entwicklungspsychologie, wo Annette Karmiloff-Smith die Frage der Angeborenheit der Module problematisiert hat (Karmiloff-Smith, 1994). Sie hält nach ihren Beobachtungen zwar eine angeborene Prädisposition für bestimmte Wahrnehmungsstrukturen für wahrscheinlich, betont andererseits aber die Herausbildung von Modulen im Laufe der Entwicklung. Modularisierung der kognitiven Prozesse ist nach ihrer Ansicht zusammen mit einem Prozeß der verstärkten Explizitheit und Bewußtheit von Wissen das wesentliche Ingredienz der kognitiven Entwicklung. Modularisierung wird dabei auf der Verhaltensebene diagnostiziert und bedeutet besonders eine verstärkte Automatisierung und Perfektionierung. Insbesondere verteidigt sie die Idee des Lernens und der Entwicklung in spezialisierten Bereichen gegenüber Piaget'schen Ideen von bereichsunabhängigen allgemeinen Entwicklungsstadien. Außerdem plädiert sie für ein Konzept, das weniger rigide auf der Trennung von Wahrnehmungsmodulen und höheren Prozessen besteht und bezweifelt auch die Existenz einer einheitlichen Sprache des Denkens.
Was bedeutet Modularität also in der Neurowissenschaft? Die einflußreichste Beschreibung von kognitiven Prozessen als einem modularen System stammt von Fodor. Die relative Unabhängigkeit der Module und ihre schwache Verbindung nach außen werden von ihm unter die Begriffe informationelle Kapselung und seichte Ausgabe gefaßt. Die

funktionale Spezialisierung von Modulen findet sich hier als Bereichsspezifität der Module beschrieben. Wichtig ist, daß Fodor die Struktur gedanklicher Prozesse beschreibt und damit nicht notwendigerweise die Struktur der neuronalen "Hardware". Wie diese zusammenhängen ist eine der umstrittenen Fragen des Gebietes.
Wozu dient Modularität in kognitiven Systemen und wie entsteht sie? Fodor nennt als Vorteil von Wahrnehmungsmodulen vor allem die Schnelligkeit der Informationsverarbeitung. Ähnliches wird auch von Karmiloff-Smith ausgedrückt, wenn sie Automatisierungsprozesse als Modularierungsprozesse beschreibt. Die Frage der Entstehung ist umstritten. Fodor hält die Modularität für angeboren, andere halten sie für eine erworbene Eigenschaft.
Antworten auf diese Fragen lassen sich auch mit Hilfe von mathematischen Modellen suchen. Das nächste Kapitel beschäftigt sich mit einer besonderen Klasse von Modellen, den "Künstlichen neuronalen Netzen", die dazu dienen kognitive Prozesse mit Hilfe von vereinfachten Modellen der neuronalen "Hardware" zu simulieren und auch eine genauere Untersuchung von Modularität erlauben.

2.4 Zusammenfassung

Aus der Verwendung des Begriffs Modularität in den verschiedenene Gebieten, wie ich sie in diesem Kapitel dargestellt habe, will ich nun versuchen eine Art allgemeiner Definition zu destillieren. Das Kernkonzept von Modularität läßt sich meiner Ansicht nach in folgender notwendigen und hinreichenden Bedingung zusammenfassen:

Definition: 1 *Die Struktur eines Systems ist modular, genau dann, wenn es aus Komponenten besteht, die in sich stärker zusammenhängen als untereinander. Diese Komponenten werden Module genannt.*

Module müssen nach dieser Definition eine komplexe innere Struktur besitzen, d.h. selbst aus Teilen bestehen. Diese Teile hängen auf vergleichbare Art miteinander zusammen, wie es die Module untereinander tun, aber der "Intra-Zusammenhang" ist in jedem Modul größer als der "Inter-Zusammenhang" zwischen den Modulen. Was "Zusammenhang" bedeutet, hängt dabei vom betrachteten System ab. In elektronischen Geräten manifestiert sich der Zusammenhang von Teilen durch Leitungen, in Programmen durch Funktionsaufrufe, in Organismen durch chemische Reaktionswege und physikalische Wirkungsketten und in neuronalen Systemen speziell durch elektrische Signalübertragung, bzw. auf der kognitiven Ebene betrachtet durch Information.

Zu dieser Kerndefinition treten weitere Merkmale hinzu, die zwar in modularen Systemen häufig anzutreffen sind, die aber nicht notwendig sind, um ein System modular nennen zu dürfen:

- Module haben unterschiedliche interne Strukturen.
- Module haben unterschiedliche Funktionen.
- Funktional oder strukturell gleichartige Module treten in einem modularen System wiederholt auf.
- Module sind ineinander geschachtelt und bilden eine in diesem Sinne hierarchische Struktur.

Die Eigenschaft von Modulen, unterschiedliche Funktionen haben zu können, wird in der Neuropsychologie, wie im letzten Abschnitt beschrieben, von einigen Autoren fast wie eine Definition von Modulen verwendet: genau dann, wenn "Bereichsspezifität" nachgewiesen werden kann, hat man ein Modul identifiziert - zumindest aber einen hervorragenden Kandidaten gefunden. Die Eigenschaft der informationellen Kapselung und damit der "Klumpenstruktur", die nach obiger Definition Modularität ausmacht, hat man damit aber genaugenommen noch nicht gezeigt, geschweige denn eine modulare Verknüpfungsstruktur auf neuronaler Ebene. Auf diese Problematik wird am Anfang des nächsten Kapitels im Zusammenhang mit einer Simulation durch künstliche neuronale Netze noch einmal eingegangen. Der Nachweis einer geschachtelten Struktur reicht dagegen zur Feststellung eines modularen Systems aus, da eine Schachtelung auf jeder Ebene eine Aufteilung in voneinander unabhängige Teilstrukturen impliziert. Sie ist aber keine notwendige Bedingung, d.h. es gibt nicht-geschachtelte modulare Systeme.

Wie läßt sich nun feststellen, ob und in welchem Maße ein System modular ist? Gefragt ist nach obiger Definition nach einem Maß für die "Klumpigkeit" der Struktur, also einer Funktion, die den Intra- und den Inter-Zusammenhang mißt und vergleicht. Unter Umständen werden sich dafür ähnliche Maße verwenden lassen, wie sie in sogenannten Clusterverfahren verwendet werden, also in Algorithmen, die in hochdimensionalen Datenräumen die Datenpunkte in Gruppen von ähnlichen Punkten fassen. Die genannten zusätzlichen Merkmale lassen sich dagegen nur schwer quantifizieren, insbesondere läßt sich die Funktionalität von Modulen nur schwer quantitativ vergleichen. Für die Fragen, wie ähnlich sich Module strukturell sind und wie stark geschachtelt die Gesamtstruktur ist, sind quantitative Vergleiche eher denkbar.

Kapitel 3

Modulare neuronale Netze

3.1 Ein Überblick

Künstliche neuronale Netze sind vereinfachte Modelle natürlicher Nervensysteme, die zum einen dem besseren Verständnis der biologischen Vorbilder dienen, zum anderen aber als Berechnungsmodelle für eine Vielzahl von technischer Aufgaben immensen Nutzen haben. Im folgenden wird die Darstellung den im vorigen Kapitel beschrittenen Weg in umgekehrter Richtung zurücklegen, also mit einigen Beispielen der Verwendung von künstlichen neuronalen Netzen in der kognitiven Neurowissenschaft beginnen, dann auf einige weniger zerebral angelegte Modelle eingehen, die das Laufverhalten von Tieren simulieren, und schließlich über stärker an technischen Anwendungen orientierte Modelle sprechen. In allen drei Gebieten zeigt sich, daß Modelle mit modularen Eigenschaften eine wichtige Rolle spielen.

3.1.1 Neuronale Netze in der Diskussion um kognitive Modularität

Es könnte sein, daß die Annahme, die Kognition sei aus Modulen zusammengesetzt, deshalb so einflußreich werden konnte, weil die Computermodelle ihrer Zeit eine solche Struktur nahelegten. Die Vorstellung, die sich Fodor von der Funktionsweise des Geistes macht, ist wohl durch die Methoden und Vorstellungen der symbolischen künstlichen Intelligenz (KI) mitgeprägt worden. Besonders auffällig ist die geistige Nähe in Fodors Analogie von der Manipulation von symbolischen Repräsentationen durch formale Grammatiken, wie sie der KI zugrundeliegen, mit der Funktionsweise der höheren kognitiven Prozesse. Aber auch die Aufteilung in eine zentrale Verarbeitungseinheit, an die einzelne Peripheriegeräte angeschlossen sind, erinnert an den Aufbau von für diese Zeit typischen Großrechenanlagen. Diese Anlagentechnik ist inzwischen durch Netzwerke von kleineren Rechnern in vielen Bereichen überholt worden und auch als Computermodelle der Kognition haben sich Netzwerke als Alternative etabliert.
Diese in der neurowissenschaftlichen Literatur oft mit Rumelhart (Rumelhart and McClelland, 1986) als PDP-Modelle oder als konnektionistische Modelle bezeichneten Netzwerke sind Verschaltungen von neuronenähnlichen Elementarfunktionen, die die Eingaben, die sie von anderen Neuronen oder von außen erhalten, gewichten, summieren und mit einer Aktivierungsfunktion verrechnet wieder an andere Neurone oder nach außen weitergeben. Das Verhalten der Netze wird dabei durch die Verknüpfungsstruktur und besonders durch die Gewichte der Verbindungen bestimmt: diese werden durch einen iterativen "Lern"-Algorithmus so angepaßt, daß ein gewünschtes Verhalten eintritt. Die Neuronen sind

in vielen Modellen in Schichten organisiert, d.h. in Gruppen von untereinander nicht verbundenen Neuronen, die mit jeweils allen Neuronen der Vorgänger- und Nachfolgeschicht verknüpft sind. Werden die berechneten Werte dabei immer nur in eine Richtung weitergegeben, d.h. liegt ein System ohne Rückkopplungen vor, wird durch ein solches Netz eine mathematische Funktion dargestellt. Gibt es dagegen Rückkopplungen, wird das Netz zu einem dynamischen System, in dem sich die Ausgaben der Neurone dauernd ändern können.
Solche rekurrenten Netze sind typische Modelle der Neurowissenschaft, mit deren Hilfe versucht wird, das Verhalten von gesunden oder hirnverletzten Menschen und gelegentlich auch anderen Tieren zu simulieren. In der Debatte um die Modularität des Geistes dienen diese Modelle häufig dazu, die Annahme von kognitiven Modulen zu widerlegen, z.B. indem gezeigt wird, daß mit ihnen bereichsspezifisches Verhalten simuliert werden kann, ohne daß die Netze selbst modular strukturiert sind. Plaut und Shallice haben dazu ein aus mehreren Schichten bestehendes rekurrentes Netz konstruiert, in dem die Ausgaben der Neuronen dreier Schichten als jeweils unterschiedliche Kodierungen von englischen Wörtern interpretiert werden (Plaut, 1995; Plaut and Shallice, 1993). So repräsentiert die erste Schicht die Rechtschreibung, die zweite die Bedeutung und die dritte die Aussprache eines Wortes. Außer diesen Schichten gibt es noch einige nicht interpretierte Zwischen- und Nebenschichten. Mit geeignet erzeugten Gewichten der Verbindungen verhält sich dieses Netz so, daß wenn man die Ausgaben der Rechtschreibeschicht auf der Repräsentation eines bestimmten Wortes festhält, das Netz in einen solchen Zustand läuft, daß in den anderen Schichten ebenfalls die diesem Wort entsprechenden Ausgaben erzeugt werden. Auf diese Weise wird der geistige Prozess des Lesens simuliert.
Aus diesem Netz werden dann Verbindungen entfernt, ohne daß die Gewichte der verbleibenden Verbindungen neu eingestellt würden. Sind diese "Verletzungen" stark genug, läuft das Netz nicht mehr in die korrekten Zustände, sondern macht "Fehler". Plaut und Shallice stellen fest, daß je nachdem, zwischen welchen Schichten Verbindungen zerstört werden, das System quantitativ unterschiedliche Fehler macht, nämlich einmal häufiger "abstrakte" und einmal häufiger "konkrete" Wörter falsch liest. Die Erklärung für dieses spezifisch abweichende Verhalten ist, daß die Repräsentationen der Wörter so gewählt worden sind, daß sich die Repräsentationen dieser beiden Wortgruppen in der Bedeutungsschicht stark voneinander unterscheiden.
Wenn ein verletztes Gehirn, je nachdem, wo es verletzt worden ist, unterschiedliche Arten von Fehlern macht, wird dies in der Neuropsychologie "doppelte Dissoziation" genannt und von Anhängern der Modultheorie - wie im vorigen Kapitel bereits erwähnt - als Beleg für Bereichsspezifität und damit für die Existenz jeweils getrennter Module für die Verarbeitung der beiden Arten von Daten gewertet. Plaut und Shallice zeigen nun, daß es nicht der Postulierung unterschiedlicher Module für das Lesen abstrakter und das Lesen konkreter Wörter bedarf, um eine solche doppelte Dissoziation zu erklären, sondern lediglich der Annahme stark unterschiedlicher Repräsentationen in einem ansonsten einheitlichen neuronalen System. Sie zeigen also, daß der Nachweis von Bereichsspezifität, also von funktional unterschiedlichen Teilsystemen nicht ausreicht, um auf eine zugrundeliegende strukturelle Modularität zu schließen.

Ein anderer Aspekt der Fodor'schen Modultheorie wird ebenfalls durch neuronale Netze in Frage gestellt, nämlich die Angeborenheit der Module. Allerdings wird von den Neurowissenschaftlern in diesem Punkt auf sehr unterschiedliche Modelle hingewiesen. Allen gemein ist der Hinweis darauf, daß neuronale Netze durch Interaktion mit ihrer Umwelt lernen und so ihre Struktur in der Form ihrer Gewichte ändern und auf diese Weise auch

modular werden können. Worin diese Modularität dann aber genau besteht, hängt von dem jeweils betrachteten Modell ab.
Eine in diesem Zusammenhang häufig zitierte Arbeit stammt von Jacobs und Jordan (Jacobs and Jordan, 1992). Sie lassen darin ein Netz ohne Rückkopplungen das "Was-und-Wo" - Problem lernen: die Ausgabeneuronen des Netzes sollen, nachdem die richtigen Gewichte gefunden worden sind, bei Eingabe der Repräsentation einer auf einem 5 mal 5 Kästchengitter situierten bestimmten Figur (etwa eines "T" aus 5 Kästchen links oben im Gitter) die korrekte Ausgaberepräsentation liefern. Diese besteht darin , daß genau eines der Neurone der einen Hälfte und eines der anderen Hälfte der Ausgabeneurone einen hohen Wert hat, wobei in der einen Hälfte auf diese Weise angezeigt wird, um welche Figur es sich handelt, in der anderen Hälfte dagegen, wo sich die Figur befindet. Rueckl hat gezeigt, daß diese Aufgabe besser von zwei getrennten als von einem gemeinsamen Netz gemeistert wird, und begründet damit die Existenz getrennter Informationsströme bei der Verarbeitung dieser Aspekte einer visuellen Wahrnehmung im Gehirn (Rueckl et al., 1989; VanEssen et al., 1990). Jacobs und Jordan lassen nun ein solch getrenntes Netz aus einem gemeinsamen Netz entstehen, indem sie den Neuronen Positionen im Raum zuordnen, die Ausgabeneuronen der beiden Hälften räumlich auseinanderziehen und dann bei der Einstellung der Gewichte die Gewichte besonders langer" Verbindungen besonders klein zu werden zwingen. Auf diese Weise wird durch ein besonders modifiziertes Lernverfahren ein in seiner Gewichtsstruktur modulares Netz erzeugt.

Karmiloff-Smith (Karmiloff-Smith, 1994) verweist in ihrer Arbeit über die Entwicklung von Kognition und Modularität auf Elman (Elman, 1990). Dieser hat ein ansonsten einfaches, aus einer Eingabe-, einer inneren und einer Ausgabeschicht bestehendes, vorwärtsgerichtetes Netz mit einer zusätzlichen Schicht ausgestattet, die mit der inneren Schicht in beide Richtungen verbunden ist, wodurch das Netz also zu einem dynamischen System wird. Die Eingabeneuronen werden dann in jedem Zeittakt mit Repräsentationen von in Sätzen geordneten aufeinanderfolgenden Wörtern gespeist. Der Zustand der rekurrent verbundenen Schicht beeinflußt die Berechnung der Werte der Ausgabeneurone und wird selber durch die gesamte Historie der eingegeben Wörter bestimmt. Dadurch wird dem Netz ein Gedächtnis verliehen, das es ermöglicht, bei der Eingabe gleicher Wörter unterschiedliche Folgewörter vorauszusagen, je nachdem, an welcher Position des Satzes sich das Wort befindet. Dies ist an sich schon ein besonderes Modell kognitiver Architektur, indem hier ein Gedächtnis nicht durch ein separates und passives "Informationslager", sondern durch eine integrierte, in jedem Zeitschritt aktiv an der Berechnung der Ausgaben teilhabende Schicht implementiert ist. Karmiloff-Smith kommt es besonders auf einen anderen Aspekt dieser Arbeit an, nämlich auf die Repräsentationen der Wörter in der versteckten Schicht, also die Ausgaben der Neurone dieser Schicht, wenn gerade ein bestimmtes Wort eingegeben wird. Diese Vektoren liegen nämlich nicht gleichmäßig im Raum der möglichen Aktivierungen verteilt, sondern haben eine Ähnlichkeitsstruktur, die Elman durch ein hierarchisches Clusterverfahren herausarbeitet und die der syntaktischen Rolle der Wörter in den Beispielsätzen entspricht. So führt die Eingabe von Wörtern, die in den Beispielsätzen auf ähnliche Weise verwendet werden, zu ähnlichen Aktivierungen der inneren Schicht, während unterschiedlich verwendete Wörter zu weniger ähnlichen Aktivierungen führen. Diese "Klumpenstruktur" im Raum der Aktivierungen ist ein Beispiel für eine Form gelernter Modularität und Karmiloff-Smith betont besonders den besonderen Status der Wort-Repräsentationen, erst im Laufe des Lernens aufgebaut zu werden.

Karmiloff-Smith weist aber auch auf Eigenschaften von konnektionistischen Modellen hin,

die die These von der Angeborenheit der Modularität unterstützen, und zwar besonders auf die Bereichsspezifität der meisten Modelle: in der Regel wird ein bestimmtes Netz nur für eine bestimmte Art von Daten verwendet, z.B. nur linguistische oder nur visuelle. Eine solche Bereichsspezifität könnte auch in der Natur der Fall sein. Auf jeden Fall aber müßten komplexere Netze als heute verwendet werden - zusammengesetzte, evtl. mit symbolischen Verfahren kombinierte Netze mit reichhaltigeren Eingabevektoren -, um höhere Prozesse wie die Übertragung von Wissen aus verschiedenen Gebieten und die im Laufe der Entwicklung größere Explizitheit von Wissen zu modellieren.

Von Goldberg (Goldberg, 1995) schließlich wird als ein Beleg dafür, daß es nicht notwendig ist, angeborene Module zu postulieren, noch auf eine besondere Form neuronaler Modelle hingewiesen, nämlich auf die Selbstorganisierten Karten oder Kohonen-Netze (Kohonen, 1984). Dieses sind (meist zweidimensionale) Gitternetze, die in den höherdimensionalen Raum der Eingaben eingebettet sind. Die Kreuzungspunkte des Gitters werden im Laufe des Lernens so verschoben, daß ihre Lage den Zentren von natürlichen "Klumpen" (Clustern) in den Eingabedaten entspricht. Das Gitter verzerrt sich dabei zwar, aber im Gitter benachbarte Punkte bleiben auch in ihrer räumlichen Lage benachbart, wodurch umgekehrt die Nachbarschaftsbeziehungen der Eingabedaten in jene des Gitters abgebildet werden ("Topologieerhaltung"). Solche Netze werden in der Neurowissenschaft als Modelle für die Entstehung von senso-motorischen Karten im Cortex betrachtet, wie dem bekannten Homunculus der taktilen Empfindung und können auch die Entstehung von okularen Dominanzstreifem im visuellen Cortex erklären (Erwin et al., 1995). Ritter und Kohonen zeigen, daß solche Karten auch die Clusterstruktur ähnlich verwandter Wörter abbilden können, also auf gewisse Weise jene Struktur explizit machen, die in der Gedächtnisschicht eines Elman-Netzes nur implizit in der Verteilung der Aktivierungen vorhanden ist (Ritter and Kohonen, 1989). Goldberg argumentiert, daß die adaptive topologieerhaltende Eigenschaft dieser Karten unnötig wäre, wenn die geistigen Funktionen von vornherein in strukturelle Module gegliedert wären.

3.1.2 Modulare motorische Netze

Die Modelle der Neurowissenschaft sind "kopflastig" in dem Sinne, daß in erster Linie die Umwandlung von sensorischen Eingaben in Wissen untersucht wird. Für die Ausgabeseite, die Handlungen, also die Aktivierung von motorischen Neuronen und Muskeln interessieren sich dagegen eher Biologen, die das Verhalten von Tieren untersuchen. Besonders das Laufverhalten von Insekten ist gut untersucht, weil diese Tiere mit einem relativ einfachen Nervensystem bereits ein immens gut koordiniertes und angepaßtes Laufen zeigen. Interessant ist das Laufverhalten besonders auch deshalb, weil Sensorik, Motorik und zentrale Prozesse besonders gut abgestimmt sein müssen, um einerseits planvolle und koordinierte Bewegungen ausführen zu können, andererseits aber auch ausreichend schnell auf Hindernisse und wechselnde Bodenbeschaffenheit zu reagieren.

Cruse zeigt, daß das in der Natur beobachtete Laufverhalten eines sechsbeinigen Insekts, der Stabheuschrecke Carausius morosus, durch ein Modell simuliert werden kann, das weitgehend ohne zentrale Prozesse auskommt (Cruse et al., 1995). Dies wird erreicht, indem jedes Bein als ein eigenständiges Modul modelliert wird, das das Bein zunächst unabhängig vom Zustand der anderen Beine schwingen und stemmen läßt. Die Gesamtkoordination des Ganges wird dann durch rein lokale, nur je benachbarte Beine betreffende, Wechselwirkungen erreicht, und zwar indem die vorderen und hinteren Umkehrpunkte

der Einzelbeinbewegung abhängig vom Zustand der Nachbarbeine nach vorne oder hinten verschoben werden. Die Einzelbeinmodule bestehen wiederum aus Untermodulen für die Steuerung der Schwing- bzw. der Stemmphase und einem Modul, das zwischen diesen Modi abhängig von den aktuellen Umkehrpunkten umschaltet. Diese Module sind als neuronale Netze implementiert. Obwohl alle Koordinationseinflüsse rein lokal sind, sind alle Beine des Tieres über den Laufuntergrund mechanisch verkoppelt. Dennoch bedarf es trotz dieses globalen Effektes in Cruses Modell keiner zentralen Steuerung, um eine korrekte Positur des Gesamtkörpers aufrechtzuerhalten. Global vorgegebene Grössen sind allein die Richtung und Geschwindigkeit der Gesamtbewegung. Cruses Modell vereint viele typische Eigenschaften modularer Systeme, wie sie im vorigen Kapitel beschrieben wurden. Die wichtigste ist die relative Unabhängigkeit der Beinmodule, die nur über eine schmale Schnittstelle, nämlich ihre Umkehrpunkte, mit den anderen Modulen verbunden sind. Diese Beinmodule sind wiederholte gleichartige Module, man findet mit den Schwing- und Stemmnetzen aber auch funktionale Spezialisten . Die Gliederung in ein hierarchisches System mit Untermodulen ist ebenfalls eine typische Eigenschaft modularer Systeme. Eine weitere wichtige Eigenschaft von Cruses Modell ist, daß seine modulare Struktur fest vorgegeben ist. Einzelne Mechanismen sind aus Verhaltensexperimenten abgeleitet, so die spezielle Form der Koordinationseinflüsse. Andere Mechanismen, etwa die Funktionsweise des Stemmnetzes sind eher hypothetischer Natur. Allein die Schwingmodule sind typische neuronale Netze, insofern als deren Gewichte anhand von Beispieldaten eingestellt wurden.

Andere Autoren haben versucht, neuronale Modelle sechsbeinigen Laufens aufzustellen, die mit weniger genauen Strukturvorgaben arbeiten und in denen mehr "gelernt" wird. Beer und Gallagher stellen ein Modell vor, in dem zunächst die Gewichte eines in sich unstrukturierten Einzelbeincontrollers gelernt werden, der dann sechsfach repliziert von den Autoren zu einem Gesamtsystem verbunden werden - mit zu lernenden Verbindungsgewichten zwar, aber doch auf sehr strukturierte Weise (Beer and Gallagher, 1992). Insbesondere ist die grundlegende Modularität des Systems auch hier vorgegeben. Eine Eigenart der von Beer verwendeten rekurrenten neuronalen Netze ist, daß sie nicht in diskreten Zeitschritten ablaufen, sondern durch ein kontinuierliches Differentialgleichungssystem beschrieben werden, das mit einer Runge-Kutta Methode numerisch gelöst wird. Für solche Netze existieren zwar den diskreten Lernverfahren entsprechende Verfahren, die aber noch kaum bekannt und schwer zu handhaben sind (Pearlmutter, 1995). Beer jedenfalls stellt die Gewichte in seinem Modell mit Hilfe eines "genetischen Algorithmus" ein. Ein genetischer Algorithmus (oder evolutionärer Algorithmus) ist ein sehr allgemein verwendbares Optimierungverfahren, das eine Menge von Datenstrukturen manipuliert, die jede eine mögliche Lösung des Optimierungsproblem darstellen. Die Datenstrukturen werden dabei Individuen genannt und die Menge der Individuen eine Population. Die Güte ("Fitneß") jedes Individuums wird ermittelt, die besten Individuen ausgewählt, zufällig verändert, Teile von ihnen untereinander vertauscht und schließlich von jedem Individuum der so entstandenen neuen Population wieder die Güte bestimmt - und so weiter, bis die Güte des besten Individuums der Population ausreichend hoch ist. Bei Beer sind die Individuen nun Felder mit sovielen Einträgen, wie es Gewichte für das Netz zu bestimmen gilt, und die Güte der Gewichte wird bestimmt, indem das Modell gestartet und das resultierende Laufen bewertet wird.

Andere Autoren benutzen genetische Algorithmen, um nicht nur die Gewichte der Verbindungen eines neuronalen Netzes, sondern gleichzeitig auch die Verbindungsstruktur des

Netzes selbst zu lernen. Es gibt viele Möglichkeiten, die Verbindungsstruktur eines Netzes als Datenstruktur zu repräsentieren. Die einfachste Form der Repräsentation ist ein Feld mit sovielen Einträgen, wie es mögliche Verbindungen gibt, wo die An- bzw. Abwesenheit einer Verbindung durch Nullen und Einsen angezeigt wird. In dieser direkten Kodierung ist die Anzahl der zu optimierenden Parameter sehr hoch und der logische Zusammenhang der Gesamtstruktur nicht wiedergegeben, was besonderen Aufwand bei der Veränderung und Vertauschung von Teilstücken nach sich zieht. Gruau benutzt eine kompaktere Kodierung, die einem Wachstumsprozeß nachempfunden ist (Gruau, 1995): Ein Individuum ist bei ihm eine Sammlung von Regeln (eine "Graph-Grammatik"), nach denen sukzessive aus einem Anfangsneuron Töchterneurone entstehen, die mit ihrem Elternneuron verbunden werden. Aufgrund der rekursiven Natur dieser Regeln ist es möglich, daß einzelne Unterstrukturen im Netz mehrfach identisch vorkommen, aber nur einmal in den Regeln kodiert sind. Gruau definiert Modularität als diesen Zusammenhang von Geno- und Phänotyp. Er verwendet sein Verfahren zur Optimierung der Gewichte und der Verbindungsstruktur eines durch kontinuierliche Differentialgleichungen beschriebenen rekurrenten neuronalen Netzes, das sechsbeiniges Laufen simuliert. Dabei entsteht die modulare Struktur des Netzes, nämlich die wiederholte Verwendung der gleichen Unterstruktur für jedes Bein, im Gegensatz zum Modell von Beer völlig selbständig im Laufe der Optimierung.

Kodjabachian und Meyer vereinfachen das Verfahren von Gruau und fügen ihm eine räumliche Komponente hinzu: in den Regeln ist jetzt auch die räumliche Position eines Neurons codiert (Kodjabachian and Meyer, 1996). Die Verbindungen entstehen nicht mehr aufgrund von logischen Eltern/Tochter- Beziehungen, sondern aufgrund von räumlicher Nähe. Auch ihnen gelingt es, mit diesem Verfahren ein neuronales Netz zu erzeugen, das Laufen simulieren kann. Allerdings ist die repetitive Grundstruktur jetzt wieder künstlich angelegt, indem sechs sinnvoll positionierte Anfangsneurone verwendet werden. Kodjabachian und Meyer zeigen insbesondere auch, wie das Verfahren der Graph-Grammatik-Codierung zur schrittweisen Evolution von hierarchisch organisierten Netzen benutzt werden kann, indem nämlich eine einmal evoluierte Struktur samt ihren Gewichten in den Regeln als Ganzes referenziert werden kann. Auf diese Weise fügen sie dem reinen Laufnetz Steuerungsnetze für Richtung und Geschwindigkeit hinzu.

Bei Beer, Gruau und Kodjabachian/ Meyer ist die verwendete Definition von Laufen eine stark vereinfachte Karikatur der in einem realen System ablaufenden Bewegungen. Die Beine haben jeweils nur zwei Freiheitsgrade, hoch und runter bzw. nach vorne und zurück, während in einem echten Tier mehrere über verschiedene Arten von Gelenken verbundene Beinsegmente sich im dreidimensionalen Raum bewegen. Auch wird die aus den Beinstellungen resultierende Positur des Tieres in diesen Simulationen nicht berücksichtigt, sondern nur das Phasenverhältnis der einzelnen Beinoszillatoren. Eine naturnähere Simulation ist bisher nur durch manuell weitgehend vorstrukturierte Modelle wie Cruses möglich. Auch dieses macht aber noch viele Abstraktionen von der Wirklichkeit, z.B. indem es die Massen, die bewegt werden, und die damit verbundenen Kräfte vernachlässigt.

3.1.3 Neuronale Netze in technischen Anwendungen

Die technischen Anwendungen neuronaler Netze sind vielfältig und den oben beschriebenen Simulationen der Natur oft sehr verwandt. So stellen die Modelle von Wahrnehmungsprozessen wie dem Sehen oder dem Erkennen von gesprochenen Wörtern und Sätzen aus technischer Sicht Lösungen von schwierigen Mustererkennungsaufgaben dar, die mit

herkömmlichen Algorithmen oft nicht ausreichend gut gemeistert werden können. Von großer Bedeutung ist dabei, daß die Modelle anhand von Beispieldaten gelernt werden, so daß man nicht auf aus allgemeinen Überlegungen und Vorwissen abgeleitete Formeln angewiesen ist, die in einer konkret betrachteten Aufgabe ohne weitere Anpassung u.U. nicht mehr angemesssen sind.

Das weitaus häufigste Netzmodell ist das vorwärtsgerichtete, in Schichten organisierte Netz, das wie oben beschrieben eine mathematische Funktion darstellt. Dieses Netzmodell wird auch als Multilagenperzeptron (MLP) bezeichnet. Durch das Lernen der Gewichte erhält man eine Approximation einer in Form von einzelnen Eingabe-Ausgabepaaren gegebenen Funktion. Da diese Beispiele in der Regel Stichproben einer zugrundeliegenden Gesamtmenge sind, hat man es mit einem statistischen Schätzproblem zu tun: die Parameter (Gewichte) eines durch die Verbindungsstruktur und die Aktivierungsfunktion gegebenen Funktionen-Modells sollen so geschätzt werden, daß die Beispieldaten möglichst gut erklärt und neuen Eingaben aus der Grundmenge möglichst gute, d.h. dem zugrundeliegenden Prozeß entsprechende, Ausgaben zugeordnet werden. Solche Schätzverfahren haben früher oft lineare Zusammenhänge zwischen den (evtl. noch nicht-linear transformierten) Eingabedaten angenommen. Die in der Regel schwellwertartigen Aktivierungsfunktionen machen ein neuronales Netz jedoch zu einem flexibleren, nicht-linearen Modell. Gleichzeitig ist diese Nichtlinearität aber durch den beschränkten Wertebereich der Aktivierungsfunktionen und die verteilte Struktur des Modells so weit gezähmt, daß robuste Schätzungen noch möglich sind.

Ein klassisches Beispiel für eine solche Anwendung ist das maschinelle Sehen, z.B. die Erkennung von Adressen auf Briefen oder die Klassifizierung von gefertigten Bauteilen als Ausschuß in der Qualitätskontrolle. Auch die Umwandlung der akustischen Signale gesprochener Sprache in Lautkategorien und schließlich Buchstaben und Texte, wie sie in automatischen Diktiergeräten zu bewerkstelligen ist, ist ein komplexes Mustererkennungs/ Klassifizierungsproblem, bei dem sich die Eigenschaft neuronaler Netze, sich an die Eigenheiten eines Sprechers "gewöhnen" zu können, besonders bezahlt macht. Ein weiterer breiter Anwendungsbereich für nicht-lineare statistische Regression ist die Vorhersage von Zeitreihen jeglicher Art, von Wetterdaten bis zu Börsenkursen, um gleich die schwierigsten Aufgaben zu nennen. Wichtige Anwendungen haben auch Netze, die darauf trainiert sind, ihre Eingabe wieder auszugeben, nämlich als nicht-lineare Filter und Datenkomprimierer.

Abgesehen von der Gliederung in Schichten untereinander nicht verbundener Berechnungselemente sind vorwärtsgerichtete Netze sonst in der Regel nicht weiter strukturiert, d.h. alle Neurone einer Schicht sind mit allen Neuronen der Vorgänger- und der Nachfolgeschicht verbunden. Gerade dieser hohe Verknüpfungsgrad, die Verteiltheit der Repräsentationen, die massive Parallelität sind es, die die flexible Robustheit der Netze ausmachen. Dennoch kann das Prinzip der Modularität, das in anderen Bereichen der Technik und Informatik so fruchtbare Verwendung findet, auch hier einen Gewinn ermöglichen. Am Beispiel der Laufnetze haben wir gesehen, daß die angemessen modulare Struktur der Modelle - manuell vorgegeben oder automatisch gefunden - entscheidend für die Bewältigung der Laufaufgabe durch rekurrente neuronale Netze ist. In diesem und dem nächsten Kapitel möchte ich zeigen, welche Möglichkeiten es gibt, auch vorwärtsgerichtete Netze manuell oder automatisch modular zu strukturieren und wie die Leistung der Netze dadurch verbessert werden kann.

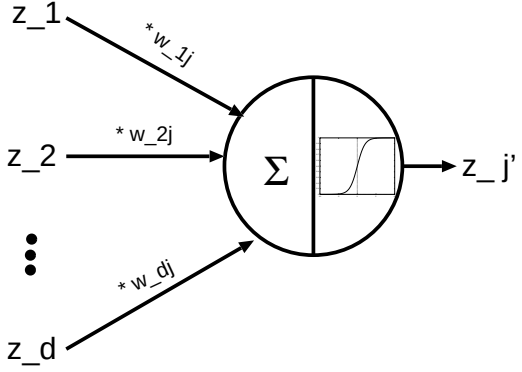

Abbildung 3.1: Funktionsweise eines Neurons mit logistischer Aktivierungsfunktion. Der Bias-Term ist nicht dargestellt.

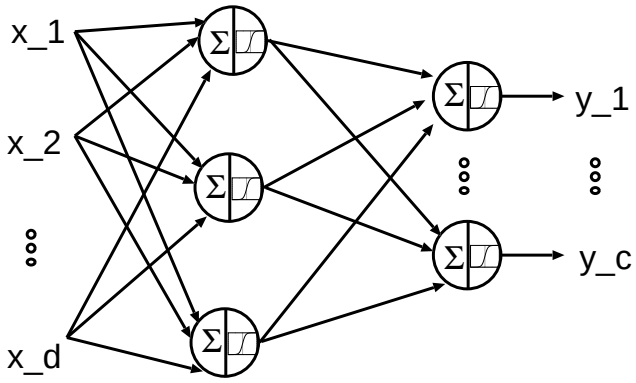

Abbildung 3.2: Aufbau eines Multilagen-Perzeptrons.

3.2 Multilagenperzeptrone (MLP): Nicht-lineare statistische Regression und Klassifikation

Mit diesem Abschnitt möchte ich eine kurze Einführung in die Theorie der Multilagen-Perzeptrone geben, soweit sie zum Verständnis des Rests des Arbeit notwendig ist. Eine vollständigere Darstellung findet man z.B. in den Lehrbüchern von Bishop, Ripley und Rojas (Bishop, 1995; Ripley, 1996; Rojas, 1993).

3.2.1 Aufbau und Aufgabe

Mulitilagenperzeptrone (MLP) sind Verschaltungen von neuronenähnlichen Elementarfunktionen, die die Eingaben, die sie von anderen Neuronen oder von außen erhalten, gewichten, summieren und mit einer Aktivierungsfunktion verrechnet wieder an andere Neurone oder nach außen weitergeben (siehe Abbildung 3.1. Die Neuronen sind dabei in Schichten organisiert, d.h. in Gruppen von untereinander nicht verbundenen Neuronen, die mit jeweils allen Neuronen der Vorgänger- und Nachfolgeschicht verknüpft sind (siehe Abbildung 3.2.
Seien $z_i, i = 1 \ldots, d$ die Ausgaben von d Neuronen einer Vorgängerschicht, $g : \Re \to \Re$ eine Aktivierungsfunktion genannte, meist sigmoidale (s-förmige) Transferfunktion und

$w_{ij}, i = 0, \ldots, d, j = 1, \ldots, c$ "Gewichte" genannte reelle Zahlen. Dann berechnen sich die Ausgaben von c Neuronen der betrachteten Schicht aus den Ausgaben der Vorgängerschicht durch

$$z_j^{'}(z_1, \ldots, z_d) = g(\sum_{i=1}^{d} w_{ij} z_i + w_{0j}), j = 1, \ldots, c.$$

Eine typische Aktivierungsfunktion ist z.B. die logistische Aktivierungsfunktion (Fermi-Funktion) $g(x) = 1/(1 + \exp(-x))$. Das Gewicht w_{0j} wird auch als "bias" bezeichnet.

Die berechneten Werte werden im MLP immer nur in eine Richtung weitergegeben, d.h. es liegt ein System ohne Rückkopplungen vor. Ein solches vorwärtsgerichtetes Netz stellt eine Funktion dar. Die Eingabestellen dieser Funktion werden oft als eine Schicht von "Eingabeneuronen" dargestellt, deren Ausgabe der Wert der Eingabe ist, die die Eingaben also nur "durchreichen". Die erste eigentliche Berechnungsschicht heißt "innere Schicht", ebenso wie evtl. folgende Schichten bis auf die letzte Schicht, die Ausgabeschicht genannt wird. Ist d die Anzahl der Eingabeneuronen und c die Anzahl der Ausgabeneuronen, werden durch das Netz d-dimensionale reelle Eingabevektoren $x = (x_1, \ldots, x_d)$ auf c-dimensionale reelle Ausgabevektoren $y = (y_1, \ldots, y_c)$ abgebildet. Die Funktionswerte y hängen außer von den Eingaben noch von den Parametern w ab, den "Gewichten", so daß sich schreiben läßt:

$$y = y(x, w).$$

Die "Lernaufgabe" für das Netz ergibt sich aus der Angabe einer endlichen Menge von Wertepaaren $\mathcal{D} = \{(x_i, t_i) \in \mathcal{T} \mid i = 1, \ldots, n\}$, der sogenannten "Trainingsmenge". Diese Menge ist eine Stichprobe einer meist als unendlich angenommenen unbekannten Grundgesamtheit $\mathcal{T} \subset \Re^d \times \Re^c$ von Wertepaaren, die als "Testmenge" bezeichnet wird. Ziel ist es, mit Hilfe der Trainingsmenge und Annahmen über die Testmenge, Gewichte zu finden, so daß die Wertepaare der Testmenge $\mathcal{T}$ durch das resultierende Netz optimal angenähert werden. Die Lernaufgabe eines neuronalen Netzes ist damit eine statistische Regressionsaufgabe.
Einen wichtigen Sonderfall der statistischen Regression bildet die Klassifikation. Hier bestehen die Zielvektoren t_i nicht aus kontinuierlichen, sondern aus diskreten Variablen, die die Klassenzugehörigkeit der Eingaben anzeigen.
Die Gewichte w werden durch Minimierung eines auf der Trainingsmenge definierten Fehlermaßes bestimmt, das den Abstand der Netzausgaben zu den Zielwerten mißt. Häufig ist das die mittlere quadratische Abweichung

$$E = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{c} (y^j(x_i, w) - t_i^j)^2.$$

Die trainingsfehler-minimalen Gewichte können in der Regel nicht exakt ausgerechnet, sondern nur durch iterative Lern-Verfahren angenähert ermittelt werden.
Eine alternatives Netz-Modell zur Lösung von nicht-linearen Regressionsaufgaben ist das "Radiale-Basis-Funktionen" -Netz (RBF-Netz). Hier werden wie beim MLP zunächst Aktivierungen der inneren Schicht berechnet. Anders als beim MLP wird nicht das Skalarprodukt des Eingabevektors mit dem Gewichtsvektor des jeweiligen inneren Neurons gebildet, sondern der euklidische Abstand des Eingabevektors zu einem Gewichtsvektor bestimmt, der also eine Art "Prototyp" darstellt. Während beim MLP der ermittelte

Wert durch eine sigmoide, also s-förmige, schwellwertartige Aktivierungsfunktion transformiert wird, ist die Aktivierungsfunktion eines RBF-Netzes eine radiale Basisfunktion, meist die Gaußsche Glockenfunktion, die mit zunehmendem Abstand des Eingabevektors vom Prototypen rasch gegen Null abfällt. Die so berechneten Aktivierungen werden dann wie im MLP linear überlagert. Im RBF-Netz sind im Gegensatz zum MLP bei einer gegebenen Eingabe immer nur wenige innere Neuronen aktiv. In diesem Sinne spricht man von lokalen Repräsentationen im RBF-Netz im Vergleich zu verteilten Repräsentationen im MLP. RBF-Netze werden im Rahmen dieser Arbeit nicht weiter behandelt, darum verzichten ich an dieser Stelle auf einen weitergehenden Vergleich der beiden Netzarten.

3.2.2 Lernverfahren

Das klassische Verfahren zum Training von MLP wird "Backpropagation" genannt und ist ein Gradientenabstiegs- Verfahren, d.h. es minimiert den Fehler, indem es die Gewichte in kleinen Schritten in Richtung des negativen Gradienten des Fehlers verändert, also die Fehlerlandschaft quasi hinuntersteigt. Sei E der oben definierte mittlere quadratische Fehler, dann berechnet sich die in einem Lernschritt vorgenommene Änderung eines Gewichtes Δw_{ij} als

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}}.$$

Dabei ist η ein Parameter, der die Lernschrittweite angibt. Um den Fehler E zu bestimmen, muß man die gesamte Trainingsmenge einmal durchlaufen. Möchte man die Gewichte sofort nach der Präsentation eines Beispiels verändern, benutzt man stattdessen den Fehler E_i auf einem Trainingsbeispiel, für den gilt

$$E = \sum_{i=1}^{n} E_i$$

und bildet

$$\Delta w_{ij} = -\eta \frac{\partial E_i}{\partial w_{ij}}.$$

Die partielle Ableitung läßt sich nun als das Produkt zweier Terme berechnen, der Aktivierung z_i des Neurons i und eines am Neuron j ankommenden, von den Ausgabeneuronen zurückgeleiteten, Fehlersignals δ_j:

$$\Delta w_{ij} = -\eta z_i \delta_j.$$

Das Fehlersignal δ_j berechnet sich rekursiv aus den Fehlersignalen δ_k der näher an der Ausgabe gelegenen Nachbarschicht nach der Formel

$$\delta_j = g'(a_j) \sum_k w_{kj} \delta_k.$$

Dabei ist $g'(a_j)$ die Ableitung der Aktivierungsfunktion an der Stelle, an der auch die Aktivierung des Neurons j berechnet wird. Für die Ausgabeneuronen gilt $\delta_k = y^k - t^k$. Ist g die logistische Aktivierungsfunktion $g(x) = 1/(1 + \exp(-x))$ gilt einfach

$$g'(a) = g(a)(1 - g(a)).$$

Problematisch ist im Backpropagation-Verfahren die Größe der Lernschrittweite η. Ist η sehr klein, benötigt man unzumutbar viele Iterationen. Ist η aber zu groß kann es zu Oszillationen kommen, wenn das Fehlerminimum übersprungen wird. Verbesserungen von Backpropagation machen die Lernschrittweite daher variabel oder führen einen Trägheitsterm ein, der Oszillationen unterdrückt. Das heute modernste numerische Optimierungsverfahren, die "Methode der konjugierten Gradienten" ("Conjugate-Gradient" oder CG-Methode). Es führt bei quadratischen Fehlerlandschaften zu genau so vielen Lernschritten wie es Gewichte gibt (wie der Raum der Gewichte Dimensionen hat). Man kann dieses Verfahren als eine Art Backpropagation-Verfahren mit Trägheitsterm ansehen, in dem die optimale Lernschrittweite in jedem Schritt mittels einer Suche entlang einer Richtung ("line-search") gefunden wird und der Koeffizient des Trägheitsterms ebenfalls dynamisch angepaßt wird, so daß die Suchrichtungen auf optimale Weise aufeinander folgen (Masters, 1995).

3.2.3 Das Problem der Generalisierung

Die Gewichte eines MLP werden eingestellt, indem ein Fehlermaß auf der Trainingsmenge minimiert wird. Jedoch ist selbst, wenn ein Satz von Gewichten existieren sollte, der den Trainingsfehler verschwinden läßt, nicht sichergestellt, daß das entsprechende Netz auch die Testmenge gut annähert, daß es, wie man sagt, gut "generalisiert". Zum einen ist schon die Lösung der Minimierungsaufgabe bei einer gegebenen Trainingsmenge oft nicht eindeutig. Möglicherweise minimieren viele sehr unterschiedliche Netze den Trainingsfehler, auch solche, die der zugrundeliegenden Testmenge nur sehr wenig entsprechen. Doch auch ein eindeutiges Fehlerminimum würde noch keine gute Generalisierung gewährleisten, da Lage und Wert des Minimums von der Zusammensetzung der Stichprobe, der Trainingsmenge, abhängen. Erst wenn gewährleistet ist, daß über viele verschiedene Stichproben ein gleichmäßig geringer Fehler erzielt wird, ist sichergestellt, daß auch der Fehler auf der Testmenge gering sein wird. Genauer gesagt, müssen gleichzeitig sowohl der Erwartungswert als auch die Varianz des minimalen Trainingsfehlers über alle möglichen Trainingsmengen minimal sein.

Ein Beispiel für eine Art von Daten, bei denen der vollständige "Fit" einer einzelnen Trainingsmenge sogar immer zu einer schlechten Generalisierung führen muß, sind verrauschte Meßwerte. Man stelle sich einen Prozeß vor, der durch eine zugrundeliegende Funktion h(x) beschrieben ist, wo aber bei jeder Messung von Wertepaaren die Ausgabevektoren durch eine zufällige additive Abweichung überlagert sind, so daß sich schreiben läßt:

$$t(x) = h(x) + \epsilon,$$

wobei ϵ eine normalverteilte Zufallsvariable mit Mittelwert Null ist. Um eine gute Generalisierung zu erzielen, muß in einem solchen Fall eigentlich die Funktion h(x) approximiert werden, gegeben sind aber nur verrauschte Wertepaare $\{(x_i, t_i(x_i))\}$. Praktisch heißt das, daß beim Training ein durch das Rauschen verursachter Restfehler in Kauf genommen werden muß. Ein verschwindender Trainingsfehler würde lediglich bedeuten, daß das Netz auch an das prinzipiell nicht modellierbare, da unsystematische Rauschen ϵ, und nicht nur an die allein für Vorhersagen relevante Funktion h(x) angepaßt worden ist. Eine zu genaue Anpassung an verrauschte Daten wird als "Overfitting" bezeichnet.

Der wichtigste Ansatzpunkt, um das Problem der Generalisierung in den Griff zu bekommen, besteht darin, die Flexibilität des Netzes so klein wie möglich zu halten, d.h. die

Menge der durch das Netz darstellbaren Funktionen einzuschränken. Wenn ein Netz nur noch eine kleine Menge von Funktionen darstellen kann, wird es weniger unterschiedliche fehlerminimale Gewichtssätze bei gegebener Trainingsmenge geben und die Varianz des minimalen Fehlers über verschiedene Trainingsmengen aus der gleichen Testmenge wird geringer sein. Ist die Flexibilität des Netzes aber zu gering, erkauft man sich dies durch einen höheren minimalen Fehler bei gegebener Trainingsmenge, bzw. durch einen erhöhten Erwartungswert des minimalen Fehlers über verschiedene Trainingsmengen. Dieser Zusammenhang ist als "bias-variance" -Dilemma bekannt, das aber kein Dilemma im eigentlichen Sinne ist, da in einem geeignet begrenzten Modell durchaus sowohl der Erwartungswert als auch die Varianz klein sein können.

3.2.4 Struktur-Optimierung

Es gibt verschiedene Möglichkeiten, die Flexibilität eines MLP einzuschränken. Man kann die Anzahl der inneren Neurone klein wählen, man kann weniger Verbindungen zwischen den Schichten zulassen als bei einer vollständigen Verknüpfung und man kann die Wertebereiche der Gewichte einschränken. Diese Einschränkungen müssen so gemacht werden, daß die gewünschte optimale Funktion durch das Netz noch ausreichend gut angenähert werden kann. Die Annahmen über die Testmenge, die durch diese Einschränkungen realisiert werden, müssen also realistisch sein. Außerdem muß der Einfluß, den solche Einschränkungen auf den iterativen Algorithmus, der den Trainingsfehler minimiert, haben, berücksichtigt werden. Unter Umständen kann eine für das Generalisierungsverhalten theoretisch günstige Einschränkung das Lernen der Trainingsmenge erleichtern, sie kann es aber auch erschweren. Aus diesen Gründen werden Verfahren, die nach der optimalen Struktur im Raum der Modelle, d.h. nach den gewinnbringendsten Einschränkungen, suchen, oft mit den Verfahren, die nach den optimalen Gewichten bei gegebenen Einschränkungen suchen, also den Lernverfahren, kombiniert. Vier Klassen von Struktur-Optimierungs Algorithmen lassen sich besonders herausstellen, nämlich Pruning-Verfahren, Konstruktions-Verfahren, Regularisierungs-Verfahren und Topologie-Evolutions-Verfahren.

In Pruning-Verfahren wird das Netz zunächst zuende trainiert. Dann werden nacheinander die "unwichtigsten" Verbindungen aus dem Netz entfernt, wobei nach jeder Wegnahme einer Verbindung das Netz nachtrainiert wird. Die verschiedenen Verfahren unterscheiden sich vor allem darin, wie die Wichtigkeit einer Verbindung bemessen wird. In einer einfachen und wenig rechenintensiven Methode wird die Verbindung mit dem betragsmäßig kleinsten Gewicht als die unwichtigste angesehen. Aufwendigere Methoden bemühen dazu eine approximative Berechnung der Hesse-Matrix, also der zweiten Ableitungen des Fehlers nach den Gewichten (Hassibi and Stork, 1993; Y. LeCun, 1993). In Konstruktions-Verfahren werden einem anfangs kleinen Netz im Verlaufe des Lernens "nach Bedarf" zusätzliche Neuronen und Verbindungen hinzugefügt. Das bekannteste Verfahren dieser Art für MLP ist das "Cascade-Correlation" Verfahren (Fahlman and Lebiere, 1990). Es erzeugt typischerweise eine recht spezielle "schmale" Architektur, die aus vielen Schichten mit jeweils wenigen Neuronen besteht.

In Regularisierungsverfahren werden keine Verbindungen entfernt oder hinzugefügt, sondern die Entwicklung des Gewichte-Vektors durch zusätzliche Terme in der Fehlerfunktion, bzw. in der Lernregel, so beeinflußt, daß die vom Netz dargestellte Funktion bestimmten zusätzlichen Anforderungen genügt. In einigen Verfahren wird explizit die Glattheit der

Funktion gefordert. In diesen Verfahren werden Zusatz-Terme in der Fehlerfunktion benutzt, die große Werte der Ableitung der Netz-Funktion nach den Eingabe-Werten bestrafen. Eine einfachere Form der Regularisierung ist der sogenannte "Weight Decay". Hier wird als Straf-Term die Summe der Gewichts-Quadrate verwendet. Beim Gradientenabstieg bedeutet das, daß alle Gewichte in jedem Lernschritt zusätzlich gegen Null gedrückt werden, und zwar proportional zur Größe des Gewichts. Wäre der Weight Decay Straf-Term der einzige Term der Fehlerfunktion, würde deshalb beim Gradienten-Abstieg jedes Gewicht exponentiell abfallen, woher der Name Weight Decay, also "Gewichts-Zerfall" rührt. Auch ein Weight Decay Strafterm führt zu weniger stark gekrümmten Funktionen, da die sigmoide Aktivierungs-Funktion der Neuronen im Bereich kleiner Argumente annähernd linear ist. Der Weight Decay Straf-Term wird durch einen Parameter α gegenüber dem Fehler-Term gewichtet, der also die "Stärke" des Decays bestimmt. Ich werde diesen Parameter im folgenden als Decay Faktor bezeichnen. Die Veränderung eines Gewichtes in einem Lernschritt berechnet sich damit nach der modifizierten Lern-Regel

$$\Delta w_{ij} = -\eta \frac{\partial E_i}{\partial w_{ij}} - \alpha w_{ij}.$$

Die vierte Klasse von Struktur-Optimierungs Verfahren bilden die Topologie-Evolutions-Verfahren. In diesen Verfahren geht es wieder darum, Verbindungen aus dem Netz zu entfernen, bzw. allgemeiner eine optimale Netz-Topologie zu finden, und zwar im Gegensatz zu den Pruning-Verfahren bereits bevor das Training beginnt. Dazu existieren eine Vielzahl von evolutiven Verfahren, in denen eine Reihe verschiedener Netz-Topologien hinsichtlich der Qualität des mit ihnen erzielten Lernerfolges bewertet werden, um dann in einer Folge von Generationen stochastisch selektiert, variiert und wieder bewertet zu werden.

Ich werde im nächsten Kapitel sowohl mit Weight Decay als auch mit evolutiven Verfahren experimentieren. Dabei werde ich untersuchen, wie diese Verfahren modifiziert werden können, um eine spezielle Annahme über die Testmenge zu realisieren, nämlich daß die Test-Menge durch ein "modulares" Netz, d.h. ein Netz, in dem jeweils nur bestimmte Gruppen von Neuronen der verschiedenen Schichten miteinander verbunden sind, am besten angenähert werden kann. Zuvor möchte ich aber noch einige Arbeiten aus der Fachliteratur vorstellen, die sich mit modularen Multilagenperzeptronen befassen. Dabei gehe ich zunächst auf eine besondere Form der Modularisierung ein, die nicht die Neurone n eines Netzes in Gruppen aufteilt, sondern die Datenmenge.

3.3 Modulare MLP I: Aufteilung des Datenraumes in Regionen

Modularität ist, wie im ersten Kapitel beschrieben, ein Begriff, der vor allem dadurch gekennzeichnet ist, daß ein Gesamtsystem in relativ unabhängige Teilsysteme zerfällt. Ich behaupte, daß es zwei wesentlich unterschiedliche Arten gibt, wie ein MLP aus solchen unabhängigen Teilnetzen aufgebaut sein kann. Man kann entweder die Trainingsbeispiele in Gruppen, die jeweils durch ein separates Netz bearbeitet werden, aufgeteilen. Oder es kann jedes Teilnetz zwar jedes Trainingsbeispiel sehen, aber von diesem jeweils nur einige Eingabe- bzw. Ausgabestellen. Im ersten Fall wird der Datenraum in Regionen,

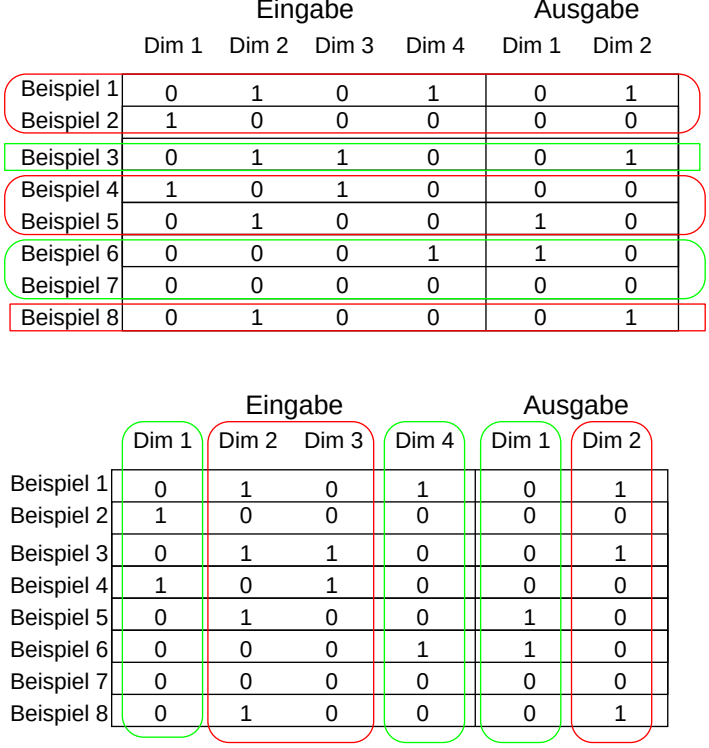

Abbildung 3.3: Zwei Arten der Aufteilung einer Lernaufgabe. Oben: Aufteilung in Regionen. Unten: Aufteilung in Unterräume

im zweiten Fall entlang der Dimensionen in Unterräume aufgeteilt. Wenn man den Trainingsdatensatz als Tabelle vor sich hat, in der jede Zeile einem Trainingsbeispiel und jede Spalte einer Dimension der Eingabe bzw. der Ausgabe entspricht, bedeutet eine Aufteilung in Regionen eine Zusammenfassung der Zeilen in Gruppen, während eine Aufteilung in Unterräume eine Zusammenfassung der Spalten bedeutet (siehe Abbildung 3.3 auf Seite 32.

3.3.1 Mögliche Vorteile und Probleme

Eine Aufteilung des Datenraumes in Regionen ist besonders dann sinnvoll, wenn sich die anzunähernde Funktion in den verschiedenen Regionen stark unterscheidet. Getrennte Netze bieten dann verschiedene Vorteile, vgl. auch (Jacobs et al., 1991a):

- **Schnelleres und besseres Lernen** Modulare Netze benötigen weniger Verbindungen als nicht-modulare. Schon dadurch ist die absolute Trainingszeit kürzer. Aber auch die Anzahl der Durchgänge durch die Trainingsmenge (der "Epochen"), bis ein kleiner Fehler erreicht ist, kann kürzer sein. Der Grund dafür liegt darin, daß das Netz mit weniger widersprüchlicher Information umgehen muß. Wird ein Netz zuerst nur mit Daten aus einer Region trainiert und dann mit solchen aus einer anderen, sehr verschiedenenen Region "vergißt" das Netz die zuerst gelernten Gewichte ein wenig im Versuch, sich den neuen Daten anzupassen. Dieser Effekt wird als zeitliche Interferenz ("temporal crosstalk") bezeichnet und durch ein angemessen regional-modulares Netz behoben.

- **Bessere Generalisierung** Ein Netz generalisiert, wie oben besprochen, dann besonders gut, wenn es gerade so flexibel ist, daß es die anzunähernde Funktion noch darstellen kann, aber nicht flexibler. Ist die Funktion in verschiedenen, deutlich getrennten Regionen sehr einfach, die Gesamtfunktion aber dennoch sehr komplex, wird ein aus sehr einfachen Modulen aufgebautes Netz in der Regel besser generalisieren: der erwartete Fehler der Module wird, wenn die Regionen jeweils hinreichend einfach sind, klein sein und die zusätzliche Varianz, die durch den Aufteilungsmechanismus eingeführt wird, wird u.U. die Varianz, die ein großes Netz hätte, nicht erreichen. Problematisch wird diese Annahme, wenn die Regionen selber schon so komplex sind, daß ein großes Netz nur wenig flexibler zu sein bräuchte als ein Modul. Dann würde u.U. die zusätzliche Varianz, die durch die Aufteilung erzeugt wird, sogar ein schlechteres Generalisierungsverhalten erzeugen. Eine verbesserte Generalisierung aufgrund einer geeignet gewählten Struktur bedeutet gleichzeitig immer auch, daß weniger Trainingsbeispiele benötigt werden. werden

- **Wiederverwendbarkeit** Die Möglichkeit, Komponenten wiederzuverwenden ist, wie im ersten Kapitel geschildert, in vielen Bereichen der Technik eine Motivation zur Schaffung modularer Systeme. Auch für neuronale Netze ist eine Technik wünschenswert, die es ermöglicht, ein einmal für eine bestimmte Teilaufgabe trainiertes Modul in einem anderen Zusammenhang wiederzuverwenden, ohne es neu trainieren zu müssen. Voraussetztung dafür ist ein Mechanismus, der das Modul sinnvoll in den neuen Zusammenhang integriert.

- **Verständlichkeit** Auch die erhöhte Verständlichkeit der Struktur eines modularen Systems ist in anderen technischen Bereichen, wie beschrieben, ein wesentlicher Aspekt. Für manche Anwendungen von neuronalen Netzen stellt es ein Problem dar, daß ihre komplexe und verteilte Struktur, die zahlreichen Verbindungen und Gewichte, keine Einsicht in den Zusammenhang der Variablen gewähren, obwohl oder gerade weil die Netze in der Lage sind, sehr genaue Voraussagen zu machen. Sofort verständliche Interpretationen erhält man fast nur aus linearen Modellen. Die Aufteilung des Datenraums in lineare Regionen gibt also eine Möglichkeit, die Verständlichkeit zu erhöhen, u.U. mit nur geringen Einbußen bei der Vorhersagekraft.

Die geschilderten Vorteile können geringfügig sein, aber u.U. die Verwendung eines neuronalen Netzes überhaupt erst ermöglichen. Voraussetzung zur Realisierung eines jeglichen Vorteils ist immer, daß das zugrundeliegende Problem sich zu einer Aufteilung überhaupt eignet.

3.3.2 Manuelle und halb-automatische Aufteilung

In dem im ersten Abschnitt beschriebenen Laufmodell von Cruse ist eine Aufteilung in Regionen als eine a priori Eigenschaft des Modells eingebaut, nämlich indem ein Beinmodul, das aus einer aktuellen Beinstellung eine neue Beinstellung im nächsten Zeitschritt zu berechnen hat, aufgeteilt ist in ein Schwingnetz, ein Stemmnetz sowie ein Selektornetz, das zwischen den beiden Modi hin- und herschaltet. Das Selektornetz bestimmt, ob die aktuelle Beinstellung vom Schwingnetz oder vom Stemmnetz bearbeitet wird, genauer: es bestimmt, da beide Netze Ausgaben liefern, welche Ausgabe, also welche neue Beinstellung, letztlich wirklich ausgegeben wird. Dies geschieht, indem das Selektornetz Ausgaben von 0 oder 1 liefert, mit denen dann die Ausgaben der Spezialisten multipliziert

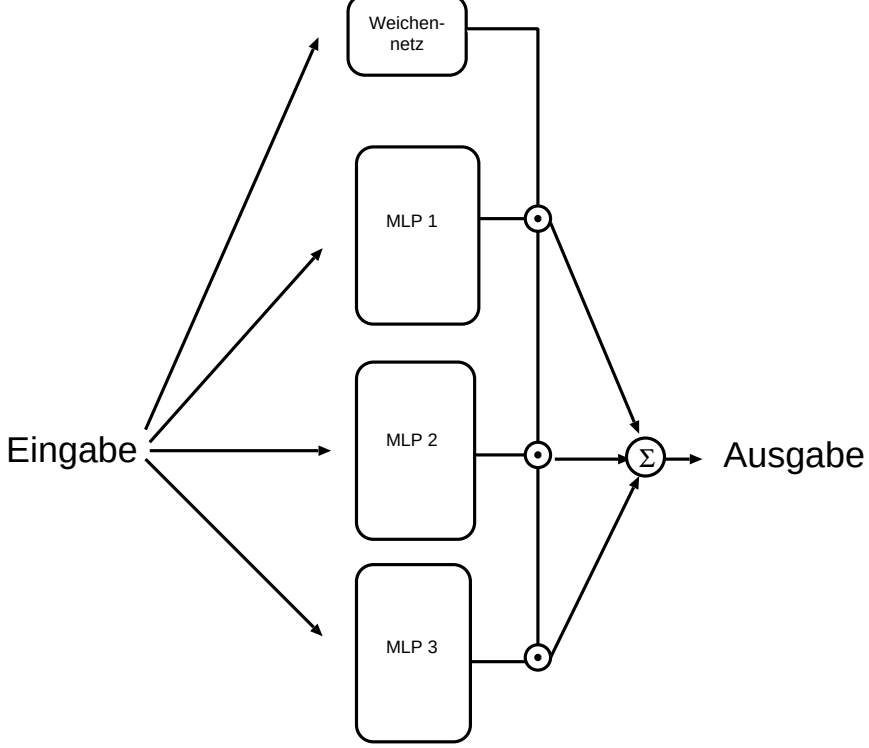

Abbildung 3.4: Eine Experten-Netz Architektur

werden. Solche Selektor- oder Weichennetze ("gating networks"), die durch multiplikative Verbindungen mit den Ausgaben der verschiedenen Spezial- oder Expertennetzen ("expert networks") die Ausgabe des Gesamtsystems bestimmen, sind typische Bestandteile von regional-modularen Netzen, die den Eingaberaum aufteilen (siehe Abbildung 3.4. In den herkömmlichen, manuell aufgeteilten Modellen, werden diese Komponenten getrennt voneinander trainiert. Die Expertennetze erhalten Trainingsbeispiele aus "ihrem" Teil des Eingaberaumes und das Weichennetz wird darauf trainiert, den Datenraum entsprechend den Vorgaben des Experimentators aufzuteilen.

Bennani teilt den Eingaberaum nicht manuell auf, sondern benutzt dazu ein "k-means" Clusterverfahren (Bennani, 1995). Seine Lernaufgabe ist die Wiedererkennung von Sprechern aufgrund von Sprachproben. In einem ersten Schritt clustert er die Sprachproben aufgrund ihrer Ähnlichkeit in sechzehn Gruppen. Für jede dieser Gruppen wird dann ein separates Netz trainiert. Um neue Muster bearbeiten zu können, wird das Weichennetz darauf trainiert, den Sprachproben ihre Clusternummer zuzuordnen. Dieses Verfahren ist insofern halb-automatisch, als es die Aufteilung zwar automatisch, aber immer noch getrennt vom Training der Experten durchführt.

Waibel hat sich schon sehr früh mit der Frage beschäftigt, wie aus MLP, die auf bestimmte Eingabedaten spezialisiert sind, größere Netze aufgebaut werden können, ohne daß die Experten neu trainiert werden müssen (Waibel et al., 1989). Er beschäftigt sich mit der Erkennung von gesprochenen Konsonanten. Die Aufteilung der Trainingsdaten erfolgt hier rein manuell in verwandte Konsonantengruppen. Dann werden in einem ersten Schritt die einzelnen Experten trainiert. Waibel verwendet dann aber kein Weichennetz, daß abhängig von der Eingabe, durch Multiplikation mit 0 oder 1 einen Experten auswählt, sondern integriert die Ausgaben der Experten auf komplexere Weise in eine Gesamtaus-

gabe, indem er sie als Eingaben für ein nachgeschaltetes MLP verwendet. Zusätzlich erhält dieses nachgeschaltete MLP als Eingabe die Ausgaben eines weiteren MLP, das auf der gleichen Ebene steht wie die Experten, also die ursprünglichen Sprachdaten als Eingabe erhält. Dieses zusätzliche Modul, das als "Klebe" bezeichnet wird ("connectionist glue"), und das nachgeschaltete, integrierende MLP werden dann nachtrainiert, während die Gewichte der Experten unverändert ("eingefroren") bleiben. Diese Konstruktion entspricht also den vorigen, mit dem Unterschied, daß ein komplexeres Weichennetz benutzt wird, das nicht nur die Sprachdaten (indirekt durch das Klebemodul), sondern auch die Ausgaben der bereits trainierten Experten als Eingabe erhält und aus diesen dann die endgültige Ausgabe baut, statt wie eine schlichte Weiche, die Ausgaben nur auszuwählen. Trotz dieses komplexen nachgeschalteten Netzes sind durch die Modulbauweise immer noch weniger Verbindungen einzustellen, als in einem großen Gesamtnetz, das alle Konsonanten zu klassifizieren lernt. Ein Vorteil von Waibels Methode ist, daß sie durch das Nachtraining die Wiederverwendung von einmal trainierten Modulen in anderem Zusammenhang erlaubt.

3.3.3 Automatische Aufteilung: Expertenmischungs-Systeme

Jacobs und Jordan stellen ein Verfahren vor, wie die Gewichte der Expertennetze und des Weichennetzes, das die Aufteilung der Trainigsbeispiele bestimmt, gleichzeitig gelernt werden können (Jacobs et al., 1991a; Jordan and Jacobs, 1994). Die Zuteilung von Trainingsbeispielen zu Experten ist also nicht a priori gegeben, sondern wird durch das Lernverfahren aus den Trainingsbeispielen selbst berechnet. Man kann dann der Lernaufgabe mit einem Arsenal von potentiellen Experten unterschiedlicher Größe und Flexibilität begegnen und das Verfahren wird die Daten vorteilhaft auf die Netze verteilen.
Zwei verschiedene Arten von Trainingsverfahren sind für diese Expertenmischungs-Systeme ("mixtures of experts") bekannt. Das etwas ältere Verfahren beruht auf der Idee des Wettbewerbs der Experten während des Lernens (Jacobs et al., 1991a). Das Weichennetz wird so initialisiert, daß zunächst alle Experten auf alle Eingaben gleichberechtigt antworten, d.h. das Weichennetz wichtet alle Ausgaben mit dem Faktor 0.5. Nach der ersten Eingabe werden die Gewichte der Weiche so verändert, daß der Experte mit der besten Ausgabe auf dieser Eingabe bevorzugt wird, d.h. sein Wichtungsfaktor wird bei einer erneuten Eingabe des gleichen oder eines ähnlichen Vektors auf Kosten der anderen Experten erhöht sein. Nun wichten die Faktoren nicht nur die Weiterleitung der Ausgabe in Vorwärtsrichtung, sondern auch die Weiterleitung des Fehlersignals in Rückwärtsrichtung, das zur Berechnung der Gewichtsänderung in den Experten verwendet wird. Die Überlegenheit des für eine bestimmte Eingabe bereits bevorteilten Experten wird dadurch noch verstärkt: bei einer erneuten Eingabe des gleichen oder eines ähnlichen Beispiels wird das Netz, das bereits auf diesem Netz gelernt hat, wieder eine bessere Ausgabe liefern. Andererseits wird ein sehr unähnliches Beispiel möglicherweise besser von einem anderen Netz bearbeitet. Auf diese Weise bilden sich Experten für die Bearbeitung von ähnlichen Beispielen heraus, der Eingaberaum wird so in Regionen geteilt. Die Grenzen der Regionen sind nicht allein durch die Ähnlichkeitsstruktur im Eingaberaum, die auch durch ein reines Clusterverfahren erfaßt werden könnte, bestimmt, sondern werden durch das Lernverhalten der Experten festgelegt. Insbesondere kommt es darauf an, wie schnell die Experten lernen. Oft könnten durch ein ausreichend großes Netz alle Trainingsdaten sehr gut angenähert werden. Die Chance eines kleinen Netzes besteht dann darin, daß es schneller lernen kann als das große, vorausgesetzt es findet eine Region, die seiner Struktur angemessen ist. Eine anfangs schnell mit Beschlag belegte Region kann dann auch

von einem mächtigeren Netz nicht mehr zurückerobert werden, da das Weichennetz dem großen Netz kein Lernen in dieser Region mehr erlaubt.

Der zweite Ansatz formuliert das Lernproblem als ein statistisches Parameterschätzproblem (Jordan and Jacobs, 1994). In einer solchen Aufgabe wird eine Verteilung, aus der eine Stichprobe gezogen wird, durch ein parametrisiertes Modell beschrieben. Die Gauß-Funktion mit ihren beiden Parametern Mittelwert und Varianz ist z.B. ein solches parametrisiertes Modell. Im sogenannten "Maximum-Likelihood"-Verfahren werden dann diejenigen Parameter des Modells gesucht, die das Ziehen gerade der vorliegenden Stichprobe am wahrscheinlichsten machen. Die Wahrscheinlichkeit der Stichprobe wird "Likelihood" genannt und ist eine Funktion der Stichprobe und der Parameter. Im Falle, daß die Elemente der Stichprobe unabhängig gezogen werden, errechnet sich die Wahrscheinlichkeit der Stichprobe aus dem Produkt der Wahrscheinlichkeiten ihrer Elemente. Ich werde jetzt als Beispiel für das Vorgehen die Schätzung der Parameter einer eindimensionalen Normalverteilung besprechen, um dann zu zeigen, wie man ein Modell aufstellen kann, daß einem erlaubt, auf ähnliche Weise auch die Gewichte eines MLP zu optimieren. Dabei stellt sich heraus, daß man unter bestimmten Annahmen als Fehlerfunktion die übliche quadratische Abweichung erhält. Nach diesen Vorbereitungen gebe ich dann das kompliziertere Likelihood-Modell für Expertenmischungssysteme an.

Sei also eine eindimensionale reelle Zufallsvariable x normalverteilt, also gelte $p(x) = \mathcal{N}(\mu^*, \sigma^*)$ und es sei eine Stichprobe $\mathcal{X} = \{x_1, \ldots, x_n\}$ gezogen worden. Dann berechnet sich die Likelihood der Stichprobe $\mathcal{L}$ aus dem Produkt der Wahrscheinlichkeiten $p(x_1), \ldots, p(x_n)$, die von den zu optimierenden Parametern μ und σ abhängen:

$$\mathcal{L}(\mathcal{X}, \mu, \sigma) = \prod_{i=1}^{n} p(x_i) = \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} e^{-(x_i-\mu)^2/2\sigma^2}.$$

Zur Berechnung des Optimums wird der negative natürliche Logarithmus der Likelihood gebildet, so daß in unserem Beispiel folgende Funktion zu minimieren ist:

$$E = -\ln \mathcal{L} = \frac{1}{2\sigma^2} \sum_{i=1}^{n} (x_i - \mu)^2 + n \ln \sigma + \frac{n}{2} \ln(2\pi).$$

Das Minimum dieses Fehlers liegt bei $\mu = 1/n \sum_{i=1}^{n} x_i$ und $\sigma^2 = 1/n \sum_{i=1}^{n} (x_i - \mu)^2$. Auf diese Weise erhält man also Schätzwerte μ und σ der wahren Parameter μ^* und σ^*.

Auch die Trainingsmenge $\mathcal{D} = \{(x_1, t_1), \ldots, (x_n, t_n)\}$ eines MLP läßt sich nun als Stichprobe einer Verteilung ansehen und sie hat somit auch eine Likelihood. Da die Trainingsmenge aus Paaren von Eingabe- und Zielvektoren besteht, berechnet sich die Likelihood hier als ein Produkt der entsprechenden Paarwahrscheinlichkeiten:

$$\mathcal{L} = \prod_{i=1}^{n} p(x_i, t_i).$$

Die Paarwahrscheinlichkeiten lassen sich schreiben als das Produkt der bedingten Wahrscheinlichkeit des Zielvektors, wenn der zugehörige Eingabevektor bereits vorliegt, mit der Wahrscheinlichkeit dieses Eingabevektors:

$$\mathcal{L} = \prod_{i=1}^{n} p(t_i|x_i) p(x_i).$$

Eine häufige Annahme ist jetzt, daß die Eingabevektoren gleichverteilt sind, die Zielvektoren bei Vorliegen der Eingabe dagegen normalverteilt, und zwar mit vom Eingabevektor unabhängiger Varianz und vom Eingabevektor abhängigen Mittelwert. Unter der weiteren Annahme, daß die Stellen $t_i^k, i = 1, \ldots, n, k = 1, \ldots, c$ der Zielvektoren unabhängig sind, erhält man damit die Fehlerfunktion:

$$E = -\sum_{i=1}^{n} \ln p(t_i|x_i) = -\sum_{i=1}^{n}\sum_{k=1}^{c} \ln p(t_i^k|x_i) = \frac{1}{2\sigma^2}\sum_{i=1}^{n}\sum_{k=1}^{c}(t_i^k - \mu_i^k)^2 + nc\ln\sigma + \frac{nc}{2}\ln(2\pi).$$

Modelliert man jetzt die von der Eingabe abhängigen Mittelwerte $\mu_i^k(x)$ durch die MLP-Ausgaben $y_i^k(x, w), i = 1 \ldots n, k = 1 \ldots c$ erhält man einen von den Netzgewichten abhängigen Ausdruck für die Likelihood, der, von den nicht von den Gewichten abhängigen Termen entkleidet, genau das übliche Fehlermaß der mittleren quadratischen Abweichung darstellt:

$$E = \sum_{i=1}^{n}\sum_{k=1}^{c}(t_i^k - y_i^k(x, w))^2.$$

In einem Expertenmischungs-System ist die bedingte Wahrscheinlichkeit einer Ausgabe jetzt nicht mehr als eine einfache Normalverteilung angenommen, sondern als eine gewichtete Summe von M Normalverteilungen, so daß man als Fehlermaß erhält:

$$E = -\sum_{i=1}^{n} \ln \sum_{j=1}^{M} \alpha_j(x_i)\phi_j(t_i|x_i),$$

wobei die Normalverteilungen $\phi_j(t|x)$ gegeben sind durch:

$$\phi_j(t|x) = \frac{1}{(2\pi)^{c/2}} exp(-\frac{||t - \mu_j(x)||^2}{2}).$$

Die α_j sind dabei die durch die Softmax-Funktion normierten Ausgaben γ_j des Weichennetzes. Es gilt:

$$\alpha_j = \frac{\exp(\gamma_j)}{\sum_{l=1}^{M}\exp(\gamma_l)}.$$

Dieses kompliziertere Fehlermaß kann jetzt auf zweierlei Weise minimiert werden. Zum einen läßt sich auch hier noch, wenn auch aufwendiger, ein Gradient des Fehlers in Bezug auf die Netzgewichte und Wichtungsfaktoren berechnen, an dem entlang man die Funktion hinabsteigen kann. Zum anderen kann man in das Modell zusätzliche Scheinvariablen einführen, die die Maximierung der Likelihood erleichtern. Die Scheinvariablen sind unbekannte Zufallsvariablen, was dazu führt, daß jetzt statt der Likelihood der Erwartungswert der Likelihood maximiert werden muß. Dieser Wert ist ebenfalls von den in Frage stehenden Parametern abhängig, so daß sich Schritte, in denen der Erwartungswert berechnet wird, mit Maximierungsschritten abwechseln müssen. Solche Verfahren heißen daher "Expectation-Maximization"-Algorithmen oder EM-Algorithmen. Dempster et al. haben gezeigt, daß solche Verfahren auch die "eigentliche" Likelihood (ohne die Scheinvariablen) maximieren.

Als Testanwendung eines Expertenmischungs-Systems haben Jordan, Jacobs, Nowlan und Hinton die Erkennung von vier unterschiedlichen gesprochenen Vokalen ([i],[I],[a],[A]) anhand der ersten beiden Formanten gewählt (Jacobs et al., 1991b). Die Beispiele für die

Vokale [i] und [I], bzw. [a] und [A] liegen in dem Datensatz jeweils eng benachbart. Die angebotenen vier, bzw. acht Experten spezialisieren sich auf eine der beiden Regionen. Die Autoren geben an, daß das System von sich aus versucht, mit wenigen der angebotenen Experten auszukommen. Das Mischungssystem lernt schneller als ein nicht-modulares MLP. Beide erreichen in dieser einfachen Aufgabe den gleichen Fehler. Jordan und Jacobs testen ein etwas verändertes System, in dem die Ausgaben der Experten noch durch ein ganzes System von mehreren baumartig hintereinanderliegenden Weichen gemischt werden (Jordan and Jacobs, 1994). Sie verwenden ein Roboterarm-Kontrollproblem, in dem es gilt, die Abbildung von Gelenk-Drehmomenten auf Gelenk-Beschleunigungen für einen viergelenkigen Roboterarm in drei Raumdimensionen zu lernen.

3.4 Modulare MLP II: Aufteilung des Datenraumes in Unterräume

Im Fall der Modularisierung des Datenraumes in Regionen wählt das Weichennetz einen Experten aus, der für ein Trainingsbeispiel zuständig ist. Im Fall der Modularisierung in Unterräume dagegen, ist jedes Trainingsbeispiel für die Ausgabe aller Module relevant - aber je nach Modul nur einige seiner Dimensionen (siehe Abbildung 3.5). Als Beispiel kann man sich zunächst zwei Netze vorstellen, die jeweils ganz unterschiedliche Aufgaben zu lernen haben. Man kann dann die beiden Aufgaben und ihre Netze "aneinanderkleben", indem jedem Eingabevektor der Trainingsmenge des ersten Problems alle Eingabevektoren aus der Trainingsmenge des anderen Problems angehängt werden. Entsprechend verfährt man mit den zugehörigen Zielvektoren. Auf diese Weise erhält man eine neue Trainingsmenge. Die Dimension der neuen Vektoren ist dann die Summe der Dimensionen der alten Vektoren und die Anzahl der Trainingsbeispiele ist das Produkt der Kardinalitäten der alten Trainingsmengen. Die neue Aufgabe ist jetzt mit einem Netz zu lösen, das entsprechend mehr Eingabe- und Ausgabeneuronen und damit auch wesentlich mehr Gewichte hat. Wie wird sich das neue Netz verhalten? Wird es die neue Aufgabe eher schlechter oder eher besser lösen als die beiden einzelnen Netze? Wird das Lernverfahren die Gewichte des Netzes so einstellen, daß es quasi wieder in zwei Teile zerfällt? Wie werden dabei die inneren Neurone auf die beiden Aufgaben verteilt werden? Oder kann man die modulare Struktur des Problems auf andere Weise in die Struktur des Netzes übertragen? Welchen Einfluß hat die Entdeckung der modularen Struktur auf Lernen und Generalisierung?
Diesen Fragen werde ich anhand von Experimenten mit einigen Testproblemen der geschilderten und anderer Art ausführlich im nächsten Kapitel nachgehen. Zunächst möchte ich wieder auf denkbare Vorteile und Probleme einer Aufteilung des Datenraumes in Unterräume eingehen und dann zum Abschluß dieses Abschnitts einige Arbeiten vorstellen, die sich bereits mit ähnlichen Fragen beschäftigt haben.

3.4.1 Mögliche Vorteile und Probleme

Eine Aufteilung des Datenraumes in Unterräume ist dann sinnvoll, wenn die anzunähernde Funktion im wesentlichen aus mehreren Funktionen zusammengesetzt ist, d.h einige der Ausgabestellen nur von einigen der Eingabestellen abhängen, bzw., wie im unten beschriebenen Was-und-Wo Problem, nur von einigen Aktivierungen der inneren Neuronen. Getrennte Netze bieten ähnlich wie bei einer Aufteilung in Regionen einige Vorteile:

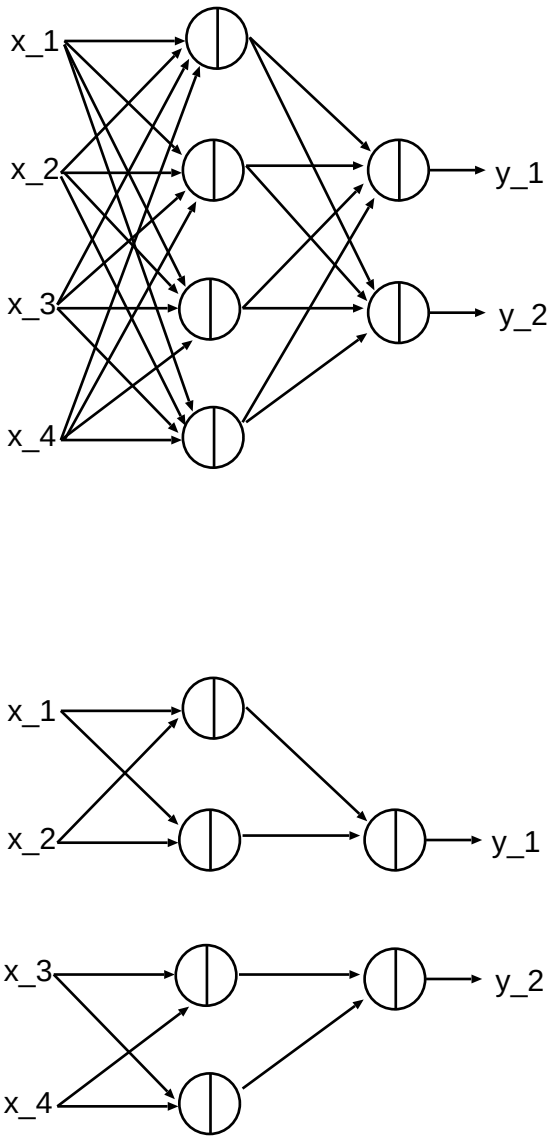

Abbildung 3.5: Oben: Ein nicht-modulares Netz. Unten: Ein Netz, das aus zwei Modulen besteht.

- **Schnelleres und besseres Lernen**

  Auch ein Unterraum-modulares Netze hat weniger Verbindungen und schon dadurch kürzere Trainingszeiten. Und auch hier muß das Netz mit weniger widersprüchlicher Information umgehen. Einer eventuellen zeitlichen Interferenz ("temporal crosstalk") ist es zwar voll ausgesetzt, da alle Teilnetze auf allen Regionen trainiert werden, aber eine andere Art von Interferenz kann durch eine Aufteilung entlang der Dimensionen behoben werden, nämlich räumliche Interferenz ("spatial crosstalk"). Dabei handelt es sich um den Effekt, daß sich die Gewichtsänderungen, die beim Lernen durch zwei Trainingsbeispiele während derselben Epoche hervorgerufen werden, annähernd auslöschen können. Wenn nämlich ein inneres Neuron mit mehreren Ausgabeneuronen verbunden ist, verlangt in einer unglücklichen Situation das Fehlersignal des einen Ausgabeneurons evtl. eine Änderung eines der Gewichte des inneren Neurons in eine Richtung, das Fehlersignal des anderen Ausgabeneurons aber gerade das Gegenteil. Eine solche Situation wird durch eine angemessene Aufteilung der Ausgabedimensionen entschärft. In dem Extremfall z.B., daß ein Netz nur ein Ausgabeneuron hat, kann es keine räumliche Interferenz geben.

- **Bessere Generalisierung**

  Auch die Generalisierung sollte durch eine Aufteilung entlang der Dimensionen verbessert werden können. Ein modulares Teilnetz ignoriert die für die Annäherung der Teilfunktion irrelevanten Eingabestellen und wird nicht durch Artefakte, die durch verrauschte Daten oder Fluktuationen beim Ziehen der Stichprobe entstehen, zur Modellierung von Scheinzusammenhängen veranlaßt. Auch hier bedeutet eine verbesserte Generalisierung gleichzeitig, daß weniger Trainingsbeispiele benötigt werden.

3.4.2 Das Was-und-Wo-Problem

Das Was-und-Wo Problem ist von verschiedenen Autoren als ein Testproblem zur Untersuchung von modularen Netzen gewählt worden (Rueckl et al., 1989; Jacobs et al., 1991a; Jacobs and Jordan, 1992; Pennington et al., 1995). In dieser Lernaufgabe geht es um eine Aufteilung der Ausgabeneuronen, aber anders als im von mir oben als Beispiel geschilderten aus zwei Trainingsmengen konstruierten Problem geht es nicht gleichzeitig um eine Aufteilung der Eingabeneuronen, sondern um eine Aufteilung der inneren Neuronen. Beim Was-und-Wo Problem sollen Figuren auf einem Raster erkannt werden. Dabei soll sowohl erkannt werden, wo sich die Figur auf dem Raster befindet, als auch um was für eine Figur es sich handelt.
Zur Beantwortung beider Fragen werden jeweils alle Eingabestellen benötigt, die Antworten werden aber jeweils an zwei unterschiedlichen Gruppen von Ausgabeneuronen abgelesen. Zwei getrennte Netze unterscheiden sich deshalb in ihrer Struktur von einem einzigen großen Netz nur dadurch, daß die inneren Neurone aus zwei Gruppen bestehen, die jeweils nur mit einer der beiden Ausgabeneuronen-Gruppen verbunden sind.

3.4.3 Manuelle Aufteilung

Rueckl hat als erster das Was-und-Wo Problem formuliert (Rueckl et al., 1989). Er zeigt, daß ein modulares Netz die Aufgabe schneller und besser lernt als ein Gesamtnetz. Voraussetzung dafür ist, daß die beiden Gruppen von Ausgabeneuronen jeweils mit sovielen inneren Neuronen verbunden sind, wie es der Schwierigkeit ihrer Teilaufgabe entspricht.

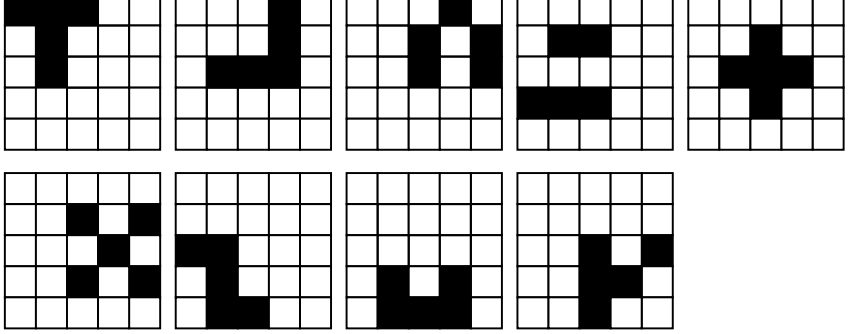

Abbildung 3.6: Die Figuren des Was-und-Wo Problems. Jede der 9 Figuren tritt an jeder der 9 Positionen auf dem Raster auf.

Rueckl experimentiert mit verschiedenen Aufteilungen von insgesamt 18 inneren Neuronen und findet die besten Ergebnisse bei 4 inneren Neuronen für die einfache Wo-Aufgabe und 14 Neuronen für die Was-Aufgabe. Rueckl erklärt die Vorteile des modularen Netzes durch die Möglichkeit, in den alleine genutzten inneren Neuronen der Teilaufgabe angepaßte Umkodierungen der Eingabe lernen zu können. Er bietet mit seinem Modell eine Erklärung für die Existenz unterschiedlicher Verarbeitungsströme (P- und M-Ströme) im visuellen Cortex des Primatengehirns an, siehe (VanEssen et al., 1990).

3.4.4 Halb-automatische Aufteilung

Jacobs und Jordan haben ein Verfahren entwickelt, wie sich die Aufteilung der Neuronen im Verlaufe eines Lernverfahrens automatisch herausbildet (Jacobs and Jordan, 1992). Sie setzen dazu aber die Kenntnis der korrekten Aufteilung der Ausgabeneuronen noch voraus. In diesem Sinne ist das Verfahren halb-automatisch. Sie bedienen sich des "Tricks", den Neuronen Positionen im Raum zuzuordnen. Die beiden Gruppen von Ausgabeneuronen werden räumlich weit voneinander entfernt positioniert. Während des Lernens werden dann die Gewichte aller Verbindungen gegen Null gedrückt (Weight Decay), und zwar je stärker je weiter die verbundenen Neurone voneinander entfernt liegen. Auf diese Weise werden Verbindungen zwischen inneren Neuronen, die sich entfernt von einer der beiden Ausgabeneuronen-Gruppen befinden, nur geringes Gewicht besitzen und nahegelegene Verbindungen bevorzugt. Bei einer angemessenen Wahl der Parameter können auch innere Neurone, die eigentlich näher an den Ausgabeneuronen der Wo-Gruppe liegen, noch starke Verbindungen zu den Was-Neuronen ausbilden und so kann insgesamt eine der Lernschwierigkeit entsprechende Aufteilung entstehen.
Jacobs und Jordan versuchen die Idee des Weichennetzes, wie es zur Aufteilung des Datenraumes in Regionen verwendet wird, auch für die Aufteilung des Was-und-Wo-Problems zu verwenden (Jacobs and Jordan, 1992). Sie stellen für jede der beiden Teilaufgaben jeweils ein großes und ein kleines Netz zur Verfügung, insgesamt also vier Netze. Diese Netze werden mit entsprechend den Teilaufgaben "durchgeschnittenen" Zielvektoren trainiert. Für jede Teilaufgabe gibt es ein separates Weichennetz, das entscheidet, ob die Teilaufgabe durch das große oder das kleine Netz bearbeitet wird. Die Weichennetze werden mit dem oben beschriebenen Wettbewerbs-Mechanismus trainiert. In beiden Teilaufgaben lernt das kleine Netz schneller und gewinnt den Wettbewerb. Erst wenn die Spezialisierung der Weichennetze ausreichend verzögert wird, gewinnt schließlich für die komplexere Was-Aufgabe das große Netz. Auch dieses Verfahren setzt die Kennt-

nis der korrekten Aufteilung der Ausgabeneuronen voraus. Außerdem ist die Aufteilung der inneren Neurone nicht frei, sondern wird durch die Größe der angebotenen Experten bestimmt.

3.4.5 Aufteilung durch einen evolutionären Algorithmus

Pennington et al lassen die Verbindungsstruktur und die Gewichte eines MLP, das das Was-und-Wo Problem zu lösen hat, in einem evolutionären oder genetischen Algorithmus entstehen (Pennington et al., 1995). Zur Bewertung der Fitneß verwenden sie die mittlere quadratische Abweichung der Netzausgaben von den Zielvektoren. Die Netzstruktur ist in den Individuen als eine Reihe von Koeffizienten kodiert, die das jeweilige Gewicht von sogenannten "Basis-Netzen" darstellen, deren Überlagerung die Netzstruktur bildet. Die Basis-Netze sind spärlich verbundene Zufallsnetze. Zur Bewertung der Modularität des evoluierten Netzes betrachten sie für jedes innere Neuron das Verhältnis der Gewichtssummen der Verbindungen zu den beiden Gruppen von Ausgabeneuronen. Sie stellen fest, daß sich die Gewichtssummen nur für insgesamt 5 von 18 Neuronen um einen Faktor größer als 5 unterscheiden. Die Verteilung der Verhältnisse ist allerdings deutlich in Richtung eines stärkeren Was-Anteils verschoben, d.h. die inneren Neuronen sind im Mittel stärker mit den Was- als mit den Wo-Neuronen verbunden. Das Verfahren von Pennington et al ist zwar vollständig automatisch, allerdings führt es nicht zu einer Aufteilung der inneren Neuronen im eigentlichen Sinne, da ihre große Mehrzahl stark mit beiden Gruppen von Ausgabeneuronen verknüpft bleibt.

Teil II

Kapitel 4

Experimente zur Modularisierung von Multilagen-Perzeptronen

In diesem Kapitel beschreibe ich Experimente zur Modularisierung von Multilagen-Perzeptronen in Teil-Netze "entlang der Dimensionen", also in Teil-Netze, die sich auf Unterräume des Datenraumes spezialisieren. Dazu verwende ich Test-Probleme unterschiedlicher Art, die aufgrund ihrer Konstruktion erwarten lassen, daß modulare Netze zu ihrer Lösung geeignet sein könnten. Zum einen untersuche ich das "Modulare Encoder Problem", eine aus mehreren Encoder-Aufgaben durch Konkatenieren der Daten "entlang der Dimensionen" zusammengesetzte Aufgabe, die in Abschnitt 4.2 genau beschrieben wird. Zum anderen verwende ich Datensätze, die mit Hilfe von modular strukturierten "Vorbildnetzen" erzeugt werden. Die Erzeugung dieser Datensätze wird in Abschnitt 4.3 beschrieben. In beiden Aufgaben interessiert mich zunächst, ob eine bereits vor Beginn des Trainings aufgrund der Kenntnis der Aufgaben-Konstruktion festgelegte modulare Netzstruktur beim Netz-Training wirklich vorteilhaft ist. Dazu vergleiche ich den Lern-Verlauf bei verschiedenen Arten von modularer Vorstrukturierung mit dem Lern-Verlauf bei einem nicht vorstrukturierten Netzes.
Im Fall des Modularen Enocers ergibt sich aus den Ergebnissen dieser Versuche eine weitergehende Frage nach der Entstehung modularer Netzstrukturen in nicht vorstrukturierten Netzen im Verlaufe des Lernens. Beim Modularen Encoder zeigt sich nämlich, daß eine solche Modularisierung für den Lernerfolg entscheidend ist. Da sie aber bei einfachem Backpropagation-Training nicht immer "von selbst" erfolgt, entwickle ich einige Trainings-Varianten, die die Modularisierung und damit den Lern-Erfolg beim Modularen Encoder verbessern.
Die beste Möglichkeit, das Lernen modularer Datensätze zu begünstigen, bleibt u.U. aber die Vorgabe der richtigen modularen Struktur. Da diese in der Regel nicht bekannt ist, entwickele ich ein evolutionäres Verfahren, das die optimale modulare Vorstrukturierung allein aus den Daten herleitet.
Der Rest dieses Kapitels ist folgendermaßen aufgebaut: Im nächsten Abschnitt stelle ich zunächst Verfahren zur Berechnung der Modularität eines MLP vor. Dann schildere ich die Experimente mit dem Modularen Encoder und schließlich die Experimente mit den Vorbildnetzen.

4.1 Maße für Modularität

Als Definition des Kernkonzepts von Modularität habe ich in Kapitel 2 vorgeschlagen: *Die Struktur eines Systems ist modular, genau dann, wenn es aus Komponenten besteht, die in sich stärker zusammenhängen als untereinander. Diese Komponenten werden Module genannt.*
Als Maß für eine solche Form von Modularität schlage ich eine Funktion vor, die für eine gegebene Aufteilung des Systems in Komponenten den Intra- und den Inter-Zusammenhang der Komponenten mißt und mittels eines Parameters γ gegeneinander wichtet. Ist die Aufteilung des Systems nicht bekannt, können die Komponenten mit Hilfe dieses Maßes bestimmt werden, indem diejenige Aufteilung gesucht wird, die das Maß optimiert.

4.1.1 γ-Modularität einer Neuronen-Partition

Ein neuronales Netz läßt sich in Komponenten aufteilen, indem die Neuronen in Gruppen zusammengefaßt werden, also duch eine Partitionierung der Neuronenmenge. Für eine gegebene Partition schlage ich vor, den Intra- und den Inter-Zusammenhang der Neuronen aufgrund der Beträge der Gewichte ihrer Verbindungen zu berechnen: Zwei Neuronen die durch ein betragsmäßig hohes Gewicht verbunden sind, nehme ich als stark zusammenhängend an.
Sei $\mathcal{N}$ die Menge der Neuronen und sei eine Partition von $\mathcal{N}$ in n Teilmengen (Gruppen) gegeben: $\mathcal{N} = \mathcal{G}_1 \uplus \mathcal{G}_2 \uplus \ldots \uplus \mathcal{G}_n$. Dann berechne ich zunächst für jede Neuronengruppe $\{\mathcal{G}_k\}, k = 1, \ldots, n$ den mittleren Gewichtsbetrag der Intra-Verbindungen, und bestimme den gesamten Intra-Zusammenhang als den Mittelwert dieser Größen über alle Gruppen. Sei also $\{w_i^k\}, i \in \mathcal{I}_k$ die Menge der Intra-Gewichte der Gruppe k. Dann definiere ich den Intra-Zusammenhang *intra* als

$$intra = \frac{1}{n} \sum_{k=1}^{n} intra_k$$

mit

$$intra_k = \frac{1}{|\mathcal{I}_k|} \sum_{i \in \mathcal{I}_k} |w_i^k|.$$

Für eine analoge Bestimmung des Inter-Zusammenhangs betrachte ich alle Paare von Neuronen-Gruppen. Für jedes Paar wird der mittlere Gewichtsbetrag der zwischen den beiden Neuronengruppen verlaufenden Verbindungen berechnet. Diese Größen werden zur Bestimmung des gesamten Inter-Zusammenhangs über alle Paare gemittelt. Seien also $\mathcal{P}_k, k = 1, \ldots, \binom{n}{2}$ alle Paare von Neuronen-Gruppen und $\{w_i^k\}, i \in \mathcal{J}_k$ die Menge der dem Paar $\mathcal{P}_k$ entsprechenden Inter-Gewichte. Dann definiere ich den Inter-Zusammenhang *inter* als

$$inter = \frac{1}{\binom{n}{2}} \sum_{k=1}^{\binom{n}{2}} inter_k$$

mit

$$inter_k = \frac{1}{|\mathcal{J}_k|} \sum_{i \in \mathcal{J}_k} |w_i^k|.$$

Für einen gegebenen Parameter γ berechne ich schließlich die γ-Modularität mod_γ eines Netzes als die Differenz von Intra-Zusammenhang und dem mit γ gewichteten Inter-Zusammenhang:

$$mod_\gamma = intra - \gamma \cdot inter.$$

Um einen skalierungs-invarianten Wert der γ-Modularität zu erhalten, werden die Gewichte vor der Berechnung durch die Gesamt-Summe der Gewichts-Beträge dividiert.

4.1.2 Auffinden der optimalen Neuronen-Partition

Die so definierte γ-Modularität ist eine Funktion zweier Argumente, nämlich der Gewichte des Netzes und der Partition der Neuronen. Es läßt sich nun auch eine γ-Modularität allein der Gewichte definieren, nämlich als der maximale Wert der γ-Modularität über alle Neuronen-Partitionen. Die Partition, die die γ-Modularität maximiert, ist in diesem Sinne die "richtige" Partition, d.h. die Maximierung der γ-Modularität kann als Verfahren dienen, die Komponenten des Netzes zu identifizieren.
Ich habe zum Auffinden der optimalen Neuronen-Partition zwei heuristische Verfahren entwickelt. Das erste ist ein evolutionärer Algorithmus, das zweite ein divisives hierarchisches Cluster-Verfahren, das anhand der γ-Modularität auf einer bestimmten Zerfallsstufe gestoppt wird.

Evolutionärer Algorithmus

Das erste Verfahren zur Neuronen-Clusterung ist ein Algorithmus, der die Zusammensetzung eines Tupels von Partitionen (einer "Population") schrittweise mit Hilfe von Zufalls-Operatoren ("Mutation" und "Rekombination") und einem Auswahlverfahren in Richtung größerer γ-Modularität der Partitionen in der Population verändert ("evoluiert").

Die Ausgangspopulation wird zunächst initialisiert, indem allen Neuronen gleichförmig zufällig eine Gruppen-Zugehörigkeit in eine von 2 Gruppen zugelost wird. Die so erhaltenen Partitionen in der Ausgangspopulation unterliegen dann drei Arten von Mutationen (sihe Abbildung 4.3 auf Seite 48). Die erste faßt zwei zufällig gewählte Gruppen zu einer zusammen (Merge-Operator), die zweite zerteilt eine zufällig ausgewählte Gruppe wiederum zufällig in zwei neue Gruppen (Split-Operator) und die dritte läßt ein zufällig ausgewähltes Neuron zu einer zufällig gewählten anderen Gruppe wechseln (Switch-Operator). Eine Partition wird in diesem Verfahren durch ein Tupel natürlicher Zahlen repräsentiert, das so viele Stellen hat, wie das Netz Neurone. Die ersten Stellen des Tupels entsprechen dabei den Eingabe-, die letzten den Ausgabeneuronen. Die Einträge des Tupels geben die Nummer der Gruppe an, der das dieser Stelle entsprechende Neuron angehört (siehe Abbildungen 4.1 und 4.2 auf den Seiten 48 und 48). Auf dieser Repräsentation definieren wir einen Rekombinations-Operator, der zufällig eine Stelle auswählt, an der zwei zu rekombinierende Tupel durchschnitten werden, und die beiden so definierten "Enden" der Tupel vertauscht (one-cut-uniform-crossover, siehe Abbildung 4.4 auf Seite 48).

Im nächsten Schritt wird die γ-Modularität der aus der Anwendung der Mutations- und Rekombinations-Operatoren hervorgegangenen Partitionen berechnet. Dann wird die Population durch eine Nachfolge-Population ersetzt (generational change), die in der Folge wieder der Mutation und Rekombination unterliegt. Die Mitglieder der Nachfolge-Population werden dazu aus den mutierten Partitionen so gelost, daß die Selektionswahrscheinlichkeit einer Partition proportional zu seiner γ-Modularität ist. Zuvor werden zur besseren Separierung die Modularitäts-Werte der Population auf das Intervall

001010110110
001111001100
011022200112
010110012222

Abbildung 4.1: Eine Population von Partitionen

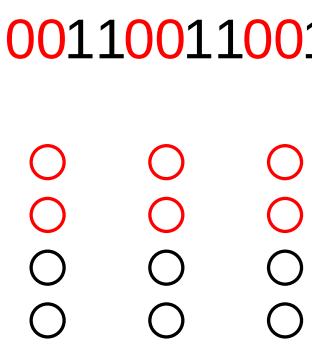

Abbildung 4.2: Eine Partition, dekodiert

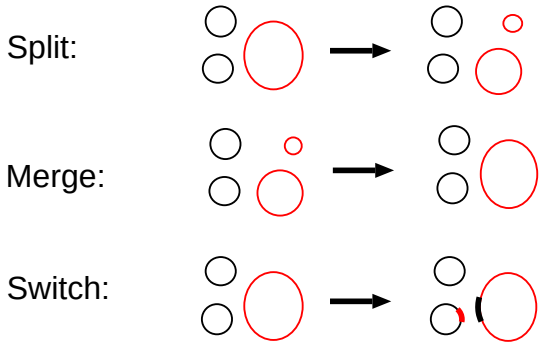

Abbildung 4.3: Die drei Mutations-Operatoren

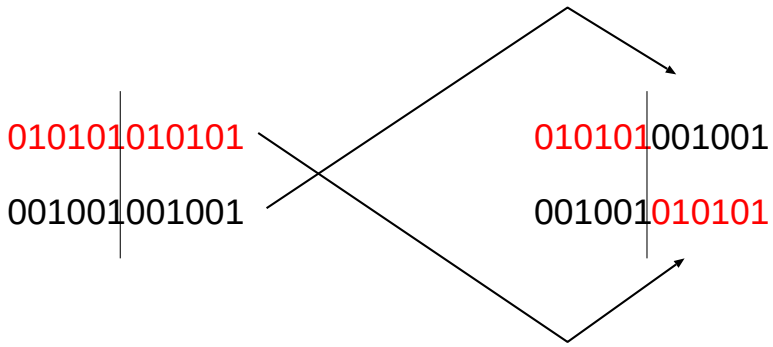

Abbildung 4.4: Der Rekombinations-Operator

[0,1] skaliert. (proportional selection with scaling). Ein bestimmter Prozentsatz der besten Mitglieder der aktuellen Population wird unverändert in die Nachfolge-Population übernommen. Diese Partitionen unterliegen erst dann wieder der Mutation und Rekombination, wenn sie in einer späteren Population nicht mehr zu den besten gehören (Elite). Der Zyklus von Variation, Bewertung (Modularitäts-Bestimmung) und Selektion wird solange fortgesetzt, bis ein Abbruchkriterium, z.B. das Erreichen einer festgelegten Anzahl von Generationen, erfüllt ist. Die Partition mit der höchsten γ-Modularität in der Population wird dann heuristisch als optimale Partition angenommen und die erreichte γ-Modularität als γ-Modularität der Gewichts-Struktur.
Wichtige Parameter des Verfahrens sind die Anzahl der Partitionen in der Population, die Häufigkeit der Mutation und der Rekombination, die Größe der Elite sowie das Abbruchkriterium. Als allgemeine Strategie verwende ich einen sehr hohen Grad von Mutation und Rekombination, vor dem die Besten durch die Elite geschützt werden. Die genauen Werte der Parameter sind im Prinzip abhängig von der betrachteten Aufgabe einzustellen. In meinen Versuchen unterliegt jede Partition in jeder Generation der Rekombination, mit einer Wahrscheinlichkeit von 0.5 der Anwendung des Switch-Operators, sowie mit einer Wahrscheinlichkeit von 0.05 dem Split und dem Merge-Operator. Insgesamt konvergiert die Population nicht zu einer Partition, sondern es bleibt ständig ein großes Maß an Varianz erhalten. Durch die unveränderte Übernahme der besten 10 % bleibt aber gesichert, daß die Modularität der besten Partition monoton steigt. Anders als in einer Strategie, die zu einer Konvergenz der Population zu einer Partition führt, kann eine solche Konvergenz hier nicht als Abbruch-Kriterium verwendet werden. Stattdessen kann man abbrechen, wenn sich die beste Modularität lange Zeit nicht verbessert hat oder einfach, wenn eine festgelegte Anzahl von Generationen erreicht ist. Ich breche die Evolution nach 100 Generationen ab und wiederhole das Verfahren 3 mal. Als Populationsgröße hat sich eine Anzahl von 20 Partitionen als geeignet erwiesen.

Divisiv hierarchisches Clustern

In diesem Verfahren zum Auffinden der optimalen Clusterung wird zunächst unabhängig von der γ-Modularität eine Folge von Partitionen erzeugt. Dies geschieht, indem nacheinander die Verbindungen mit dem kleinsten Gewichtsbetrag aus dem Netz entfernt werden. Das Netz zerfällt dadurch nach einigen Schritten zunächst in zwei Zusammenhangs-Komponenten. Setzt man das Entfernen der kleinsten Verbindungen weiter fort, zerfallen diese Komponenten dann selbst wieder in Sub-Komponenten. Dies setzt sich fort, bis das Netz ganz zerfallen ist, d.h. jede Komponente nur noch aus einem Neuron besteht.
Auf diese Weise erhält man eine Hierarchie von Partitionen. Auf jeder Hierarchie-Stufe, also nach jedem Zerfall in Sub-Komponenten, wird die γ-Modularität der neu entstandenen Partition berechnet. Der Zerfalls-Prozeß wird gestoppt, wenn die γ-Modularität nach dem Zerfall gesunken ist. Die Partition vor diesem letzten Zerfall wird dann heuristisch als die optimale Partition angenommen.

Vergleich

Das divisiv hierarchische Cluster-Verfahren ist schneller als der evolutionäre Algorithmus, da weniger Partitionen betrachtet werden und es ist insofern zuverlässiger, als beständig das gleiche Ergebnis berechnet wird. Während aber im evolutionären Algorithmus als stochastischem Verfahren immer eine gewisse Wahrscheinlichkeit besteht, das Optimum zu finden, kann dies im deterministischen Verfahren schon dadurch ausgeschlossen sein,

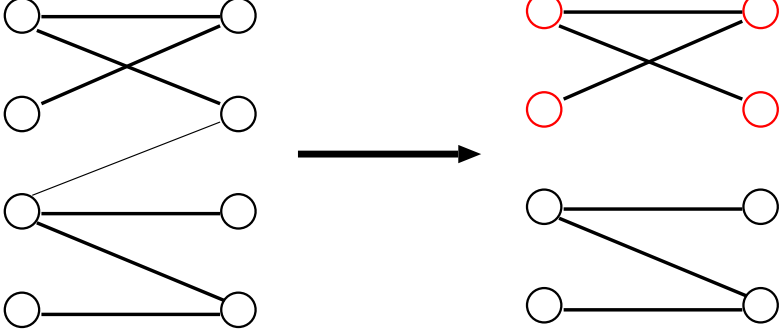

Abbildung 4.5: Divisives hierarchisches Clustern: Zerfall in zwei Komponenten durch Wegnahme der Verbindung mit kleinstem Gewicht.

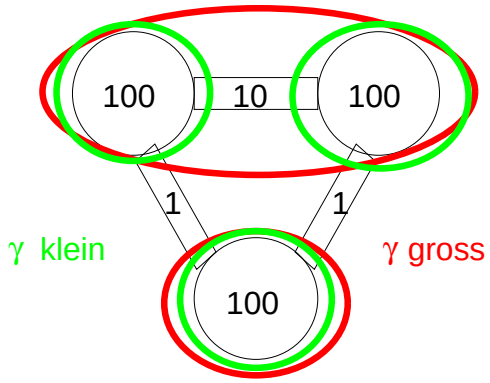

Abbildung 4.6: Rolle des γ-Parameters

daß die optimale Partition keine der durch den Zerfalls-Prozeß erzeugten ist. In einigen Testversuchen hat sich gezeigt, daß bei stark modularen Gewichts-Strukturen das divisive Clustern immer die optimale Partition findet. Bei nur schwacher Modularität findet dagegen das evolutive Verfahren oft geringfügig bessere Partitionen. Für die Verwendung in den folgenden Versuchsreihen reicht das divisive Verfahren aber völlig aus. Das evolutionäre Verfahren ist aus zwei Gründen dennoch hier so ausführlich dargestellt worden. Zum einen stärken die Kontroll-Versuche mit ihm das Vertrauen in das divisive Verfahren. Zum anderen bildet es die Grundlage für ein ähnliches Verfahren, mit dessen Hilfe in den nächsten Kapiteln eine optimale modulare Vorstrukturierung gesucht wird.

4.1.3 Rolle des γ-Parameters

Der γ-Parameter dient dazu, die Gestalt der optimalen Partition und indirekt die Anzahl der Gruppen zu beeinflussen. Ist γ groß, hat also der Inter-Zusammenhang ein großes Gewicht, sind jene Partitionen optimal, bei denen die Gruppen deutlich voneinander getrennt sind. Bei gleicher Gewichtsstruktur des Netzes ist bei kleinem γ dagegen eine Partition optimal, bei der die Gewichtskonzentration innerhalb der Gruppen besonders stark ist. Dies können unter Umständen verschiedene Partitionen sein.
Als Beispiel kann man sich eine Gewichtsstruktur vorstellen, die sich in drei Gruppen mit gleichem, großem mittlerem Gewichtsbetrag der Intra-Verbindungen aufteilen läßt. Zwei dieser drei Gruppen seien aber durch nicht unerhebliche Inter-Gewichte miteinander verbunden, während die Verbindungen beider zur dritten im Mittel eher geringes Gewicht

haben. Je nach Wahl von γ wird entweder eine Aufteilung in drei oder in zwei Gruppen optimal sein. Ist γ klein, sind drei Gruppen optimal, da die Konzentration der Gewichte innerhalb der drei Gruppen am größten ist. Eine Zusammenfassung der zwei noch relativ stark verbundenen Gruppen zu einer wäre nicht optimal, da dadurch der ausschlaggebende Intra-Zusammenhang sinken würde. Anders ist dies bei hohem γ. Jetzt fällt die saubere Trennung der Gruppen stärker ins Gewicht und eine Aufteilung in nur zwei Gruppen ist optimal, weil der Inter-Zusammenhang geringer ist.

4.1.4 Ein komponenten-unabhängiges Struktur-Maß

Die bisher betrachtete γ-Modularität entspricht sehr genau dem Kern-Konzept von Modularität, wie ich es im ersten Kapitel herausgearbeitet habe, indem es nämlich Bezug nimmt auf eine Aufteilung des Systems in Komponenten. Daraus resultiert die Messung der γ-Modularität eines neuronalen Netzes mittels der optimalen Partition. Dieses Maß hat aber den Nachteil, daß sich die optimale Partition bereits ab einer geringen Anzahl von Neuronen nicht mehr durch erschöpfende Suche finden läßt, sondern nur noch heuristisch berechnet werden kann. Ich stelle aus diesem Grund noch ein alternatives Struktur-Maß S vor, das komponenten-unabhängig ist und damit direkt, d.h. ohne Verwendung eines Cluster-Verfahrens, berechnet werden kann. Ein solches Maß macht aber keine Aussage über Modularität im eigentlichen Sinne mehr.
Zur Berechnung des Struktur-Maßes S wird für jedes Neuron die Verteilung der Beträge seiner Gewichte betrachtet. Ist das Neuron mit allen Vorgänger-Neuronen über den gleichen Gewichtsbetrag verbunden und ist dies für alle Neuronen einer Schicht der Fall, kann man die Verbindung der Schichten als sehr unstrukturiert ansehen. Umgekeht gilt im Falle einer sehr modularen Struktur, daß jedes Neuron nur mit einigen seiner Vorgänger über betragsmässig große Gewichte, zu den anderen dagegen über kleine Gewichte verbunden ist. Das Struktur-Maß S mißt diese Ungleichverteilung der Gewichtsbeträge. Dazu wird zunächst für jedes Neuron die Standard-Abweichung der Gewichts-Beträge berechnet. Zur Normierung werden diese anschließend durch die Summe der Gewichts-Beträge dividiert. Die normierten Standard-Abweichungen werden dann über alle Neuronen der inneren und der Ausgabe-Schicht gemittelt.

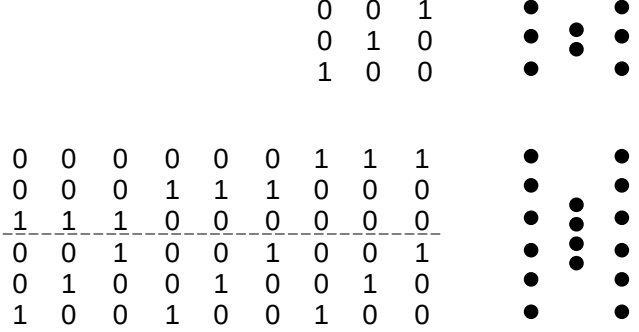

Abbildung 4.7: Beispiel für das Zusammensetzen zweier Encoder-Trainingsmengen zu einer Modularen Encoder Aufgabe. Links oben die zu assoziierenden Vektoren einer 3-Bit-Encoder Aufgabe. Unten die daraus gebildete Trainingsmenge des Modularen Encoders. Rechts die dazu korrespondierenden Netz-Architekturen.

4.2 Der Modulare Encoder

4.2.1 Die Konstruktion der Aufgabe

Die modulare Encoder-Aufgabe entsteht aus der Zusammensetzung von mehreren identischen Encoder-Aufgaben. Bei der Encoder-Aufgabe handelt es sich um ein autoassoziatives Problem, d.h. die Eingabevektoren und die Zielvektoren in den Lernbeispielen sind identisch. Die Anzahl der inneren Neurone ist in dieser Aufgabe typischerweise viel kleiner als die Anzahl der Eingabe- bzw. Ausgabeneuronen, man spricht von einer Flaschenhals-Architektur. Die Aufgabe heißt Encoder, weil die Aktivitäten der inneren Neuronen im trainierten Netz nicht-lineare Umkodierungen der Trainingsvektoren in einen niedriger dimensionalen Raum darstellen. Als zu assoziierende Vektoren wählen wir für unsere Zwecke die n-dimensionalen Einheitsvektoren.
Zwei Lernaufgaben lassen sich zusammensetzen, indem das kartesische Produkt der Trainigsmengen gebildet wird. Die so erhaltenen Paare von Lernbeispielen werden zu einer neuen Trainingsmenge vereinigt, indem die beiden Eingabevektoren eines Paares ebenso wie die beiden Zielvektoren durch "Hintereinanderschreiben" zu jeweils einem neuen Vektor vereinigt werden. Mit Hintereinanderschreiben (oder "Konkatenieren") meine ich dabei den binären Operator $C : \Re^n \times \Re^m \rightarrow \Re^{n+m}$, der einem Paar von Vektoren $(a, b) = ((a_1, \ldots, a_n), (b_1, \ldots, b_m))$ den Vektor $c = C(a, b) = (a_1, \ldots, a_n, b_1, \ldots, b_m)$ zuordnet (siehe Abbildung 4.7 auf Seite 52).
Ich verwende eine aus drei 8-Bit Endoder Trainingsmengen zusammengesetzte Aufgabe. Den resultierenden 512-elementigen Datensatz teile ich zufällig in eine Trainings- und eine Testmenge mit je 256 Elementen. Die Anzahl der Ein- und Ausgabeneuronen beträgt 24. Als Anzahl der inneren Neuronen wähle ich 12.

4.2.2 Lernen in modular vorstrukturierten Architekturen

Fragestellung und Aufbau

Zuerst möchte ich klären, welchen Einfluß eine der Konstruktion der Aufgabe entsprechende modulare Vorstrukturierung auf das Lernverhalten hat. Ich unterscheide dabei drei Arten der modularen Vorstrukturierung:

- In der ersten Art, werden die Neuronen des Netzes werden zuerst in 3 gleich große Gruppen eingeteilt, so wie es der Konstruktion der Aufgabe entspricht. Durch Entfernen der Inter-Verbindungen wird das Netz dann in vollständig getrennte Unter-Netze aufgeteilt (Split, **Sp**).

- In der zeiten Art werden die Gewichte der Inter-Verbindungen mit Null initialisiert, können aber im Verlaufe des Trainings andere Werte annehmen (modulare Initialisierung, Softsplit, **So**).

- In der dritten Art wirkt zusätzlich zur Initialisierung mit Null wirkt auf die Gewichte der Inter-Verbindungen während des gesamten Trainings ein Weight Decay. Als Decay-Faktor wähle ich $\alpha = 10^{-4}$ (Softsplit mit Weight Decay, **So4**). Hier handelt es sich also nicht mehr um eine reine Vortrukturierung. Dennoch benötigt diese Trainings-Variante das Vorwissen um die Aufgaben-Konstruktion und wird deshalb in diesem Zusammenhang behandelt.

- Zum Vergleich wird ein nicht vorstrukturiertes Netz auf gleiche Weise trainiert (ungeteiltes Netz, Unsplit, **Un**).

Mit den so vorstrukturierten Netzen wird jetzt ein Backpropagation-Training mit einer Lernrate von $\eta = 0.8$ durchgeführt. Das Training wird 100-mal aus verschiedenen zufälligen Initialisierungen der Netz-Gewichte heraus wiederholt und jeweils 1000 Epochen lang durchgeführt. Die Anfangs-Gewichte werden gleichverteilt zufällig aus dem Intervall (-1,1) gezogen. Während des Trainings werden in jeder Epoche folgende Kennzahlen aufgenommen:

- Der Trainingsfehler, angegeben als RMS-Fehler (root mean squared error). Der RMS-Fehler ist die Wurzel des durch die Anzahl der Elemente der Datenmenge normierten mittleren quadratischen Fehlers: $RMS = \sqrt{(E/\|\mathcal{D}\|}$.

- Der binarisierte Trainingsfehler. Den binarisierten Fehler definiere ich als die Summe der pro Epoche an allen Ausgabeneuronen vorkommenden Abweichungen, die größer als ein Schwellwert von 0.1 sind.

- Der Testfehler, d.h. der Fehler auf der Testmenge, angegeben als RMS-Fehler.

- Der binarisierte Testfehler.

- Die Modularität der ersten Schicht.

  Es hat sich gezeigt, daß die Modularität eines MLP am besten schichtenweise berechnet wird, da die Gewichtsbeträge je nach Schicht oft sehr unterschiedlich sind, ohne daß in der Schicht mit den höheren Beträgen von einem stärkeren "Zusammenhang" der Neuronen gesprochen werden könnte.

  Für γ wähle ich in allen Experimenten einen mittleren Wert 6. Zur Berechnung verwende ich das schnellere divisive Cluster-Verfahren.

- Die Modularität der zweiten Schicht.

- Das komponenten-unabhängige Struktur-Maß S.

Ergebnisse

Abbildung 4.8 auf Seite 56 zeigt Netz-Gewichte und Neuronen-Clusterungen in verschiedenen Lernsituationen. Die Abbildung gibt einen qualitativen Eindruck der Phänomene, die im folgenden quantitativ beschrieben werden. Sie besteht aus 4 Teil-Abbildungen, die wiederum aus verschiedenen Bereichen zusammengesetzt sind.

Bevor ich auf die dargestellten Inhalte eingehe, erkläre ich zunächst den Aufbau der Teil-Abbildungen: Die Raster aus roten und grünen Kästchen oben links und unten rechts veranschaulichen die Gewichte der Verbindungen. Die Farbe eines Kästchens kodiert das Vorzeichen eines Gewichts und die Größe des Kästchens den Betrag des Gewichtes relativ zum maximalen Betrag über alle Verbindungen. Der maximale Betrag ist als "maxW" unten links als Text angegeben. Eine Zeile des oberen linken Rasters repräsentiert die Verbindungen eines Eingabe-Neurons zu allen inneren Neuronen. Eine Zeile des unteren rechten Rasters repräsentiert die Verbindungen eines inneren Neurons zu allen Ausgabeneuronen.
Das Diagramm oben rechts veranschaulicht das Ergebnis des Cluster-Verfahrens, also die Einteilung der Neuronen in Gruppen. Neuronen einer Gruppe werden in gleicher Farbe dargestellt. Die inneren Neuronen sind doppelt dargestellt, da die Clusterung nach Schichten getrennt vorgenommen wird und jedem inneren Neuron somit zweimal eine Gruppen-Nummer zugeteilt wird. Die Modularität der ersten und der zweiten Schicht sind unten links als Text als "mod1" bzw. "mod2" angegeben.

In der oberen linken Teil-Abbildung von Abbildung 4.8 ist ein nicht vorstrukturiertes Netz vor Beginn des Trainings dargestellt. Viele Verbindungen haben nach der Initialisierung sehr ähnliche (kleine) Gewichtsbeträge, entsprechend sind viele Kästchen ähnlich groß. Die Modularität ist sehr gering. In der oberen rechten Teil-Abbildung ist dasselbe Netz am Ende des Trainings dargestellt. Die Verbindungen der ersten Schicht haben jetzt gegenüber der zweiten Schicht deutlich geringere Gewichte. Die Modularität ist in beiden Schichten höher als am Anfang, in der ersten Schicht größer als in der zweiten, aber in beiden nicht sehr deutlich ausgeprägt. Das Clusterverfahren gruppiert so auch nur einige aber nicht alle der Neuronen so, wie es der Lern-Aufgabe entsprechen würde. In der unteren linken Teil-Abbildung ist ein vollständig aufgeteiltes Netz am Ende des Trainings dargestellt. Das maximale Gewicht ist geringer als im ungeteilten Netz. Die Module werden vom Cluster-Verfahren vollständig korrekt erkannt. In der unteren rechten Teil-Abbildung ist ein nicht vorstrukturiertes Netz, das mit einem der im nächsten Abschnitt vorgestellten Trainings-Varianten (Cluster-Training) trainiert wurde, dargestellt. Das maximale Gewicht ist noch geringer als im geteilten Netz. Die Module werden vom Cluster-Verfahren vollständig korrekt erkannt. Die Zuordnung von inneren Neuronen zu Modulen ist hier zufällig anders als jene, die für das geteilte Netz gewählt wurde. Die erreichte Modularität ist sehr hoch.

Abbildung 4.9 auf Seite 57 zeigt die Entwicklung der Fehler- und Modularitäts-Maße während des Trainings. Die Kurven zeigen die Mittelwerte der 100 Läufe. In der oberen Zeile ist der Fehler auf der Trainings-Menge dargestellt. Die linke Seite zeigt die Entwicklung bis zur zehnten Epoche, die rechte Seite die Entwicklung danach. In der zweiten Zeile ist auf gleiche Weise die Entwicklung des Fehlers auf der Testmenge dargetellt. Die anderen drei Abbildungen zeigen die Entwicklung der Modularität der ersten Schicht, der zweiten Schicht und des Struktur-Maßes S.

Der mittlere Trainings-Fehler sinkt im ungeteilten Netz in den ersten Epochen am schnellsten, gefolgt von den Softsplit-Netzen und dem geteilten Netz. Nach etwa 5-6 Epochen hat sich diese Reihenfolge gerade umgekehrt. Am Ende liegt der Trainings-Fehler für das ungeteilte Netz deutlich am höchsten. Das geteilte Netz erreicht einen noch etwas geringeren Trainings-Fehler als die Softsplit-Netze. Der mittlere Test-Fehler sinkt ebenfalls beim ungeteilten Netz zunächst am schnellsten. Bereits nach 2 Epochen liegt der Test-Fehler aber beim geteilten Netz niedriger. Nach 4 Epochen liegt auch der Test-Fehler der Softsplit-Netze niedriger als jener des ungeteilten Netzes. Am Ende hat das ungeteilte Netz den höchsten Test-Fehler, mit einem Wert, der wesentlich über dem Wert seines Trainings-Fehlers liegt. Das geteilte Netz hat auch am Ende den geringsten Test-Fehler. Die Softsplit-Netze liegen knapp darüber, wobei das Softsplit-Netz mit Weight Decay noch einen etwas niedrigeren Test-Fehler als das reine Softsplit-Netz erreicht.
Die Modularität, die das ungeteilte Netz erreicht, ist nur sehr gering, das Softsplit-Netz erreicht eine mittlere Modularität, während das Softsplit-Netz mit Weight Decay am Ende des Trainings im Mittel fast die Modularität des geteilten Netzes erreicht.

In **Tabelle 4.1** auf Seite 58 sind die End-Werte der in Abbildung 4.9 dargestellten Kurven zusammengefaßt, zusammen mit den zugehörigen Standardabweichungen. Dabei steht RMS für den Trainings-Fehler, tRMS für den Test-Fehler, mod1 und mod2 für die Modularität der ersten, bzw. der zweiten Schicht und S für das Struktur-Maß.

Eine sehr drastische Darstellung der unterschiedlichen Leistungsfähigkeit der verschieden vorstrukturierten Netze erhält man, wenn man nicht Mittelwerte betrachtet, sondern den Anteil der Läufe, die einen binarisierten Testfehler von 0 erreichen, also den Anteil der in diesem Sinne "erfolgreichen" Läufe. In **Tabelle 4.2** auf Seite 58 sind diese Werte zusammen mit der Epoche aufgeführt, in der der binarisierte Test-Fehler im Mittel zum ersten Mal 0 erreicht hat. Man erkennt die deutlichen Unterschiede sowohl in der Schnelligkeit wie auch in der Zuverlässigkeit des Lernens. Während im ungeteilten Netz nie ein binarisierter Testfehler von Null erreicht wird, ist dies in allen Läufen mit dem vollständig geteilten Netz der Fall, wobei dazu im Mittel 32.8 Epochen benötigt werden. Modulare Anfangs-Gewichte bringen bereits eine wesentlichen Fortschritt gegenüber dem normal initialisierten ungeteilten Netz mit einer Rate von 40 % und einer mittleren Epoche von 631.4. Der zusätzliche Weight Decay verbessert das Ergbnis weiter auf eine Rate von 86 % und eine mittlere Epoche von 179.8.

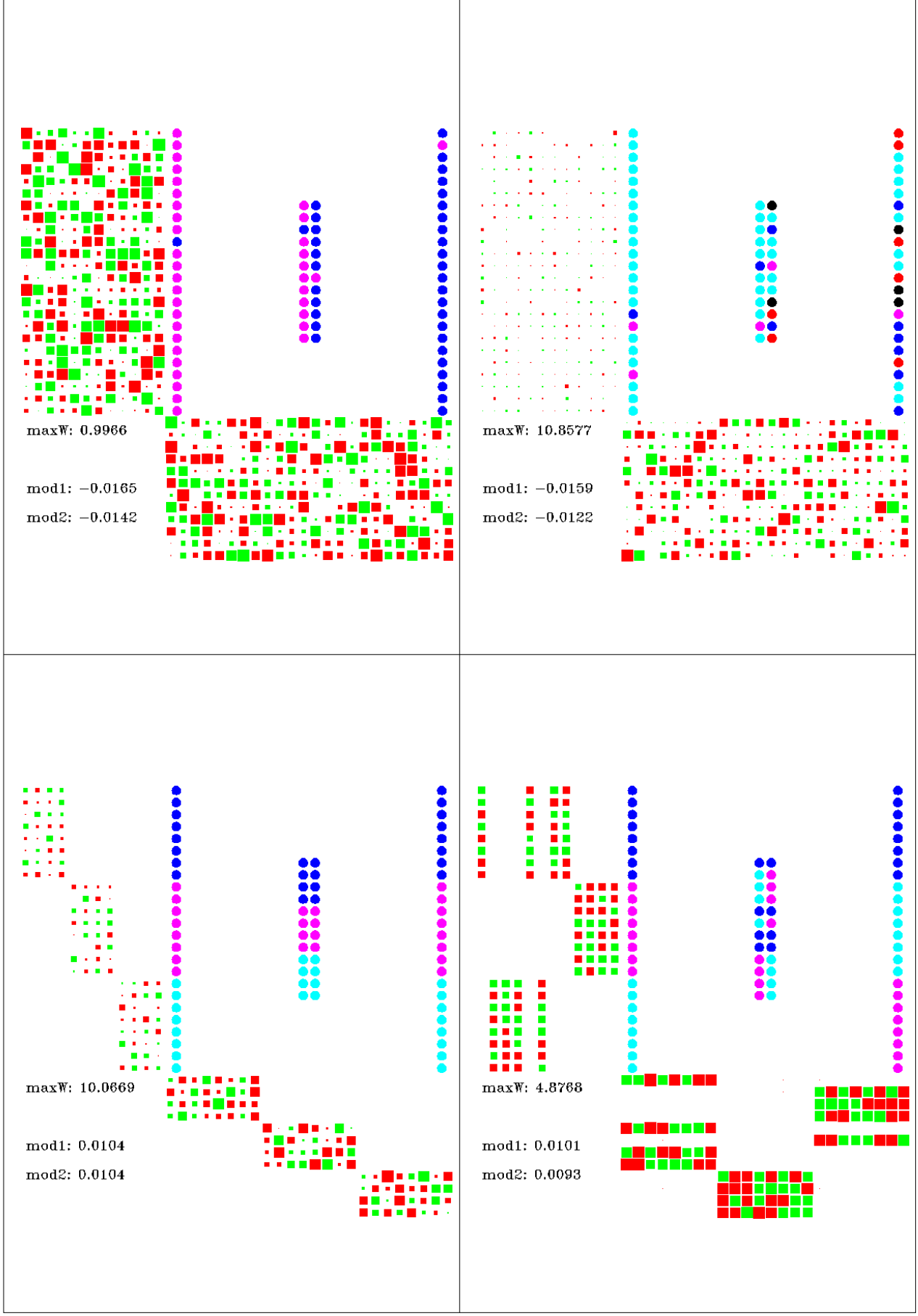

Abbildung 4.8: Netz-Gewichte und Neuronen-Clusterung in verschiedenen Lernsituationen. Erläuterungen im Text.

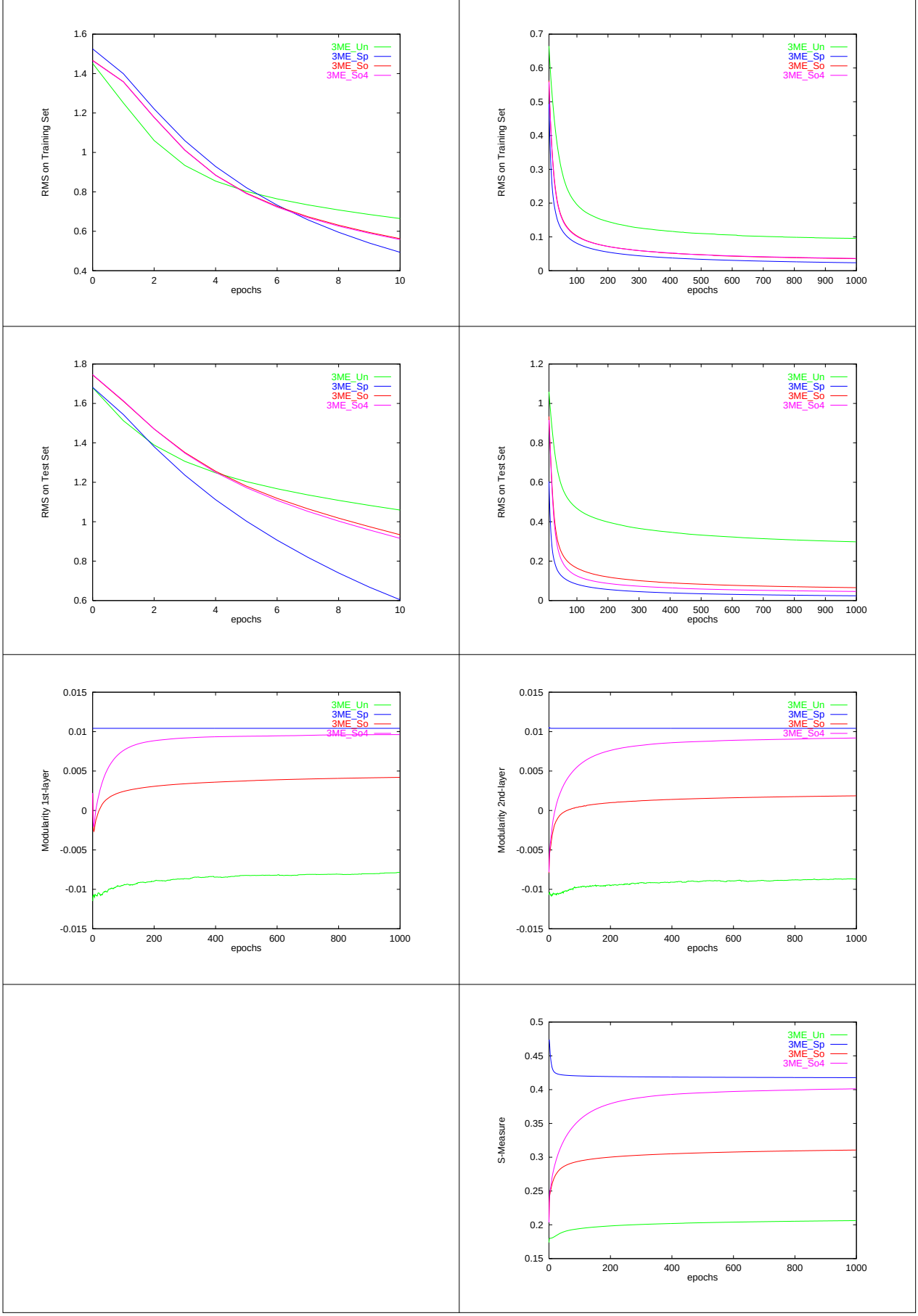

Abbildung 4.9: Entwicklung von Fehler und Modularität während des Trainings. Vergleich modulare Vorstrukturierung - keine Vorstrukturierung. Erläuterungen im Text.

	Unsplit (**Un**)	Split (**Sp**)	Softsplit (**So**)	Softsplit & Weight Decay (**SoWD**)
RMS	0.10 ± 0.05	0.02 ± 0.00	0.04 ± 0.03	0.04 ± 0.03
tRMS	0.30 ± 0.11	0.02 ± 0.00	0.07 ± 0.05	0.05 ± 0.05
mod1 $\cdot 10^4$	-78.5 ± 22.7	104.2± 0	42.0 ± 15.7	96.2 ± 13.6
mod2 $\cdot 10^4$	-86.7 ± 26.3	104.2± 0	18.7 ± 16.9	92.0 ± 15.8
S $\cdot 10^2$	20.6 ± 1.6	41.8 ± 1.0	31.1 ± 1.8	40.1 ± 2.7

Tabelle 4.1: Fehler- und Modularitäts-Maße am Ende des Trainings. Mittelwerte und Standardabweichungen. Vergleich modulare Vorstrukturierung - keine Vorstrukturierung. Erläuterungen im Text.

	Unsplit (**Un**)	Split (**Sp**)	Softsplit (**So**)	Softsplit & Weight Decay (**SoWD**)
%	0	100	40	86
t_0	-	32.8	631.4	179.8

Tabelle 4.2: Anteil erfolgreicher Läufe (%) und Anzahl der Epochen (t_0) bis zum Erreichen eines binarisierten Test-Fehlers von 0 (Mittelwert über erfolgreiche Läufe). Vergleich modulare Vorstrukturierung - keine Vorstrukturierung. Erläuterungen im Text.

4.2.3 Modularisierung in nicht vorstrukturierten Architekturen

Fragestellung

Im letzten Abschnitt ist der Vorteil modular vorstrukturierter Netze beim Training des Modularen Encoder Problems deutlich geworden. Um eine Vorstrukturierung vornehmen zu können, wird aber die Kenntnis der Aufgaben-Konstruktion benötigt. In diesem Abschnitt wird nun der Fall betrachtet, daß die Modularität der Aufgabe nicht vor Beginn des Trainings bekannt ist. Ein einfaches Backpropagation Training reicht dann, wie gesehen, nicht aus, um einen binarisierten Test-Fehler von Null zu erreichen. Wie kann also das Backpropagation-Verfahren verändert werden, damit ein Training auch ohne a priori Information gelingt?

Aufbau und Vorversuche

In Abschnitt 3.2.4 "Struktur-Optimierung" des letzten Kapitels ist bereits das Regularisierungs-Verfahren des Weight Decay beschrieben worden. Ich versuche nun in einem ersten Ansatz mit Hilfe dieses Verfahrens eine Modularisierung der Gewichtsstruktur zu erzeugen. Der Decay wirkt jetzt anders als im letzten Abschnitt gleichmäßig auf alle Verbindungen. Wie sich zeigt, ist dieses Verfahren geeignet, eine modulare Netz-Struktur zu erzeugen, wenn der Decay Faktor α groß genug gewählt wird. Ein Weight Decay Term in der Fehlerfunktion bedeutet gleichzeitig jedoch immer auch ein gewisses Maß an Fehlinformation, d.h. das Verfahren neigt dazu, in lokale Optima mit insgesamt zu niedrigen Gewichten zu geraten.

Ein naheliegender zweiter Ansatz ist also ein Weight Decay, dessen Stärke im Verlauf des Lernens abnimmt. Auf diese Weise kann zunächst eine modulare Gewichts-Struktur entstehen, aus der heraus dann die Gewichte ansteigen können. Wie im letzten Abschnitt

anhand des Softsplit-Netzes gezeigt wurde, bedeutet eine modulare Anfangsstruktur bereits einen wesentlichen Vorteil. War dort die Anfangsstruktur aber nur modular und sonst zufällig, ist eine durch Weight Decay erzeugte Gewichtsstruktur darüberhinaus der Aufgabe sogar schon ein Stück weit angepaßt. Ich lasse also in meinem zweiten Ansatz den Weight Decay Faktor α abnehmen, und zwar von einem hohen Anfangswert aus linear.

Eine nähere Untersuchung der erfolgreichen Läufe mit dem linear abnehmenden Weight Decay zeigt, daß die Summe der Gewichtsbeträge des Netzes, eine Größe, die ich als "Energie" bezeichne, in diesen Läufen einen deutlich vom einfachen Backpropagation oder von Backpropagation mit konstantem Weight Decay verschiedenen Verlauf nimmt. Dies ist in **Abbildung 4.10** auf Seite 61 dargestellt. Bei einfachem Backpropagation steigt die Energie steil an und verharrt dann auf hohem Niveau. Bei Backpropagation mit Weight Decay steigt die Energie ebenfalls steil an, wird dann aber langsamer wieder abgebaut, was zu einer charakteristischen "Rutschen"- Form führt. Im Gegensatz dazu steigt bei den erfolgreichen Läufen mit abnehmendem Weight Decay die Energie relativ langsam und gleichmäßig an.

Dies führt zu einem dritten Ansatz, nämlich den langsamen Anstieg der Energie explizit zu erzwingen. Dies geschieht, indem nach jeder Epoche eines einfachen Backpropagation-Trainings die Gewichte durch Division mit einem geeigneten Wert so umskaliert werden, daß die Energie des Netzes langsam linear anwächst. Der "natürlichen Neigung" der Gewichte, steil anzusteigen, wird dadurch eine künstliche Verknappung entgegengesetzt. Auf diese Weise wird sehr rasch eine starke Modularität erzeugt. Ich nenne dieses Verfahren "Weight Growth".

Der vierte Ansatz schließlich geht wieder aus von der Idee der durch einen starken Weight Decay erzeugten Modularität. Statt aber den Decay kontinuierlich abnehmen zu lassen, wird er in diesem Verfahren solange konstant hoch gehalten, bis die Modularität der Struktur ausreichend ausgeprägt ist. Dann macht man sich die durch das Cluster-Verfahren erhaltene zusätzliche Information zunutze und läßt den Weight Decay auf den Intra-Gewichten komplett wegfallen, während man ihn auf die Inter-Gewichte weiter wirken läßt. Dies schafft eine Situation, wie sie etwa dem Softsplit-Netz mit Weight Decay im letzten Abschnitt entspricht, wieder mit dem Unterschied, daß die Gewichte nicht nur modular strukturiert, sondern bereits auch ein Stück weit trainiert sind, wenn der Intra-Decay wegfällt. Ich nenne dieses Verfahren im folgenden "Cluster-Training".

Alle vier Ansätze hängen von zusätzlichen Parametern ab, die möglichst gut gewählt werden müssen, um einen fairen Vergleich der Verfahren zu ermöglichen. Zu diesem Zwecke führe ich zunächst jeweils eine Versuchsreihe mit 100 Läufen aus unterschiedlichen Initialisierungen und mit zufällig aus sinnvollen Intervallen gezogenen Parametern durch. Anschließend betrachte ich jeweils die Abhängigkeit des Test-Fehlers von den Parametern in einem Histogramm. **Abbildung 4.11** auf Seite 62 zeigt die erhaltenen Histogramme.

Das Histogramm in der ersten Zeile zeigt die Abhängigkeit des Test-Fehlers vom Weight Decay Faktor α bei Backpropagation mit Weight Decay. In den folgenden Experimenten wähle ich $\alpha = 4.0$.
Die zweite Zeile zeigt links das Histogramm des Weigt Decay Anfangswerts und rechts das Histogramm des Parameters " Abnahme pro Epoche" im Verfahren des abnehmenden Weight Decay. Ich wähle den Anfangs-Decay als $3.2 \cdot 10^{-4}$ und die Abnahme als $0.08 \cdot 10^{-4}$.

Die dritte Zeile zeigt die Abhängigkeit des Parameters "Energie-Anstieg pro Epoche" im Weight-Growth Verfahren. Ich wähle den Wert 1.3.

In der vierten Zeile schließlich finden sich die Histogramme der Parameter des Cluster-Trainings-Verfahrens, der Weight Decay Faktor und die "Modularitäts-Schwelle", d.h. diejenige Modularität, die (in der ersten Schicht) erreicht sein muß, um das Aufheben des Decays auf den Intra-Verbindungen zu bewirken. Das Histogramm des Weight Decay Faktors legt den sehr niedrigen Wert von etwa $\alpha = 10^{-5}$ nahe. Ein solcher Wert führt aber nie zu einer hohen Modularität. Hier versagt also diese Methode der Parameter-Bestimmung. Der Grund dafür ist, daß ein sehr starker Weight Decay nur in Verbindung mit einer gut gewählten Schwelle zu guten Ergebnissen führt, ansonsten aber zu sehr schlechten Ergebnissen. Im Mittel ist daher ein niedriger Weight Decay erfolgreicher. Zur Veranschaulichung dieses Sachverhalts zeigt **Abbildung 4.12** auf Seite 63 die Entwicklung von Modularität und Test-Fehler für einen erfogreichen und einen nicht erfolgreichen Fall bei Cluster-Training. In letzterem Fall wurde die Modularitätschwelle nicht erreicht und der anhaltend sehr hohe Weight Decay führt zu einem hohen Test-Fehler. Im erfogreichen Fall sinkt der Fehler dagegen sehr rasch nach Erreichen der Schwelle. In der folgenden Versuchsreihe nehme ich für die Parameter des Cluster-Trainings Werte an, die mir nach einigem Experimentieren geeignet erschienen, nämlich für den Weight Decay Faktor den Wert $\alpha = 10^{-3.5}$ und für die Modularitäts-Schwelle den Wert 0.007.

Nach diesen Vorüberlegungen wird mit jedem der vier Verfahren ein Backpropagation-Training mit einer Lernrate von $\eta = 0.8$ durchgeführt. Das Training wird 100-mal aus verschiedenen zufälligen Initialisierungen der Netz-Gewichte heraus wiederholt und jeweils 1000 Epochen lang durchgeführt. Die Anfangs-Gewichte werden gleichverteilt zufällig aus dem Intervall (-1,1) gezogen. Als Kennzahlen werden der Test- und Trainingsfehler, die Modularität der ersten und der zweiten Schicht und das Struktur-Maß S aufgenommen.

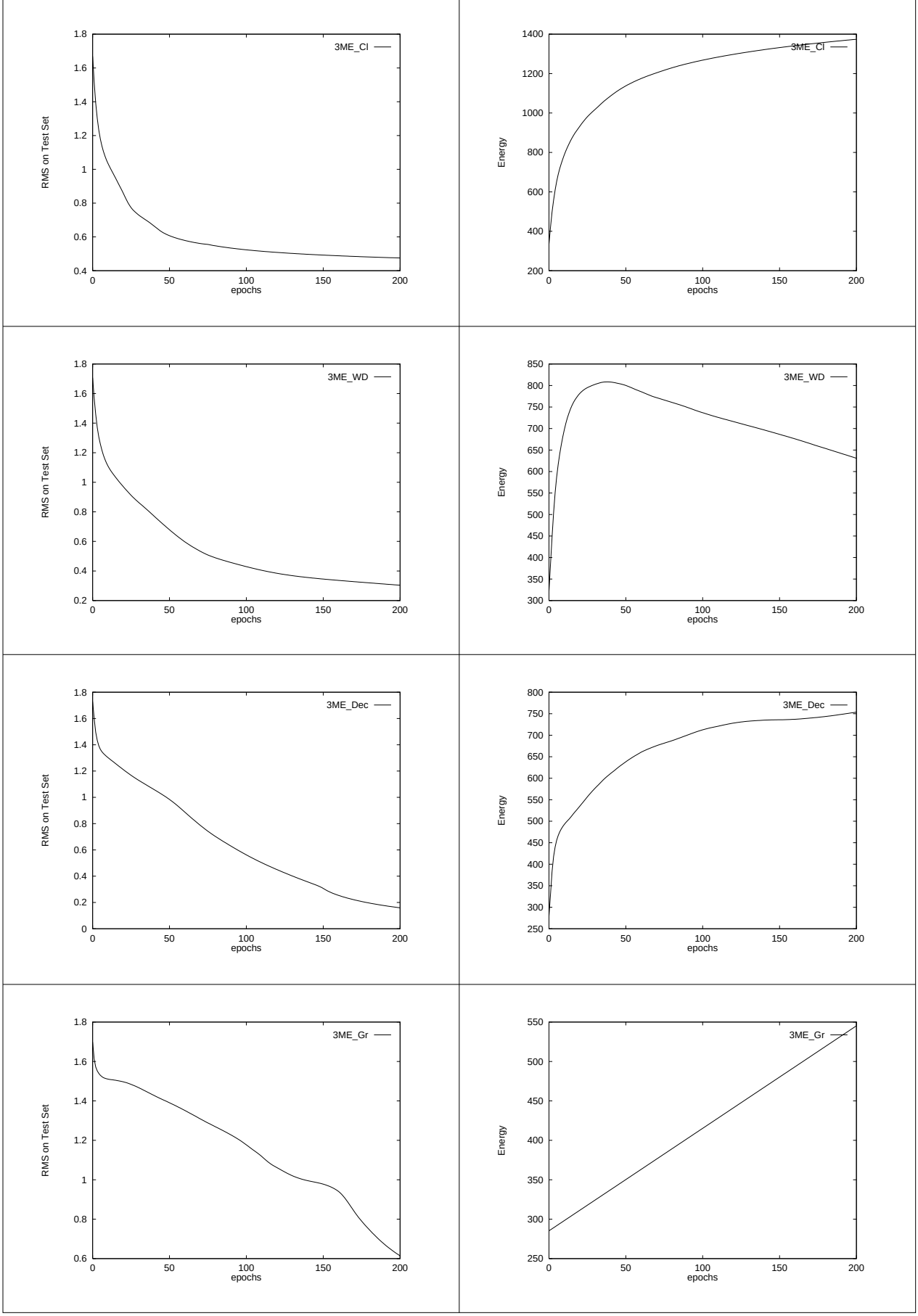

Abbildung 4.10: Entwicklung der Netz-Energie am Trainings-Anfang. Vergleich verschiedener Verfahren. Erläuterungen im Text.

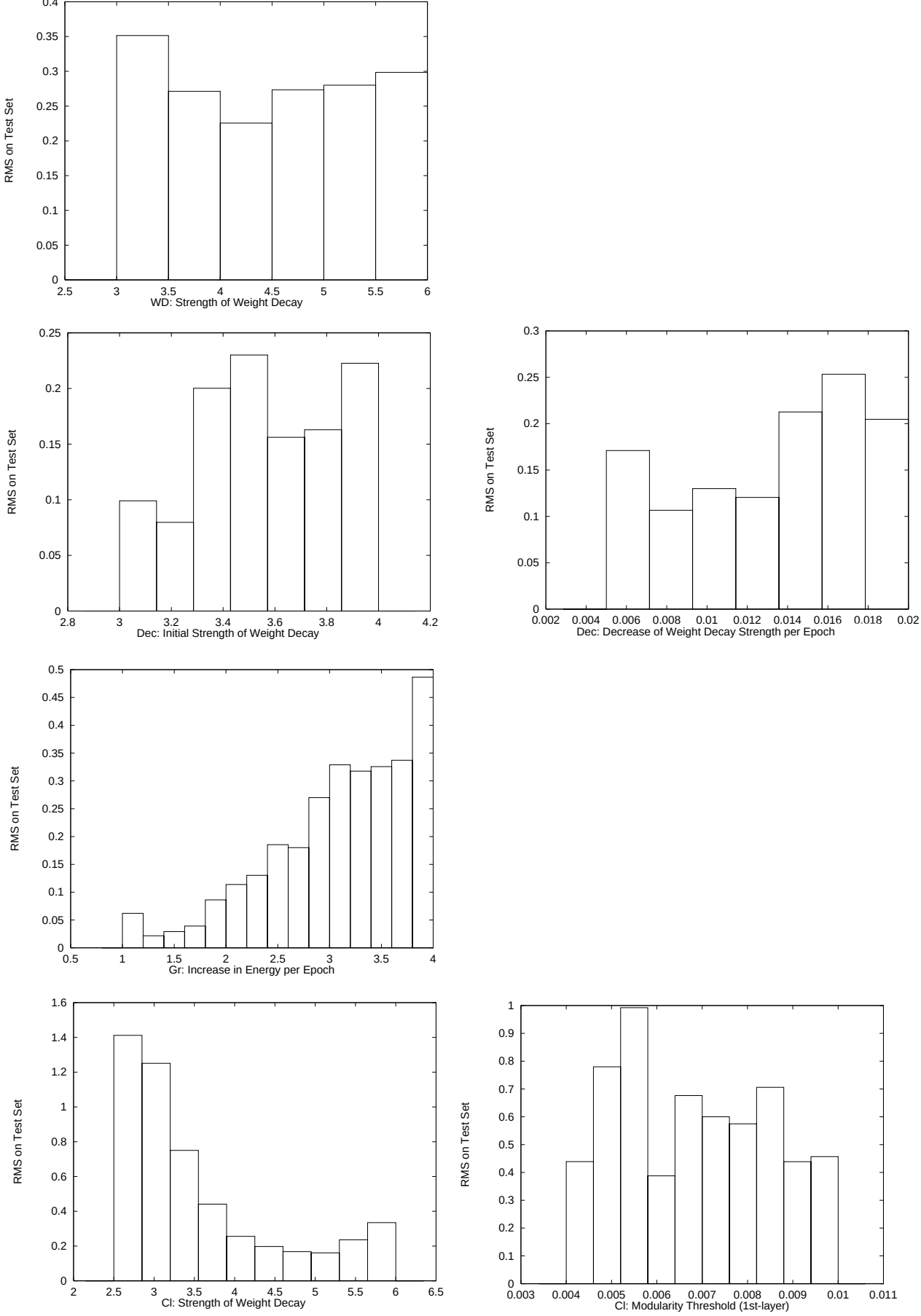

Abbildung 4.11: Abhängigkeit des Test-Fehlers von den Parametern der Verfahren. Die Höhe der Boxen entspricht dem Mittelwert der in diese Box fallenden Observationen. Erläuterungen im Text.

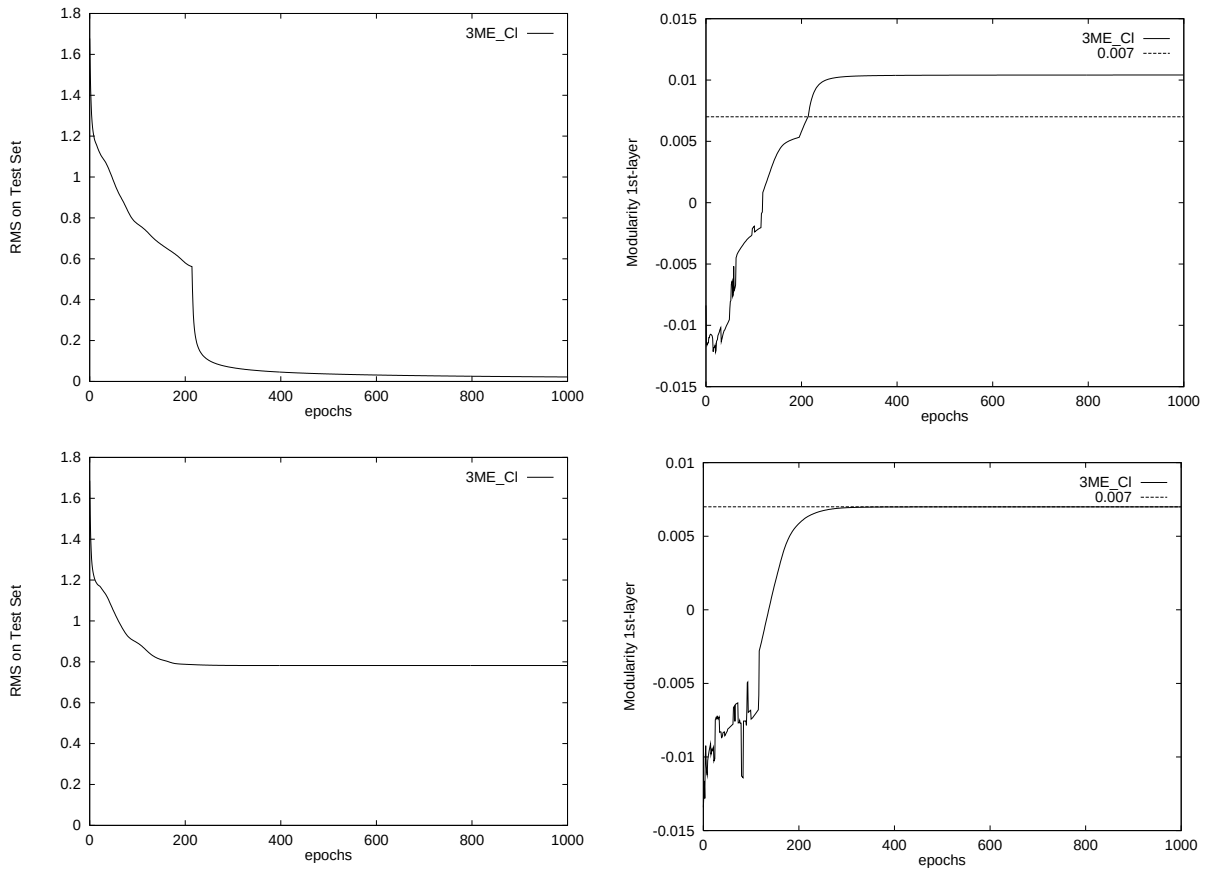

Abbildung 4.12: Entwicklung von Test-Fehler und Modularität im Cluster-Trainingsverfahren. Vergleich erfolgreicher Lauf - nicht erfogreicher Lauf. Erläuterungen im Text.

Abbildung 4.13 auf Seite 66 zeigt wieder die Entwicklung der Fehler- und Modularitäts-Maße während des Trainings. Dargestellt sind Backpropagation mit konstantem Weight Decay (**WD**), mit abnehmendem Weight Decay (**Dec**), Weight Growth (**Gr**) und Cluster-Training (**Cl**). Außerdem sind die Kurven des einfachen Backpropagation Trainings im ungeteilten Netz (**Un**) aus dem letzten Abschnitt zum Vergleich wieder mit eingetragen. Die Kurven zeigen die Mittelwerte der 100 Läufe. In der oberen Zeile ist der Fehler auf der Trainings-Menge dargestellt. Die linke Seite zeigt die Entwicklung bis zur zehnten Epoche, die rechte Seite die Entwicklung danach. In der zweiten Zeile ist auf gleiche Weise die Entwicklung des Fehlers auf der Testmenge dargetellt. Die anderen drei Abbildungen zeigen die Entwicklung der Modularität der ersten Schicht, der zweiten Schicht und des Struktur-Maßes S.

Die Entwicklung der Trainings- und Test-Fehler in der Anfangsphase (Diagramme links oben) wird wesentlich durch die Höhe des anfänglichen Weight Decay bestimmt. Dies führt zu der anfänglichen Reihenfolge: einfaches Backprop, Backprop mit Weight Decay, Cluster-Training und Backprop mit abnehmendem Weight Decay. Am langsamsten sinken anfangs beide Fehlerarten beim Weight Growth. Diese Reihenfolge ändert sich dann drastisch im weiteren Verlauf des Trainings (Diagramme rechts oben). Auffallend ist zunächst, daß Backprop mit Weight Decay am Ende zwar einen deutlich höheren Trainings-Fehler als einfaches Backprop aufweist, aber dennoch einen niedrigeren Test-Fehler. Insofern bringt bereits ein konstanter Weight Decay eine wesentliche Verbesserung. Die Fehler der anderen Verfahren liegen aber noch niedriger. Den niedrigsten Trainings- und Testfehler erzielt im Mittel das Weight Growth- Verfahren, gefolgt von Backprop mit abnehmendem Weight Decay und dem Cluster-Training, die am Ende im Mittel sehr ähnliche Werte erreichen.
Die Entwicklung der Modularität wird anfangs ebenfalls durch die Stärke des Weight Decays bestimmt - je stärker der Decay, umso höher die Modularität. Am steilsten steigt die Modularität beim Weight Growth Verfahren an. Im weiteren Verlauf flacht dann als erstes die Modularitäts-Kurve des abnehmenden Weight Decay ab, gefolgt von der Kurve des Weight Growth, dessen Modularität nach Überschreiten eines Maximums wieder abfällt und am Ende nur knapp über der Kurve des abnehmenden Decays liegt. Die zweithöchste Modularität erreicht Backpropagation mit Weight Decay und die höchste das Cluster-Training. Einfaches Backpropagation führt dagegen nur zu geringer Modularität.
In **Tabelle 4.4** auf Seite 67 sind wieder die End-Werte der in Abbildung 4.13 dargestellten Kurven zusammengefaßt, zusammen mit den zugehörigen Standardabweichungen. Dabei steht RMS für den Trainings-Fehler, tRMS für den Test-Fehler, mod1 und mod2 für die Modularität der ersten, bzw. der zweiten Schicht und S für das Struktur-Maß.

Eine sehr drastische Darstellung der unterschiedlichen Leistungsfähigkeit der verschieden vorstrukturierten Netze erhält man auch hier, wenn man nicht Mittelwerte betrachtet, sondern den Anteil der Läufe, die einen binarisierten Testfehler von 0 erreichen, also den Anteil der in diesem Sinne "erfolgreichen" Läufe. In **Tabelle 4.3** auf Seite 67 sind diese Werte zusammen mit der Epoche aufgeführt, in der der binarisierte Test-Fehler im Mittel zum ersten Mal 0 erreicht hat. Man erkennt die deutlichen Unterschiede sowohl in der Schnelligkeit wie auch in der Zuverlässigkeit des Lernens. Einfacher Weight Decay reicht in keinem Lauf aus, um einen Testfehler von Null zu erreichen. Abnehmender Weight Decay mit einer Rate von 54 % und einer mittleren Epoche von 402.8 sowie das Weight Growth

Verfahren mit einer Rate von 79 % bei einer mittleren Epoche von 366.9 bringen bereits erhebliche Verbesserungen und sind erfolgreicher als die einfache modulare Initialisierung (Softsplit) im letzten Abschnitt. Die mit einer Rate von 93 % und einer mittleren Epoche von 250.8 erfolgreichste Methode ist aber das Cluster-Training. In den Mittelwert-Kurven ist diese Überlegenheit nicht deutlich geworden, weil die nicht erfolgreichen Läufe mit ihrem beim Cluster-Training sehr hohen Fehler den Mittelwert stark verzerren.

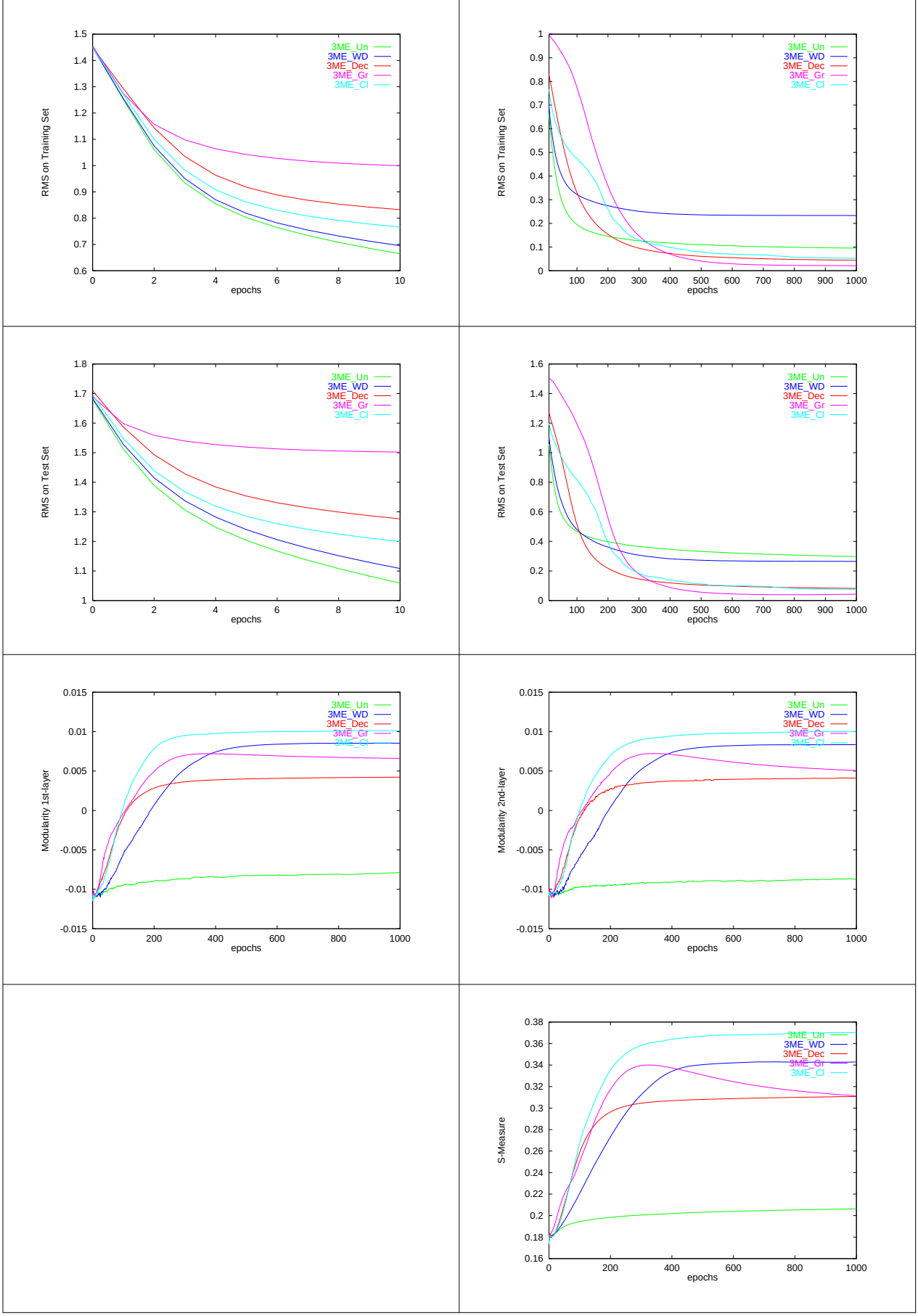

Abbildung 4.13: Entwicklung von Fehler und Modularität während des Trainings. Vergleich verschiedener Trainings-Varianten im nicht vorstrukturierten Fall.

	Weight Decay (**WD**)	Decreasing Weight Decay (**Dec**)	Weight Growth (**Gr**)	Weight Decay & Clustern (**Cl**)
%	0	54	79	93
t_0	-	402.8	366.9	250.8

Tabelle 4.3: Anteil erfogreicher Läufe und Anzahl der Epochen bis zum Erreichen eines binarisierten Test-Fehlers von 0 (Mittelwert über erfolgreiche Läufe). Vergleich verschiedener Trainings-Varianten im nicht vorstrukturierten Fall. Erläuterungen im Text.

	Weight Decay (**WD**)	Decreasing Weight Decay (**Dec**)	Weight Growth (**Gr**)	Weight Decay & Clustern (**Cl**)
RMS	0.23 ± 0.01	0.04 ± 0.04	0.02 ± 0.02	0.05 ± 0.12
tRMS	0.26 ± 0.02	0.08 ± 0.08	0.04 ± 0.04	0.08 ± 0.19
mod1 $\cdot 10^4$	85.4 ± 5.5	42.3 ± 26.5	66.0 ± 14.1	101.1 ± 11.6
mod2 $\cdot 10^4$	83.5 ± 4.5	41.1 ± 32.0	50.7 ± 8.8	100.2 ± 13.6
S $\cdot 10^2$	34.3 ± 0.5	31.1 ± 3.0	31.1 ± 1.1	37.0 ± 1.6

Tabelle 4.4: Fehler- und Modularitäts-Maße am Ende des Trainings. Mittelwerte und Standardabweichungen. Vergleich verschiedener Trainings-Varianten im nicht vorstrukturierten Fall. Erläuterungen im Text.

4.2.4 Auffinden der optimalen modularen Vorstrukturierung

Fragestellung

Die im vorigen Abschnitt vorgestellten Trainings-Varianten beruhen auf der geschickten Manipulation des Weight Decay und im Falle des Cluster-Trainings auf einer Analyse der durch den Weight Decay hervorgebrachten Modularitäts-Struktur durch ein Cluster-Verfahren. Dies setzt voraus, die zu lernende Aufgabe ist von solcher Art, daß ein nicht vorstrukturiertes Netz während des Lernens durch den Weight Decay modular wird. Ist dies nicht oder nicht ausreichend stark der Fall, benötigt man andere Verfahren, um eine dennoch möglicherweise vorhandene modulare Struktur der Aufgabe zu erkennen. Ich schlage dazu ein evolutionäres Verfahren vor, das anhand einer Bewertung der Lern-Performanz eine optimale Vorstrukturierung der Neuronen findet.

Allgemeine Methode

Das verwendete evolutionäre Verfahren entspricht weitgehend dem bereits in Abschnitt 4.1.2 auf Seite 47 im Zusammenhang mit der Berechnung der γ-Modularität beschriebenen Verfahren zum Auffinden einer optimalen Neuronen-Partition. Insbesondere wird die gleiche Art der Repräsentation von Partitionen verwendet, die gleichen Split-, Merge- und Switch-Mutationsoperatoren, die gleiche Form der Rekombination und das gleiche Reproduktions-Schema mit proportionaler Selektion und Elite. Ein Unterschied zu dem Verfahren, wie es in Abschnitt 4.1.2 verwendet wurde, betrifft die genaue Kodierung der Partition. Ich bediene mich nämlich zur Verringerung der Anzahl der zu evoluierenden Parameter der Symmetrie der Encoder-Aufgabe, indem ich festlege, daß die sich entsprechenden Ein- und Ausgabeneuronen immer der gleichen Gruppe zugehören.

Der wesentliche Unterschied betrifft aber die Fitneß-Funktion. In Abschnitt 4.1.2 war die Fitneß einer Partition die Modularität der Partition bei gegebenen Netz-Gewichten. Das

Ziel der Evolution ist jetzt ein anderes. Entsprechend dem Ziel, eine für das Lernverhalten optimale Vorstrukturierung zu finden, wird die Fitneß einer Partition aus der Effektivität des Lernens bestimmt, die eine Vorstrukturierung mit dieser Partition bewirkt. Zur Bestimmung der Fitneß einer Partition wird also ein Netz zunächst entsprechend dieser Partition vorstrukturiert. Dann wird das Netz wiederholt aus jeweils verschiedenen Gewichts-Initialisierungen heraus trainiert und der Mittelwert eines noch näher zu bestimmenden Performanz-Kriteriums zur Bewertung der Fitneß herangezogen. Es hat sich gezeigt, daß es aufgrund der nur geringen Anzahl von Wiederholungen, die zeitlich möglich sind, sehr wichtig ist, für alle zu vergleichenden Partitionen die gleiche Folge von Gewichts-Initialisierungen zu verwenden, und zwar in jeder Generation. Wechselnde Netz-Initialisierungen bei nur wenigen Wiederholungen haben eine statistisch stark schwankende Fitneß-Landschaft zur Folge, die eine Evolution massiv beeinträchtigt.

Als Art und Weise der Vorstrukturierung kommen ein vollständiger Split, ein Softsplit, also modulare Anfangs-Gewichte, und ein Softsplit mit Weight Decay in Frage, also jene Arten der Vorstrukturierung, die bereits in Abschnitt 4.2.2 auf Seite 52 beschrieben wurden. Bei Verwendung der Vorstrukturierungs-Methode des Softsplit mit Weight Decay wähle ich die Stärke des Weight Decay nicht als festen Wert, sondern lasse sie mit der Partition mitevoluieren. Der Decay Faktor wird dazu auf einem zusätzlichen Chromosom des Individuums kodiert und das Verfahren mit einem zusätzlichen Mutationsoperator ausgestattet. Ich habe den Decay Faktor als ganze Zahl zwischen (einschließlich) 2 und 8 kodiert. Diese Zahl wird dann beim Training als negativer 10er-Logarithmus des zu verwendenden Decay Faktors aufgefaßt. Ein Wert von 2 bedeutet also einen sehr starken Weight Decay von $\alpha = 10^{-2}$, während ein Wert von 8 lediglich einen sehr geringen Weight Decay von $\alpha = 10^{-8}$ bewirkt. Der neue Mutationsoperator verändert den aktuellen Wert um 1 in eine zufällige Richtung unter Einhaltung der Intervall-Grenzen 2 und 8.

Als allgemeine Strategie verwende ich wieder die Kombination aus einem sehr hohen Grad am Mutation und Rekombination, dem die Elite stabilisierend entgegenwirkt ("nichtkonvergierende Strategie"). Jede Partition unterliegt in jeder Generation der Rekombination und der Anwendung des Switch-Operators, sowie mit einer Wahrscheinlichkeit von 0.1 dem Split und dem Merge-Operator. Die Größe der Elite beträgt 10 % der Größe der Population. Die Größe der Population beträgt 30 Individuen.

Ich habe Versuche mit einer Aufteilung der Population in ringförmig angeordnete Unter-Populationen ("Inseln") durchgeführt. Diese evoluieren getrennt und tauschen nur von Zeit zu Zeit die besten Individuen mit den Nachbar-Inseln aus. Eine solche Aufteilung dient vor allem dazu, eine vorschnelle Konvergenz der Gesamt-Population zu nur einer einzigen Partition zu verhindern. Sie hat aber bei der von mir verwendeten "nichtkonvergierenden" Strategie in einigen Testläufen keinen Vorteil erbracht. Ich habe daher nur eine Gesamt-Population verwendet.

Verwendete Performanzkriterien

Ich interessiere mich zunächst dafür, ein Performanz-Kriterium zu finden, das eine Evolution zu der der Konstruktion der Aufgabe entsprechenden und in diesem Sinne "korrekten" Partition bewirkt, also einer Partition in drei gleich große Teile mit einer korrekten Gruppierung der Ein-und Ausgabeneuronen. Ein solches Kriterium muß mehrere Eigenschaften besitzen:

- Zum einen muß die Fitneß der korrekten Partition nach diesem Kriterium höher sein als die eines nicht vorstrukturierten Netzes. Die Ergebnisse aus Abschnitt 4.2.2 legen einige Kandidaten für solche Kriterien nahe.

- Zum anderen muß außerdem gewährleistet sein, daß die korrekte Partition nach dem gewählten Kriterium auch besser abschneidet als alle anderen kombinatorisch möglichen Partitionen. Dies läßt sich aus den bisherigen Ergebnissen genau genommen nicht herauslesen. Aufgrund der Konstruktion der Aufgabe sollte man zwar vermuten, daß die "korrekte" Partition jeder anderen Partition überlegen ist. Da aber nur wenige und immer die gleichen Initial-Gewichte für das Netz-Training verwendet werden, kann es vorkommen, daß bei genau diesen Initial-Gewichten eine andere als die erwartete Partition optimal ist.

- Schließlich ist noch eine weitere Eigenschaft des Performanz-Kriteriums wichtig. Es reicht nicht aus, daß die korrekte Partition das Optimum des Kriteriums bildet, sondern es muß auch in realistischer Zeit eine Evolution hin zu diesem Optimum möglich sein. Das bedeutet zum einen, daß die durch das Kriterium erzeugte Fitneß-Landschaft hinreichend einfach sein muß, so daß die Anzahl der erforderlichen Generationen klein ist. Zum anderen darf jede einzelne Fitneß-Auswertung nicht übermäßig viel Zeit in Anspruch nehmen, das Training des Netzes also nicht zu lange dauern.

Der letzte Gesichtspunkt lenkt das Augenmerk auf Unterscheidungs-Kriterien, die bereits nach wenigen Lern-Schritten deutlich ausgeprägt sind, und zwar so deutlich, daß das Training nur wenige Male wiederholt werden muß, um eine zuverlässige Unterscheidung zu ermöglichen. In Abbildung 4.14 sind noch einmal die Entwicklung von Trainings- und Test-Fehler in den ersten 100 Epochen dargestellt. Man kann erkennen, daß sich die Entwicklung der Fehler hinsichtlich der Unterscheidbarkeit der verschiedenen Kurven ungefähr in drei Phasen einteilen läßt: In der ersten Phase fallen alle Kurven steil ab und liegen eng beieinander. In der folgenden Übergangsphase flachen die Kurven stark ab, wobei die Kurven in dieser Phase stark voneinander abweichen. In der dritten Phase verlaufen die Kurven dann fast parallel zur x-Achse und alle Kurven außer jener des ungeteilten Netzes liegen wieder nahe beieinander.
Ich habe mich entschieden, die Performanz in der Übergangsphase zu messen, die den besten Kompromiß zwischen kurzer Trainings-Dauer und guter Unterscheidbarkeit bietet. Die Unterscheidbarkeit der vorstrukturierten Netze untereinander ist dort sogar noch größer als in der End-Phase. Zwei Möglichkeiten der Bewertung stehen zur Wahl. Man kann entweder eine bestimmte Epoche festlegen, zu der der Trainings- bzw. der Test-Fehler der verschiedenen Partitionen der Population gemessen wird. Oder man legt einen Fehler fest und mißt die Anzahl der Epochen, die zum Erreichen dieses Fehlers benötigt werden. Ich habe mich für letztere Möglichkeit entschieden, und zwar bei Verwendung eines zu erreichenden Trainingsfehlers von 0.34. Dieser Wert ist auch in Abbildung 4.14 eingetragen. Der Grund für diese Wahl ist, daß die Kurven der Softsplit-Netze nach diesem Kriterium deutlicher von der Kurve des ungeteilten Netzes getrennt liegen, als bei Verwendung einer festen Epochen-Zahl oder des Testfehlers. Diese Eigenschaft ist nützlich, da ich als Vorstrukturierungs-Art Softsplit mit Weight-Decay verwende. Es wird der Mittelwert des Kriteriums aus 3 Läufen aus verschiedenen (aber im Laufe der Evolution gleichbleibenden) Netz-Initialisierungen verwendet. Die variable Dauer des Trainingsläufe, die das Kriterium zur Folge hat, bewirkt eine Beschleunigung des Verfahrens im Laufe der Generationen in dem Maße, wie die mittlere Fitneß der Population steigt.

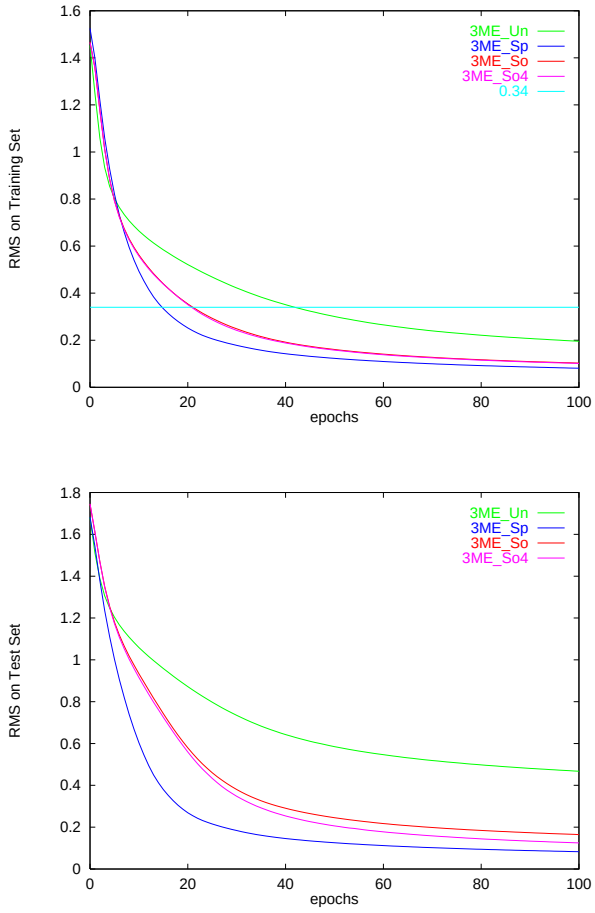

Abbildung 4.14: Entwicklung von Trainings-Fehler (oben) und Test-Fehler (unten) in den ersten 100 Epochen bei modularer Vorstrukturierung. Vergleiche Abb. 4.9. Oben ist der Fehler eingezeichnet, den es bei dem verwendeten Fitneß-Maß möglichst schnell zu erreichen gilt.

Ergebnisse

In Abbildung 4.15 auf Seite 72 ist im oberen Teil die Entwicklung der Fitneß der besten Neuronen-Partition der Population im Laufe der Generationen dargestellt. Beim Abbruch der Evolution benötigt also das Training eines entsprechend der besten Partition der Population vorstrukturierten Netzes im Mittel über 3 Läufe nur noch knapp 16 Epochen, bis der Grenz-Trainingsfehler erreicht wird. Im unteren Teil von Abbildung 4.15 ist die Entwicklung des zur besten Partition gehörenden Decay Faktors dargestellt. Der Weight Decay, der zur besten Partition gehŕt, wird also im Verlauf der Evolution immer stärker.

In den Abbildungen 4.16 und 4.17 auf Seite 73 und Seite 74 sind einige markante "Stationen der Evolution dargestellt, d.h. einige der jeweils zu einer bestimmten Generation besten Partitionen. Die Schraffuren kennzeichnen die Gruppenzugehörigkeit der Neuronen, "gen" gibt die Generation an, "wd" bezeichnet den negativen 10-er Logarithmus des Decay Faktors α und "fit" steht für die Fitneß der Partition, also die mittlere Anzahl der benötigten Epochen.
Die beste Partition der Anfangs-Generation, in der alle Partitionen noch aus zwei Gruppen bestehen, zeichnet sich durch eine Aufteilung der inneren Neuronen in zwei gleich große Gruppen aus. Die Gruppenzugehörigkeit der äußeren Neuronen erscheint dagegen noch zufällig. In Generation 13 hat sich bereits eine ungefähre Dreiteilung der äußeren Neuronen etabliert, das untere Drittel hat aber noch keine entsprechenden Neuronen in der inneren Schicht, d.h. auf die Verbindungen zu den Neuronen dieses Drittels wirkt der - mit $\alpha = 10^{-6}$ allerdings sehr schwache - Weight Decay. In Generation 78 haben alle äußeren Neuronen Gruppen-Partner in der inneren Schicht, die Gruppengrößen sind aber noch sehr unterschiedlich und die Aufteilung der äußeren Neuronen noch recht fehlerhaft. In Generation 130 ist die Aufteilung der äußeren Neuronen fast korrekt (ein Fehler in der mittleren Gruppe) und der Weight Decay ist stärker geworden. Die Gruppengrößen in der inneren Schicht sind aber noch unterschiedlich. In Generation 154 sind die inneren Neuronen dann gleichmäßig aufgeteilt, in den äußeren Neuronen ist immer noch ein Fehler. In Generation 261 ist die Partition dann "perfekt", in dem Sinne, daß sie der Konstruktion der Aufgabe entspricht. Allerdings wird in Generation 267 eine Partition gefunden, die, obwohl sie nicht i.d.S. perfekt ist, bei gleichem Weight Decay eine leicht höhere Fitneß hat. In Generation 296 ist dann wieder eine "perfekte" Partition optimal, und zwar mit einem noch stärkeren Weight Decay.

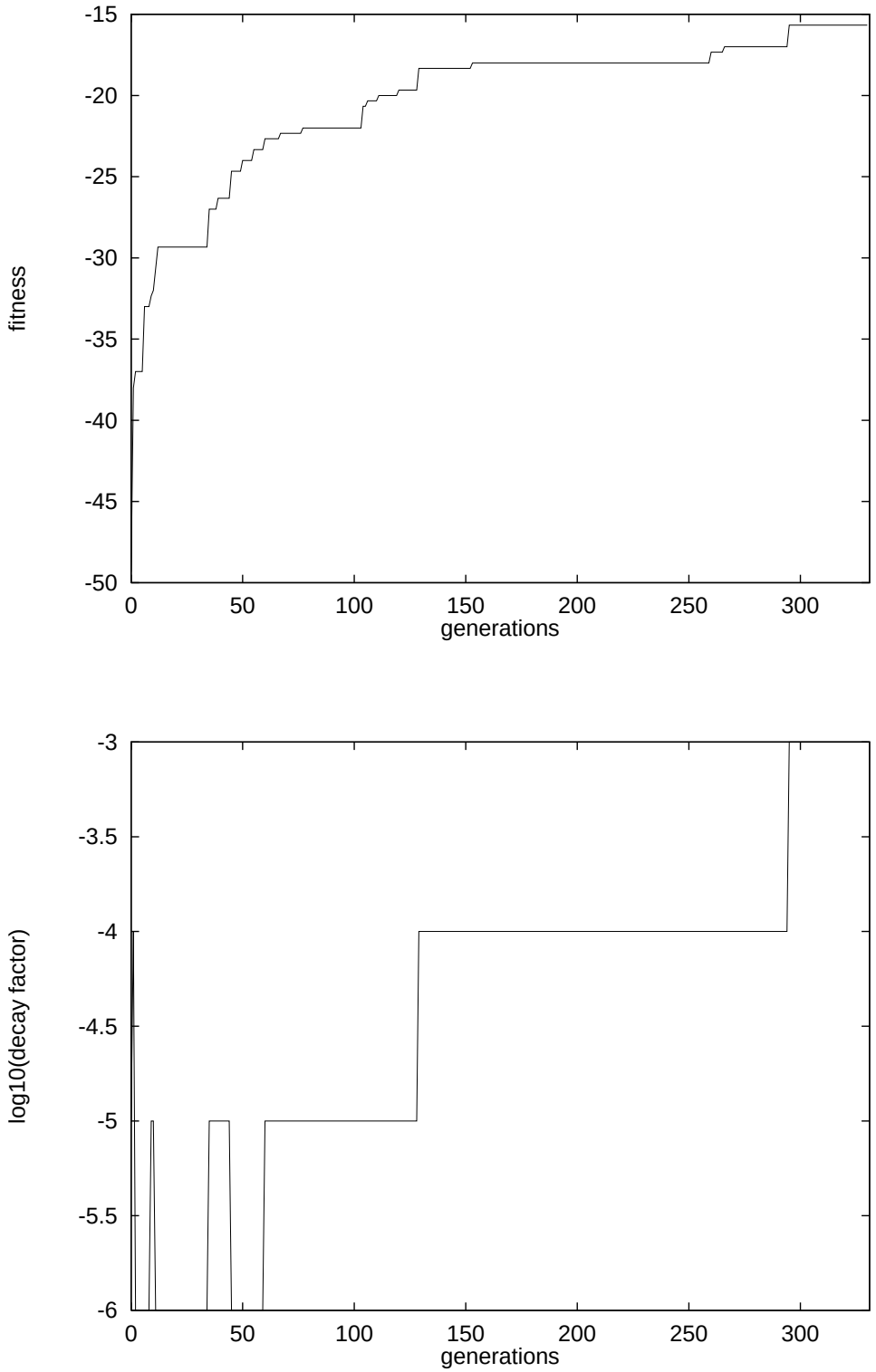

Abbildung 4.15: Oben: Entwicklung der Fitneß des besten Individuums über die Generationen. Die Kurve ist aufgrund der Verwendung der Elite monoton steigend. Unten: Entwicklung des Weight Decay Faktors. Der Decay wird im Laufe der Evolution stärker.

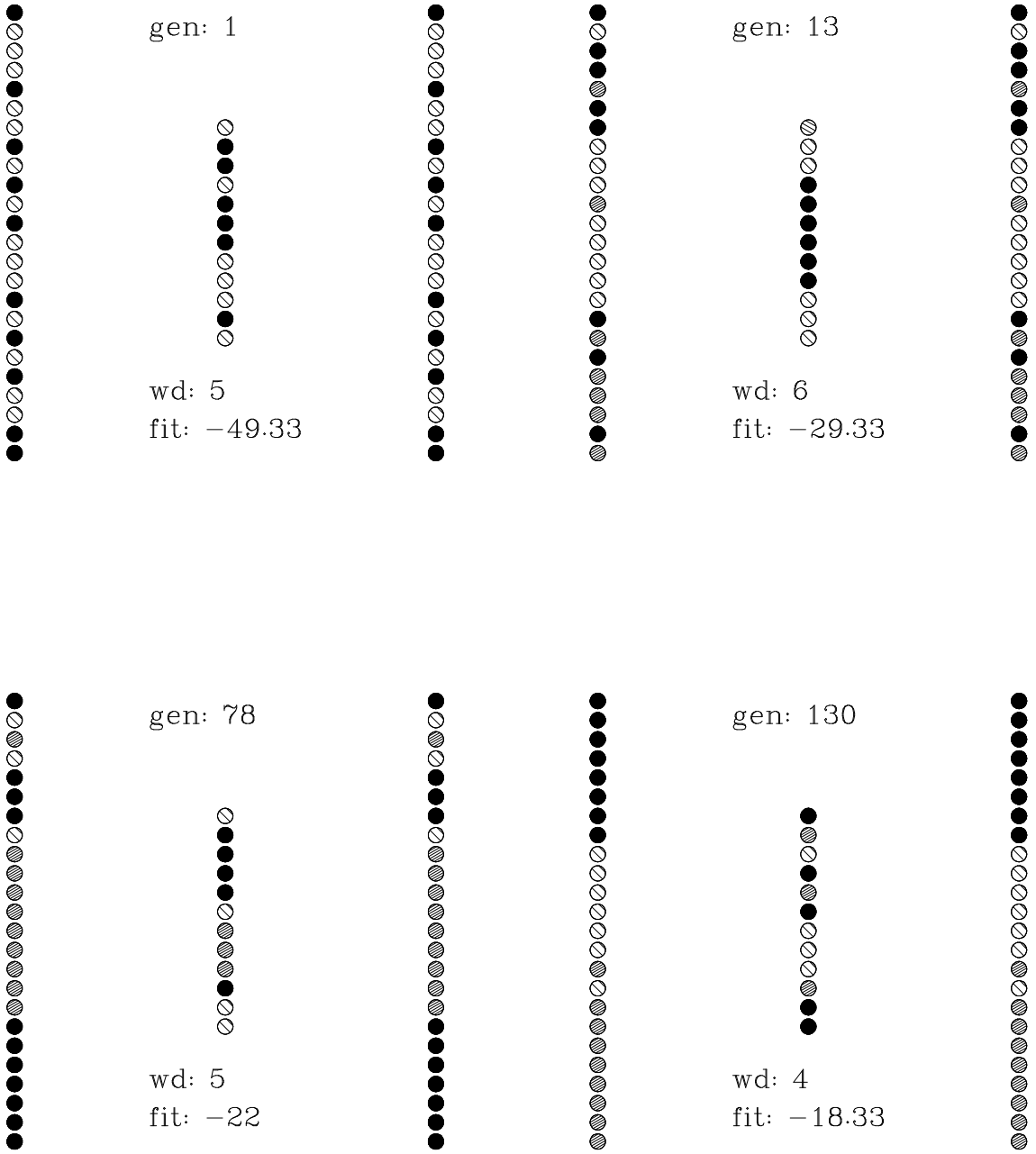

Abbildung 4.16: Stationen der Evolution, erste Seite. Es ist jeweils das beste Individuum der Population in der angegebenen Generation dargestellt. “wd” gibt den negativen 10er-Logarithmus des Decay Faktors an.

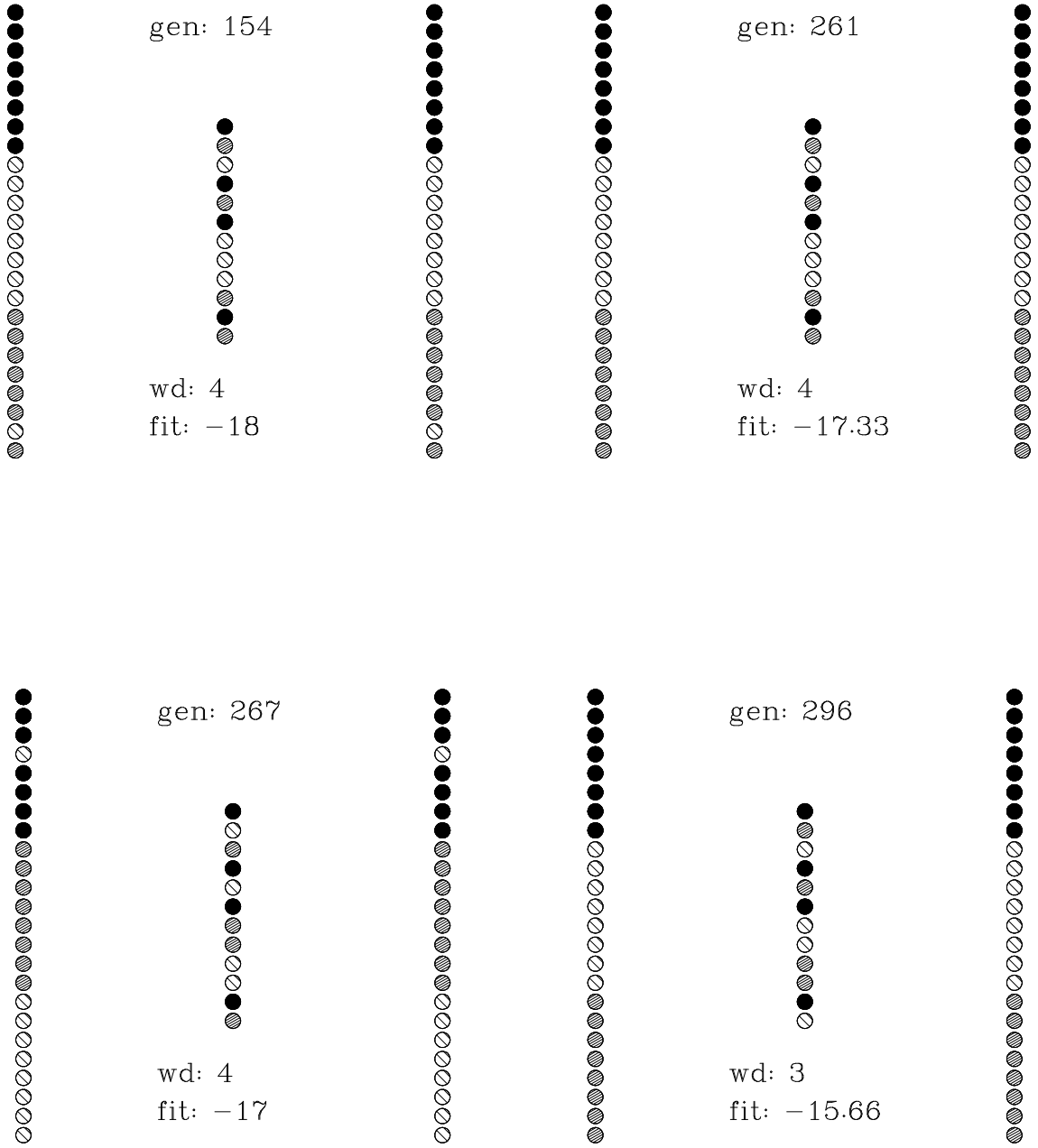

Abbildung 4.17: Stationen der Evolution, erste Seite. Es ist jeweils das beste Individuum der Population in der angegebenen Generation dargestellt. “wd” gibt den negativen 10er-Logarithmus des Decay Faktors an.

4.3 Vorbildnetze

4.3.1 Modulare Daten aus Vorbildnetzen

Beim Modularen Encoder hat man es aufgrund der Konstruktion mit deutlich getrennten Datendimensionen zu tun. Möchte man die Art und Stärke der Modularität der Daten feiner steuern, kann man die Daten durch ein bereits in gewünschter Weise modular strukturiertes "Vorbildnetz" erzeugen. Die Neuronen eines Netzes werden dazu in mehrere Gruppen geteilt. Die Gewichte der Verbindungen, die innerhalb von Neuronengruppen verlaufen (Intra-Gewichte), werden normalverteilt zufällig gesetzt, und zwar mit Mittelwert Null und einer relativ hohen Varianz. Die Gewichte der Verbindungen von Neuronen aus unterschiedlichen Gruppen (Inter-Gewichte) werden dagegen mit einer relativ geringen Varianz gesetzt. Auf diese Weise erhält man ein modular strukturiertes Netz. Die Modularität des Netzes kann mit den im Abschnitt 4.1 beschriebenen Methoden gemessen werden und hängt von der Gruppierung der Neurone und den Varianzen der Intra- und Inter-Gewichte ab. Aus einem "Vorbildnetz" werden dann Daten erzeugt, indem reelle Eingabewerte gleichverteilt zufällig aus dem Intervall (-1,1) gezogen und die korrespondierenden Ausgabewerte an den Ausgabeneuronen abgelesen werden. Diese Ausgaben werden mittels eines Schwellwertes von 0.5 zu binären Zielvektoren umgewandelt.

Die in den Experimenten betrachteten Vorbildnetze haben eine 12-12-12 Architektur, die in zwei gleich große Teile partitioniert wird. Ich betrachte sechs Kombinationen von Intra- und Intervarianz. Bei den ersten drei Kombinationen liegt die Varianz der Inter-Gewichte mindestens bei 1, und zwar in den Kombinationen 50:1, 50:5 und 5:1. In den drei anderen Kombinationen liegt die Inter-Varianz dagegen nur bei 0.1, und zwar in den Kombinationen 50:0.1, 5:0.1 und 1:0.1. Die Vorbildnetze sind in den Abbildungen 4.18 und 4.19 auf den Seiten 76 und 77 dargestellt. Die jeweilige Modularität der Vorbildnetze ist auch der Tabelle 4.5 auf Seite 78 zu entnehmen. Aus diesen Vorbildnetzen werden Datensätze mit jeweils 2000 Elementen generiert, die in eine je 1000-elementige Trainings- bzw. Test-Menge aufgeteilt werden. Die "Nachahmernetze" haben wie die Vorbildnetze eine 12-12-12 Architektur.

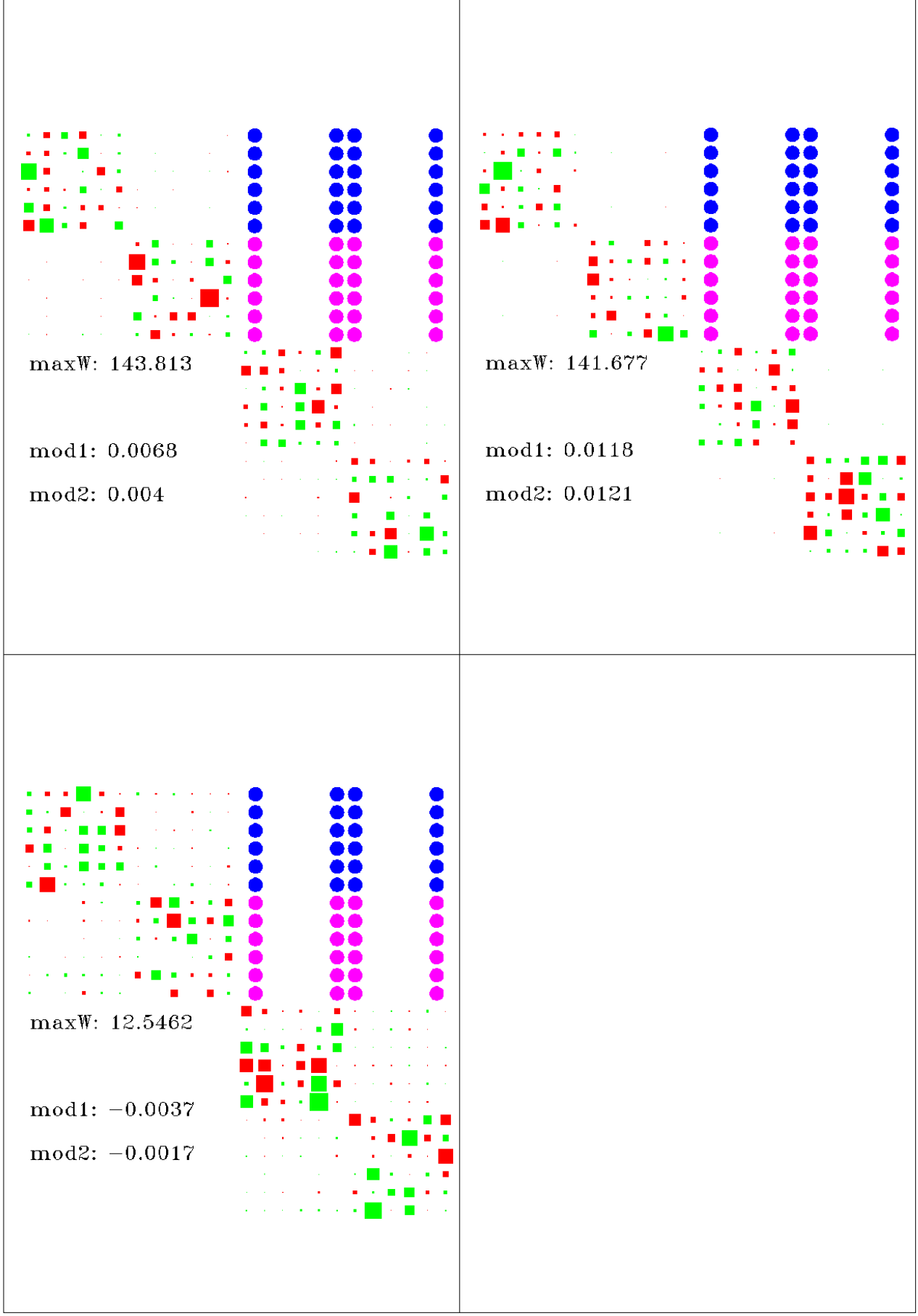

Abbildung 4.18: Die 3 Vorbildnetze mit hoher Inter-Varianz. Links oben 50:5, rechts oben 50:1, unten 5:1.

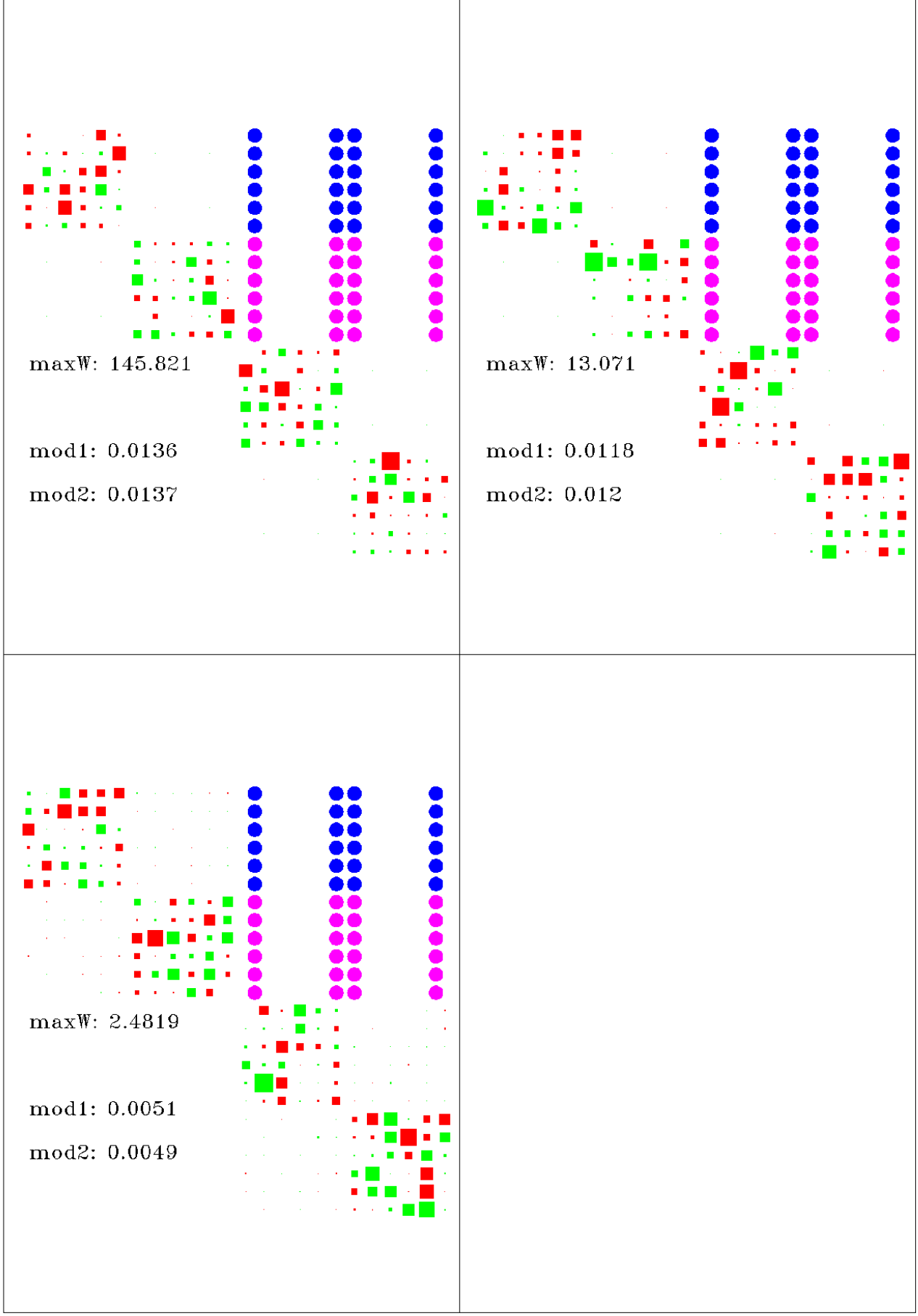

Abbildung 4.19: Die 3 Vorbildnetze mit niedriger Inter-Varianz. Links oben 50:0.1, rechts oben 5:0.1, unten 1:0.1.

Intra:Inter	50:5	50:1	5:1	50:0.1	5:0.1	1:0.1
Modularität 1. Schicht $\cdot 10^4$	68	118	-37	136	118	51
Modularität 2. Schicht $\cdot 10^4$	40	121	-17	137	120	49

Tabelle 4.5: Modularität der verwendeten Vorbildnetze. Erläuterungen im Text.

	50:5	50:1	5:1
RMS-Anfang	Sp—So,So4—Un	Sp—So—So4—Un	So4—So—Un——Sp
RMS-Ende	So—Un—So4——Sp	So,Un—So4—Sp	So4,So,Un——Sp
tRMS-Anfang	Sp—So,So4—Un	Sp—So—So4—Un	So4—So—Un——Sp
tRMS-Ende	So—So4—Un——Sp	So,So4—Un—Sp	So4——So,Un——Sp

Tabelle 4.6: Reihenfolge der Vorstrukturierungs-Arten hinsichtlich der erreichten Fehler. Erläuterungen im Text.

4.3.2 Lernen in modular vorstrukturierten Netzen

Fragestellung und Aufbau

Wie beim Modularen Encoder Problem interessiere ich mich für die Unterschiede im Lernverhalten von modular vorstrukturierten und nicht vorstrukturierten Netzen. Ich verwende wieder die gleichen Arten der Vorstrukturierung wie beim Modularen Encoder, also vollständig getrennte Netze, Softsplit-Netze mit modularen Anfangs-Gewichten und Softsplit-Netze mit zusätzlichem Weight Decay der Stärke $\alpha = 10^{-4}$. Das Training wird mit der Lernrate $\eta = 0.8$ durchgeführt und die Initialisierung der Gewichte erfolgt gleichverteilt zufällig aus dem Intervall(-1,1). Anders als beim modularen Encoder wird das Training 10 mal wiederholt und 500 Epochen lang durchgeführt.

Als Kennzahlen werden während des Trainings der Trainings- und der Test-Fehler, sowie die Modularitäten und das Struktur-Maß aufgenommen. Auf eine Einbeziehung der binarisierten Fehler in die Darstellung wird verzichtet, da, wie sich zeigt, keiner der Läufe einen binarisierten Trainings- oder Test-Fehler von 0 erreicht.

Ergebnisse

Die Ergebnisse der Simulationen sind in den Abbildungen 4.21 bis 4.25 auf den Seiten 81 bis 85 dargestellt. In den Tabellen 4.6 und 4.6 auf Seite 78 stelle ich die Ergebnisse noch einmal verkürzt dar, indem ich für jedes Vorbildnetz nur die Reihenfolge der verschiedenen Vorstrukturierungs-Arten in Bezug auf die erreichten Fehler aufführe, und zwar getrennt die Reihenfolge am Anfang und die Reihenfolge am Ende. Dabei bedeutet '—' einen deutlichen, '——' einen sehr deutlichen und ',' einen geringen Abstand.

	50:0.1	5:0.1	1:0.1
RMS-Anfang	Sp—So,So4—Un	Sp—So,So4—Un	So,So4—Un——Sp
RMS-Ende	Sp—So,So4—Un	Sp—So,So4—Un	So—Un—So4——Sp
tRMS-Anfang	Sp—So,So4—Un	Sp—So,So4—Un	So4—So——Un—Sp
tRMS-Ende	Sp—So4—So—Un	Sp——So4—So—Un	So4—So—Un,Sp

Tabelle 4.7: Reihenfolge der Vorstrukturierungs-Arten hinsichtlich der erreichten Fehler. Erläuterungen im Text.

Die Ergebnisse lassen sich folgendermaßen zusammenfassen:

- Geteilte Netze haben bei allen Vorbildern mit *hoher* Inter-Varianz am Ende höhere Trainings- und Test-Fehler als ungeteilte Netze (sie "lernen schlechter"). Dagegen lernen bei den Vorbildern mit *geringer* Inter-Varianz geteilte Netze wesentlich besser als ungeteilte, allerdings mit der deutlichen Ausnahme von Vorbild 1:0.1.

- **Softsplit-Netze mit Weight-Decay lernen überall besser als ungeteilte Netze, auch dort wo die geteilten Netze schlechter lernen!** Besonders deutlich ist dies bei den Vorbildern 5:1 und 1:0.1.

- Softsplit-Netze ohne Weight-Decay und ungeteilte Netze lernen oft ähnlich gut, wobei die Softsplit-Netze aber immer etwas besser liegen. Die Softsplit-Netze ohne Weight-Decay lernen immer schlechter als die Softsplit-Netze mit Weight-Decay, mit Ausnahme von Vorbild 50:5.

- Softsplit-Netze ohne Weight Decay erreichen am Ende oft nur die Modularität von ungeteilten Netzen. Softsplit-Netze mit Weight-Decay erreichen dagegen deutlich höhere Modularität.

- Die Softsplit-Netze lernen auch am Anfang immer besser als ungeteilte Netze! Am besten lernt aber am Anfang noch das geteilte Netz, mit Ausnahme der Vorbilder 5:1 und 1:0.1: dort lernt das geteilte Netz von Anfang an sehr schlecht. Softsplit-Netze ohne Weight Decay lernen am Anfang besser als jene mit Weight Decay, mit Ausnahme der Vorbilder 5:1 und 1:0.1: dort ist Softsplit mit Weight Decay überlegen.

In einem Satz zusammengefaßt bedeuten diese Ergebnisse: Geteilte Netze sind in den geschilderten Versuchen unterlegen, wenn die Modularität des Vorbilds nicht glasklar ist. Softsplit mit Weight Decay ist dann die Methode der Wahl. Sie ist bei allen Vorbildern den ungeteilten Netzen sowohl am Anfang als auch am Ende des Trainings überlegen.

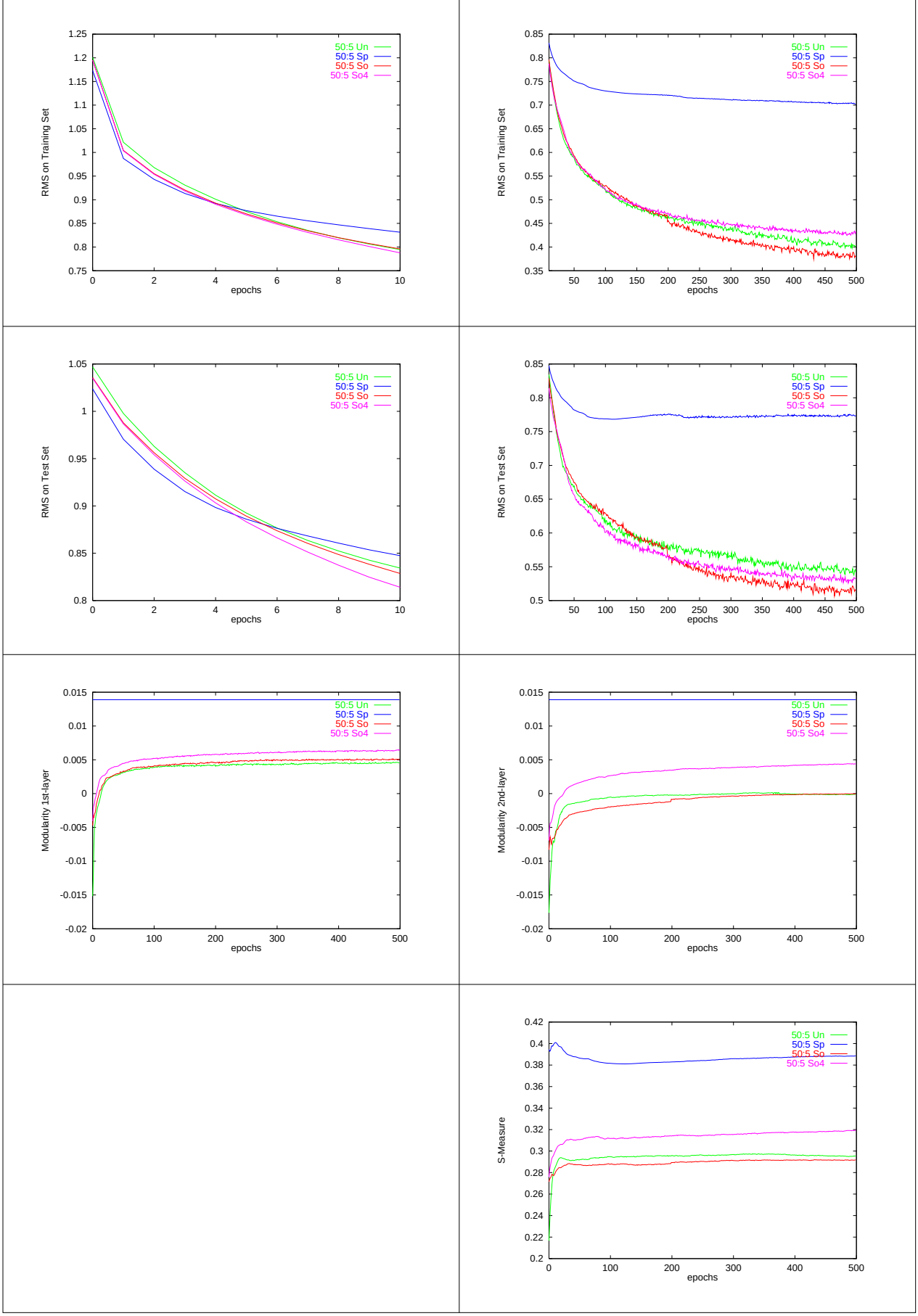

Abbildung 4.20: Vorbildnetz 50:5. Entwicklung von Fehler und Modularität während des Trainings. Vergleich modulare Vorstrukturierung - keine Vorstrukturierung. Erläuterungen im Text.

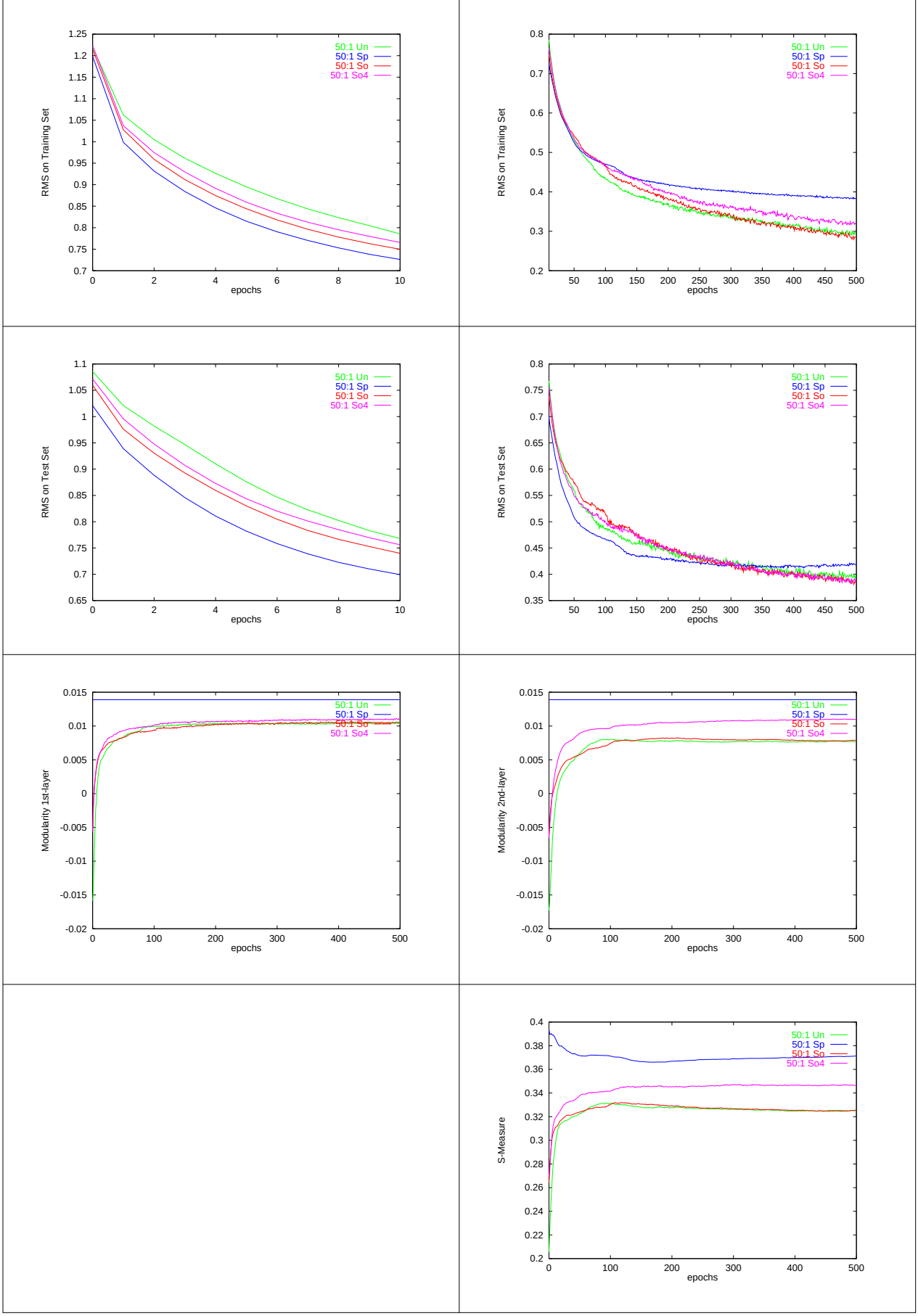

Abbildung 4.21: Vorbildnetz 50:1. Entwicklung von Fehler und Modularität während des Trainings. Vergleich modulare Vorstrukturierung - keine Vorstrukturierung. Erläuterungen im Text.

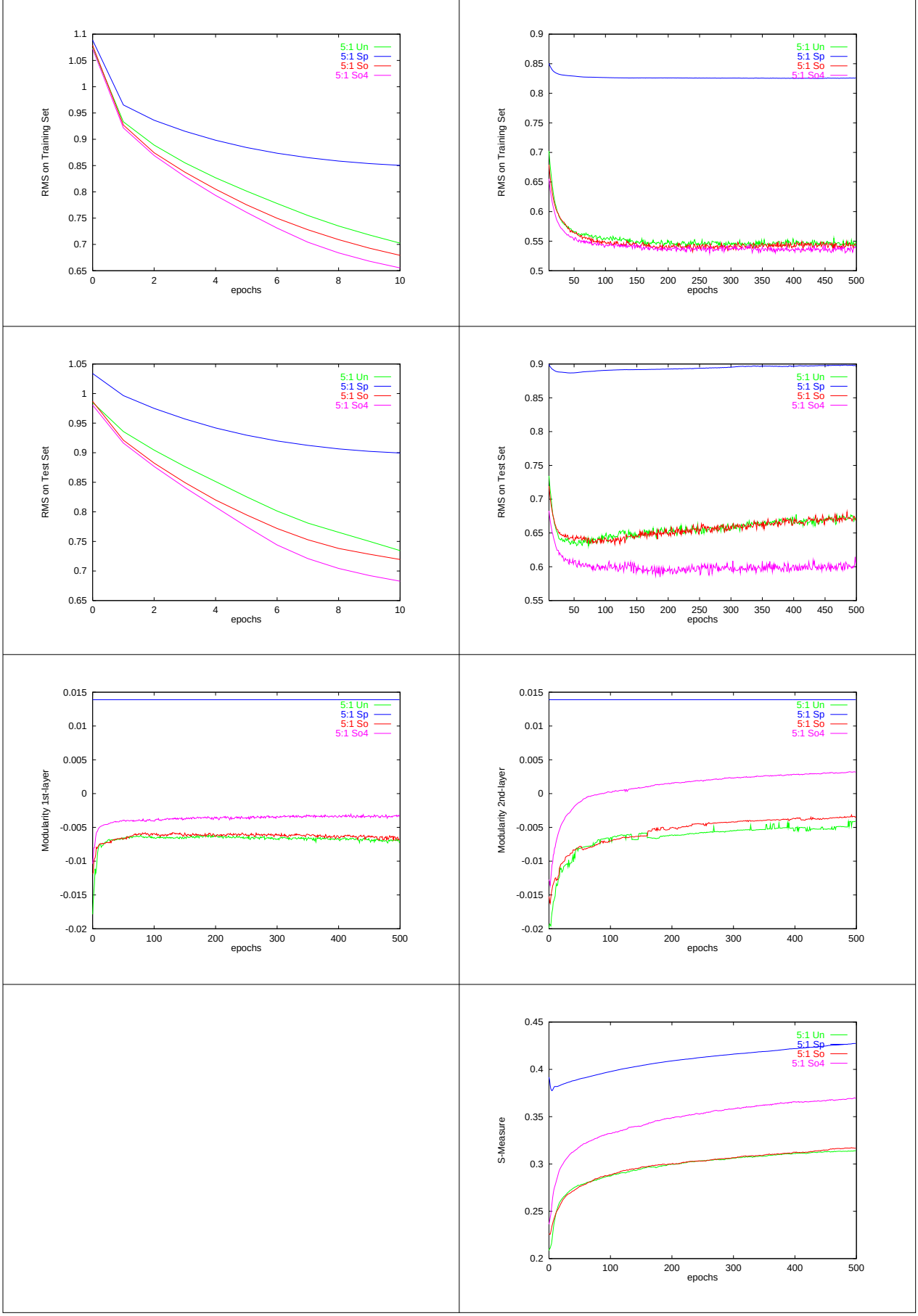

Abbildung 4.22: Vorbildnetz 5:1. Entwicklung von Fehler und Modularität während des Trainings. Vergleich modulare Vorstrukturierung - keine Vorstrukturierung. Erläuterungen im Text.

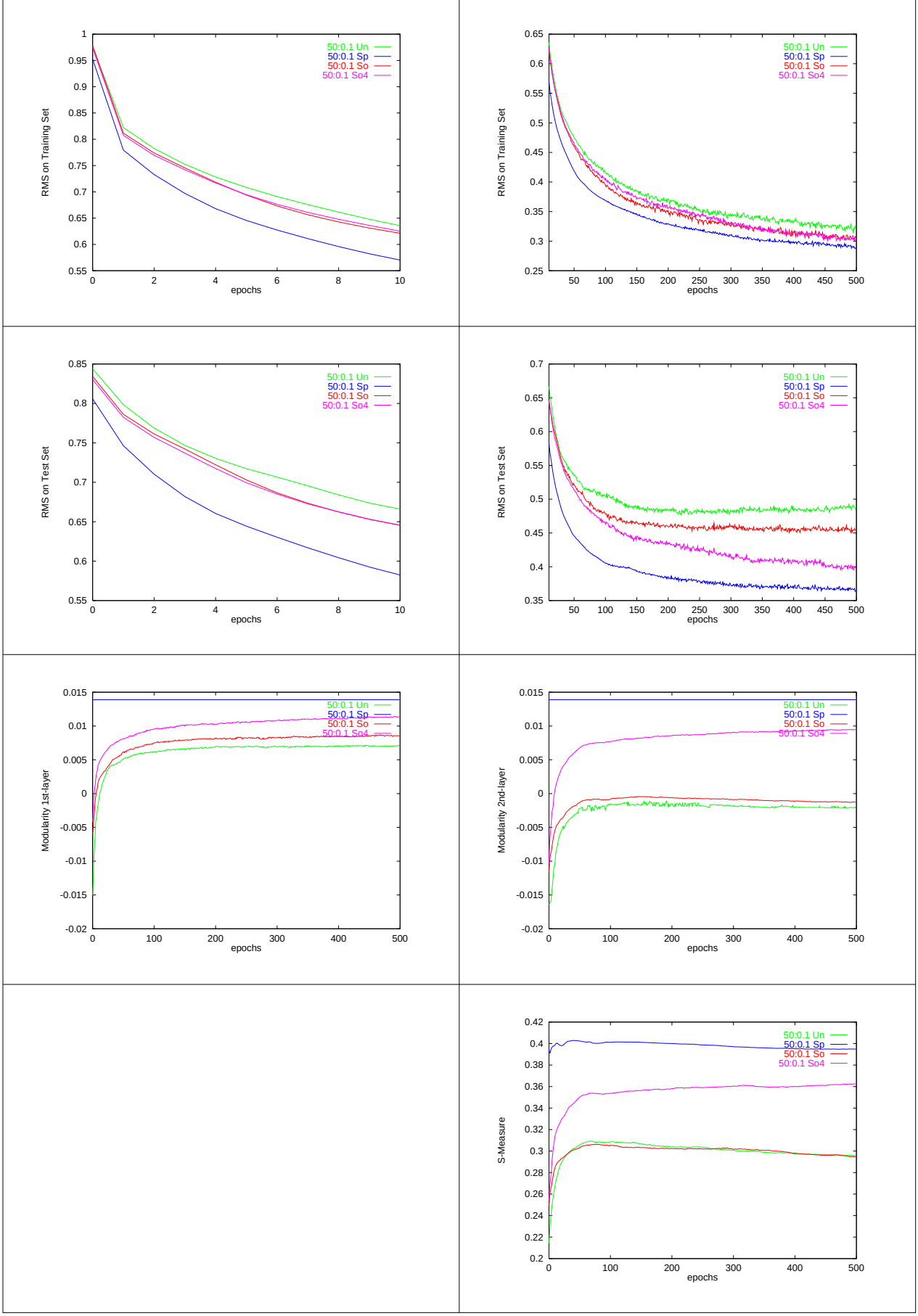

Abbildung 4.23: Vorbildnetz 50:0.1. Entwicklung von Fehler und Modularität während des Trainings. Vergleich modulare Vorstrukturierung - keine Vorstrukturierung. Erläuterungen im Text.

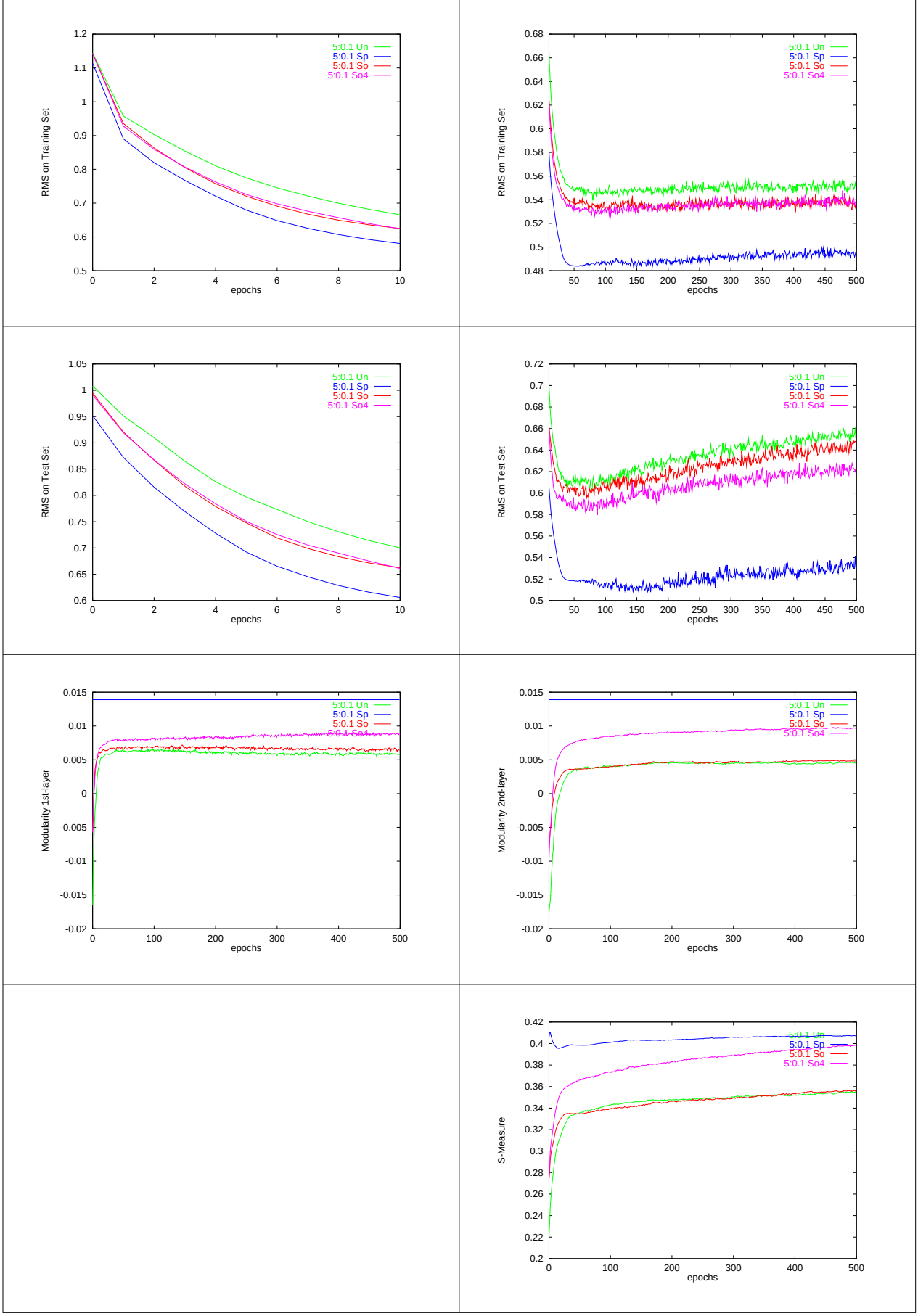

Abbildung 4.24: Vorbildnetz 5:0.1. Entwicklung von Fehler und Modularität während des Trainings. Vergleich modulare Vorstrukturierung - keine Vorstrukturierung. Erläuterungen im Text.

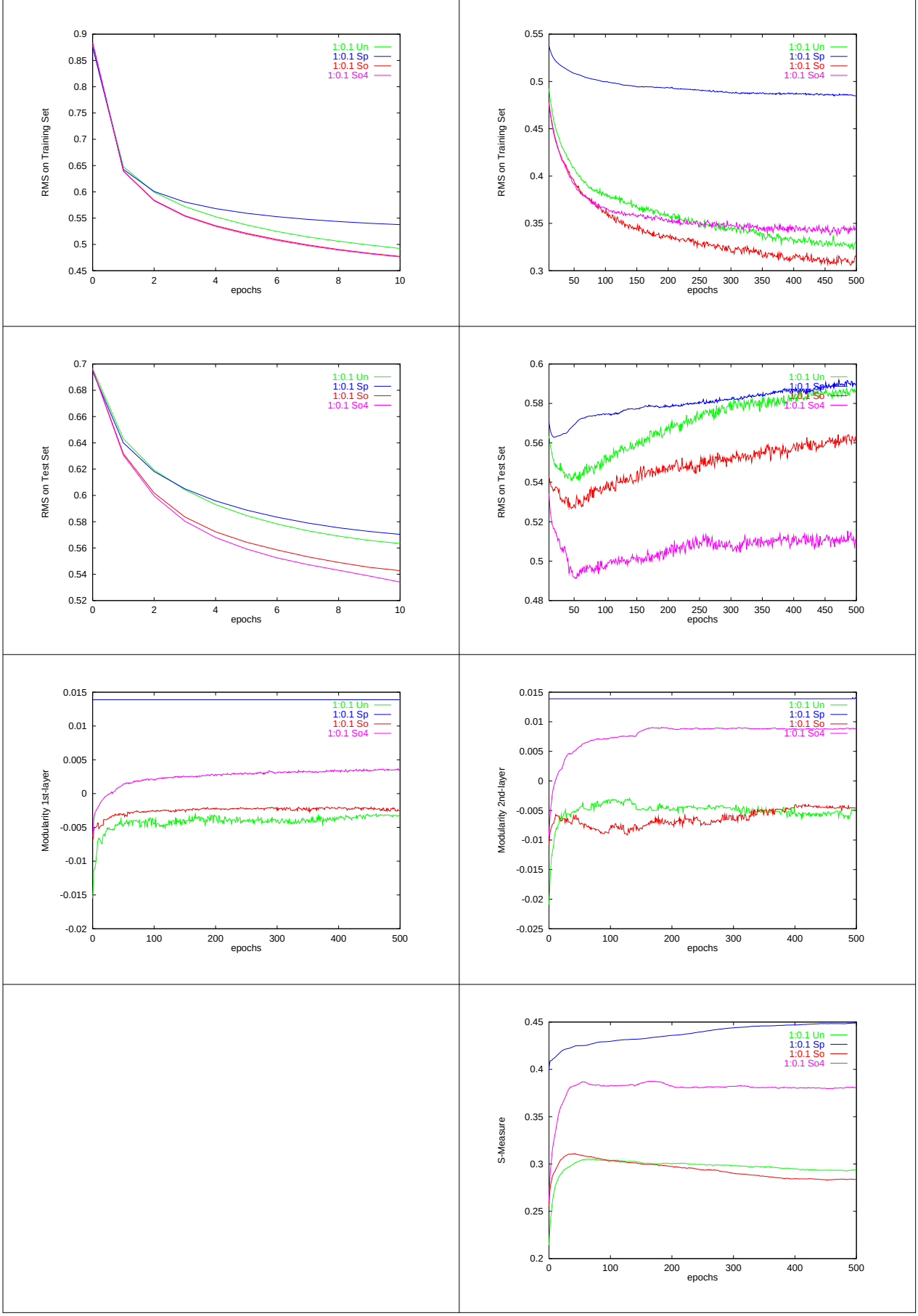

Abbildung 4.25: Vorbildnetz 1:0.1. Entwicklung von Fehler und Modularität während des Trainings. Vergleich modulare Vorstrukturierung - keine Vorstrukturierung. Erläuterungen im Text.

4.3.3 Auffinden der optimalen modularen Vorstrukturierung

In diesem Abschnitt untersuche ich, ob sich mit Hilfe des in Abschnitt 4.2.4 verwendeten evolutionären Verfahrens auch bei den Vorbildnetz-Datensätzen die modulare Struktur der Aufgabe wiederfinden läßt. Ich untersuche dazu als Extrem-Beispiele das am stärksten modulare Vorbildnetz 50:0.1 und das am wenigsten modulare und mit hoher Inter-Varianz versehene Vorbildnetz 5:1.

In der ersten Versuchsreihe verwende ich den Trainings-Fehler nach nur 2 Epochen als Performanz-Kriterium. Die optimale Partition ist damit jene, die zum steilsten Abfall des Fehlers auf der Beispiel-Menge am Anfang des Lernens führt. Der Test-Fehler wird nicht betrachtet. Als Vorstrukturierungs-Art wird entweder modulare Initialisierung (Softsplit) mit zusätzlichem (mit-evoluierendem) Weight Decay oder eine vollständige Aufteilung (Split) verwendet. Die vorstrukturierten Netze werden jeweils aus 10 festen Initialisierungen mit Backpropagation (Lernrate $\eta = 0.8$) trainiert.
In der zweiten Versuchsreihe verwende ich als Performanz-Kriterium den Test-Fehler, und zwar erst nach 50 Epochen. Ich verwende verkleinerte Datensätze, nämlich je 50-elementige Stichproben der Trainings- und Testmengen aus den Vorbildnetzen 50:0.1 und 5:1. Außerdem haben die Nachahmernetze jetzt 18 innere Neuronen, also 6 mehr als ihre Vorbilder. Es werden nur noch 3 Wiederholungen aus unterschiedlichen Initialisierungen pro Fitneß-Bewertung durchgeführt.

Ergebnisse

In Abbildung 4.26 auf Seite 88 sind die Entwicklung der Fitneß und des Decay Faktors des besten Individuums in den Evolutionen der ersten Versuchsreihe dargestellt. Die oberen zwei Zeilen beziehen sich auf die Evolution bei Verwendung von Vorbild 50:0.1, und zwar die erste Zeile bei einer Softsplit-Vorstrukturierung und die zweite Zeile bei einer Split-Vorstrukturierung, wo also kein Weight Decay verwendet wird. Die unteren zwei Zeilen zeigen entsprechend die Evolutionsverläufe bei Verwendung von Vorbild 5:1. Man erkennt, daß bei beiden Vorbildern die Split-Variante nach nur wenigen Generationen auf einem im Vergleich zur Softsplit-Variante relativ schlechten Fitneß-Wert verharrt. Nicht dargestellt ist, daß es sich bei der erreichten Partition in beiden Fällen um die "triviale Partition" handelt, in der alle Neuronen der gleichen Gruppe angehören, also keine Aufteilung stattfindet!
In den Softsplit-Varianten evoluiert bei beiden Vorbildern am Ende ein recht geringer Weight Decay mit $\alpha = 10^{-7}$ (Vorbild 50:0.1) bzw. $\alpha = 10^{-6}$ (Vorbild 5:1). Die Fitneß-Kurven haben eine typische Form mit im Laufe der Evolution länger werdenden Plateaus, die durch kurze Übergangsstufen voneinander getrennt sind. Die absoluten Differenzen der Fitneßwerte zwischen den Stufen sind sehr gering, sie sind etwa in der 3. bis 4. Nachkomma-Stelle festzustellen. Da das Vorbild 50:0.1, wie aus den Ergebnissen des letzten Abschnitts abzulesen ist, eine leichtere Lernaufgabe als das Vorbild 5:1 darstellt, ist die erreichte Fitneß hier entsprechend größer.

Die Abbildungen 4.27 und 4.28 auf Seite 89 und Seite 90 zeigen einige Stationen der Evolution bei den Softsplit Varianten. In beiden Fällen entspricht die am Ende erreichte Aufteilung nicht genau der Aufteilung der Vorbilder in zwei Hälften. Im Falle des Vorbilds 50:0.1 zerfällt die untere Hälfte noch einmal in zwei Gruppen und die Aufteilung der inneren Neuronen auf die beiden Hälften ist nicht ganz gleichmäßig (7:5, bzw. 7:3:2). Die Zwischen-Station in Generation 47 zeigt eine "bessere" Aufteilung bei einem höheren

Weight Decay, ist aber in ihrer Fitneß suboptimal. In der bei Vorbild 5:1 erreichten Partition zerfällt ebenfalls die untere Hälfte noch einmal in zwei Gruppen. Außerdem ist ein äußeres Neuron der oberen Hälfte "fälschlich" einer der unteren Gruppen zugeschlagen. Die Zuordnung der inneren Neuronen zu den Gruppen ist ebenfalls ungleichmäßig (4:8, bzw. 4:7:1). Anders als bei Vorbild 50:1 zerfallen die Zwischen-Stationen in den früheren Generationen bei schwachem Weight Decay eher in noch mehr Gruppen.

In Abbildung 4.29 auf Seite 91 sind Entwicklung von Fitneß und Weight Decay für die Evolutionen der zweiten Versuchsreihe (reduzierter Datensatz, Testfehler nach 50 Epochen, zusätzliche innere Neuronen) dargestellt. Der wichtigste Unterschied zur vorhergehenden Versuchsreihe ist zunächst, daß auch bei Split-Vorstrukturierung andere als die triviale Partition evoluieren. Die dabei erreichten Fitneß-Werte sind jenen, die bei Softsplit-Strukturierung erreicht werden, vergleichbar. Die Evolution verläuft zudem bei Split-Vorstrukturierung um eine Größenordnung schneller als bei Softsplit. Die bei Vorbild 50:0.1 erreichten Fitneß-Werte sind wie in der ersten Versuchsreihe höher als bei Vorbild 5:1, was die unterschiedliche Schwierigkeit der Aufgabe wiederspiegelt. Die Evolution des Weight Decay verläuft bei Vorbild 50:0.1 so, daß sehr schnell der maximal mögliche Decay Faktor von $\alpha = 10^{-2}$ erreicht wird, der Softsplit also sehr stark zum Split hin tendiert. Im Gegensatz dazu ist ein Anstieg in der Entwicklung des Decays bei Vorbild 5:1 nur in der ersten ca. 300 Generationen zu verzeichnen. Danach macht die Evolution einen großen Sprung und der Weight Decay sinkt wieder und ist im Folgenden auf niedrigerem Niveau stark schwankend.

In den Abbildungen 4.30 bis 4.33 sind für jede der vier Evolutionen jeweils vier markante Stationen dargestellt. In allen vier Evolutionen wird ungefähr die modulare Struktur der Vorbilder wiedergefunden. Zudem wird auch ungefähr die Komplexität der Vorbilder respektiert. Die innere Schicht besteht in dieser Versuchsreihe aus 18 Neuronen, also 6 Neuronen mehr als im Vorbild, und einige dieser Neuronen werden nun nicht zu äußeren Neuronen gruppiert, so daß sie unverbunden sind (Split), bzw. ihre Verbindungen nach außen einem Weight Decay unterliegen(Softsplit). Bei Vorbild 50:0.1 sind es 9 (Softsplit), bzw. 10 (Split) innere Neuronen, die so isoliert sind, bei Vorbild 5:1 dagegen nur 5 (Softsplit), bzw. 4 (Split). Die Einteilung der äußeren Neuronen ist in allen vier Evolutionen leicht fehlerhaft, beim gleichen Vorbild werden aber jeweils fast die gleichen Fehler gemacht, was vermuten läßt, daß einige dieser "Fehler" eher eine Eigenschaft der Datensätze sind, als Ausdruck einer unzureichenden Evolution. Isolierte innere Neuronen treten bei Vorbild 50:0.1 deutlich früher im Evolutionsverlauf auf als bei Vorbild 5:1. Der Sprung in der Fitneß bei Vorbild 5:1, Softsplit, geht nicht nur wie oben beschrieben mit einer Abschwächung des Weight Decays einher, sondern auch mit dem Auftreten isolierter innerer Neuronen.

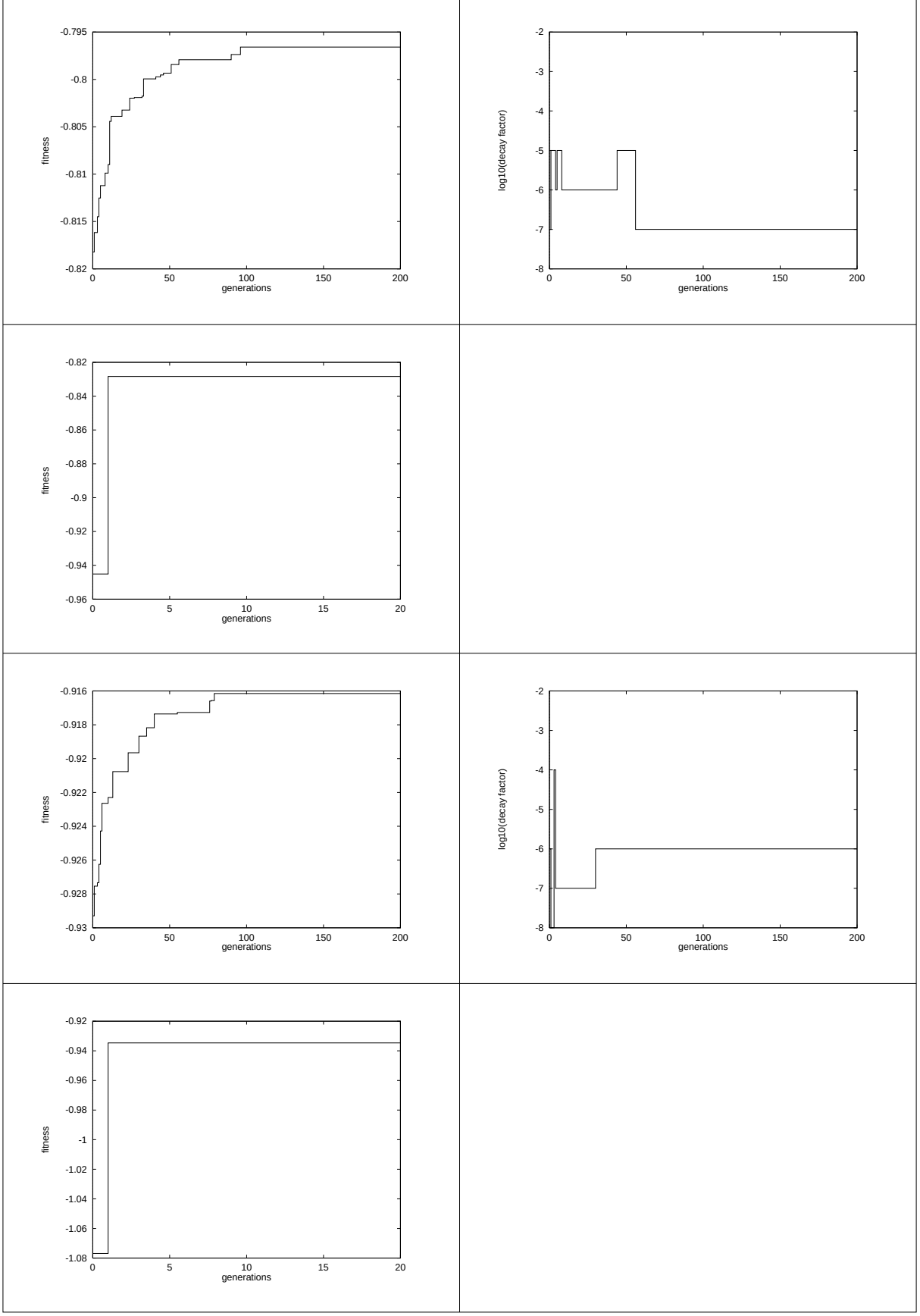

Abbildung 4.26: Entwicklung der Fitneß und des Decay Faktors des besten Individuums über die Generationen. Fitneß ist der negative Trainingsfehler nach zwei Epochen. Obere Hälfte: Vorbild 50:0.1. Erste Zeile: Softsplit (links Fitneß rechts Decay Faktor). Zweite Zeile: Split (Fitneß). Untere Hälfte: Vorbild 5:1.

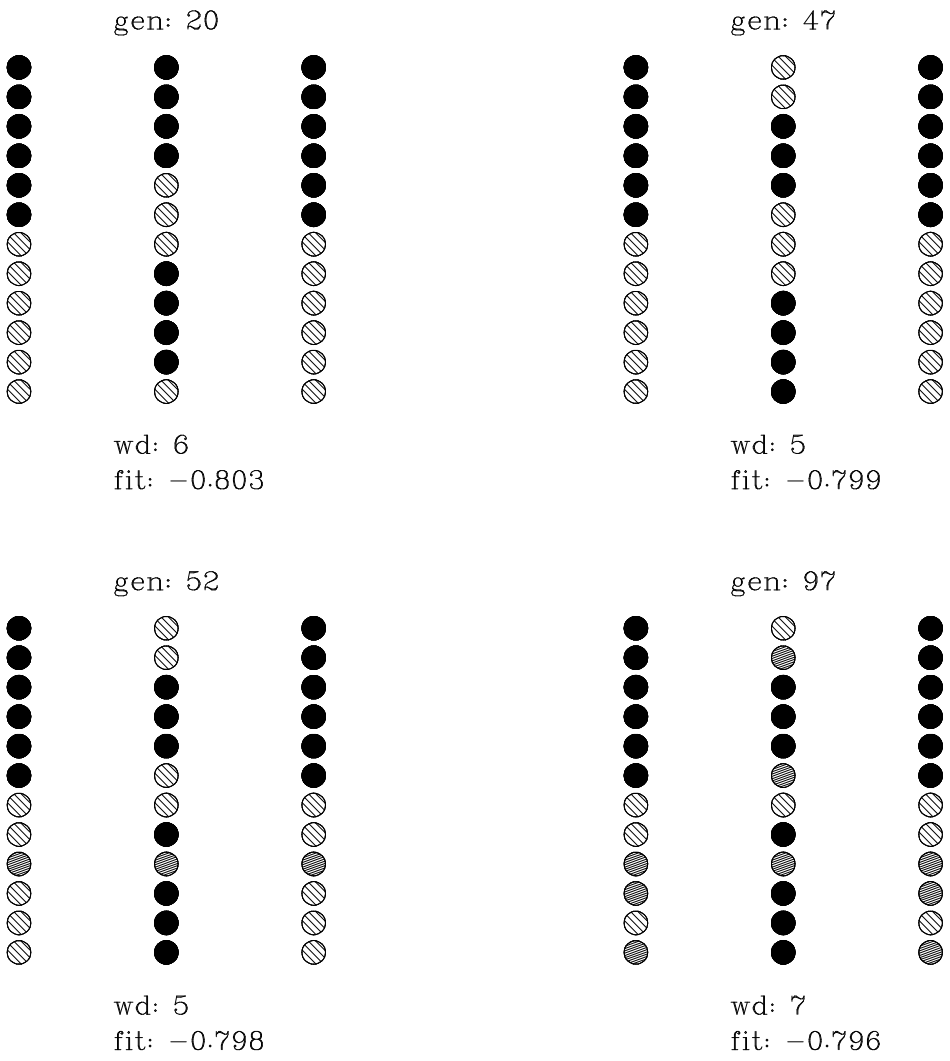

Abbildung 4.27: Stationen der Evolution. Es ist jeweils das beste Individuum der Population in der angegebenen Generation dargestellt. “wd” gibt den negativen 10er-Logarithmus des Decay-Faktors an. Vorbild 50:0.1. Fitneß: neg. Trainingsfehler nach 2 Epochen.

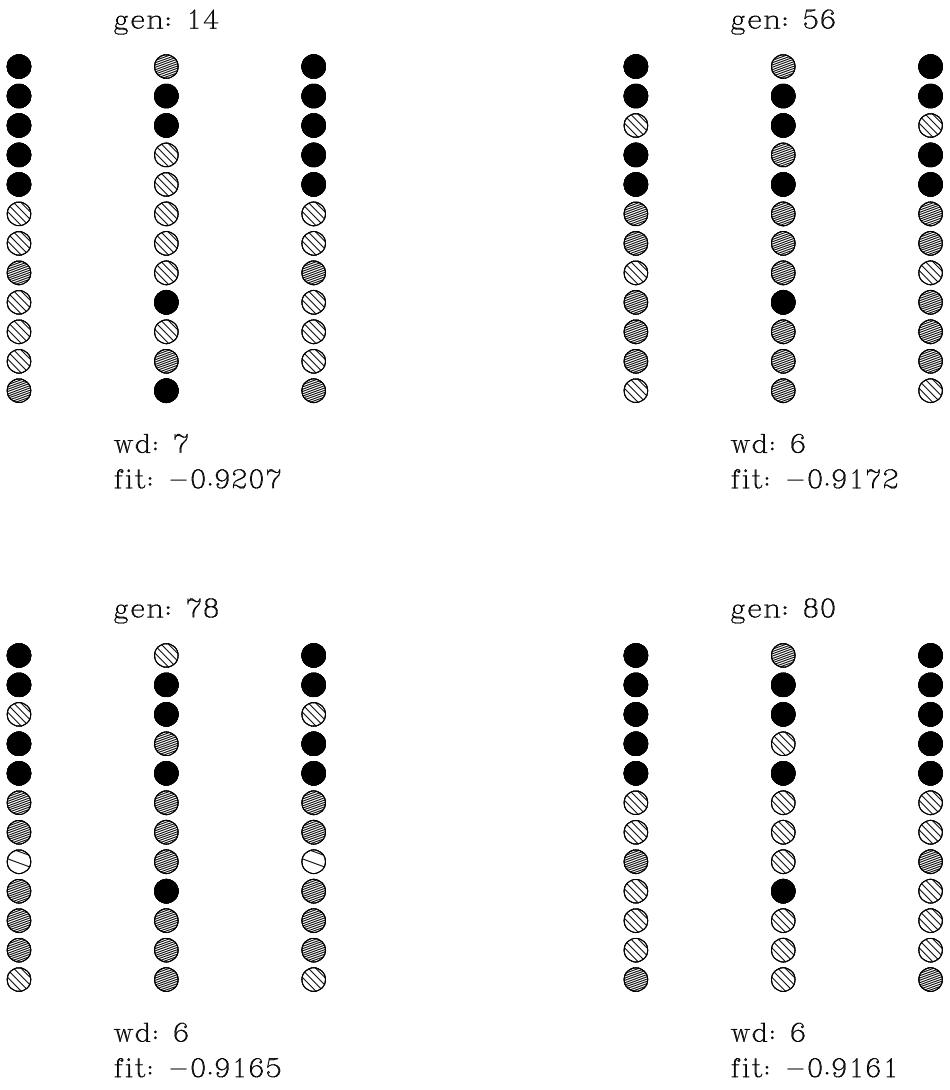

Abbildung 4.28: Stationen der Evolution. Vorbild 5:1, Fitneß: neg. Trainingsfehler nach 2 Epochen. Softsplit.

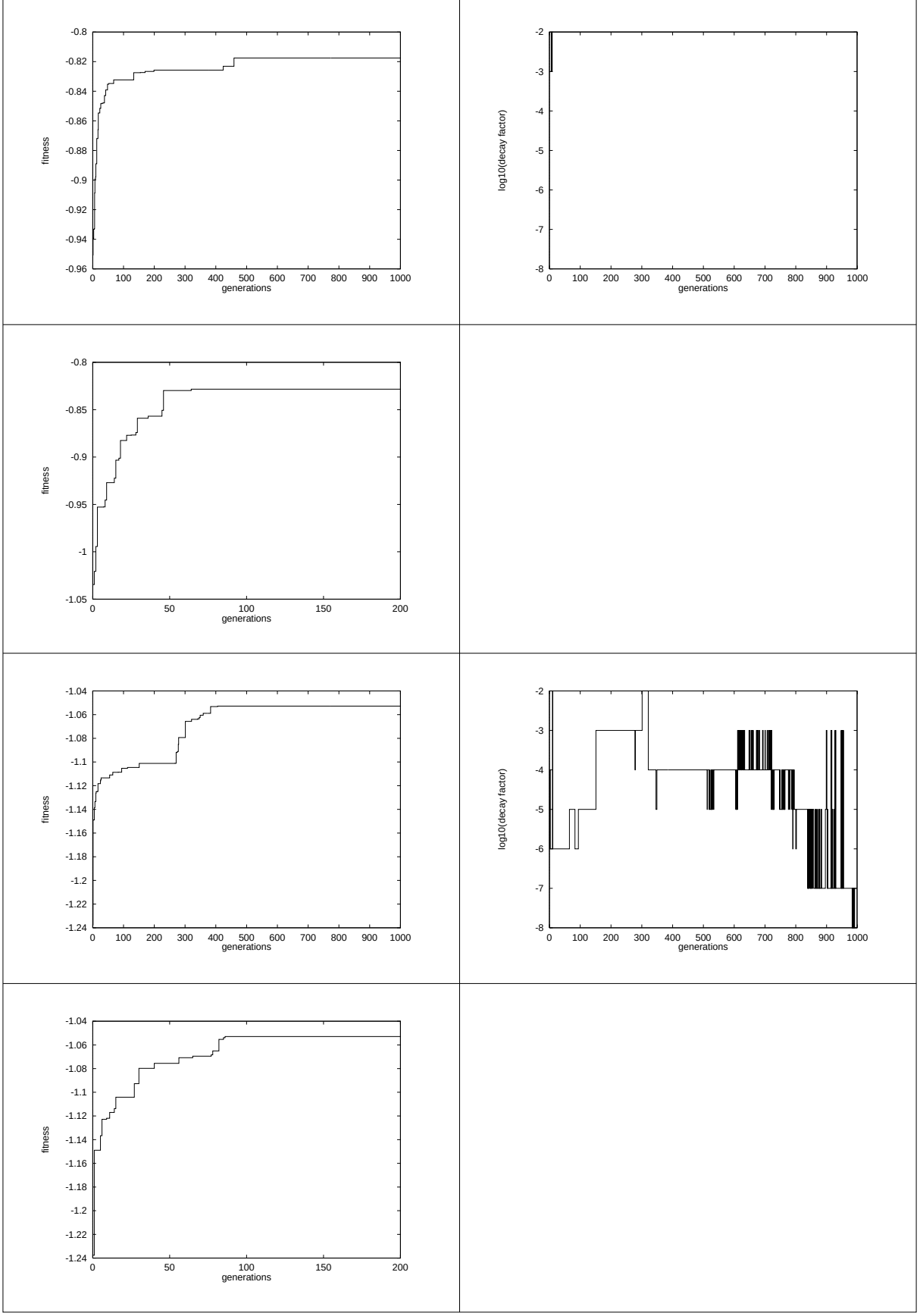

Abbildung 4.29: Entwicklung der Fitneß und des Decay Faktors des besten Individuums über die Generationen. Fitneß ist der negative Testfehler nach 50 Epochen, reduzierter Datensatz. Obere Hälfte: Vorbild 50:0.1. Erste Zeile: Softsplit (links Fitneß rechts Decay Faktor). Zweite Zeile: Split (Fitneß). Untere Hälfte: Vorbild 5:1.

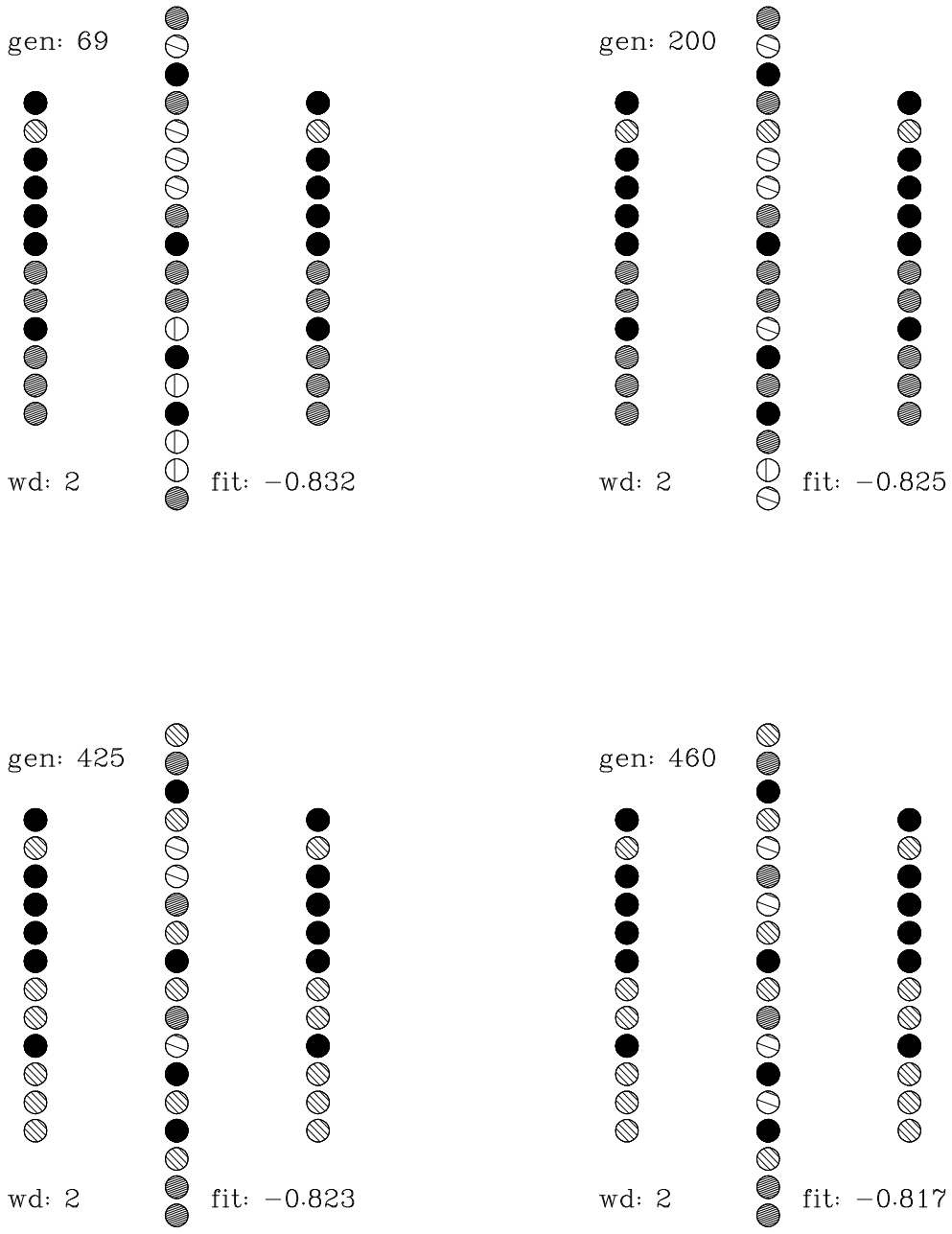

Abbildung 4.30: Stationen der Evolution. Vorbild 50:0.1, reduzierter Datensatz. Fitneß: negativer Test-Fehler nach 50 Epochen. Softsplit.

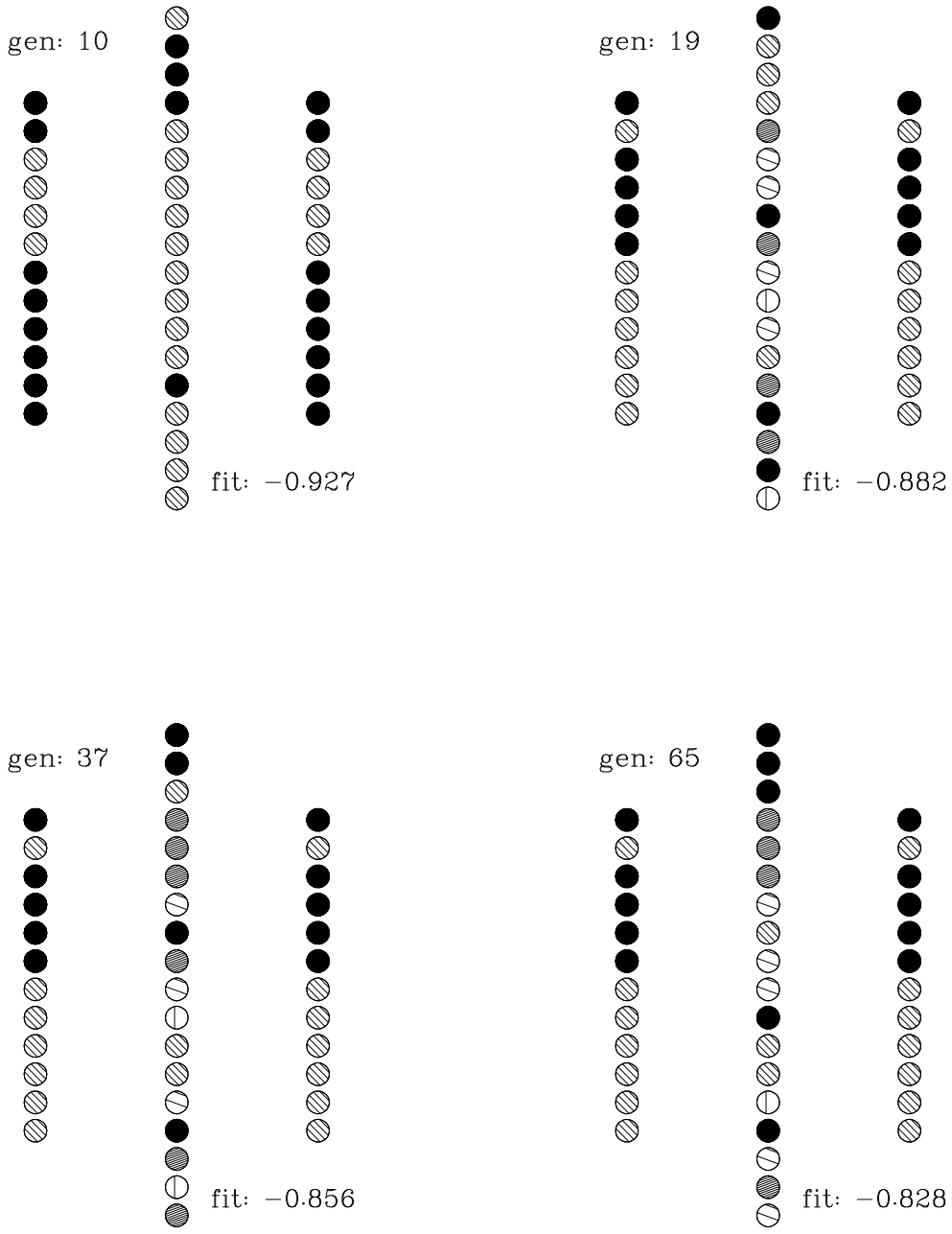

Abbildung 4.31: Stationen der Evolution. Vorbild 50:0.1, reduzierter Datensatz. Fitneß: negativer Test-Fehler nach 50 Epochen. Split.

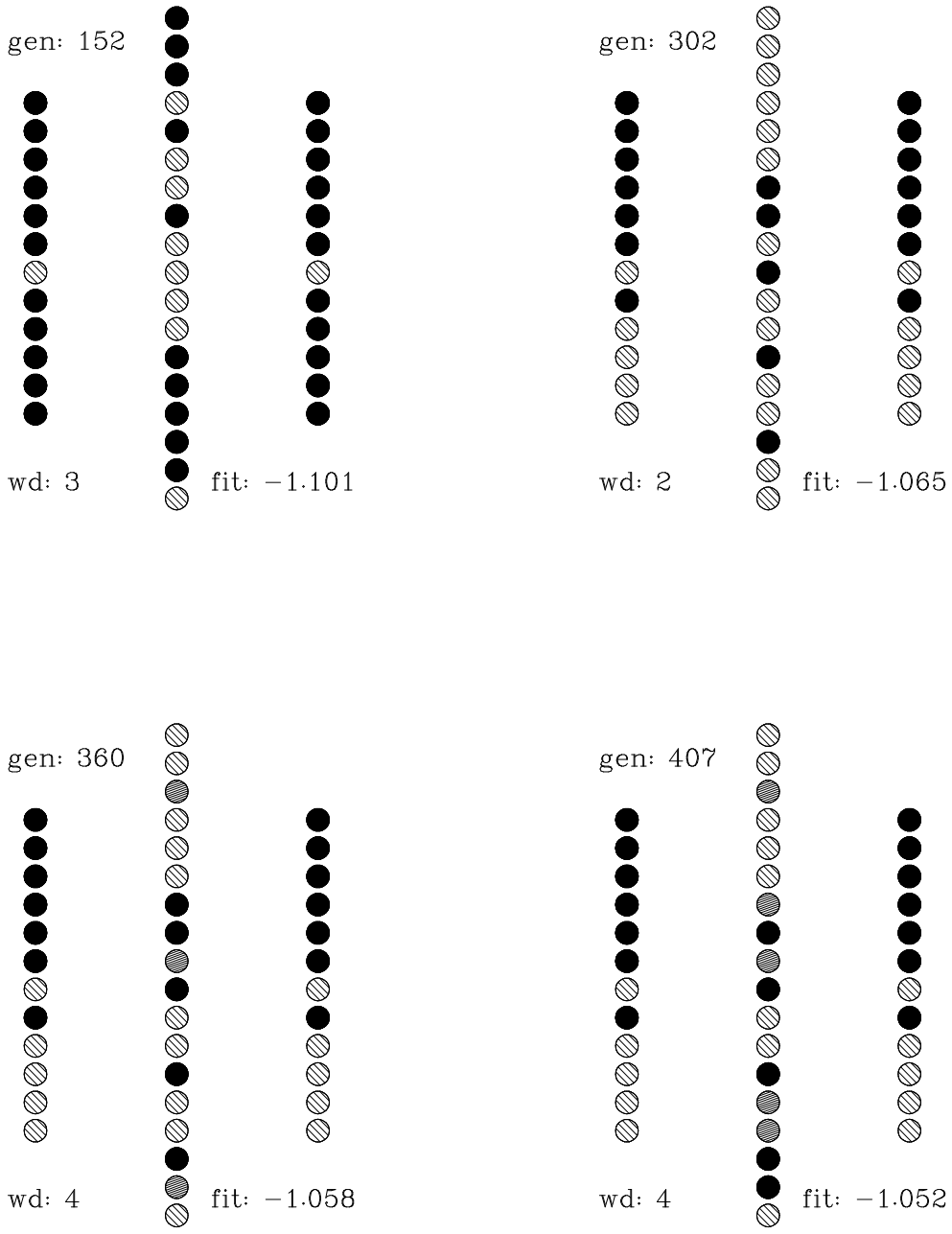

Abbildung 4.32: Stationen der Evolution. Vorbild 5:1, reduzierter Datensatz. Fitneß: negativer Test-Fehler nach 50 Epochen. Softplit.

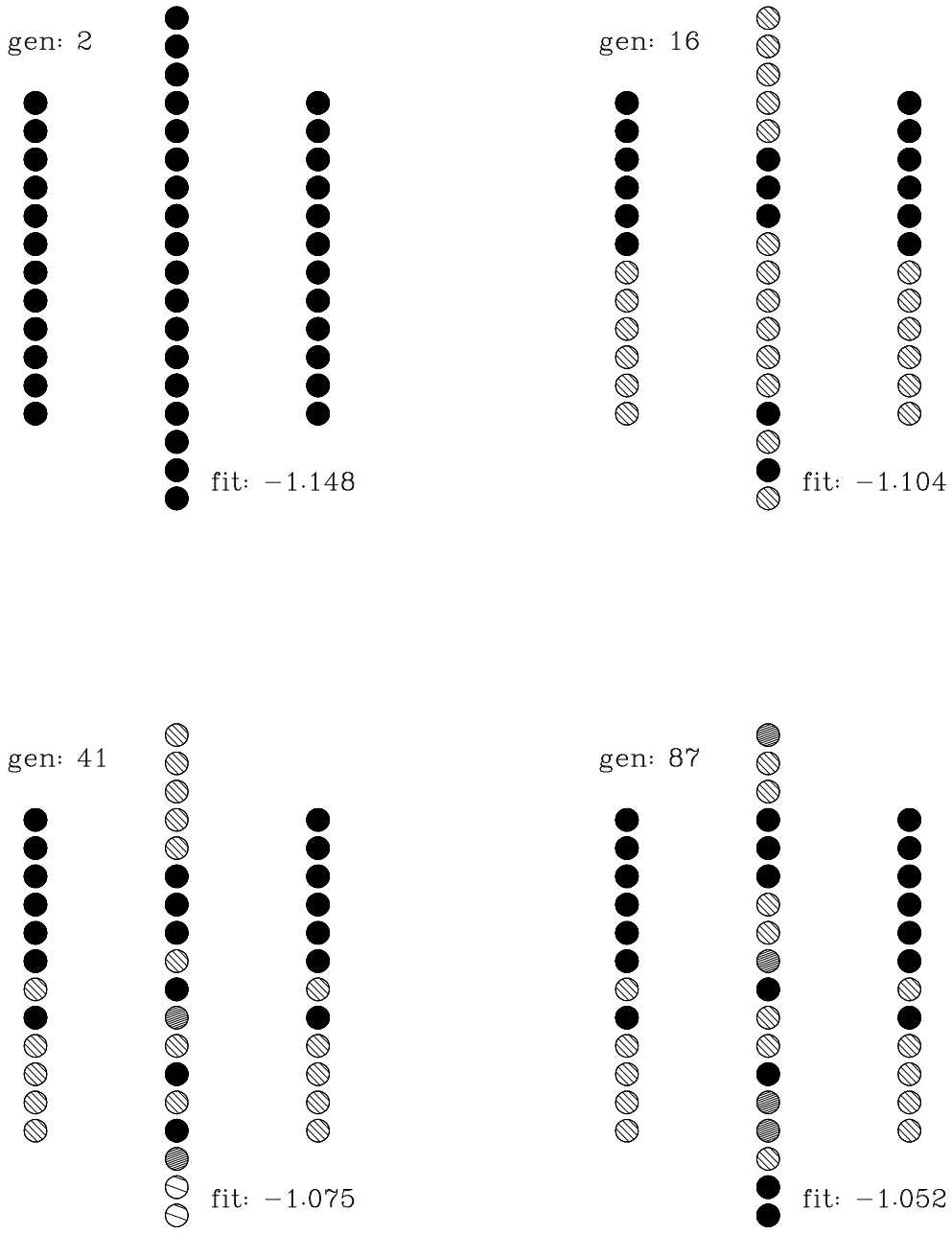

Abbildung 4.33: Stationen der Evolution. Vorbild 5:1, reduzierter Datensatz. Fitneß: negativer Test-Fehler nach 50 Epochen. Split.

Kapitel 5

Zusammenfassung und Diskussion

5.1 Hauptaussagen: Modularität und Lernen

Im ersten Teil der Arbeit wurde der Kern des Modularitäts-Begriffs herausgearbeitet: Modularität wurde definiert als die Eigenschaft eines in Komponenten teilbaren Systems, bei hohem Intra-Zusammenhang einen niedrigen Inter-Zusammenhang zu besitzen. Zu dieser notwendigen Bedingung können dann weitere charakteristische Eigenschaften hinzutreten, wie funktionale Differenziertheit, Repetitivität oder hierarchische Anordnung der Module. Am Ende des ersten Teils wurde aufgezeigt, daß Multilagen-Perzeptrone auf zwei Weisen in Module zerfallen können, nämlich durch Aufteilung des Datenraumes entweder in Regionen oder in Unterräume.

Im experimentellen Teil der Arbeit wurden dann der Einfluß einer unterraum-modularen Netzstruktur auf die Lern-Performanz eines Multilagen-Perzeptrons bei bestimmten modularen Test-Daten bestimmt, sowie Ansätze für eine automatische Modularisierung von Netzstrukturen vorgestellt. Dabei sind mit der Modularen Encoder Aufgabe und den Vorbildnetz-Daten zwei unterschiedliche Arten von modularen Daten untersucht worden. Bei der klar modularen Encoder-Aufgabe und auch bei stark modularen Vorbildnetz-Daten lernen Netze, die durch Wegnahme von Verbindungen entsprechend der Struktur der Daten in Module aufgeteilt sind ("Split"), besser als vollverbundene Netze. Bei weniger stark modularen Vorbildnetz-Daten ist eine derartige Aufteilung zu drastisch, aber eine weiche Art der Netzstrukturierung durch modulare Anfangsgewichte und intermodularen Weight Decay ("Softsplit") ist immer noch in der Lage, Vorwissen über die Datenmodularität zur Verbesserung des Lernens auszunützen. Ist solches Vorwissen nicht vorhanden, kann ein evolutionäres Verfahren verwendet werden, um die optimalen Module zu identifizieren. Als Bewertungsfunktion dient dabei die anfängliche Schnelligkeit des Lernens. Beim Modularen Encoder führt auch der im Vergleich zu einem evolutionären Verfahren effizientere Ansatz, die Entwicklung der Netzgewichte während des Trainings so zu kontrollieren, daß eine angemessen modulare Struktur hervortritt und evtl. verstärkt wird, zu verbessertem Lernen.

5.2 Nebenaussagen: Methoden

5.2.1 Modularitätsmaße

Zur Bestimmung der Modularität eines Multilagen-Perzeptrons sind zwei Ansätze vorgestellt worden: ein Neuronen-Cluster-Verfahren, das die Gewichtsbeträge als Zusammenhangs-Maß verwendet und hohen intra- gegen niedrigen intermodularen Zusammenhang abwägt, sowie eine Berechnungs-Vorschrift, die die neuronen-lokale Variabilität der Gewichtsbeträge quantifiziert (Struktur-Maß S). Die Bestimmung der Modularität mit Hilfe des Cluster-Verfahrens hat sich in den betrachteten Testaufgaben als zuverlässig und nützlich herausgestellt. Es erlaubt nicht nur, den Modularitäts-Grad der Gewichts-Struktur zu quantifizieren - hier erzielt auch das Struktur-Maß S vergleichbare Ergebnisse - sondern es liefert außerdem zusätzliche Information über die Zugehörigkeit der Neuronen zu verschiedenen Modulen, die zur Beeinflussung des Weight Decay in einem Lernverfahren herangezogen werden kann. Mit Hilfe des Parameters γ, der Intra- und Inter-Zusammenhang gegeneinander wichtet, kann außerdem eine a priori Annahme über die Gewichts-Struktur in das Cluster-Verfahren eingebracht, bzw. die Struktur auf unterschiedlichen Schachtelungs-Ebenen analysiert werden.

5.2.2 Weight Decay Varianten

Die verschiedenen Weight Decay Varianten, die zur Modularisierung der Netz-Struktur beim Modularen Encoder eingesetzt wurden (abnehmender Decay, Weight Growth und intermodularer Weight Decay), bringen dort deutliche Verbesserungen gegenüber einem einfachen konstanten Weight Decay. Alle Varianten haben gemein, daß sie den steilen Anstieg der Gewichte zu Beginn des Trainings und damit die vorschnelle Saturierung der Aktivierungsfunktionen effektiver verhindern als dies mit einem konstanten Weight Decay möglich wäre. Sie verlangsamen damit bewußt das anfängliche Lernen zugunsten einer weniger komplexen lokalen Fehler-Landschaft, die auch dann noch eine verbesserte Generalisierung gewährleistet, wenn es den Gewichten im weiteren Verlauf des Lernens erlaubt wird, anzusteigen. Die aus dieser Vorgehensweise resultierenden s-förmigen Lernkurven lassen sich in Analogie zum subjektiv erlebten menschlichen Lernen interpretieren als ein anfängliches oberflächliches Lernen der allgemeinen Merkmale einer Aufgabe mit anschließender Vertiefung der Détails.

5.2.3 Evolutionäres Verfahren

Der vorgestellte evolutionäre Algorithmus bietet eine alternative Möglichkeit, die modulare Struktur der Testaufgaben automatisch zu erkennen. Innovativ ist vor allem der Ansatz, die im Chromosom kodierte Modul-Struktur dem Netz bei der Fitneß-Bestimmung nicht durch die Wegnahme von Verbindungen, sondern durch modulare Anfangsbedingungen und inter-modularen Weight Decay aufzuprägen. Dies gewährt auch noch bei sehr schwach modularen Daten die Unterscheidbarkeit der zugrundeliegenden Modulstruktur. Im Falle der Vorbildnetz-Daten kommt das Verfahren bei diesem Ansatz außerdem mit extrem kurzen Trainingsläufen zur Fitneß-Bestimmung aus. Ferner ist gezeigt worden, daß der Algorithmus gleichzeitig die Komplexität des Netzes der Komplexität der Aufgabe anpaßt, indem überzählige innere Neuronen isolierten Modulen zugeordnet werden.

5.3 Verallgemeinerbarkeit

Die Frage, ob und bei welcher Art von realen Aufgaben sich die hier vorgestellten Ansätze nutzbringend einsetzen lassen, ist allein aufgrund der Erfahrungen mit den Testaufgaben nicht zu beantworten. Im Prinzip ist die Anpassung der Netzstruktur an die Struktur der Lernaufgabe immer von erheblicher Bedeutung für das Generalisierungsverhalten. Wie in Kapitel 3 geschildert, existieren bereits eine Vielzahl allgemeiner Verfahren, die einer solchen Anpassung dienen und sich in realen Aufgaben bewährt haben (Weight Decay, Pruning, topologie-optimierende evolutionäre Verfahren). Zwei der vorgestellten Ansätze, der abnehmende Weight Decay und das Weight Growth Verfahren sind nicht modulstrukturspezifische Verbesserungen, bzw. Alternativen zum konstanten Weight Decay und damit wie dieser "general-purpose" Methoden. Im Gegensatz dazu sind der intermodulare Weight Decay ("Cluster-Training") und der evolutionäre Algorithmus spezifisch auf modulare Daten zugeschnittene "special-purpose" Erweiterungen, bzw. Sonderfälle des allgemeinen Ansatzes. Es stellt sich somit die Frage, ob diese auf eine spezielle Form von Strukturiertheit, nämlich Modularität, zugeschnittenen Verfahren bei realen Aufgaben zusätzlichen Nutzen bringen. Gibt es reale Aufgaben, deren modulare Struktur einerseits ausreichend "versteckt" ist, um nicht von einem general-purpose Verfahren entdeckt zu werden, andererseits aber klar genug ausgeprägt, um den special-purpose Verfahren ihre Erkennung zu erlauben? Angesichts des Mangels an konkreter Erfahrung scheint mir die sinnvollste Haltung zu dieser Frage darin zu bestehen, die neuen Ansätze als weitere Waffen im Arsenal des Daten-Analysten zu betrachten, die ihm im Trial-and-Error Prozeß des Netz-Trainings zur Verfügung stehen.

5.4 Bezug zur Fach-Literatur

5.4.1 Unterraum-Modularisierung von MLP

Die in der Fachliteratur beschriebenen Ansätze zur Unterraum-Modularisierung von Multilagen-Perzeptronen (Jacobs et al., 1991a; Jacobs and Jordan, 1992; Pennington et al., 1995) sind wie die in dieser Arbeit vorgestellten Ansätze lediglich an Testaufgaben, und zwar speziell an der Was-und-Wo Aufgabe erprobt worden. Sie sind aber nicht "vollautomatisch, d.h. sie benötigen a priori Information über den Aufbau der Module, und zwar insbesondere über die Zuordnung der Ausgabeneuronen zum Was-, bzw. zum Wo-Teil der Aufgabe.

5.4.2 Neuropsychologie

Arbeiten über modulare neuronale Netze werden, wie in Kapitel 3 geschildert, gelegentlich in der neuropsychologischen Modularitäts-Diskussion zitiert. Können auch die Ergebnisse der vorliegenden Arbeit einen Beitrag in dieser Diskussion liefern? Zwei Standpunkte werden in dieser Diskussion oft mit neuronalen Modellen belegt: Zum einen kann man mit ihnen zeigen, daß funktional differenzierte Systeme ohne strukturell modulare neuronale Grundlage existieren können (Plaut, 1995; Plaut and Shallice, 1993). Mit den Ergebnissen dieser Arbeit läßt sich hier entgegenhalten, daß es auch Aufgaben gibt, in denen eine modulare Struktur des Netzes Voraussetzung für effizientes Lernen ist. Der zweite neuropsychologische Standpunkt, zu dessen Verteidigung neuronale Netze herangezogen werden, betont die Entwicklung modularer Struktur durch Lernen, im Gegensatz

zu angeborener Modularität. Die Experimente dieser Arbeit bestätigen, daß modulare Strukturen im Verlaufe des Lernverfahrens entstehen können, wenn die Aufgabe - und das heißt in diesem Zusammenhang: die Umwelt- entsprechend modular ist. Wichtig scheint dabei zu sein, daß sich die Modularität in frühen Stadien des Lernens ausprägt. Andererseits zeigen die Ergebnisse aber auch, daß eine "angeborene" modulare Aufteilung des Netzes vor Beginn des Lernens das Lernen mancher stark modularer Aufgaben wesentlich erleichtern kann. Bei nur schwach modularen Aufgaben/Umwelten können modulare Vorstrukturierungen ebenfalls noch vorteilhaft sein, hier aber in Form von Anfangsstrukturen, die im Laufe des Lernens verändert werden können. Insgesamt scheint die in dieser Arbeit untersuchte Form struktureller Modularität, die für unterschiedliche Dimensionen der Umwelt unterschiedliche Module bereitstellt (Unterraum-Modularität), für die Diskussion um Wahrnehmungsmodule relevanter zu sein, als eine Modularität, wo immer alle Dimensionen der Umwelt von allen Modulen gesehen werden und je nach "Inhalt entschieden wird, welches Modul die Ausgabe liefert, so wie es in den Experten-Netz-Architekturen der Fall ist (Regionen-Modularität). Diese Form der Modularität entspricht möglicherweise eher den höheren Verarbeitungsstufen nach der Integration der verschiedenen Sinneseindrücke.

5.4.3 Rekurrente Netze

Zu Beginn des zweiten Kapitels ist mit der Simulation von Laufverhalten durch rekurrente Netzen ein anderer Bereich vorgestellt worden, in dem modulare neuronale Netze eine wichtige Rolle spielen. Die Ergebnisse dieser Arbeit sind auf solche Aufgaben zunächst nicht direkt übertragbar, da sie im Zusammenhang vorwärtsgerichteter Netze gefunden worden sind. Sie könnten aber in jenen Verfahren Anwendung finden, die das Training rekurrenter Netze auf eine Zeitreihenvorhersage mit einem vorwärtsgerichteten Netz zurückführen.

5.4.4 Clusterverfahren

Das zur Analyse der Modularität der MLP-Gewichte entwickelte Clusterverfahren läßt sich mit verschiedenen bekannten Verfahren vergleichen, wie sie z.B. von Everitt beschrieben werden (Everitt, 1993). Zum einen ähnelt es in gewisser Hinsicht den Optimierungs-Clusterverfahren, in denen ebenfalls eine optimale Partition gesucht wird. Die zu partitionierenden Entitäten (Taxa) sind dort aber hoch-dimensionale Datenvektoren und nicht dimensionslose Neuronen. Das Optimalitätskriterium wird aus den aus einer Partition resultierenden Intra- und Inter-Varianzen der Datenvektoren berechnet: je geringer die Intra-Varianz und je höher die Inter-Varianz, desto besser ist die Partition. Die Intra-Varianz ist dabei dem in dieser Arbeit verwendeten Intra-Zusammenhangs-Maßen sehr verwandt, da sich die Varianz einer Menge von Vektoren aus den mittleren paarweisen Abständen der Vektoren berechnen läßt. Diese paarweisen Abstände entsprechen in unserem Verfahren bis auf das Vorzeichen den Gewichten der Neuronen-Verbindungen: hier wird Intra-Gewicht maximiert, dort Intra-Abstand minimiert. Ein Unterschied der Verfahren liegt auch darin, daß der Inter-Zusammenhang anders bestimmt wird als die Inter-Varianz. Während diese als Varianz der Cluster-Schwerpunkte definiert wird, ist für jene aus der lediglich paarweise gegebenen Zusammenhangs-Information kein Schwerpunkt zu berechnen, so daß der Inter-Zusammenhang zwischen zwei Clustern aus den Gewichten aller Interverbindungen zwischen diesen zwei Clustern berechnet wird.

Zum anderen ähnelt das divisive Verfahren, das in dieser Arbeit als eine Heuristik zur Berechnung der optimalen Partition eingführt wurde, den sehr verbreiteten hierarchischen Cluster-Verfahren. Neu ist allerdings, daß die Schachtelungs-Tiefe, an der der Zerfall abgebrochen wird -und damit die Anzahl der Cluster- aufgrund des Modularitätsmaßes a posteriori bestimmt und nicht, wie in hierarchischen Clusterverfahren üblich, a priori vorgegeben wird!

Als Maß für den Zusammenhang zweier Neuronen wird in dem Cluster-Verfahren der Betrag des Gewichts der Verbindung zwischen den beiden Neuronen verwendet. Die Annahme, Verbindungen mit großem Gewicht seien wichtiger als kleine, ist allerdings in gewissem Maße naiv. Problematisch ist vor allem die Interpretation der Gewichte der ersten Schicht. So können Eingaben, die mit einem hohen Gewicht an ein inneres Neuron geleitet werden, dennoch unwichtig für die Ausgabe sein, wenn das innere Neuron selbst nur mit einem geringen Gewicht mit der Ausgabe verbunden ist. Eine Eingabe, die mit zwei inneren Neuronen mit hohem Gewicht verbunden ist, kann dennoch unwichtig sein, wenn diese inneren Neuronen mit verschiedenen Vorzeichen mit der Ausgabe verbunden sind, so daß sich ihre Beiträge auslöschen. Andererseits kann eine Eingabe, die mit geringem Gewicht mit vielen inneren Neuronen verbunden ist, insgesamt einen erheblichen Einfluß auf die Ausgabe haben (Masters, 1995). Schließlich setzt ein sinnvoller Vergleich der Gewichte der ersten Schicht normierte Eingaben voraus (Bishop, 1995). Eine ausgefeiltere Variante würde deshalb als Zusammenhangs-Maß den Effekt messen, den eine kleine Variation des Gewichtes auf die Fehlerfunktion hat und dabei u.U. auch Interaktionen zweiter Ordnung mit anderen Gewichten, also die Hesse-Matrix berücksichtigen, so wie es die Pruning-Verfahren des Optimal Brain Damage, bzw. Optimal Brain Surgeon tun.

5.5 Mögliche Erweiterungen

Eine denkbare Erweiterung der Ansätze wäre eine differenziertere Gestaltung des intermodularen Weight Decays, sowohl beim "Cluster-Training" als auch bei der Evolution mit "Softsplit"-Strukturierung, indem man für die verschiedenen Gruppen von Inter-Verbindungen unterschiedliche Decay Faktoren zuläßt. Außerdem könnte man zusätzlich intra-modularen Decay zulassen. Eine Verallgemeinerung des evolutionären Algorithmus auf nicht modulare Strukturen, die aber den "Softsplit"-Ansatz beibehält, könnte man mit Hilfe von Chromosomen erreichen, die unterschiedliche Decay Faktoren für jede einzelne Verbindung kodieren. Schließlich kann man daran denken, die harte Zuordnung von Neuronen zu Clustern aufzuweichen, indem ein Neuron in unterschiedlichem Grade verschiedenen Clustern zugehören kann (fuzzy clustering).

Die vorgestellten Ansätze berücksichtigen außerdem lediglich den in Kapitel 2 als grundlegend herausgearbeiteten Zusammenhangs-Aspekt von Modularität und z.B. nicht das wiederholte Auftreten von Teilstrukturen oder andere Symmetrien innerhalb der Gewichts-Struktur. Eine Erweiterung der Ansätze um eine solche solche Erkennung von redundanter Struktur könnte die Lernaufgabe vereinfachen, die Komplexität des Netzes verringern und damit die Generalisierungsfähigkeit erhöhen.

5.6 Warum man diese Arbeit lesen sollte

Die Arbeit thematisiert den im Zusammenhang modularer neuronaler Netze weitgehend vernachlässigten Bereich der Unterraum-Modularität. Sie untersucht Vorteile dieser Form von Modularität und stellt Ansätze zu ihrer Erzeugung vor, die ohne a priori Information über die Aufgabenstruktur auskommen. Die Arbeit hat weitgehend explorativen Charakter und soll vor allem zur Ideen-Entwicklung in diesem Bereich beitragen. Teile der Arbeit sind in einem Journal-Beitrag zusammengefaßt worden (Wagner, 1999).

Literaturverzeichnis

Beer, R. D. and Gallagher, J. C. (1992). Evolving dynamical neural networks for adaptive behavior. *Adaptive Behavior*, 1(1).

Behrmann, M., Black, S. E., and Murji, S. (1995). Spatial attention in the mental architecture: Evidence from neuropsychology. *Journal of Clinical and Experimental Neuropsychology, Special Issue: Modularity and the Brain*, 17(2):220–242.

Bennani, Y. (1995). A modular and hybrid connectionist system for speaker identification. *Neural Computation*, 7(4):791–798.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press.

Bonner, J. T. (1988). *The Evolution of Complexity*. Princeton University Press, Princeton, N.J.

Brockhaus (1971). *Brockhaus-Enzyklopädie*. Brockhaus-Verlag, Wiebaden.

Cruse, H., Bartling, C., G.Cymbalyuk, Dean, J., and Dreifert, M. (1995). A modular artificial neural net for controlling a six-legged walking-system. *Biological Cybernetics*, 72:421–430.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14:179–211.

Erwin, E., Obermayer, K., and Schulten, K. (1995). Models of orientation and ocular dominance columns in the visual cortex: A critical comparison. *Neural Computation*, 7:425–68.

Everitt, B. S. (1993). *Cluster Analysis*. London, 3 edition.

Fahlman, S. E. and Lebiere, C. (1990). The cascade-correlation learning architecture. In *Advances in Neural Information Processing*, volume 2, pages 524–532, San Mateo, CA. Morgan Kaufmann.

Faust, M., Babkoff, H., and Kravetz, S. (1995). Linguistic processes in the two cerebral hemispheres: Implication for modularity vs interactionism. *Journal of Clinical and Experimental Neuropsychology, Special Issue: Modularity and the Brain*, 17(2):171–192.

Fodor, J. A. (1975). *The language of thought*. Thomas Crowell.

Fodor, J. A. (1983). *The modularity of mind*. MIT Press.

Fodor, J. A. (1985). Precis of the modularity of mind. *Behavioral and Brain Sciences*, 8:1–42.

Goldberg, E. (1995). Rise and fall of modular orthodoxy. *Journal of Clinical and Experimental Neuropsychology, Special Issue: Modularity and the Brain*, 17(2):193–208.

Gould, S. (1977). *Ontogeny and Phylogeny*. Harvard University Press, Cambridge, MA.

Gruau, F. (1995). Automatic definition of modular neural networks. *Adaptive Behavior*, 3(2):152–183.

Hassibi, B. and Stork, D. (1993). Second order derivatives for network pruning: optimal brain surgeon. In *Advances in Neural Information Processing*, volume 5, pages 164–171, San Mateo, CA. Morgan Kaufmann.

Hesse, R. (1935). *Tierbau und Tierleben in ihrem Zusammenhang betrachtet*, volume 1, chapter 6, pages 844–859. Verlag von Gustav Fischer, Jena.

Jacobs, R. A. and Jordan, M. (1992). Computational consequences of a bias toward short connections. *Journal of Cognitive Neuroscience*, 4(4):323–336.

Jacobs, R. A., Jordan, M., and Barto, A. G. (1991a). Task decomposition through competition in a modular connectionist architecture: The what and where vision tasks. *Cognitive Science*, 15:219–250.

Jacobs, R. A., Jordan, M., Nowlan, S. J., and Hinton, G. E. (1991b). Adaptive mixtures of local experts. *Neural Computation*, 3:79–97.

Jordan, M. and Jacobs, R. A. (1994). Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6(2):181–214.

Karmiloff-Smith, A. (1994). Precis of beyond modularity: A developmental perspective on cognitive science. *Behavioral and Brain Sciences*, 17:693–745.

Knaur's Lexikon der Technik (1972). *Knaur's Lexikon der Technik*. Droemer-Knaur, München/Zürich.

Kodjabachian, J. and Meyer, J.-A. (1996). Evolutionary design of an artificial neural network controlling the locomotion of a six legged animat. In *ICML'96 Workshop on Evolutionary Computation and Machine Learning*.

Kohonen, T. (1984). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69.

Masters, T. (1995). *Practical Neural Network Recipes in C++*. Academic Press.

Moscovitch, M. (1995). Recovered consciousness: A hypothesis concerning modularity and episodic memory. *Journal of Clinical and Experimental Neuropsychology, Special Issue: Modularity and the Brain*, 17(2):276–290.

Moscovitch, M. and Umiltà, C. (1990). Modularity and neuropsychology: Implications for the organization of attention and memory in normal and brain damaged people. In Schwartz, M. E., editor, *Modular processes in dementia*, pages 1–59. MIT/Bradford.

Nachson, I. (1995). On the modularity of face recognition: The riddle of domain specifity. *Journal of Clinical and Experimental Neuropsychology, Special Issue: Modularity and the Brain*, 17(2):256–275.

Needham, J. (1933). On the dissociability of the fundamental processes in ontogenesis. *Biological Reviews*, 8:180–223.

Pearlmutter, B. A. (1995). Gradient calculation for dynamic recurrent neural networks: a survey. *IEEE Transactions on Neural Networks*, 6(5):1212–1228.

Pennington, R., Snoad, N., and Bossomaier, T. (1995). Evolution of computational efficiency in visual processing. In *Proc. IEEE Conf. on Evolutionary Computation*, volume I, page 199, Perth, Australia.

Plaut, D. C. (1995). Double dissociation without modularity: Evidence from connectionist neuropsychology. *Journal of Clinical and Experimental Neuropsychology, Special Issue: Modularity and the Brain*, 17(2):291–321.

Plaut, D. C. and Shallice, T. (1993). Deep dyslexia: A case study of connectionist neuropsychology. *Cognitive Neuropsychology*, 10(5):377–500.

Ripley, B. D. (1996). *Pattern recognition and Neural Networks*. Cambridge University Press.

Ritter, H. and Kohonen, T. (1989). Self-organized semantic maps. *Biological Cybernetics*, 61:241–254.

Rojas, R. (1993). *Theorie der neuronalen Netze*. Springer-Verlag.

Rueckl, J. G., Cave, K. R., and Kosslyn, S. M. (1989). Why are "what" and "where" processed by separate cortical visual systems? a computational investigation. *Journal of Cognitive Neuroscience*, 1(2):171–186.

Rumelhart, D. and McClelland, J. (1986). *Parallel Distributed Processing*. MIT Press.

Soroker, N., Calamaro, N., and Myslobodsky, M. S. (1995). Ventriloquist efferct reinstates responsiveness to auditory stimuli in the 'ignored' space in patients with hemispatial neglect. *Journal of Clinical and Experimental Neuropsychology, Special Issue: Modularity and the Brain*, 17(2):243–255.

Umiltà, C. (1995). Domain-specific forms of neglect. *Journal of Clinical and Experimental Neuropsychology, Special Issue: Modularity and the Brain*, 17(2):202–219.

VanEssen, D., Fellemann, D., DeYoe, E., J.Olavarria, and J.Knierim (1990). Modular and hierachical organization of extrastriate visual cortex in the macaque monkey. In *Cold Spring Harbor Symposia on Quantitative Biology*, volume LV, pages 679–696. Cold Spring Harbor Laboratory Press.

Wagner, G. (1995). Adaptation and the modular design of organisms. In Morán, F., Morán, A., Merelo, J., and Chacón, P., editors, *Advances in Artificial Life*, pages 317–328. Springer Verlag.

Wagner, H. (1999). Helping a multilayered perceptron discover modular structure. *Neural Processing Letters*. submitted.

Waibel, A., Sawai, H., and Shikano, K. (1989). Modularity and scaling in large phonemic neural networks. *IEEE Transactions on Acoustic, Speech and Signal Processing*, 37(12).

Y. LeCun, J.S. Denker, S. S. (1993). Optimal brain damage. In *Advances in Neural Information Processing*, volume 5, pages 598–605, San Mateo, CA. Morgan Kaufmann.