Algebraische Einbettungen Neuronaler Netze

Diplomarbeit

am Lehrstuhl für Kognitive Systeme
des Instituts für Informatik und Parktische Mathematik
der Christian–Albrechts–Universität zu Kiel
Prof. Dr. Gerald Sommer

vorgelegt von

Sven Buchholz

Oktober 1997

Betreuer: Dr. Eduardo Bayro

Zusammenfassung

In dieser Diplomarbeit werden Clifford–Algebren als einheitliches und umfassendes Modell algebraischer Einbettungen reeller Vektorräume als neuer mathematischer Rahmen für den Entwurf Neuronaler Netze untersucht. Im Mittelpunkt der Arbeit steht dabei die Entwicklung von Multilayer Perceptrons und ihrer Lernverfahren in diesem Rahmen, wobei auch bisherige Ansätze systematisiert und bewertet werden. Es erfolgen vergleichende experimentelle Studien zur Beurteilung der Leistungsfähigkeit der entwickelten Netze, sowie zu den generellen Möglichkeiten des Ansatzes zur Kodierung und Verarbeitung geometrischer Sachverhalte.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Spezielle Symbole und Abkürzungen

$\mathbb{R}$	Körper der reellen Zahlen
$\mathbb{C}$	Körper der komplexen Zahlen
$\mathbb{H}$	Schiefkörper der Quaternionen
M^n	n–faches Cartesisches Produkt von M
$\mathcal{P}(M)$	Potenzmenge non M
X^Y	Menge der Abbildungen von X nach Y
$[x]_i$	i–te Komponente eines (Multi–) Vektors
$\langle\ ,\ \rangle$	Skalarprodukt
$\times$	Vektorprodukt
$\wedge$	Äußeres Produkt
$\odot$	komponentenweises Produkt zwischen (Multi–) Vektoren
${}^n\mathcal{R}$	Ring $(\mathbb{R}^n, +, \odot)$
${}^n\mathcal{C}$	Ring $(\mathbb{C}^n, +, \odot)$
${}^n\mathcal{H}$	Ring $(\mathbb{H}^n, +, \odot)$
$R(n)$	Algebra der $n \times n$–Matrizen über dem Ring R
$\mathcal{C}_{n,s}$	2^n–dimensionale reelle Clifford Algebra mit Signatur s
$\otimes$	Algebra–Multiplikation
$\otimes_{n,s}$	Geometrisches Produkt in $\mathcal{C}_{n,s}$
$\bar{x}$	zu x konjugierter Multivektor
$\forall$	für alle
$\exists$	es existiert
$\nexists$	es existiert kein
$\exists_1$	eindeutige Existenz
MLP	Multilayer Perceptron
CMLP	Komplexes Multilayer Perceptron
QMLP	Quaternionen Multilayer Perceptron
CA–MLP	Clifford–Algebra Multilayer Perceptron
$\mathcal{C}_{n,s}$–MLP	Clifford–Algebra Multilayer Perceptron über $\mathcal{C}_{n,s}$

Einleitung

Bei einer Vielzahl von Problemen bedarf es einer Erweiterung der reellen Zahlen, um eine adäquate Repräsentation zu finden, die eine Lösung erlaubt. Die komplexen Zahlen entstanden historisch aus dem Bedürfnis, gewissen in $\mathbb{R}$ unlösbaren Gleichungen, wie etwa $x^2 + 1 = 0$, eine Lösung zu verschaffen. Die Mathematiker begannen daher auch nicht mit ihrer Fundierung, sondern rechneten einfach mit ihnen. Der Rest ist Geschichte.

Die zweite fundamentale Erweiterung reeller Zahlen, die Quaternionen sind hingegen unter Nichtmathematikern und Nichtphysikern weit weniger bekannt, hielten aber im Verlauf der letzten Jahre nun auch verstärkt Einzug in verschiedene Gebiete, darunter Robotik und Computergraphik. Nichtzuletzt deshalb, weil sie eine elegante Möglichkeit zur Darstellung von Rotationen im Raum bieten, also eine ausgezeichnete geometrische Bedeutung besitzen. Bei komplexen Zahlen wird die entsprechende aus der Polardarstellung ableitbare geometrische Bedeutung durch ihre Anwendungen selbst noch überstrahlt.

Die algebraische Struktur ist in beiden Fällen also Träger von Geometrie, sie erlaubt uns die geometrische Manipulation von Vektoren der Ebene bzw. des Raumes durch einfaches Rechnen.

Da die erste sich stellende Frage bei der Entwicklung autonomer künstlicher Systeme, die mit ihrer Umwelt über einen Wahrnehmungs–Aktions–Zyklus interagieren, die nach der internen Repräsentation der äußeren Welt ist, wobei eine geometrische sicher vorteilhaft sein dürfte — von Koenderink sogar das Postulat
„Das Gehirn ist eine geometrische Maschine“
stammt — sollte man annehmen, daß algebraische Strukturen, die dies im „kleinen“ ermöglichen, wie die komplexen Zahlen und Quaternionen, bereits breiten Eingang in die Neuroinformatik gefunden haben.

Dies trifft aber eher nicht zu.
Zwar wurden Neuronale Netze für komplexe Zahlen bereits Anfang der 90iger Jahre entwickelt, aber ein Quaternionen MLP wurde erst kürzlich [Arena96a] [Arena96b] vorgestellt. Die komplexen Netze lieferten kaum über sie selbst hinausgehende nennenswerte Impulse.

Das Fehlen eines allgemeineren Modells mit einer entsprechenden Motivierung war sicher

der Hauptgrund, warum die Entwicklung Neuronaler Netze in Erweiterungen der reellen Zahlen nahezu zum Stillstand kam.

In [Pearson94] wurde nun erstmals das alle relevanten algebraischen Erweiterungen der reellen Zahlen umfassende Modell der Clifford–Algebren benutzt, um MLPs in diesem Rahmen zu formulieren. Bedauerlicherweise sind die dort vorgeschlagenen Netze aber nicht korrekt, wie wir nachweisen werden, obwohl das Modell selbst richtig ist und auch ein weitaus größeres Potential besitzt, als in der zitierten Arbeit zum Vorschein kommt. Dieses Potential ist auch die eigentliche Motivation, warum wir die Entwicklung von Neuronalen Netzen in Clifford–Algebren in dieser Arbeit betreiben wollen.

Werfen wir einen Blick auf die generelle Probleme mit denen man beim Entwurf von Neuronalen Netzen in algebraischen Erweiterungen unausweichlich konfrontiert wird.

- Zuallererst gelten viele uns von den reellen Zahlen vertrauten und nützlichen Regeln in den algebraischen Erweiterungen nicht, weil diese bestimmte Eigenschaften wie zum Beispiel Kommutativität nicht mehr besitzen. Damit werden naturgemäß Lernverfahren und Korrektheitsbeweise dann mathematisch aufwendiger und schwerer.
- Die potentiellen Vorteile der algebraischen Strukturen könnten in Neuronalen Netzen nur schwer nutzbar sein, d.h. an deren Berechnungsparadigmen scheitern. Allgemein gesprochen approximiert in diesen ein Neuronales Netz eine Funktion, indem es einen Fehler minimiert, wobei das Lernen anhand von Beispielen erfolgt, also eine Approximation durchgeführt wird. Dies alles spielt sich im Rahmen eines reellen Vektorraumes ab und die benutzten Mittel der Optimierung bedürfen auch eines normierten Raumes.
- Mit dem vorherigen Problem geht dann schnell ein weiteres einher. Für reelle Netze stehen starke Approximationssätze zur Verfügung, so daß wir, wenn wir ihren Rahmen nicht verlassen können, mit ihnen auf der Ebene der Effizienz konkurrieren müssen.

Das letzte Problem ist aber eigentlich nur ein psychologisches. Es spiegelt einen Trend wieder, der aus dem sehr positiven Faktum der Etablierung und schnell fortschreitenden Verbreitung Neuronaler Netze in vielen Anwendungen resultiert, Neuronale Netze zunehmend nur noch als algorithmisches Werkzeug, denn als Modell für biologische Abläufe zu sehen.

Das Ziel dieser Arbeit ist es zu zeigen, daß Neuronale Netze in Clifford–Algebren konzeptionelle Vorteile gegenüber herkömmlichen Netzen haben, die sich aus dem Vorhandensein zusätzlicher Möglichkeiten zur Verarbeitung und Hervorbringung geometrischer Strukturen ergeben. Diese sind sozusagen Bestandteil der Clifford–Algebren selbst, da diese aus entsprechenden Einbettungen reeller Vektorräume hervorgehen, wobei zusätzliche geometrische Entitäten erzeugt werden. Die Beobachtung der Wirksamkeit derartig plausibler Konzepte auf der Ebene Neuronaler Netze erlaubt dann weitere interessante Schlußfolgerungen.

Gliederung der Arbeit

Am Anfang steht die Einführung des bereits erwähnten algebraischen Modells. Dieses wird in Kapitel 1 Schritt für Schritt von den elementaren mathematischen Grundlagen Neuronaler Netze ausgehend erarbeitet. Aus einfachen mathematischen Sachverhalten wird eine erste algebraische Spezifikation des Modells gewonnen, die dann konstruktiv aus unserer Motivation der Erzeugung neuer Entitäten aus Vektoren präzisiert und vervollständigt wird. Als Resultat dieses Einbettungsprozesses erhalten wir Clifford–Algebren, in deren Grundlagen dann eingeführt wird.

Kapitel 2 untersucht, welche herkömmlichen vorwärtsgerichteten Neuronalen Netze zu Neuronalen Clifford–Algebra Netzen sinnvoll erweiterbar sind und wie dies geschehen kann. Nach der Behandlung wichtiger Grundlagen vorwärtsgerichteter Neuronaler Netze und der formalen Definition eines Neuronalen Clifford–Algebra Netzes widmen wir uns zunächst den RBF–Netzen und diskutieren ihre Erweiterbarkeit. Als Ergebnis der Diskussion können wir diesen Ansatz nicht weiterverfolgen und MLPs bilden von da ab den Gegenstand unserer Untersuchungen. Es werden zuerst die Grundlagen über Aktivierungsfunktionen und den Backpropagation–Algorithmus bei reellen MLPs vermittelt. Anschließend befassen wir uns mit CMLPs und arbeiten an ihnen prinzipielle Probleme und Möglichkeiten des Übergangs von MLPs zu Clifford–Algebra MLPs heraus.

Den theoretischen Kern dieser Arbeit bildet Kapitel 3. Die zuvor bei den CMLPs gewonnen Einsichten werden zu einer aus 2 disjunkten Klassen von Aktivierungsfunktionen bestehenden Systematik formalisiert. Von der einen Klasse, zu der auch die Netze aus [Pearson94] gehören, wird nachgewiesen, daß man mit ihren Aktivierungsfunktionen keine korrekten Clifford–Algebra MLPs erhalten kann. Aufbauend auf dem QMLP [Arena96a] [Arena96b], welches eine Aktivierungsfunktion der anderen Klasse benutzt, werden korrekte MLPs in beliebigen Clifford–Algebren entwickelt. Für 4–dimensionale Clifford–Algebren werden nahezu alle Schritte explizit angegeben, für höherdimensionale Algebren kann oft nur die Idee skizziert werden, da die Beweise sonst den Rahmen der Arbeit sprengen würden.

Kapitel 4 ist ganz experimentellen Untersuchungen zu Clifford–Algebra MLPs gewidmet, wobei zuerst die Illustration und Bestätigung wichtiger theoretischer Aussagen des vorhergehenden Kapitels im Vordergrund stehen, was anhand des XOR–Problems erfolgt. Danach werden Beispiele zur Funktionsapproximation mit Clifford–Algebra MLPs in 4 und 8 Dimensionen untersucht. Aussagen zur Effizienz der von uns entwickelten Netze im Vergleich zu herkömmlichen MLPs werden dann am Beispiel der Vorhersage einer chaotischen Zeitreihe gewonnen.

In Kapitel 5 werden die vom geometrischen Produkt erzeugten Kodierungen näher betrachtet und hinsichtlich ihrer Eignung zur Erkennung von Drahtgittermodellen mit anderen geometrischen Merkmalen experimentell verglichen. Es erfolgt auch ein Vergleich der Funktionsprinzipien unserer Netze mit denen bei Neuronalen Netzen höherer Ordnung.

Kapitel 6 enthält die Zusammenfassung und Diskussion der Resultate dieser Arbeit.

Kapitel 1

Das algebraische Modell der Arbeit

Dieses Kapitel hat zum Ziel, nicht nur einfach das algebraische Modell der Arbeit einzuführen, sondern es insbesondere als das Ergebnis eines natürlichen, direkt auf der Ebene der Neuronalen Netze nachvollziehbaren, Einbettungsprozesses darzustellen. Erst auf diesem Wege wird es uns nämlich gelingen nachzuweisen, daß es sich dabei gewissemaßen schon um das allgemeinste Modell überhaupt handeln wird, was man betrachten kann, welches zudem noch eine geometrische Interpretierbarkeit besitzt. Die komplexen Zahlen und die Quaternionen dienen uns dabei nicht nur als statische Vorgaben, sondern auch als nützliche Orientierungspunkte. Den Ausgangspunkt bilden die endlich dimensionalen reellen Vektorräume, die das mathematische Fundament klassischer vorwärtsgerichteter Neuronaler Netze bilden.

1.1 Grundlagen

Beginnen wir also mit dem in unserem Fall Grundlegensten, den reellen Zahlen $\mathbb{R}$, genauer mit dem Begriff des *Körpers*, dem die reellen Zahlen als algebraische Struktur zugeordnet werden können.

Definition 1.1: **(Körper)**
Ein *Körper* ist ein Tripel $(K, +, \cdot)$, bestehend aus einer Menge K und zwei Verknüpfungen $+$ und $\cdot$ auf K (Addition und Multiplikation), d.h. einer Abbildung

$$+ : K \times K \to K,\ (a, b) \mapsto a + b\,,$$

und einer Abbildung

$$\cdot : K \times K \to K,\ (a, b) \mapsto a \cdot b\,,$$

mit folgenden Eigenschaften (*Körperaxiome*):

(K 1) $(K, +)$ ist eine abelsche Gruppe.

(K 2) $(K\backslash\{0\}, \cdot)$ ist eine abelsche Gruppe.

(K 3) Es gelten die *Distributivgesetze*:

$$(a+b)\cdot c = a\cdot c + b\cdot c \text{ und}$$
$$c\cdot(a+b) = c\cdot a + c\cdot b\,.$$

Eine nicht–leere Teilmenge T von K, welche die Körperaxiome erfüllt, heißt ein *Teilkörper* von K.

Aufgrund von (K 1) existiert zu jedem Element $a \in K$ ein eindeutig bestimmtes additiv inverses Element, welches mit $-a$ bezeichnet wird. Statt $a+(-b)$ schreibt man kurz $a-b$. Ebenso gibt es nach (K 2) zu jedem $a \in K\backslash\{0\}$ ein eindeutig bestimmtes multiplikativ inverses Element a^{-1}. Für $a\cdot b^{-1}$ schreibt man kurz $\frac{a}{b}$. Subtraktion und Division sind also Ableitungen aus den entsprechenden Körperaxiomen.

Die reellen Zahlen bilden offensichtlich einen Körper, die gewohnten Regeln nach denen wir mit ihnen Rechnen haben durch diese Definition sozusagen nur ihre axiomatische Kennzeichnung erfahren. Auch weitere, der uns von den reellen Zahlen vertrauten Rechenregeln sind bereits Folgerungen aus den Körperaxiomen, wie jene für die Multiplikation mit der Null:

$$0\cdot a = a\cdot 0 = 0\,. \tag{1.1}$$

Die Null nimmt also bei der Multiplikation eine spezielle Stellung ein. Da sie kein multiplikativ inverses Element besitzt, muß sie in (K 2) ausgeschlossen werden. Diese Nichtexistenz eines multiplikativ Inversen der Null liefert somit auch den Grund für die bekannte Regel: „Durch 0 darf man nicht teilen!“

In Körpern ist sie das einzige derartige Element (K 2), d.h. für alle $a, b \in K$ gilt:

$$a\cdot b = 0 \Rightarrow (a = 0 \vee b = 0)\,. \tag{1.2}$$

Diese Eigenschaft wird als *Nullteilerfreiheit* bezeichnet.

Nachdem der Begriff des Körpers erläutert wurde, wollen wir als nächsten Schritt die Regeln für das Rechnen mit n–Tupeln reeller Zahlen formalisieren, was uns allgemeiner auf Vektoren und die Struktur des Vektorraumes führt.

Definition 1.2: (Vektorraum)

Sei K ein Körper. Ein *K–Vektorraum* ist ein Tripel $(V, +, \cdot)$, bestehend aus einer Menge V, einer Verknüpfung (Addition)

$$+ : V\times V \to V,\ (v,w) \mapsto v+w\,,$$

und einer Verknüpfung (Multiplikation mit Skalaren)

$$\cdot : K\times V \to V,\ (\lambda, v) \mapsto \lambda\cdot v\,,$$

so daß folgendes gilt (*Vektorraumaxiome*):

(V 1) $(V,+)$ ist eine abelsche Gruppe.

(V 2) Für alle $v,w \in V$, $\lambda,\mu \in K$ gilt:

$$\begin{aligned}(\lambda+\mu)\cdot v &= (\lambda\cdot v)+(\mu\cdot v),\\ \lambda\cdot(v+w) &= (\lambda\cdot v)+(\lambda\cdot w),\\ (\lambda\mu)\cdot v &= \lambda\cdot(\mu\cdot v),\\ 1_K\cdot v &= v.\end{aligned}$$

Ein Vergleich der Axiome eines Körpers mit denen eines Vektorraumes zeigt sofort, daß jeder Körper über sich selbst ein Vektorraum ist, da die Axiome der Addition (K 1) und (V 1) identisch sind, und die skalare Multiplikation des Vektorraumes in diesem Fall gerade die Multiplikation des Körpers ist. Also sind die ersten beiden Gleichungen von (V 2) die Distributivgesetze des Körpers (K 3) und die beiden verbliebenen Gleichungen folgen unmittelbar aus (K 2) und (1.1). Damit haben wir praktisch den nachstehenden einfachen Satz bewiesen.

Satz 1.1: (Körper als Vektorräume)
Sei K ein Körper und T ein Teilkörper von K. Dann ist K ein T–Vektorraum.

Wir führen nun die komplexen Zahlen ein, und wollen sie sodann mit unseren bisher erarbeiteten Instrumentarium als algebraische Einbettung von $\mathbb{R}$ untersuchen.

Definition und Satz 1.3: (Komplexe Zahlen)
Das Tripel $(\mathbb{R}\times\mathbb{R},+,\otimes)$ mit der für alle $a_1,b_1,a_2,b_2 \in \mathbb{R}$ durch

$$(a_1,b_1)+(a_2,b_2) := (a_1+a_2,b_1+b_2) \tag{1.3}$$

definierten Addition und der durch

$$(a_1,b_1)\otimes(a_2,b_2) := (a_1a_2-b_1b_2,a_1b_2+a_2b_1) \tag{1.4}$$

definierten Multiplikation ist ein Körper, der sogenannte *Körper der komplexen Zahlen* und wird mit $\mathbb{C}$ bezeichnet.

Komplexe Zahlen sind also Paare von reellen Zahlen, mit einer zusätzlich erklärten Multiplikation, durch die $\mathbb{R}^2$ zu einem Körper wird, was man leicht durch einfaches Nachrechnen der Körperaxiome verifiziert.

Mittels Identifizierung der reellen Zahlen gemäß

$$\mathbb{R} := \{(a,b)\in\mathbb{C} \mid b=0\}, \tag{1.5}$$

erhält man, daß $\mathbb{R}$ als Teilkörper in $\mathbb{C}$ eingebettet ist. Nach Satz 1.1 ist dann insbesondere $\mathbb{C}$ auch ein zweidimensionaler $\mathbb{R}$–Vektorraum. Komprimiert kann dies alles durch die Notation

$$(\mathbb{R}=)\ \mathbb{R}^1 \prec ((\mathbb{R}^2,+,\cdot_{\mathbb{R}}),\otimes)\ (=\mathbb{C}), \tag{1.6}$$

zum Ausdruck gebracht werden, wobei $\prec$ das Einbettungssymbol sein soll.

Verallgemeinern wir den konkreten Fall $\mathbb{R} \prec \mathbb{C}$, so erhalten wir als charakteristische Eigenschaften unserer Einbettungen:

- Übergang zu einem höherdimensionalen Raum
  (quantitative Erweiterung; $\mathbb{R}^1 \to \mathbb{R}^2$)
- Einführung/Benutzung zusätzlicher Operationen in diesem Raum
  (qualitative Erweiterung; komplexe Multiplikation $\otimes$)
- Strukturerhaltung (Vektorraumeigenschaften)

Mit der in (1.6) eingeführten Notation können wir diese Merkmale wie dann folgt zusammenfassen:

$$\mathbb{R}^n \prec ((\mathbb{R}^m, +, \cdot_{\mathbb{R}}), \otimes) \quad (n < m)\,. \tag{1.7}$$

Unsere Aufgabe besteht nun darin, Schritt für Schritt die einzelnen noch freien Parameter von (1.7) so festzulegen, daß der Einbettungsprozeß $\prec$ wohldefiniert wird.

Bevor wir damit beginnen, wollen wir zum Abschluß dieses Grundlagenabschnittes noch das wichtigste Hilfsmittel bei der Untersuchung algebraischer Strukturen einführen, welches dann auch häufig auf unsere Einbettungen angewandt werden wird. Es handelt sich um den Begriff der *Isomorphie*.

Zwei mathematische Verknüpfungsgebilde V und W vom Typ T (z.B. Körper) heißen zueinander *isomorph* (in Zeichen $V \cong W$), falls es wenigstens eine bijektive Abbildung gibt, die V strukturerhaltend nach W abbildet. Eine solche Abbildung wird dann ein *Isomorphismus* genannt. Isomorphie ist eine Äquivalenzrelation.

Bei Vektorräumen zum Beispiel sind die Isomorphismen genau die bijektiven linearen Abbildungen, dabei heißt eine Abbildung F zwischen zwei K–Vektorräumen V und W K–linear, falls

$$F(\lambda \cdot v + \mu \cdot w) = \lambda F(v) + \mu F(w) \tag{1.8}$$

für alle $v, w \in V$ und $\lambda \in K$ gilt.

Damit haben wir alle grundlegenden algebraischen Begriffe und Sachverhalte behandelt, und können mit der eigentlichen Modellierung unserer Einbettungen beginnen.

1.2 Spezifikation der algebraischen Struktur

Unser nächstes Ziel ist es, die algebraische Struktur der rechten Seite von (1.7) zu spezifizieren. Für den Spezialfall der komplexen Zahlen haben wir dies bereits getan, wobei wir

die Eigenschaften eines Körpers nachweisen konnten. Untersuchen wir also diesbezüglich unsere zweite Vorgabe — die Quaternionen $\mathbb{H}$.

Quaternionen (entdeckt 1853 von Hamilton) sind Zahlen der Form

$$a + b\,\mathrm{i} + c\,\mathrm{j} + d\,\mathrm{k} \tag{1.9}$$

mit $a, b, c, d \in \mathbb{R}$ und imaginären Einheiten i, j, k.

Für Quaternionen ist eine Multiplikation gemäß Tabelle 1.1 erklärt. Da die Multiplikati-

$\otimes$	1	i	j	k
1	1	i	j	k
i	i	-1	k	-j
j	j	-k	-1	i
k	k	j	-i	-1

Tabelle 1.1: *Multiplikationstabelle der Quaternionen*

onstabelle nicht symmetrisch ist, ist die Multiplikation der Quaternionen nicht kommutativ, also ist $\mathbb{H}$ kein Körper mehr. $(\mathbb{H}, \otimes)$ ist aber noch eine Gruppe. Strukturen, in denen an Stelle von (K 2) in Definition 1.1 lediglich

(SK 2) $\qquad (K\backslash\{0\}, \cdot)$ ist eine Gruppe.

gilt, heißen *Schiefkörper*. Die Quaternionen bilden also einen solchen.

Komplexe Zahlen und Quaternionen sind aber bereits die einzigen Beispiele für $\mathbb{R}$ echt umfassende Körper bzw. Schiefkörper, was im folgenden Satz präzisiert werden soll, dessen zweite Aussage als Satz von Frobenius bekannt ist.

Satz 1.2: (Maximale Charakterisierung)

(i) Es gibt genau einen Körper, nämlich den der komplexen Zahlen $\mathbb{C}$, der $\mathbb{R}$ als Teilkörper enthält und als $\mathbb{R}$–Vektorraum endliche Dimension hat.

(ii) Außer $\mathbb{C}$ gibt es noch genau einen Schiefkörper, nämlich den der Quaternionen $\mathbb{H}$, der $\mathbb{R}$ als Teilkörper enthält und als $\mathbb{R}$–Vektorraum endliche Dimension hat.

Wir dürfen also bei unseren Einbettungen, um nicht auf $\mathbb{C}$ und $\mathbb{H}$ beschränkt zu bleiben, bezüglich der algebraischen Multiplikation in (1.7) auch nicht mehr die Gruppeneigenschaft fordern, sondern müssen diesbezüglich zur Halbgruppe abschwächen, was uns auf die Struktur des *Ringes* führt.

Definition 1.4: (Ring mit Eins)

Ein Tripel $(R, +, \cdot)$ heißt *Ring*, wenn $(R, +)$ eine abelsche Gruppe, $(R, \cdot)$ eine Halbgruppe

ist, und wenn die Distributivgesetze gelten.
Existiert zusätzlich ein Element $1 \in R$ derart, daß für alle $r \in R$

$$1 \cdot r = r \cdot 1 = r \tag{1.10}$$

gilt, so wird $(R, +, \cdot)$ ein *Ring mit Eins* genannt.

In Ringen gilt stets (1.1), jedoch nicht notwendigerweise (1.2), d.h. es können Elemente existieren, die (1.2) verletzen. Zu ihrer exakten Kennzeichnung führen wir ein:

Definition 1.5: (Nullteiler)
Sei $(R, +, \cdot)$ ein Ring. Ein Element $a \in R$ heißt *Links–Nullteiler* (bzw. *Rechts–Nullteiler*), falls ein $b \in R \backslash \{0\}$ existiert mit $a \cdot b = 0$ (bzw. $b \cdot a = 0$).

Eine unmittelbare Konsequenz aus dem Vorhandensein von Nullteilern in einer algebraischen Struktur ist, daß die Division von Elementen nicht erklärt ist, was z.B. das Lösen von Gleichungen sehr schwierig gestalten kann. Mit den konkreten Auswirkungen für den Entwurf Neuronaler Netze im Rahmen derartiger Strukturen werden wir uns später noch ausführlich beschäftigen müssen.

Eine genauere formale Spezifikation der algebraischen Struktur unserer angestrebten Einbettungen können wir schließlich mit Hilfe der folgenden Definition angeben.

Definition 1.6: (Lineare Algebra)
Sei K ein Körper, $(R, +, \otimes)$ ein Verknüpfungsgebilde.
Falls $(R, +, \otimes)$ ein Ring mit einem Einselement und $V := (R, +, \cdot_K)$ ein Vektorraum über K ist, so heißt das Paar $\mathcal{L} = (V, \otimes)$ eine *Lineare Algebra* über K.

Ist V von endlicher Dimension, so wird diese auch die *Ordnung* von $\mathcal{L}$ genannt, desweiteren wird V als der *Vektorraumanteil* von $\mathcal{L}$ bezeichnet.

Aus unseren bisherigen Überlegungen können wir nun direkt ableiten, daß unser algebraisches Modell genau aus den endlich dimensionalen reellen *Linearen Algebren* besteht. Die unter diesen von $\mathbb{R}$, $\mathbb{C}$ und $\mathbb{H}$ verschiedenen Algebren wollen wir von nun an als *hyperkomplexe Zahlen* bezeichnen wollen.

Wir müssen uns nun noch davon überzeugen, daß unsere algebraische Spezifikation sinnvoll ist, d.h. zum einen müssen wir uns vergewissern, daß es überhaupt hyperkomplexe Zahlen gibt. Desweiteren müssen wir sicherstellen, daß wir keine relevanten Einbettungen per Definition ausgeschlossen haben. Diesen komplizierteren Fall wollen wir separat als erstes behandeln.

Note über Divisionsalgebren

In Definition 1.6 haben wir bezüglich der Multiplikation der Algebra die Assoziativität gefordert. Verzichtet man darauf und betrachtet diese Eigenschaft als Attribut, so erhält man alternativ:

Definition 1.6': (Lineare Algebra)
Sei K ein Körper, $(A, +, \otimes)$ ein Verknüpfungsgebilde.
Falls $V := (A, +, \cdot)$ ein Vektorraum ist, so heißt das Paar $\mathcal{A} = (V, \otimes)$ eine *Algebra* über K.
Ist $(A, \otimes)$ zusätzlich eine Halbgruppe, so heißt $\mathcal{A}$ weiter eine assoziative Algebra.

Ist $(A\backslash\{0\}, \otimes)$ abgeschlossen, und existiert zu jedem Element ein linksinverses und rechtsinverses Element (bezüglich $\otimes$), so wird $\mathcal{A}$ eine *Divisionsalgebra* genannt.

Nach Satz 1.2 sind $\mathbb{C}$ und $\mathbb{H}$ die einzigen assoziativen Divisionsalgebren (außer $\mathbb{R}$ selbst), die als $\mathbb{R}$–Vektorraum endliche Dimension haben. Es gibt darüberhinaus nur noch genau eine nicht–assoziative Divisionsalgebra in diesem Sinne, die sogenannten *Octonionen* [Porteous95], die die Ordnung 8 haben.

Diese Aussage können wir im Rahmen der Arbeit nicht beweisen und skizzieren daher nur kurz die Beweisidee:
Man kann zeigen, daß derartige Divisionsalgebren stets eine Zweierpotenz als Ordnung besitzen müssen. Weiter kann man dann folgern, daß wenn es keine Divisionsalgebra der Ordnung 16 gibt, auch keine höherdimensionalen Divisionsalgebren mehr existieren. Eine Divisionsalgebra der Ordnung 16 müßte die Octonionen als Subalgebra enthalten, dieses ist aber bereits nicht mehr realisierbar.

Die Octonionen selbst werden in unseren noch zu konstruierenden algebraischen Einbettungen als isomorphe Kopie enthalten sein, wobei hier Isomorphie im Sinne von Definition 1.6' gemeint ist.

Insgesamt ist damit gewährleistet, daß alle relevanten Möglichkeiten algebraischer Einbettungen des $\mathbb{R}^n$ in unserem Modell enthalten sind.

Die Frage nach der Existenz weiterer hyperkomplexer Zahlen, außer den bisher kennengelernten, gibt uns Gelegenheit zu einer ersten kleinen Anwendung des Schließens mittels Isomorphie. Nach einem Satz der Algebra ist bekanntlich jede Lineare Algebra isomorph zu einer Matrizenalgebra. Die Existenz von Matrizenalgebren beliebiger endlicher Ordnung ist wohl unzweifelhaft gegeben, womit gleichsam deutlich wird, wie grob unsere Spezifikation noch ist.

Wir wollen die Frage aber auch noch explizit beantworten. Ein sehr einfaches Beispiel liefert sofort der $\mathbb{R}^2$ mit der gewöhnlichen Addition und der komponentenweisen Multiplikation:

$$\otimes : \mathbb{R}^2 \odot \mathbb{R}^2 \to \mathbb{R}^2,\ ((a, b), (c, d)) \mapsto (ac, bd)\,. \tag{1.11}$$

Auf diese Weise läßt sich der $\mathbb{R}^n$ offensichtlich stets zu einem Ring erweitern, daher

definieren wir allgemeiner

$$ {}^{n}\mathcal{R} := (\mathbb{R}^{n}, +, \odot), \tag{1.12} $$

wobei $\odot$ jetzt allgemein die komponentenweise Multiplikation auf dem $\mathbb{R}^{n}$ bezeichne.

Die obigen Ringe ${}^{n}\mathcal{R}$ sind allerdings hinsichtlich ihrer geometrischen Interpretierbarkeit, als eher bescheiden zu bezeichnen.

Aber wir wollten ja auch nur ein erstes Beispiel für hyperkomplexe Zahlen kennenlernen, für die wir als Zusammenfassung nocheinmal ausdrücklich festhalten:

> Hyperkomplexe Zahlen sind i.a. weder frei von Nullteilern noch kommutativ.

Nachdem nun eine algebraische Spezifikation etabliert wurde, ist es an der Zeit, zu formulieren, was wir innerhalb dieser Grenzen modellieren wollen, d.h. unseren algebraischen Einbettungen mittels unserer Vorstellungen und Intentionen „Leben einzuhauchen“, womit der Übergang zu konkreten Konstruktionen eingeleitet wird.

1.3 Konstruktion der Einbettungen

Als erstes müssen wir uns natürlich überlegen, wie unser anstehendes Konstruktionsverfahren prinzipiell funktionieren soll. Wir haben den Ausgangs– und Zielpunkt aber noch zu vage bestimmt, um dabei direkt nur mittels Technik vorgehen zu können. Also nutzen wir die vorteilhafte Notwendigkeit auf Anschauungen und Vorstellungen angewiesen zu sein, und bemühen uns zunächst um inhaltliche Einsichten.

1.3.1 Vorüberlegungen

Diese gewinnt man häufig am einfachsten durch einen Blick auf bereits Bekanntes. Betrachten wir doch nocheinmal die Quaternionen $\mathbb{H}$ und ihre Entstehung. HAMILTONS' Absicht war es natürlich nicht, einen maximalen Schiefkörper zu entdecken, der $\mathbb{R}$ enthält. Vielmehr hatte er die Absicht, mit Vektoren des $\mathbb{R}^{3}$ so rechnen zu können, wie mit reellen Zahlen, insbesondere also auch Vektoren durch Vektoren teilen zu können. Durch den Satz 1.2 wissen wir, daß dies im $\mathbb{R}^{3}$ unmöglich ist. Erst dessen geschickte Einbettung in einen 4–dimensionalen Raum führte ihn zur Verwirklichung seiner Ziele. Was aber können wir dem Ergebnis seiner Bemühungen hilfreiches entnehmen?

Der Schlüssel ist die Beobachtung, daß ein Quaternion

$$ a + b\,\mathrm{i} + c\,\mathrm{j} + d\,\mathrm{k} \tag{1.13} $$

eigentlich kein homogenes Gebilde mehr ist (bzw. sein kann), sondern aus qualitativ verschiedenen Teilen besteht, nämlich aus dem *Skalarteil* a und dem *Vektorteil* $b\,\mathrm{i} + c\,\mathrm{j} + d\,\mathrm{k}$, wofür wir den Grund ja nun kennen.

Versuchen wir uns gleich am allgemeinen Fall des $\mathbb{R}^n$ und untersuchen, welche qualitativ verschiedenen Objekte wir aus seinen Vektoren prinzipiell bilden könnten. Das Projizieren in irgendeine Komponente eines Vektors liefert stets ein Skalar, womit wir die kleinstmögliche Entität erhalten. Offenbar gibt es kein Objekt, das qualitativ zwischen Skalar und Vektor liegt. Um „größere“ Objekte als Vektoren zu erzeugen, benötigen wir einen Operator mit expandierender Wirkung. Aufgrund der Endlichkeit unseres Vektorraumes können wir aber keine Entitäten erzeugen, die größer sind als die Dimension des Vektorraumes selbst. Wir brauchen als Gegenstück also noch einen kontrahierenden Operator, um nicht unbeschränkt zu expandieren. Gewiß wollen wir auch alle möglichen Entitäten erzeugen.

Mit diesen einfachen Überlegungen ist die Konstruktion ihrem Wesen nach aber bereits bestimmt. Es handelt sich einfach um eine Potenzmengenkonstruktion, und die Objekte, auf die sie angewandt wird, sind die Basisvektoren des Vektorraumes.

Für den $\mathbb{R}^n$ mit der Standardbasis $\{e_1, \ldots, e_n\}$ erhalten wir konkret die Potenzmenge

$$\mathcal{P}(n) := \{e_\emptyset, e_{\{1\}}, e_{\{2\}}, \ldots, e_{\{n\}}, e_{\{1,2\}}, \ldots, e_{\{1,\ldots,n\}}\} \tag{1.14}$$

wobei wir die Indexschreibweise (z.B. $e_{\{1,2\}}$ für $\{e_1, e_2\}$) verwendet haben, um gleichwohl anzudeuten, daß die obige Menge Basis des Vektorraumanteils einer Linearen Algebra werden soll. Die Menge $\mathcal{P}(n)$ läßt sich mittels der Mengeninklusion $\subseteq$ auf kanonische Weise (wie durch die Indizierung ebenfalls angedeutet) ordnen, was ein 2^n–Tupel ergibt. Selbiges bezeichne nun die Standardbasis des $\mathbb{R}^{2^n}$. Dann können wir mit Hilfe der Indexmenge $\mathcal{A}_n := (\mathcal{P}(n), \subseteq)$ jeden Vektor $x \in \mathbb{R}^{2^n}$ schreiben als:

$$x = \sum_{A \in \mathcal{A}_n} x_A e_A \,. \tag{1.15}$$

Für diese Indexmenge treffen wir noch zwei weitere Vereinbarungen.

Definition 1.7: (Basisindizierung)
Mit der Indexmenge $\mathcal{A}_n$ werde auch die zugrundeliegende Basis identifiziert. Weiter sei für alle $s \in \{1, \ldots, n\}$

$$\mathcal{A}_n^s := \{T \subseteq \mathcal{A}_n \mid |T| = s\} \,. \tag{1.16}$$

Die Teilausdrücke von (1.14) haben bisher noch keinerlei Bedeutung. Denken wir sie uns nun durch einen Operator $\wedge$ verknüpft, den wir als *das äußere Produkt* bezeichnen wollen, dann erhalten wir die folgende neue Gestalt von (1.14) als 2^n–Tupel:

$$(e_\emptyset, e_1, e_2, \ldots, e_n, e_1 \wedge e_2, \ldots, e_1 \wedge \ldots \wedge e_n) \,. \tag{1.17}$$

Wenn unsere Umformung konsistent gewesen sein soll, dann muß $\wedge$ ein Operator mit expandierender Wirkung sein. Genauer sollte z.B. $e_1 \wedge \ldots \wedge e_n$ eine Entität der „Größe n“

ergeben. Formal läßt sich dies sehr leicht bewerkstelligen, indem wir einfach jedem Produkt p in (1.17) die Anzahl seiner Faktoren, bezeichnet als der *Grad* von p, zuordnen. Diese Zuordnung geschähe durch den einstelligen *Gradoperator* $\langle\cdot\rangle$. Produkte p mit $n := \langle p\rangle \geq 2$ bezeichnet man auch als n–Vektoren. Gebräuchlich sind auch die Bezeichnungen *Bivektor* für 2–Vektoren und *Trivektor* für 3–Vektoren. Per Konvention gelte wie üblich $\langle e_\emptyset\rangle := 0$ (leeres Produkt) und $\langle e_i\rangle := 1$ für alle $i \in \{1,\ldots,n\}$.

Das äußere Produkt soll nun formal eingeführt werden. Als Bedeutung wird ihm dabei das von seinen Faktoren erzeugte Volumenelement zugeordnet.

Definition 1.8: (Äußeres Produkt)
Sei $m \in \mathbb{N}$. Das äußere Produkt beliebiger Vektoren $v_1,\ldots,v_m \in \mathbb{R}^n$ sei definiert als

$$\bigwedge_i^m := \det(v_1,\ldots,v_m)\,, \tag{1.18}$$

wobei für die Standardbasis $(e_1,\ldots,e_n)$ des $\mathbb{R}^n$

$$\det(e_1,\ldots,e_n) = 1 \tag{1.19}$$

gelte, d.h. *det* ist die rechtsorientierte Determinantenform des $\mathbb{R}^n$.

Abbildung 1.1 zeigt das äußere Produkt von drei linear unabhängigen Vektoren, i.e. einen Trivektor.

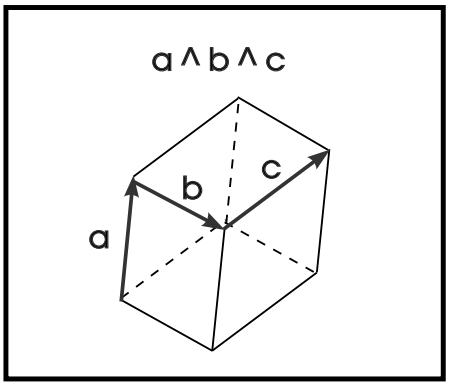


Abbildung 1.1: *Veranschaulichung eines Trivektors*

Es übertragen sich natürlich alle Eigenschaften der Determinante auf $\wedge$, insbesondere die Assoziativität und die Neutralität der Eins. Das äußere Produkt wird in unsere Einbettungen als geometrisches Fundament einfließen, für deren Konstruktionsverfahren wir jetzt nocheinmal unseren Blickwinkel ändern.

1.3.2 Das Verfahren

Den Ausgangspunkt unseres Konstruktionsverfahrens sollten wir möglichst direkt an einem Neuronalen Netz selbst bestimmen. Für unsere Konstruktion reicht dafür aber die

Beschäftigung mit einem einzelnen Neuron, wie es in Abbildung 1.2 dargestellt ist, völlig aus.

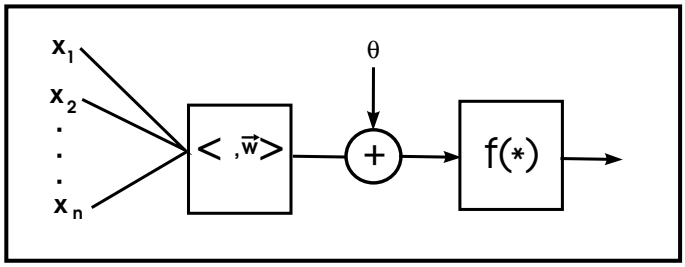


Abbildung 1.2: *Skalare Korrelation in einem Neuron*

Unser Interesse beschränkt sich im Moment sogar einzig auf die Korrelation eines Eingabedatums $\vec{x}$ mit einem Gewichtsvektor $\vec{w}$ mittels des Euklidischen Skalarproduktes $\langle \cdot, \cdot \rangle$, welches bekanntermaßen wie folgt definiert ist:

$$\langle x, y \rangle := \sum_{i=1}^{n} x_i \cdot y_i \,. \tag{1.20}$$

Der Schwellwert θ hätte in die Korrelation mit aufgenommen werden können, was aber für unsere Überlegungen hier unerheblich ist. Unser Ziel ist es, an Stelle des Euklidischen Skalarproduktes, die algebraische Multiplikation der konstruierten Einbettung als Korrelator im Neuron zu verwenden

$$f(\langle \vec{w}, \vec{x} \rangle + \theta) \rightsquigarrow f(\vec{w} \otimes \vec{x} + \theta) \,, \tag{1.21}$$

womit sich dann ein neuer Typus eines Neurons und somit auch neue Neuronale Netze ergeben würden. Die Veränderungen, die f dabei erfahren muß sind dann anschließend zu diskutieren.

Die Konstruktion unserer Einbettungen wird mit der Verallgemeinerung des Euklidischen Skalarproduktes beginnen, d.h. sie orientiert sich an dem im Neuron tatsächlich vorhandenen Korrelator und erweitert diesen. Wir geben im folgenden alle Schritte in allgemeinster Form an. Am Anfang steht dabei die Definition einer Bilinearform.

Definition 1.9: (Bilinearform)
Seien V und W K-Vektorräume. Eine Abbildung

$$F : V \times W \to K$$

heißt *Bilinearform*, wenn für alle $v, v_1, v_2 \in V$, $w, w_1, w_2 \in W$ und $\lambda \in K$ gilt:

(BF 1) Linearität in der 1.Komponente:

$$\begin{aligned} F(v_1 + v_2, w) &= F(v_1, w) + F(v_2, w) \,, \\ F(\lambda v, w) &= \lambda F(v, w) \end{aligned}$$

und

(BF 2) Linearität in der 2.Komponente:

$$\begin{aligned} F(v, w_1 + w_2) &= F(v, w_1) + F(v, w_2)\,, \\ F(v, \lambda w) &= \lambda F(v, w)\,. \end{aligned}$$

Im Falle $V = W$ heißt F eine Bilinearform auf V. Weiter existiert zu jeder solchen Bilinearform eine zugehörige quadratische Form $Q : V \to K$, die definiert ist durch

$$Q(u) := F(u, u) \text{ für alle } u \in V\,. \tag{1.22}$$

Unser Interesse gilt nur den symmetrischen Bilinearformen eines Vektorraumes V, den sogenannten *Skalarprodukten*. Dies sind Abbildungen, bei denen zu den Bedingungen (BF 1) und (BF 2) zusätzlich

$$F(u, v) = F(v, u) \tag{1.23}$$

für alle $u, v \in V$ erfüllt ist, so zum Beispiel beim Euklidischen Skalarprodukt auf dem $\mathbb{R}^n$, wie der Name schon sagt.

Skalarprodukte sind bereits eindeutig durch ihre quadratische Form bestimmt.
Es gilt:

$$F(u, v) = \frac{1}{2} \cdot (Q(u + v) - Q(u) - Q(v))\,. \tag{1.24}$$

Ein Skalarprodukt F läßt sich weiter klassifizieren als:

positiv definit	, wenn $\forall u \in V \backslash 0 :\ F(u, u) > 0$;
positiv semidefinit	, wenn $\forall u \in V :\ F(u, u) \geq 0$;
negativ definit	, wenn $\forall u \in V \backslash 0 :\ F(u, u) < 0$;
negativ semidefinit	, wenn $\forall u \in V :\ F(u, u) \leq 0$;
indefinit	, wenn F weder positiv noch negativ semidefinit ist.

Unser Standardbeispiel $\langle \cdot, \cdot \rangle$ ist positiv definit. Damit deutet sich schon an, worauf eine Verallgemeinerung des Euklidischen Skalarproduktes hinaus laufen wird.

Zu jedem Skalarprodukt F gehört noch eine ausgezeichnete Menge, das sogenannte *Radikal* von F (kurz *Rad* F), welches durch

$$Rad\ F := \{u \in V \mid F(u, v) = 0 \text{ für alle } v \in V\}\,. \tag{1.25}$$

definiert wird. Es enthält also genau die Vektoren, die zu allen anderen Vektoren senkrecht stehen, insbesondere also auch den Nullvektor. Enthält das Radikal darüberhinaus noch andere Vektoren, so spielen diese quasi die selbe Rolle wie die uns schon bekannten Nullteiler in Ringen. Die Nullteilerfreiheit unserer Einbettungen konnten wir nicht fordern, wie wir im Abschnitt 1.1 gesehen haben. Es besteht aber keine mathematische Notwendigkeit, Skalarprodukte mit Radikalen ungleich dem Nullvektor zu betrachten. Hingegen gibt es gute Gründe diesen Fall auszuschließen. So sind derartige Vektoren bei Konstruktionsverfahren generell problematisch, insbesondere würden sie aber in den konstruierten

Einbettungen weitere Nullteiler implizieren. Auch kann man derartigen Vektoren keine natürliche (bzw. von Null verschiedene) Länge mehr zuordnen, was eine Normierung des gesamten Raumes dann stets zu einem schwierigen Unterfangen macht.

Mit dem Ausschluß dieses Falls sind wir nun in der Lage den Satz anzugeben, welcher die gesamte Konstruktion tragen wird.

Satz 1.3: (Invariante Kennzeichnung Quadratischer Formen)
Sei V ein n–dimensionaler K–Vektorraum, F ein Skalarprodukt auf V mit $Rad\ F = \{0\}$, Q die zu F gehörige Quadratische Form. Dann existiert eine Basis $\{\tilde{e}_1, \ldots, \tilde{e}_n\}$ von V derart, daß für alle $v \in V$ gilt:

$$Q(v) = -v_1^2\tilde{e}_1 - v_2^2\tilde{e}_2 - \ldots - v_p^2\tilde{e}_p + v_{p+1}^2\tilde{e}_{p+1} \ldots + v_{p+q}^2\tilde{e}_{p+q}\,, \tag{1.26}$$

wobei $p, q \in \mathbb{N}$ und p+q=n gilt.

Das Paar (p, q) heißt der *Typ* der Quadratischen Form Q. Setzt man $s := p$, so ist (n, s) eine ebenfalls gebräuchliche Schreibweise für den Typ einer Quadratischen Form, welcher wir den Vorrang geben werden. In dieser Notation wird s dann auch die *Signatur* von Q, genannt, da an dieser Stelle der Vorzeichenwechsel stattfindet. Weiter wollen wir vereinbaren, daß $\mathbb{R}_{n,s}$ die gesamte Aussage von Satz 1.3 abkürzend für den $\mathbb{R}^n$ bezeichne.

Satz 1.3 erlaubt das Reduzieren einer Quadratischen Form auf ihren Typ. Wir werden nun zeigen, wie durch $\mathbb{R}_{n,s}$ zusammen mit unseren Überlegungen des vorangegangenen Teilabschnittes eine Lineare Algebra der Ordnung 2^n bezüglich der Basis (1.17) impliziert wird.

Setzt man in (1.26) die Vektoren der Standardbasis $(e_1, \ldots, e_n)$ des $\mathbb{R}^n$ ein, so erhält man sofort

$$-Q(e_i) = +1 \quad \text{falls} \quad i \leq p \tag{1.27}$$
$$-Q(e_i) = -1 \quad \text{falls} \quad i > p \tag{1.28}$$

und weiter für $1 \leq i, j \leq n$ mit $i \neq j$

$$e_i e_j = -e_j e_i\,, \tag{1.29}$$

da $(e_1, \ldots, e_n)$ orthonormal ist.

Mit den letzten Gleichungen sind wir schließlich am Ziel angelangt. Das Konstruktionsverfahren ist jetzt vollständig beschreibbar.

Es beginnt mit der Wahl eines Raumes $\mathbb{R}_{n,s}$ und soll als Ergebnis eine vollständige Multiplikationstabelle für

$$(1, e_1, e_2, \ldots, e_n, e_1 \wedge e_2, \ldots, e_1 \wedge \ldots \wedge e_n) \tag{1.30}$$

liefern, genauer eine Operation $\otimes_{n,s}$ auf dem $\mathbb{R}^{2^n}$ definieren, durch die dieser Raum zu einer Linearen Algebra wird.

Die so erzeugte Lineare Algebra wird dann die *Clifford–Algebra* von $\mathbb{R}_{n,s}$ genannt und mit $\mathcal{C}_{n,s}$ bezeichnet. Kommen wir nun aber zum eigentlichen Verfahren.

Konstruktionsverfahren
Sei also $\mathbb{R}_{n,s}$ bereits gewählt.

0) Setze für alle $i \in \{1, \ldots, n\}$:

$$e_i^2 := \begin{cases} 1 & : \quad i < s \\ -1 & : \quad i \geq s \end{cases}$$

1) Bestimme alle trivialen Produkte, d.h. die durch $\wedge$ allein festgelegten.

2) Bestimme die noch verbliebenen Produkte mittels (1.29) und 0).

Wie man damit zum Beispiel von $\mathbb{R}_{2,0}$ zu $\mathcal{C}_{2,0}$ gelangt, ist nachstehend demonstriert, wobei zur Vereinfachung $\wedge$ in den Ausdrücken weggelassen wurde.

Beispiel 1.1:

0) $\longrightarrow$ 1) $\longrightarrow$ 2) $\longrightarrow$ 3)

$\otimes_{2,0}$	1	e_1	e_2	e_1e_2
1	1	e_1	e_2	e_1e_2
e_1	e_1	**-1**	e_1e_2	
e_2	e_2	$-e_1e_2$	**-1**	
e_1e_2	e_1e_2			

$$\begin{aligned} (e_1e_2)e_1 &= -e_2e_1e_1 = e_2 \\ (e_1e_2)e_2 &= e_1e_2e_2 = -e_1 \\ e_1(e_1e_2) &= e_1e_1e_2 = -e_2 \\ e_2(e_1e_2) &= -e_2e_2e_1 = e_1 \\ (e_1e_2)(e_1e_2) &= -e_1e_1e_2e_2 = -1 \end{aligned}$$

$\otimes_{2,0}$	1	e_1	e_2	e_1e_2
1	1	e_1	e_2	e_1e_2
e_1	e_1	-1	e_1e_2	$-e_2$
e_2	e_2	$-e_1e_2$	-1	e_1
e_1e_2	e_1e_2	e_2	$-e_1$	-1

Betrachten wir das Ergebnis 3) genauer, so erkennen wir die Multiplikationstabelle der Quaternionen (Tabelle 1.1) wieder (man identifiziere einfach e_1 mit i, e_2 mit j und e_1e_2 mit k), die wir jetzt als eine spezielle Einbettung des $\mathbb{R}^2$ erhalten haben.

Mit der Angabe des Konstruktionsverfahrens ist die Charakterisierung unserer Einbettungen abgeschlossen. Der Einbettungsprozeß läßt sich nocheinmal wie folgt zusammenfassen

$$\mathbb{R}_{n,s} \prec \mathcal{C}_{n,s} \,. \tag{1.31}$$

Am Ende der Einbettungsprozesse stehen also die Clifford–Algebren, die somit das algebraische Modell für unsere Erweiterungen reeller Neuronaler Netze bilden, und nun genauer untersucht werden sollen.

1.4 Clifford–Algebren

Clifford–Algebren haben durch unser Konstruktionsverfahren bisher nur eine implizierte Charakterisierung erfahren. Wir geben für sie daher als erstes auch eine explizite Definition an.

Definition und Satz 1.10: (Clifford–Algebra)
Gegeben sei der Vektorraum $(\mathbb{R}^{2^n}, +, \cdot)$ mit Basis $\mathcal{A}_n$.
Sei weiter $s \in \{1, \ldots, n\}$ gegeben. Für alle $e_A, e_B \in \mathcal{A}_n$ definiere eine Operation $\otimes_{n,s}$ durch:

$$e_A \otimes_{n,s} e_B := (-1)^{|(A \cap B) \setminus \{1,\ldots,s\}|} (-1)^{p(A,B)} e_{A \triangle B}\,, \tag{1.32}$$

wobei $p(A,B) = \sum_{j \in B} |\{i \in A \,|\, i < j\}|$ ist und $\triangle$ die symmetrische Mengendifferenz bezeichne.
Dann ist $((\mathbb{R}^{2^n}, +, \cdot), \otimes_{n,s})$ eine Lineare Algebra, die sogenannte *Clifford–Algebra* vom Typ (n, s), welche kurz mit $\mathcal{C}_{n,s}$ bezeichnet wird.

Strenggenommen müßten wir jetzt noch zeigen, daß das Konstruktionsverfahren des letzten Abschnittes und (1.32) für festes s tatsächlich die selben Algebren erzeugen. Wir verzichten hier aber darauf, da wir ja keine reine Mathematik betreiben wollen. Uns interessiert vielmehr, wie wir von den beiden unterschiedlichen Zugängen profitieren können.

Der implizite Zugang, i.e. das Konstruktionsverfahren, beschreibt Einbettungen als einen Vorgang. Die explizite Definition 1.10 stellt hingegen die selben Einbettungen als algebraisches Modell dar. Hätten wir nur letzteres zur Verfügung, so könnten wir zwar Neuronale Netze bezüglich dieses Modells untersuchen, aber ohne jeden Unterbau und jedwede geometrische Anschauung. Letztere ließe sich sogar noch wesentlich vertiefen, wenn wir Clifford–Algebren in der in [Hestenes84] entwickelten Interpretation der *Geometrischen Algebren* betrachten würden. Warum wir uns in dieser Arbeit dann doch fast nur auf der rein algebraischen Seite bewegen werden, liegt daran, daß wir ersteinmal rein formal konkrete und korrekte Neuronaler Netze in diesem Rahmen entwickeln müssen. Das Einbringen von Geometrie in Neuronale Netze ist dabei natürlich stets unser Antrieb und Fernziel.

Kehren wir nun wieder zur technischen Seite zurück und führen noch ein paar übliche Bezeichnungen im Rahmen der Clifford–Algebra ein, die über unseren impliziten Zugang leicht nachvollziehbar sind.

Die Operation $\otimes_{n,s}$ wird *geometrisches Produkt* genannt, ein $x \in \mathcal{C}_{n,s}$ heißt *Multivektor*. Ist klar, in welchen Clifford–Algebren das geometrische Produkt gebildet wird, so schreiben wir dann auch einfach nur $\otimes$. Analog wie bei Quadratischen Formen wird s die *Signatur* von $\mathcal{C}_{n,s}$ genannt. Eine Clifford–Algebra mit Signatur 0 wird auch als Euklidische Clifford–

Algebra bezeichnet. Dies gibt uns auch gleich Gelegenheit zu:

Bemerkung 1.1:

(*i*) Den reellen Zahlen entspricht die Clifford–Algebra $C_{0,0}$.
(*ii*) Den komplexen Zahlen entspricht die Clifford–Algebra $C_{1,0}$.
(*iii*) Den Quaternionen entspricht die Clifford–Algebra $C_{2,0}$.

Diese Algebren werden auch als die fundamentalen Clifford–Algebren bezeichnet, sie sind sämtlich Euklidische Clifford–Algebren. Beide Attribute werden wir etwas später noch genauer verstehen. Zunächst noch eine weitere kleine Bemerkung.

Bemerkung 1.2:

(*i*) Jede Clifford–Algebra hat als Ordnung eine Zweierpotenz.
(*ii*) Zu jedem $n \in \mathbb{N}$ gibt es genau n Clifford–Algebren der Ordnung 2^n.
(*iii*) Die Einträge in den Multiplikationstabellen zweier verschiedener Clifford–Algebren gleicher Ordnung unterscheiden sich höchstens durch Vorzeichen.

Die letzte Aussage wird leicht ersichtlich, wenn wir eine Beispielrechnung für (1.32) betrachten.

Beispiel 1.2:

Wir wollen $e_{\{1,2\}} \otimes_{2,0} e_{\{1\}}$ berechnen. Es gilt unabhängig von $s: e_{\{1,2\}} \triangle e_{\{1\}} = e_{\{2\}}$.
Weiter ist $|(\{1,2\} \cap \{1\}) \setminus \emptyset| = 1$ und $p(\{1,2\},\{1\}) = 1$, also insgesamt:
$e_{\{1,2\}} \otimes_{2,0} e_{\{1\}} = (-1)^1 \cdot (-1)^1 \cdot e_{\{2\}} = e_{\{2\}}$,
genau wie im Beispiel 1.1 auf konstruktive Weise hergeleitet.

Aussage (*iii*) von Bemerkung 1.2 gibt zu der Vermutung Anlaß, daß es häufig Isomorphismen zwischen Clifford–Algebren gleicher Ordnung geben wird. Dabei reicht es natürlich dann, die Isomorphie der Algebren wie in Definition 1.11 zu fordern, da die Vektorraumanteile der Algebren identisch sind.

Definition 1.11: (Isomorphie von Clifford–Algebren)

Zwei Clifford–Algebren C_{n,s_1} und C_{n,s_2} heißen *isomorph* (in Zeichen $C_{n,s_1} \cong C_{n,s_2}$), wenn es eine Bijektion ϕ von C_{n,s_1} nach C_{n,s_2} derart gibt, daß für alle $x, y \in C_{n,s_1}$ gilt:

$$(x+y)\,\phi = x\phi + y\phi \tag{1.33}$$

$$(x \otimes_{n,s_1} y)\,\phi = x\phi \otimes_{n,s_2} y\phi\,. \tag{1.34}$$

Für die Angabe von Isomorphien zwischen Clifford–Algebren ist die (p,q)–Notation handlicher, zu welcher wie zu diesem Zwecke kurz übergehen wollen. Unter den Isomorphieaussagen ist das sogenannte *Periodizitätstheorem*

$$\mathcal{C}_{p,q+8} \cong \mathcal{C}_{p,q} \bullet \mathbb{R}(16) \tag{1.35}$$

von zentraler theoretischer Bedeutung, wobei $\bullet$ das Tensor–Produkt und $\mathbb{R}(16)$ die Matrizenalgebra der reellen 16×16–Matrizen bezeichne. Es besagt vereinfacht, daß sich mit einer Periodizität von 8 die inneren strukturellen Eigenschaften von Clifford–Algebren wiederholen. Uns interessieren natürlich mehr praktische Kriterien, ein sehr hilfreiches ist:

$$\mathcal{C}_{p+1,q} \cong \mathcal{C}_{q,p+1} \,. \tag{1.36}$$

Von nun ab gelte wieder die Standardnotation (n, s). Eine unmittelbare Folgerung aus dem letzten Kriterium ist dann

$$\mathcal{C}_{2,1} \cong \mathcal{C}_{2,2} \,. \tag{1.37}$$

Es gibt also nur zwei nicht–isomorphe Clifford–Algebren der Ordnung 4. Abschließend zu diesem Themenkreis wollen wir noch bemerken, daß jede Clifford–Algebra zu einer Matrizenalgebra aus nicht gemischten Produkten über $\mathbb{R}\,, \mathbb{C}$ oder $\mathbb{H}$ isomorph ist [1], weshalb diese Algebren selbst auch die fundamentalen Clifford–Algebren genannt werden.

Als letztes wollen wir jetzt die Verallgemeinerung der komplexen Konjugation in Clifford–Algebren formulieren.

Definition und Satz 1.12: (Konjugation)

Sei $x \in \mathcal{C}_{n,s}$. Der zu x *konjugierte Multivektor* $\bar{x}$ ist definiert durch

$$\bar{x} := \sum_{A \in \mathcal{A}_n} (-1)^{\frac{|A|\,(|A|+1)}{2}} x_A e_A \,. \tag{1.38}$$

Die Abbildung $^-$ ist ein Automorphismus von $\mathcal{C}_{n,s}$ und wird die *Konjugation* in $\mathcal{C}_{n,s}$ genannt.

Die Konjugation ist also alleinig durch die Ordnung der Clifford–Algebra und nicht durch deren Signatur bestimmt. Für $\mathcal{C}_{1,0} (= \mathbb{C})$ erhält man die komplexe Konjugation als Spezialfall aus der obigen Definition. In jeder Euklidischen Clifford–Algebra gilt:

$$x \otimes \bar{y} = (\langle x, y \rangle, \ldots), \tag{1.39}$$

für $x = y$ insbesondere sogar:

$$x \otimes \bar{x} = (\langle x, x \rangle, 0, \ldots, 0) \,, \tag{1.40}$$

womit dann auch der Name dieser Clifford–Algebren erklärt ist.

Diese Gleichungen haben eine sehr wichtige, positive Konsequenz für den Entwurf Neuronaler Netze in Clifford–Algebren, denn zumindest für Euklidische Clifford–Algebren ist der skalare Korrelator im geometrischen Korrelator einbettbar. Wir können unsere abstrakten theoretischen Überlegungen also ganz natürlich auf die Ebene der Neuronalen Netze übertragen und dort nun weiterentwickeln.

[1] für eine diesbezügliche Übersicht sei auf Anhang C verwiesen

Kapitel 2

Entwurf Neuronaler Netze in Clifford–Algebren

In diesem Kapitel wollen wir untersuchen, welche herkömmlichen vorwärtsgerichteten Neuronalen Netze in Clifford–Algebren sinnvoll übertragbar sind. Dazu müssen wir uns zunächst einen Überblick über den Aufbau und die Funktion gewöhnlicher vorwärtsgerichteter Neuronaler Netze verschaffen. Der Begriff des vorwärtsgerichteten Neuronalen Netzes faßt aber eigentlich noch sehr unterschiedliche neuronale Berechnungskonzepte zusammen. Die wichtigsten Vertreter darunter sind die *Multilayer Perceptrons* (MLPs) und die *Radialen–Basisfunktionen–Netze* (RBF–Netze), auf die wir uns auch im wesentlichen beschränken werden. Aufgrund ihrer verschiedenen Funktionsprinzipien hat jedes dieser beiden Netze gewisse Vorteile in bestimmten Anwendungsgebieten gegenüber dem anderen. Für Funktionsinterpolationen ist das MLP populärer als das RBF–Netz, bei Klassifizierungsaufgaben verhält sich die Sache meist umgekehrt.

Im folgenden werden wir das Attribut „vorwärtsgerichtet“ meist nicht mehr ausdrücklich erwähnen. Weiter sollen mit Neuronalen Netzen auch stets die reellen Standard–Netze gemeint seien.

2.1 Vorwärtsgerichtete Neuronale Netze

Die Aufgabe, die ein Neuronales Netz lernen soll, kann allgemein als Problem der Approximation einer Funktion formuliert werden. Die zu approximierende Funktion $F : \mathbb{R}^N \to \mathbb{R}^P$ ist dabei nicht analytisch beschrieben, sondern lediglich durch K Abtastpunkte (x^k) und zugehörige Funktionswerte (y^k) gegeben, welche von einem (häufig unbekannten) Prozeß stammen. Die Menge all dieser Prozeßdaten sei mit S bezeichnet. Wie wir erst später sehen werden, ist ein Neuronales Netz mit einer versteckten Schicht und linearen Ausgabeknoten, wie es Abbildung 2.1 zeigt, ausreichend, um universelle Approximatoren

zu erhalten. Wir können uns also ganz auf die versteckte Schicht konzentrieren.

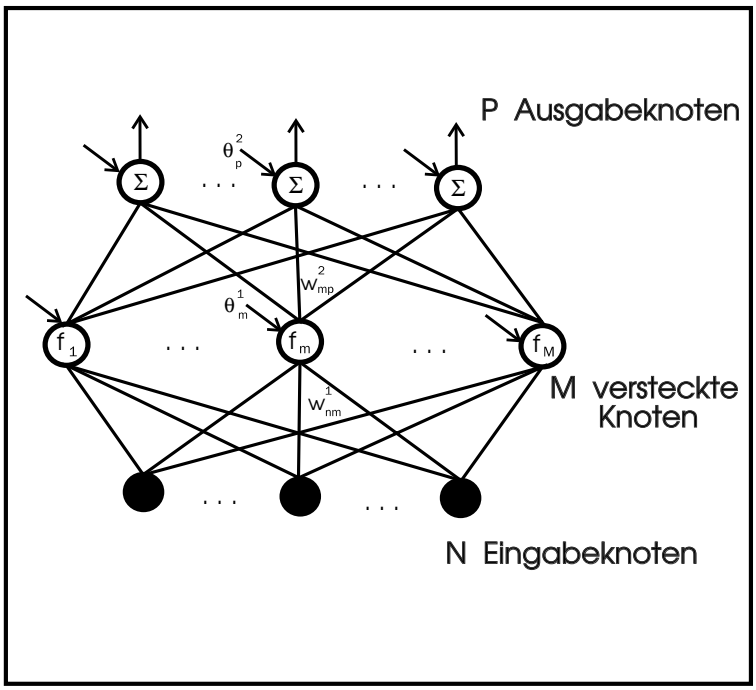


Abbildung 2.1: *Vorwärtsgerichtetes Neuronales Netz*

Jeder der M versteckten Knoten repräsentiert eine *Basisfunktion*

$$f_m : \mathbb{R} \to \mathbb{R} \qquad (1 \leq \mathrm{m} \leq \mathrm{M}). \tag{2.1}$$

Die Gewichte w^1_{nm} und die Schwellwerte θ^1_m sind die Parameter dieser Basisfunktionen. Mit den Verbindungen von der versteckten Schicht zu den Ausgabeknoten sind Gewichte w^2_{mp} assoziiert, θ^2_p sind die Schwellwerte der Ausgabeknoten. Die oberen Indizes geben die Schicht im Netzwerk an, für die unteren gilt: $(1 \leq n \leq N), (1 \leq m \leq M)$ und $(1 \leq p \leq P)$. Also bezeichnet w^2_{mp} das Gewicht vom m–ten Knoten der versteckten Schicht zum p–ten Ausgabeknoten und so weiter. Ferner sei $w^1 := \{w^1_{nm}, \theta^1_m\}$, $w^2 := \{w^2_{mp}, \theta^2_p\}$ und schließlich $\omega := \{w^1, w^2\}$ die Menge aller Parameter des Netzes. Desweiteren wollen wir noch wie üblich als Konvention vereinbaren, daß Mengenbildung über fehlende Indizes zugelassen ist.

Die Funktionalität des Netzes ist durch

$$\mathcal{N}^S_\omega : \mathbb{R}^{\mathrm{N}} \to \mathbb{R}^{\mathrm{P}}, \tag{2.2}$$

gegeben, das durch $\mathcal{N}^S_\omega$ berechnete p–te Ausgabedatum zu einem Eingabedatum $x^k = (x^k_1, \ldots, x^k_N)$ ist

$$\mathcal{N}^S_\omega(x^k)_p = \sum_{m=1}^{M} w^2_{mp} \cdot f_m(w^1_m; \theta^1_m; x^k) + \theta^2_p. \tag{2.3}$$

Im folgenden sei mit $\mathcal{N}^S_\omega$ stets auch das Netz selbst bezeichnet, d.h. wir identifizieren es mit seiner Berechnungsfunktion. Zur besseren Unterscheidung und als Abkürzung für eine Netzausgabe definieren wir zusätzlich noch $o_p := \mathcal{N}^S_\omega(x^k)_p$.

Das Netz $\mathcal{N}_\omega^S$ soll nun durch das Trainieren der Muster aus S eine möglichst gute Approximation von F lernen. Was dies genau bedeutet und wie es realisiert werden kann, soll erst später geklärt werden. Für den Moment begnügen wir uns mit der Feststellung, daß die Approximationsfähigkeiten eines Netzes durch die verwendeten Basisfunktionen bestimmt werden. Verschaffen wir uns daher einen Überblick über mögliche Varianten von Basisfunktionen mittels der folgenden charakterisierenden Eigenschaften:

0) keine Basisfunktionen

1) parametrisiert / nicht–parametrisiert

2) lokal / global

3) orthogonal / nicht–orthogonal,

die wir nun einzeln ein wenig näher untersuchen wollen.

Keine Basisfunktionen. Das ist natürlich strenggenommen gar keine Eigenschaft mehr und wurde auch nur aufgeführt, um zu betonen, daß ein Netz mit einer linearen Schicht nicht ausreichend ist. Viele einfache Probleme sind nämlich bereits nicht mehr linear separabel, also für derartige Netze unlösbar. Das zweifellos bekannteste Beispiel dafür ist das XOR–Problem.

Parametrisierung. Nicht parametrisierte Basisfunktionen sind solche, die außer der Eingabe selbst, keine weiteren variablen Parameter haben, also nicht von veränderlichen Gewichten abhängen. Dann vereinfacht sich (2.3) also zu:

$$\mathcal{N}_\omega^S(x^k)_p = \sum_{m=1}^{M} w_{mp}^2 \cdot f_m(x^k) + \theta_p^2 \,. \tag{2.4}$$

Die Basisfunktionen selbst sollten selbstverständlich nichtlinear sein, da ein Netz mit zwei linearen Schichten äquivalent zu einem Netz mit einer linearen Schicht ist. Durch ihre geschickte Wahl wird man ein spezielles Problem meist sehr gut lösen können, aber dann i.a. damit auch nur dieses. Der Ansatz ist generell zu unflexibel, wir betrachten folglich parametrisierte Basisfunktionen.

Lokalität. Lokale Basisfunktionen sind dadurch gekennzeichnet, daß sie nur in einem begrenzten Gebiet des Eingaberaumes (also lokal) einen signifikanten Output erzeugen. Das einfachste Beispiel wäre eine *Lookup*–Tabelle, d.h. die Basisfunktionen würden in dem Falle die Diskrete Metrik realisieren, wobei sich dann die Basisfunktionen natürlich auch nicht überlappen würden. Die Realisierung der Lokalität über die Euklidische Metrik führt zu RBF–Netzen. Globalität bedarf keiner gesonderten Erklärung und führt auf MLPs. Beide Typen sollen auf ihre Erweiterbarkeit für Clifford–Algebren hin untersucht werden.

Orthogonalität. Orthogonalisierung ist eine generelle Methode, um sich nicht überlappende Basisfunktionen zu erzeugen. Orthogonalität ist über den Begriff des Skalarproduktes definiert, genauer bezeichnet er das *senkrecht stehen* von Vektoren (siehe auch (1.25)),

wobei die Vektoren hier natürlich Elemente eines Funktionenraumes sind.Wir geben ein kleines Beispiel. Zwei Funktionen $f_i, f_j \in C^1[a,b]$ sind orthogonal, wenn gilt:

$$\int_a^b f_i(x)\, f_j(x)\, dx = c\delta_{i,j}\,, \tag{2.5}$$

wobei $\delta_{i,j}$ das Kronecker Symbol und $c \in \mathbb{R}$ ist. Orthogonale Basisfunktionen haben den Vorteil, daß man neue Funktionen zur Basis hinzufügen oder herausnehmen kann, ohne die Parameter der alten Basisfunktionen neu bestimmen zu müssen. Wichtige Vertreter für orthogonale Funktionen sind orthogonale Wavelets. Im Rahmen dieser Grundlagenarbeit können wir sie allerdings nicht behandeln.

Nachdem wir uns diesen Überblick über mögliche Eigenschaften von Basisfunktionen verschafft haben, ist es an der Zeit unsere vorwärtsgerichteten Neuronalen Clifford–Algebra Netze konkret einzuführen. Dazu übertragen wir einfach das Netz aus Abbildung 2.1 vom Anfang dieses Abschnittes mit all seinen Notationen in eine Clifford–Algebra.

Definition 2.1: (Neuronales Clifford–Algebra Netz)
Ein *Neuronales Clifford–Algebra Netz* (CA–Netz) ist allgemein ein vorwärtsgerichtetes Neuronales Netz, dessen Elemente Multivektoren sind oder auf Multivektoren operieren. Die Funktionalität eines CA–Netzes ist gegeben durch:

$$\mathcal{CN}_\omega^S : (\mathcal{C}_{n,s})^N \to (\mathcal{C}_{n,s})^P\,. \tag{2.6}$$

Wir betrachten im speziellen CA–Netze mit einer Architektur gemäß Abbildung 2.1, d.h. mit linksseitiger Gewichtsmultiplikation. Der durch ein derartiges Netz $\mathcal{CN}_\omega^S$ berechnete p–te Ausgabemultivektor zu einem Eingabemultivektor $x^k = (x_1^k, \ldots, x_N^k)$ ist

$$\mathcal{CN}_\omega^S(x^k)_p \;=\; \sum_{m=1}^{M} w_{mp}^2 \otimes_{n,s} F_m(w_m^1; \theta_m^1; x^k) + \theta_p^2\,, \tag{2.7}$$

wobei die Basisfunktionen die Funktionalität

$$F_m \;: \mathcal{C}_{n,s} \to \mathcal{C}_{n,s} \qquad (1 \le m \le M) \tag{2.8}$$

haben.

Die Beschränkung auf linksseitige Gewichtsmultiplikation hat keine inhaltlichen Konsequenzen. Wir mußten nur Aufgrund der i.a. gegebenen Nichtkommutativität des geometrischen Produktes eine formale Festlegung treffen, wobei wir natürlich die Konvention für Standard–Netze übernommen haben.

Befassen wir uns nun als nächstes mit der Frage nach der Erweiterbarkeit von RBF–Netzen zu Clifford–Algebra RBF–Netzen.

2.2 Diskussion von Clifford–Algebra RBF–Netzen

Die Überschrift nimmt es zum Teil schon vorweg, die Erweiterung von RBF–Netzen zu Clifford–Algebra RBF–Netzen ist nicht unbedingt angebracht. Als Argument könnten wir auch gleich angeben, daß (unseres Wissens nach) noch keine Komplexen RBF–Netze entwickelt wurden, was sicher nicht daran liegt, daß die Idee dazu noch niemandem gekommen ist. Nun, zum einen haben wir ein allgemeineres Modell zur Verfügung, zum anderen sollte man sich auch stets selbst überzeugen. Desweiteren ist es von großer Bedeutung zu erkennen, was genau der Grund für ein mögliches Scheitern sein könnte. Wäre nämlich die Ursache die Lokalität der Basisfunktionen, so wäre eine sehr große Klasse von Basisfunktionen generell für CA–Netze ausgeschlossen.

Kommen wir vom Konjunktiv jetzt direkt zu einem RBF–Netz $\mathcal{N}_\omega^S$. Ein solches Netz berechnet ein Ausgabedatum gemäß

$$\mathcal{N}_\omega^S(x^k)_p = \sum_{m=1}^{M} w_{mp}^2 \cdot f_m(\|w_m^1 - x^k\|) + \theta_p^2 \,, \tag{2.9}$$

d.h. das Argument einer Basisfunktion f_m ist der Euklidische Abstand zwischen Eingabedatum und dem Gewicht der Basisfunktion, welches auch das Zentrum von f_m genannt wird. Zu diesem wird also der *radiale* Abstand der Eingabe ermittelt. Die Basisfunktionen sind folglich translations- und rotationsinvariant.

Unsere Idee für den direkten Übergang von Standard–Netzen zu CA–Netzen ist es ja, den skalaren durch den geometrischen Korrelator zu ersetzen. Nach Gleichung (2.9) sind die Basisfunktionen aber formal gar nicht über das Skalarprodukt parametrisiert. Um einzusehen, daß dies doch der Fall ist, müssen wir uns genauer mit den Begriffen der Norm und der Metrik beschäftigen, über die sich der Abstandsbegriff formalisieren läßt.

Definition 2.2: (Norm)
Sei V ein K–Vektorraum. Eine Abbildung

$$\|\,\| : V \to \mathbb{R},\ \mathrm{v} \mapsto \|\mathrm{v}\| \tag{2.10}$$

heißt Norm auf V, falls für alle $v, w \in V$ und $\lambda \in K$ gilt:

(N 1) $\|\lambda\, v\| = \lambda\, \|v\|$

(N 2) $\|v + w\| \leq \|v\| + \|w\|$

(N 3) $\|v\| = 0 \ \Rightarrow\ v = 0$

Die reelle Zahl $\|v\|$ heißt *Norm* (auch: *Betrag*, *Länge*) des Vektors v.

Jedes Skalarprodukt impliziert eine Norm. Die Euklidische Norm läßt sich wie folgt über das Euklidische Skalarprodukt definieren:

$$\|\,\| : \mathbb{R}^n \to \mathbb{R},\ \mathrm{v} \mapsto \sqrt{\langle \mathrm{v}, \mathrm{v} \rangle}\,. \tag{2.11}$$

Definition 2.3: **(Metrik)**
Sei V ein K–Vektorraum. Eine Abbildung

$$d: M \times M \to \mathbb{R},\ (\mathrm{x},\mathrm{y}) \mapsto \mathrm{d}(\mathrm{x},\mathrm{y}) \tag{2.12}$$

heißt Metrik auf V, falls für alle $x, y, z \in V$ gilt:

(M 1) $d(x,y) = d(y,x)$

(M 2) $d(x,y) \leq d(x,z) + d(z,y)$

(M 3) $d(x,y) = 0 \Leftrightarrow x = y$

Jede Norm impliziert nun wiederum eine Metrik, indem man einfach $d(v,w) := \|v + w\|$ setzt. Wir können also für die Basisfunktionen in (2.9) auch schreiben:

$$f_m\left(\|w_m^1 - x^k\|\right) = f_m\left(\sqrt{\langle w_m^1 - x^k, w_m^1 - x^k\rangle}\right). \tag{2.13}$$

Insgesamt wird damit aber deutlich, daß wir eher ein inhaltliches als ein rein formales Problem haben, denn letztendlich soll stets der Abstand zwischen Eingabedatum und Zentrum einer Basisfunktion bestimmt werden, d.h. eine Metrik ausgewertet werden. Diese könnten wir vielleicht auch noch indirekt über das geometrische Produkt nachbilden, indem wir nach seiner Anwendung in die erste Komponente des Ergebnisses projizieren, und uns dann die Metrik geeignet zusammensetzen, aber eine notwendige oder gar sinnvolle Anwendung des geometrischen Produktes wäre dies zweifellos nicht. Insbesondere natürlich auch keine inhaltliche Erweiterung der RBF–Netze. Bedenkt man noch, daß RBF–Netze universelle, d.h. beliebig genaue Approximatoren stetiger Funktionen sind, so wird klar, daß man mit der Anwendung des geometrischen Produktes bestenfalls eine bessere Effizienz erzielen kann. Eine etwaige formal geeignete Verwendung des geometrischen Korrelators in der 2. Schicht leistet diesbezüglich aber sicher auch nichts.

Norm und Metrik sind nuneinmal Begriffe eines Vektorraumes, d.h. sie sind vollständig und alleinig über den Vektorraumanteil einer Clifford–Algebra darstellbar, also auch ohne Clifford–Algebra!

Es verbleiben zwei Möglichkeiten, diesem Fazit Rechnung zu tragen.

1) Die erste ist, man faßt RBF–Netze überhaupt nur als Anregung für gänzlich neue CA–Netze auf. Einen naheliegenden Ansatzpunkt dafür liefern dann natürlich die Basisfunktionen selbst. Die gebräuchlichste unter ihnen ist die Gaußfunktion

$$\exp\left(\frac{\|w_m^1 - x^k\|}{\sigma^2}\right), \tag{2.14}$$

wobei σ deren Varianz angibt. Da die Exponentialfunktion auch in Clifford–Algebren [Hestenes84] definiert ist, könnte man versuchen, einfach direkt weiterzuarbeiten. Ein wenig Vorsicht ist dabei aber stets geboten, so ist zum Beispiel das Additionstheorem der Exponentialfunktion für Clifford–Algebren meist nur noch eine Existenzaussage, d.h. es gilt zwar

$$\exp(x) \cdot \exp(y) = \exp(z)\,, \tag{2.15}$$

aber im allgemeinen ist $z \neq x + y$ und überhaupt nur noch sehr schwer konstruktiv zu bestimmen. Man tut also wohl gut daran, sich ersteinmal mit $\mathbb{C}$ und $\mathbb{H}$ zu beschäftigen. Unseres Erachtens nach sollten dort dann insbesondere Polarkoordinatendarstellungen und damit verbundene Phasenkonzepte von Interesse sein.

Man könnte zunächst aber auch ganz von der Ebene der Neuronalen Netze abstrahieren, und sich an der Verallgemeinerung des Regularisierungsansatzes, der den RBF–Netzen mathematisch ja zu Grunde liegt, für Funktionenräume über Clifford–Algebren versuchen. Dies dürfte aber ein nahezu unendlich schwieriges Unterfangen werden, insbesondere weil man Differentialoperatoren in diesen Räumen beherrschen muß, und selbst die Theorie der Lie–Algebren dazu wohl ziemlich ausgereizt werden muß.

2) Man kann es sich natürlich wesentlich einfacher machen, und die RBF–Netze so belassen, wie sie sind. Das dies tatsächlich noch eine Möglichkeit für Entwicklungen ist, wird deutlich, wenn man die 1.Schicht der RBF–Netze genauer betrachtet. Im allgemeinen wird diese nämlich zur Generalisierung benutzt, d.h. zum reduzieren der Anzahl der Basisfunktionen gegenüber der Anzahl der Eingabedaten. Das dies realisierende Lernverfahren ist meist ein Clustering–Algorithmus, wobei genug Spielraum für verschiedene konkrete Umsetzungen bleibt. Ein erster derartiger Algorithmus in einer Clifford–Algebra wurde von uns bereits in [Buchholz96] vorgeschlagen. Offensichtlich kann man die Generalisierung als eine Art Vorverarbeitung auffassen. Welche Möglichkeiten das geometrische Produkt diesbezüglich bietet, werden wir in Kapitel 5 ein wenig näher untersuchen.

Hier wollen wir nur noch feststellen, daß die Lokalität der Basisfunktionen in unseren Überlegungen keine Rolle gespielt hat.

2.3 Von MLPs zu Clifford–Algebra MLPs

Aus der Einleitung wissen wir bereits, daß schon MLPs für komplexe Zahlen (CMLPs), wie auch MLPs für Quaternionen (QMLPs) und sogar für allgemeine Clifford–Algebren (CA–MLPs) entwickelt worden sind. In diesem Abschnitt werden wir nur bis zu CMLPs vordringen, da sich an ihnen schon prinzipielle Probleme der Erweiterung von MLPs zu Clifford–Algebra MLPs untersuchen lassen. Am Anfang steht aber wie immer der reelle Fall, also das gewöhnliche MLP.

2.3.1 Reelle Multilayer Perceptrons

Ein reelles MLP können wir (mit den Notationen aus Abschnitt 2.1) wie folgt angeben:

$$\mathcal{N}_\omega^S(x^k)_p = \sum_{m=1}^{M} w_{mp}^2 \cdot f_m(\sum_{n=1}^{N}(w_{nm}^1 \cdot x_n^k) + \theta_m^1)) + \theta_p^2 \,. \tag{2.16}$$

Über die Basisfunktionen eines MLPs, die auch als *Aktivierungsfunktionen* bezeichnet werden, wissen wir bisher nur, daß sie nichtlinear und global seien sollten. Weitere Eigenschaften lassen sich aus dem Lernverfahren, durch das das Netz trainiert werden soll, herleiten, welchem wir uns daher auch gleich widmen wollen. Zuvor brauchen wir jedoch überhaupt ersteinmal ein Fehlermaß, das durch das Lernverfahren minimiert werden soll. Dieses ist durch den quadratischen Fehler

$$E(y,o) = \frac{1}{2}\,\|y-o\|^2 \tag{2.17}$$

zwischen erwarteter Netzausgabe y und tatsächlicher Netzausgabe o gegeben.

Das fast ausschließlich verwandte Verfahren zur Minimierung der Fehlerfunktion $E(y,o)$ ist der sogenannte *Backpropagation–Algorithmus*. Er realisiert einen Gradientenabstieg in den Gewichten ω des Netzes auf $E(y,o)$, d.h. die daraus ableitbare Lernregel ist vereinfacht

$$\Delta\,\omega = -\alpha \nabla_\omega E(y,o)\,, \tag{2.18}$$

wobei $\alpha \in \mathbb{R}$ die *Lernrate* ist. Die Gewichte der einzelnen Schichten werden nun schrittweise von der Ausgangsschicht rückwärts adaptiert, indem der Fehler zurückpropagiert wird. Zum Zwecke der einfacheren Darstellung dieses Verfahrens sei o_j jetzt allgemein die Ausgabe des j–ten Neurons in einer beliebigen Schicht und net_j dessen Netto–Eingabe.

Zunächst erfolgt die Adaption der Gewichte der Ausgabeschicht gemäß

$$\frac{\partial\,E}{\partial\,w_{mp}^2} = \delta_p \frac{\partial\,net_p}{\partial\,w_{mp}^2} \quad \text{mit} \quad \delta_p := \frac{\partial\,E}{\partial\,o_p} \cdot \frac{\partial\,o_p}{\partial\,net_p}\,, \tag{2.19}$$

danach werden die partiellen Ableitungen der Gewichte der versteckten Schicht berechnet

$$\frac{\partial\,E}{\partial\,w_{nm}^1} = \delta_m \frac{\partial\,net_m}{\partial\,w_{nm}^1} \quad \text{mit} \quad \delta_m := \sum_{p=1}^{P} \delta_p \cdot \frac{\partial\,net_l}{\partial\,o_m} \cdot \frac{\partial\,o_m}{\partial\,net_m}\,. \tag{2.20}$$

Damit der Backpropagation–Algorithmus überhaupt formuliert werden kann, müssen die Aktivierungsfunktionen also differenzierbar sein. Ihnen wenden wir uns nun direkt zu. Die gebräuchlichsten nichtlinearen Aktivierungsfunktionen sind $\tanh(x)$ und $1/(1+\exp(-x))$. Die Graphen dieser beiden Funktionen sind in Abbildung 2.2 dargestellt.

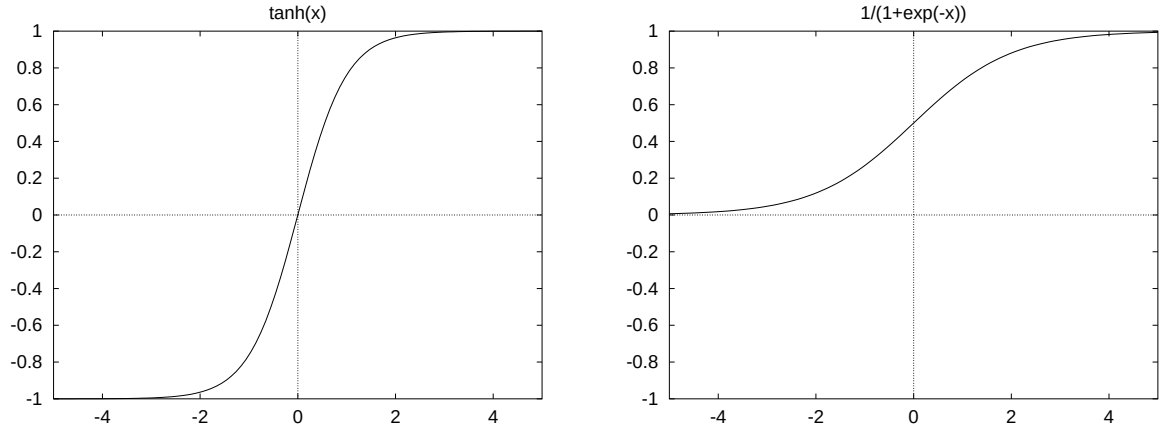


Abbildung 2.2: *Graphen gebräuchlicher Aktivierungsfunktionen*

Außer dem Wertebereich gibt es offenbar keine signifikanten Unterschiede zwischen den beiden Funktionen. Tatsächlich gehören beide Funktionen auch zur selben Klasse, den sogenannten sigmoiden (s–förmigen) Funktionen.

Definition 2.4: (Sigmoide Funktionen)
Für jedes $\beta \in \mathbb{R}_{>0}$ definiere die Funktion:

$$\sigma_\beta(x) := \frac{1}{1+\exp(-\beta x)} \qquad \forall x \in \mathbb{R}\,. \tag{2.21}$$

Die obigen Funktionen heißen *sigmoide Funktionen* und bilden nach $(0,1)$ ab.
Für $\beta = 1$ in (2.21) schreiben wir kurz nur σ und sprechen auch von der σ–Funktion.

Es gilt:

$$\tanh(x) = 2\sigma_2 - 1\,, \tag{2.22}$$

also ist tanh nur eine speziell skalierte und verschobene sigmoide Funktion.

Über den Parameter β läßt sich allgemein der lineare Anteil einer sigmoiden Funktion beeinflussen. Mit abnehmenden $\beta < 1$ erhält man von der Mitte hin zu den Rändern zunehmend flachere lineare Funktionen, die für $\beta \to -\infty$ gegen die konstante Gerade 0.5 konvergieren. Umgekehrt liefern wachsende $\beta > 1$ Funktionen mit immer steileren und kleineren linearen Anteil, für $\beta \to \infty$ bekommt man die harte Schwellwert–Funktion.

Sigmoide Funktionen haben eine sehr einfache Ableitung, so gilt für die σ–Funktion:

$$\dot{\sigma}(\cdot) \;=\; \sigma(\cdot) \;\cdot\; (\,1 - \sigma(\cdot)\,)\,, \tag{2.23}$$

d.h. die Ableitung ist alleinig durch σ selbst berechenbar, was den Rechenaufwand beim Backpropagation–Algorithmus erheblich vereinfacht.

In [Cybenko89] wurde bewiesen, daß sich mit sigmoiden Aktivierungsfunktionen jede stetige reellwertige Funktion in einer n–dimensionalen reellen Variablen beliebig genau ap-

proximieren läßt. Diese Aussage kann wie folgt formalisiert werden.

Satz 2.1: (Approximationssatz für reelle sigmoide Funktionen [Cybenko89])
Zu jeder stetigen Funktion $g : \mathbb{R}^n \to \mathbb{R}$ und jedem vorgegebenen $\epsilon \in \mathbb{R}_{>0}$ existieren für alle $j \in \{1, \ldots, n\}$ $w_j^1 \in \mathbb{R}^n$ und $w_j^2, \theta_j \in \mathbb{R}$ derart, daß

$$\| g(x) - \sum_{j=1}^{n} w_j^2 \cdot \sigma(\langle w_j^1, x \rangle) + \theta_j) \| \; < \; \epsilon \tag{2.24}$$

gilt, wobei $x \in \mathbb{R}^n$ und σ eine beliebige sigmoide Funktion ist.

Aus diesem Satz folgt, daß MLPs mit sigmoider Aktivierungsfunktion universelle Approximatoren beliebiger stetiger Funktionen von $\mathbb{R}^n \to \mathbb{R}^m$ sind, er sichert somit die Existenz einer Lösung des Funktionsapproximationsproblems für MLPs.

Für die komplexen Zahlen $\mathbb{C}$ werden die Sachverhalte aber schon wesentlich schwieriger.

2.3.2 Komplexe Multilayer Perceptrons

Betrachtet man rückblickend die Entwicklung von CMLPs, so stellt man fest, daß dabei das Hauptproblem nicht in der Aufstellung eines korrekten komplexen Backpropagation–Algorithmus lag (Angabe eines Lösungsverfahrens), sondern in der Zusicherung der Existenz von Lösungen für die vorgeschlagenen Basisfunktionen, weshalb wir uns auch auf die Erörterung des letztgenannten Aspektes beschränken werden.

Eine komplexe Aktivierungsfunktionen ist in allgemeinster Form durch

$$f(z) = u\,(x, y) + v\,(x, y)\,\mathrm{i} \tag{2.25}$$

gegeben, mit $z := x + y\,\mathrm{i} \in \mathbb{C}$, d.h. durch zwei Funktionen $u, v : \mathbb{C} \to \mathbb{R}$. Alle Forderungen an Aktivierungsfunktionen für reelle MLPs muß man selbstverständlich übernehmen. Was lag also näher, als auch gleich die Aktivierungsfunktionen selbst zu übernehmen, also anstelle der reellen σ–Funktion nun die komplexe Version

$$\sigma_{\mathbb{C}} : \mathbb{C} \to \mathbb{C}, \; z \mapsto \frac{1}{1 + \exp(-z)} \tag{2.26}$$

zu betrachten. Für $\sigma_{\mathbb{C}}$ ist Satz 2.1 in $\mathbb{C}$ formuliert aber nicht mehr richtig, was man aber erst eine gewisse Zeit nachdem $\sigma_{\mathbb{C}}$ als Aktivierungsfunktion in CMLPs vorgeschlagen und auch verwendet wurde, festgestellt hat. Das die einfache Übertragung der sigmoiden Funktionen nach $\mathbb{C}$ keine geeigneten Aktivierungsfunktionen liefert, kann man aber auch noch anders einsehen.

Betrachten wir dazu zunächst die $\sigma_{\mathbb{C}}$–Funktion. Für jedes $n \in \mathbb{N}$ ist die komplexe Zahl

$$z_s := \; 0 + \pi(2n + 1)\,\mathrm{i} \tag{2.27}$$

eine Singularität von $\sigma_{\mathbb{C}}$, d.h. für $z \to z_s$ gilt $|\sigma_{\mathbb{C}}(z)| \to \infty$, also ist $\sigma_{\mathbb{C}}$ unbeschränkt und damit keine zulässige Aktivierungsfunktion. Singularitäten von $tanh_{\mathbb{C}}$ in diesem Sinne sind alle komplexen Zahlen der Form

$$0 + \pi(n + \frac{1}{2})\,\mathrm{i}\,. \tag{2.28}$$

Die Wahl einer geeigneten komplexen Aktivierungsfunktion ist also keinesfalls trivial. In [Georgiou92] wurde die nichtlineare beschränkte Funktion

$$f_{c,r} : \mathbb{C} \to \mathbb{C},\ z \mapsto \frac{z}{c + \frac{1}{r}\,|z|}\,, \tag{2.29}$$

mit positiven reellen Konstanten c und r, vorgeschlagen, die mittlerweile auch am häufigsten Verwendung findet. Dort wird auch ein weiteres Kriterium angegeben, dem eine Aktivierungsfunktion f der Form (2.25) genügen muß. Es fordert, daß für die zugehörige Jacobi–Matrix $J(f)$

$$\det(J(f)(z)) = 0 \Rightarrow z = (0 + 0\,\mathrm{i}) \tag{2.30}$$

gilt, d.h für die partiellen Ableitungen u_x (u partiell abgeleitet nach x), u_y, v_x und v_y muß

$$u_x\, v_y \neq v_x\, u_y \tag{2.31}$$

erfüllt sein. Für die Funktion $f_{c,r}$ ist dies der Fall, da man die folgenden partiellen Ableitungen

$$u_x, v_y = \begin{cases} \dfrac{r\,(y^2 + c\,r\,|z|)}{|z|\,\left(c\,r + |z|\right)^2} & : \quad |z| \neq 0 \\ \dfrac{1}{c} & : \quad |z| = 0 \end{cases} \tag{2.32}$$

und

$$u_y, v_x = \begin{cases} -\dfrac{r\,x\,y}{|z|\,\left(c\,r + |z|\right)^2} & : \quad |z| \neq 0 \\ 0 & : \quad |z| = 0 \end{cases} \tag{2.33}$$

erhält.

Die skizzierten Probleme lassen sich gänzlich umgehen, wenn man komponentenweise wieder auf die reellen Aktivierungsfunktionen zurückgreift, also statt (2.25) nur Funktionen der Form

$$f(z) = u(x) + v(y)\,\mathrm{i} \tag{2.34}$$

betrachtet. Dabei hat man dann weder Probleme mit Singularitäten, noch muß man sich um zusätzliche Forderungen wie (2.31) Gedanken machen. Im Gegensatz zu $\sigma_{\mathbb{C}}$ gilt für die komponentenweise komplexe σ–Funktion

$$\sigma_{[\mathbb{C}]} : \mathbb{C} \to \mathbb{C},\ (x + y\,\mathrm{i}) \mapsto \frac{1}{1 + \exp(-x)} + \frac{1}{1 + \exp(-y)}\,\mathrm{i} \tag{2.35}$$

wieder ein zu Satz 2.1 analoger Approximationssatz, ebenso wie für die Funktion $f_{c,r}$, wobei der Beweis für $f_{c,r}$ natürlich wesentlich schwieriger ist. In der Praxis kann man aber beide Funktionen gleichermaßen gut benutzen.

Wie sich die Situation in allgemeineren Clifford–Algebren darstellt, wird nun ausführlich im nächsten Kapitel untersucht.

Kapitel 3

Clifford–Algebra MLPs

Ein Clifford–Algebra MLP $\mathcal{CN}_\omega^S$ wird als Spezifizierung von (2.7) durch die folgende Berechnung

$$\mathcal{CN}_\omega^S(x^k)_p = \sum_{m=1}^{M} w_{mp}^2 \otimes_{n,s} F(\sum_{n=1}^{N}(w_{nm}^1 \otimes_{n,s} x_n^k) + \theta_m^1) + \theta_p^2 \tag{3.1}$$

eines Ausgabedatums charakterisiert. Wenden wir uns gleich der entscheidenden Frage zu, welche Aktivierungsfunktionen in den versteckten Knoten des Netzes zum Einsatz kommen können. Dazu verallgemeinern wir unsere aus dem komplexen Fall stammenden intuitiven Vorstellungen zu einer formalen Systematik möglicher Aktivierungsfunktionen.

Definition 3.1: **((Nicht–) Komponentenweise Funktionen)**
Sei $F : \mathcal{C}_{n,s} \to \mathcal{C}_{n,s}$.
Die Funktion F heißt *komponentenweise*, falls

$$\forall i \in \{1,\ldots,n\}\ \exists f_i \in \mathbb{R}^{\mathbb{R}}\ \forall \mathrm{x} \in \mathcal{C}_{\mathrm{n,s}} : [\mathrm{F(x)}]_{\mathrm{i}} = \mathrm{f_i(x_i)}$$

gilt, andernfalls *nicht–komponentenweise*.

Die komplexen Aktivierungsfunktionen $\sigma_{\mathbb{C}}$ und $f_{c,r}$ sind nicht–komponentenweise Funktionen, $\sigma_{[\mathbb{C}]}$ ist selbstverständlich ein Vertreter einer komponentenweisen Funktion.

Beschäftigen wir uns zunächst mit nicht–komponentenweisen Aktivierungsfunktionen für CA–MLPs.

3.1 CA–MLPs mit nicht–komponentenweiser Aktivierungsfunktion

Bei CMLPs haben wir bereits gesehen, daß es nicht allzu viele geeignete Aktivierungsfunktionen dieses Typus gibt, und diese Klasse von Funktionen schwerer zu handhaben ist, als die komponentenweisen Funktionen. Für $\mathbb{C}$ hatte man aber grundsätzlich noch die Wahl zwischen Repräsentanten beider Funktionsklassen. Ein Hauptergebnis dieses Kapitels wird sein, daß für alle nicht–fundamentalen Clifford–Algebren nicht–komponentenweise Aktivierungsfunktionen nicht mehr geeignet sind. Die Ursache dafür werden die in diesen Algebren vorhandenen Nullteiler sein. Dessen ungeachtet wurden bereits CA–MLPs mit nicht–komponentenweiser Aktivierungsfunktion in [Pearson94] entwickelt. Mit dieser Arbeit wollen wir uns nun nur insoweit beschäftigen, wie es für den Beweis unserer obigen Behauptung erforderlich ist.

Die dort verwendete Aktivierungsfunktion ist die Verallgemeinerung von $f_{c,r}$ für beliebige Clifford–Algebren, wobei die Konstanten r und c zu 1 gesetzt wurden, womit man Funktionen

$$F_{n,s} : \mathcal{C}_{n,s} \to \mathcal{C}_{n,s},\ x \mapsto \frac{x}{1 + \|x\|}\,, \tag{3.2}$$

erhält, wobei $\|\cdot\|$ die Euklidische Norm ist.

In [Pearson94] wurde nur bewiesen, daß sich jede stetige Funktion $G_{n,0} : \mathcal{C}_{n,0} \to \mathcal{C}_{n,0}$ durch eine endliche Linearkombination von Funktionen $F_{n,0}$ beliebig genau approximieren läßt. Ungeachtet dessen, daß also einzig für Euklidische Clifford–Algebren in jener Arbeit die Existenz einer Lösung verifiziert wurde, wurden dort trotzdem auch Netze in beliebigen Clifford–Algebren entwickelt und getestet. Wir wollen uns wegen unserer eingangs erwähnten Behauptung der generellen Nichteignung nicht–komponentenweiser Aktivierungsfunktionen hier nicht die Mühe machen zu untersuchen, ob der Beweis aus [Pearson94] auf allgemeine Clifford–Algebren übertragbar ist, zumal wir dazu extra das Instrumentarium der Maßtheorie in Clifford–Algebren einführen müßten.

Der Nachweis der notwendigen Bedingung (2.30) für die partiellen Ableitungen von $F_{n,s}$ wurde in der zitierten Arbeit ebenfalls unterlassen. Wir müssen ihn jedoch erbringen, um abzusichern, daß eine Aktivierungsfunktion $F_{n,s}$ nicht schon im herkömmlichen Sinne selbst ungeeignet ist.

Dazu vereinbaren wir, daß Summation über Großbuchstaben vom Anfang des Alphabets stets Indizierung über einer vorgegebenen Indexmenge $\mathcal{A}_n$ (1.15) meint. Dann können wir unsere Aktivierungsfunktion $F_{n,s}$ schreiben als

$$F_{n,s}(x) = \sum_A u_A(x)\,, \tag{3.3}$$

mit $u_A(x) := [F_{n,s}(x)]_A$ und wir erhalten die partiellen Ableitungen

$$\frac{\partial\, u_A}{\partial\, [x]_B} = \begin{cases} -\frac{[x]_A [x]_B}{\|x\|\,(1+\|x\|^2)} & : \quad \|x\| > 0 \\ 0 & : \quad \|x\| = 0 \end{cases} \tag{3.4}$$

für den Fall $A \neq B$ und

$$\frac{\partial\, u_A}{\partial\, [x]_B} = \begin{cases} \frac{\|x\|^2 - [x]_A^2 + \|x\|}{\|x\|\,(1+\|x\|^2)} & : \quad \|x\| > 0 \\ 1 & : \quad \|x\| = 0 \end{cases} \tag{3.5}$$

falls $A = B$ gilt, womit wir jetzt (2.30) für $F_{n,s}$ beweisen können.

Satz 3.1: (Jacobi–Matrix von $F_{n,s}$)
Für alle $x = (x_1, \ldots, x_{2^n =: m}) \in \mathcal{C}_{n,s}$ mit $x \neq 0_{\mathcal{C}_{n,s}}$ gilt:

$$\det(J(F_{n,s})(x)) = 0 \;\Rightarrow\; x = 0_{\mathcal{C}_{n,s}}\,. \tag{3.6}$$

Beweis: Sei also x wie in der Voraussetzung gegeben.

Die Nenner aller partiellen Ableitungen von $F_{n,s}$ sind gleich, daher reicht es zu zeigen, daß die Behauptung für die aus $J(F_{n,s})$ durch Streichen der Nenner entstehende Matrix $\overline{J}(F_{n,s})$ erfüllt ist. Es gilt:

$$\begin{aligned} \det(\overline{J}(F_{n,s})(x)) &= \det \begin{pmatrix} \|x\|^2 - x_1^2 + \|x\| & -x_1 x_2 & \ldots & -x_1 x_m \\ -x_2 x_1 & \|x\|^2 - x_2^2 + \|x\| & \ldots & -x_2 x_m \\ \ldots & \ldots & \ldots & \ldots \\ -x_m x_1 & -x_m x_2 & \ldots & \|x\|^2 - x_m^2 + \|x\| \end{pmatrix} \\ &= \|x\|^4 \left(1 + m\,\|x\|^2 + (m-1)\,\|x\| + x_1^2\,\|x\| + \ldots + x_m^2\,\|x\|\right) \\ &> 0 \end{aligned}$$

□

Den in [Pearson94] angegebenen Backpropagation–Algorithmus wollen wir nicht extra aufführen, sondern verweisen direkt auf die Quelle. Wir beginnen stattdessen lieber gleich mit der Entwicklung unserer beabsichtigten Alternative zu den dortigen Netzen.

3.2 CA–MLPs mit komponentenweiser Aktivierungsfunktion

Glücklicherweise brauchen wir bei der Entwicklung von CA–MLPs mit komponentenweiser Aktivierungsfunktion — komponentenweise Funktionen für Clifford Neuronen wurden auch bereits in [Bayro96] vorgeschlagen — nicht ganz bei Null anfangen, da bereits ein auf

der σ–Funktion basierendes Quaternionen MLP (QMLP) in [Arena96a] und [Arena96b] vorgestellt wurde. Den Schritt hin zu derartigen allgemeinen CA–MLPs müssen wir dann allerdings selbst tun. Beginnen wir also direkt mit der Betrachtung des QMLPs, bei der wir quasi als Nebenprodukt unsere Behauptung des vorangegangenen Abschnittes über die Nichteignung von CA–MLPs mit nicht–komponentenweiser Aktivierungsfunktion beweisen werden.

3.2.1 Das QMLP

Das QMLP basiert auf der komponentenweisen Anwendung der gewöhnlichen σ–Funktion (2.21), d.h. die benutzte Aktivierungsfunktion $\sigma_{[\mathbb{H}]}$ setzt sich aus dieser wie folgt zusammen:

$$\sigma_{[\mathbb{H}]}(\cdot) = \sigma(\cdot) + \sigma(\cdot)\,\mathrm{i} + \sigma(\cdot)\,\mathrm{j} + \sigma(\cdot)\,\mathrm{k}\,. \tag{3.7}$$

Die Ableitung von $\sigma_{[\mathbb{H}]}$ wird demzufolge ebenfalls komponentenweise gebildet

$$\dot{\sigma}_{[\mathbb{H}]}(\cdot) = \dot{\sigma}(\cdot) + \dot{\sigma}(\cdot)\,\mathrm{i} + \dot{\sigma}(\cdot)\,\mathrm{j} + \dot{\sigma}(\cdot)\,\mathrm{k}\,, \tag{3.8}$$

womit eine Vereinfachung gegenüber der Situation bei einer nicht–komponentenweisen Funktion nocheinmal deutlich erkennbar wird.

Als erstes wollen wir nun das Approximationstheorem für das QMLP formulieren, welches sicher stellt, daß jede stetige Quaternionen–wertige Funktion auf einem kompakten Definitionsbereich im $\mathbb{H}^n$ beliebig genau durch eine endliche Linearkombination von $\sigma_{[\mathbb{H}]}$–Funktionen darstellbar ist.

Dazu benötigen wir zuvor jedoch die Definition eines Produktes, das an das Skalarprodukt für Vektoren erinnert, und formal auch so gebildet wird, aber für n–Tupel von Quaternionen erklärt ist.

Wir definieren dieses Produkt hier gleichwohl allgemeiner.

Definition 3.2: (Tupelprodukt)
Für alle $x, y \in (\mathbb{R}^4)^n$ und alle $s \in \{0, 1, 2\}$ sei:

$$x \cdot y \;:=\; x_1 \otimes_{2,s} y_1 \;+\; \ldots \;+\; x_n \otimes_{2,s} y_n\,. \tag{3.9}$$

Die obige Abbildung ist $\mathbb{R}$–linear und läßt sich analog auch für höherdimensionale Clifford–Algebren einführen. Die Beschränkung auf 4–dimensionale Clifford–Algebren in der obigen Definition diente einzig dem Zwecke, nocheinmal die richtige Formulierung von Aussagen über Multivektoren an einem möglichst einfachen Beispiel demonstrieren zu können. Nur bei einer festen Clifford–Algebra kann man nämlich für die Wahl eines Elementes „$x \in \mathcal{C}_{n,s}$“ schreiben, was ja bloß eine Abkürzung für „$x \in \mathbb{R}^{2^n}$ und auf dem $\mathbb{R}^{2^n}$ sei das

geometrische Produkt $\otimes_{n,s}$ gegeben" ist. Immer wenn Aussagen über mehrere Clifford–Algebren (und somit auch über mehrere geometrische Produkte) getroffen werden sollen, muß man wie in der obigen Definition verfahren.

Nach dieser Bemerkung zitieren wir nun das Approximationstheorem für das QMLP aus [Arena96b].

Satz 3.2: (QMLP–Approximationstheorem [Arena96b])
Sei $X \subseteq \mathbb{H}^n$ kompakt.
Zu jeder stetigen Funktion $g : X \mapsto \mathbb{H}$ und jedem reellen $\epsilon > 0$ existieren für alle $i \in \{1, \ldots, N\}$ $\alpha_i \in \mathbb{R}$, $y_i, x \in \mathbb{H}^n$ und $\theta_i \in \mathbb{H}$ derart, daß für alle $x \in X$

$$\| g(x) - \sum_{i=1}^{n} \alpha_i \sigma_{[\mathbb{H}]}(y_i \cdot x + \theta_i) \| < \epsilon \tag{3.10}$$

gilt.

Kompaktheit ist als topologischer Begriff natürlich nur über den Vektorraumanteil einer Clifford-Algebra definiert, also ist insbesondere die Voraussetzung „$X \subseteq \mathbb{H}^n$ kompakt" äquivalent zu „X ist kompakt in $(\mathbb{R}^4)^n$".

Der Beweis des Theorems basiert im wesentlichen auf dem folgenden Lemma, auf das wir dann später bei der Verallgemeinerung des QMLPs für beliebige Clifford–Algebren nocheinmal zurückkommen werden.

Lemma 3.1: ([Arena96b])
Sei $\phi : (\mathbb{R}^4)^n \to \mathbb{R}$ eine lineare Funktion.
Dann existieren eindeutig bestimmte $y_r, y_i, y_j, y_k \in (\mathbb{R}^4)^n$ derart, daß für alle $(\mathbb{R}^4)^n$ gilt:

$$\phi(x) = [y_r \cdot x]_1 = [y_i \cdot x]_2 = [y_j \cdot x]_3 = [y_k \cdot x]_4 \ . \tag{3.11}$$

Das Approximationstheorem besagt, daß QMLPs mit einer versteckten Schicht von $\sigma_{[\mathbb{H}]}$–Knoten universelle Approximatoren (im Sinne des Theorems) sind, wobei sogar reelle Gewichte von der versteckten Schicht zur Ausgabeschicht genügen.

Damit ist das QMLP dann auch konform zu unserer Definition 3.1 eines CA–MLPs, und wir können insbesondere auch alle dort benutzten Notationen bei der nun folgenden Angabe des Backpropagation–Algorithmus' übernehmen. Tabelle 3.1 listet die verwendeten Notationen im einzelnen nocheinmal auf.

Backpropagation–Algorithmus für das QMLP

Wir vereinbaren als Abkürzung σ für $\sigma_{[\mathbb{H}]}$ zu schreiben, da keine Mißverständnisse auftreten können, desweiteren wollen wir durchgängig Quaternionen für Gewichte und Schwellwerte, sowie statt linearer Ausgabeknoten ebenfalls σ–Knoten in der Ausgabeschicht verwenden, was natürlich beides aufgrund von Satz 3.2 erst recht möglich ist.

- N Knoten in der Eingabe–Schicht
- M Knoten in der versteckten Schicht
- P Knoten in der Ausgabe–Schicht
- I_n n–tes Eingabe-Quaternion
- w^1_{nm} Gewicht vom n–ten Eingabeknoten zum m–ten versteckten Knoten
- w^2_{mp} Gewicht vom m–ten versteckten Knoten zum p–ten Ausgabeknoten
- θ^1_m m–ter Schwellwert der versteckten Schicht
- θ^2_p p–ter Schwellwert der Ausgabe–Schicht

Tabelle 3.1: *Notationen für das QMLP*

Für die Vorwärtsphase ergeben sich dann die folgenden Gleichungen:

- Aktivierungs– und Ausgabewert eines versteckten Knotens

$$S^1_m := \sum_{n=1}^{N} w^1_{nm} \otimes I_n + \theta^1_m \tag{3.12}$$

$$X^1_m := \sigma(S^1_m) \tag{3.13}$$

- Aktivierungs– und Ausgabewert eines Ausgabe–Knotens

$$S^2_p := \sum_{m=1}^{M} w^2_{mp} \otimes X^1_m + \theta^2_p \tag{3.14}$$

$$o_p = X^2_p := \sigma(S^2_p)\,. \tag{3.15}$$

Die zu minimierende Fehlerfunktion E ist analog zu (2.17) durch

$$E = \frac{1}{2} \sum_{p=1}^{P} \langle y_p - o_p, y_p - o_p \rangle \tag{3.16}$$

gegeben, wobei $y = (y_1, \ldots, y_p) \in \mathbb{H}^P$ wieder der erwartete Ausgabewert sei.

Die Anwendung des Gradientenabstieges auf E liefert, mit Hinzunahme einer Lernkonstante $\alpha \in \mathbb{R}_{>0}$, dann die Änderung der Gewichte zu

$$\Delta w^2_{mp} = \alpha \, [\underbrace{(y_p - o_p) \odot \dot{\sigma}(S^2_p)}_{\delta^2_p}] \otimes \overline{X}^1_m \tag{3.17}$$

für die Gewichte zur Ausgabeschicht, bzw. zu

$$\Delta w^1_{nm} = \alpha \, [\underbrace{(\sum_{p=1}^{P} \overline{w}^2_{mp} \otimes \delta^2_p) \odot \dot{\sigma}(S^1_m)}_{\delta^1_m}] \otimes \overline{I}_n \tag{3.18}$$

für die Gewichte der versteckten Schicht. Die Änderung der Schwellwerte ist demzufolge:

$$\Delta \, \theta^2_p = \alpha \, \delta^2_p \quad \text{bzw.} \quad \Delta \, \theta^1_m = \alpha \, \delta^1_m \, . \tag{3.19}$$

Für die im Algorithmus benutzte Konjugation von Quaternionen gilt:

$$\forall x = (x_1, x_2, x_3, x_4) \in \mathbb{H} : \bar{\mathrm{x}} = (\mathrm{x}_1, -\mathrm{x}_2, -\mathrm{x}_3, -\mathrm{x}_4) \, , \tag{3.20}$$

wie man der allgemeinen Definition 1.12 leicht entnehmen kann.

Der Backpropagation–Algorithmus für das QMLP ist damit vollständig beschrieben.

Kehren wir nun zu der von uns im vorangegangenen Abschnitt aufgestellten Behauptung der Erkennbarkeit der Nichteignung nicht–komponentenweiser Aktivierungsfunktionen für Clifford–Algebren mit Nullteilern zurück, die wir jetzt fast mit einem Blick auf den QMLP–Backpropagation–Algorithmus einsehen können. Zum Beweis ist es nämlich ausreichend, nur die Regel (3.18) für die Änderung der Gewichte der 1.Schicht zu betrachten, genauer den Teil:

$$\Delta w^1_{nm} = \ldots (\sum_{p=1}^{P} \overline{w}^2_{mp} \otimes \delta^2_p) \odot \sigma(\dot{S}^1_m) \ldots \, . \tag{3.21}$$

Die komponentenweise Multiplikation $\odot$ in dieser Gleichung rührt natürlich von der Verwendung einer komponentenweisen Aktivierungsfunktion her, hat man stattderer eine nicht–komponentenweise Aktivierungsfunktion vorliegen, so steht an der Stelle von $\odot$ unausweichlich das geometrische Produkt der zugrundeliegenden Clifford–Algebra, unabhängig davon, welche sonstige konkrete Gestalt der Backpropagation–Algorithmus in dieser Algebra auch haben mag. In einer Clifford–Algebra $\mathcal{C}$ mit Nullteilern (also in einer Algebra ungleich $\mathbb{R}, \mathbb{C}$ und $\mathbb{H}$) hat dies zur Folge, daß $\Delta w^1_{nm} = 0$ sein kann, auch wenn ein von Null verschiedener Fehler zurückpropagiert werden sollte, d.h. es findet kein Lernen statt. Folglich sind nicht–komponentenweise Funktionen für CA–MLPs über den angegebenen Algebren ungeeignet.

Wir müssen jetzt natürlich erst noch beweisen, daß Nullteiler für CA–MLPs mit komponentenweiser Aktivierungsfunktion kein Problem darstellen.

3.2.2 Erweiterung für beliebige Clifford–Algebren

Wir wollen nun das QMLP für beliebige Clifford–Algebren erweitern, wobei wir intuitiv vorgehen wollen, beginnend bei der Verallgemeinerung seines Backpropagation–Algorithmus', womit wir schon impliziert behaupten, daß eine Modifikation notwendig und

$\otimes_{2,1}$	1	e_1	e_2	I
1	1	e_1	e_2	I
e_1	e_1	1	I	e_2
e_2	e_2	$-$I	-1	e_1
I	I	$-e_2$	$-e_1$	1

$\otimes_{2,2}$	1	e_1	e_2	I
1	1	e_1	e_2	I
e_1	e_1	1	I	e_2
e_2	e_2	$-$I	1	$-e_1$
I	I	$-e_2$	e_1	-1

Tabelle 3.2: *Multiplikationstabellen zu $\mathcal{C}_{2,1}$ und $\mathcal{C}_{2,2}$*

möglich ist. Der Nachweis der Notwendigkeit wird uns dann auch die richtige Idee für die erforderliche Modifikation liefern.

Eine Notation soll zuvor noch vereinbart werden. Für ein CA–MLP über der Clifford–Algebra $\mathcal{C}_{n,s}$ wollen wir kurz $\mathcal{C}_{n,s}$–MLP schreiben, mit Ausnahme des QMLPs.

Den Übergang von diesem zu einem $\mathcal{C}_{2,1}$–MLP oder zu einem $\mathcal{C}_{2,2}$–MLP geschieht einfach durch den Austausch der Multiplikationstabellen. Die Multiplikationstabellen von $\mathcal{C}_{2,1}$ und $\mathcal{C}_{2,2}$ sind beide in Tabelle 3.2 zusammengefaßt, wobei I:= $e_1 \wedge e_2$ gesetzt wurde.

Die aus dem Isomorphiekriterium (1.36) hergeleitete Isomorphie dieser beiden Algebren ist somit auch direkt anhand der Multiplikationstabellen erkennbar.

Das der Austausch der Multiplikationstabellen im Backpropagation–Algorithmus aber allein nicht ausreichend ist, wird deutlich wenn wir die dort zu Anwendung kommende Konjugation hinsichtlich ihrer Bedeutung hinterfragen. Beim zurückpropagieren der Fehler, sollen die Gewichte ihre Änderung natürlich proportional zu diesen erfahren, insbesondere soll keine Änderung erfolgen, wenn der Fehler bereits Null ist, was zur Konvergenz des Algorithmus notwendig ist. Übernehmen wir einfach die Konjugation beim Übergang von $\mathbb{H}$ nach $\mathcal{C}_{2,1}$ oder $\mathcal{C}_{2,2}$ so ist dies nicht mehr gewährleistet. Wir müssen also die Konjugation durch eine geeignete andere Funktion ersetzen, für die wir uns jetzt einen Kandidatenkreis überlegen wollen.

Erinnern wir uns an Bemerkung 1.2 (*iii*) aus Kapitel 1, die besagte, daß sich die Einträge in den Multiplikationstabellen gleichdimensionaler Clifford–Algebren maximal durch das Vorzeichen unterscheiden. Die Konjugation ist, wie man direkt der Definition entnimmt, nun gerade eine Funktion, die Vorzeichen von Komponenten vertauscht. Was liegt also näher, als unseren Kandidaten in der Menge aller derartigen Funktionen zu suchen, die wir jetzt formal einführen wollen.

Definition 3.3: (Vorzeichenvertauschende Abbildungen)
Sei $n \in \mathbb{N}$. Dann werden alle vorzeichenvertauschenden Abbildungen von $\mathbb{R}^n \to \mathbb{R}^n$ durch die Menge

$$F^n_{+/-} := \{f : \mathbb{R}^n \to \mathbb{R}^n \,|\, \forall \mathrm{x} \in \mathbb{R}^n \, \forall \mathrm{i} \in \{1, \ldots, \mathrm{n}\} : [\, \mathrm{f(x)} \,]_\mathrm{i} \in \{\mathrm{x_i}, -\mathrm{x_i}\}\} \tag{3.22}$$

beschrieben.

Der folgende Satz gibt alle für unsere Zwecke notwendigen Eigenschaften dieser Funktionenklasse an.

Satz 3.3: (Vorzeichenvertauschende Abbildungen in Clifford–Algebren)
Für je zwei n-dimensionale Clifford-Algebren $\mathcal{C}_{n,s}$ und $\mathcal{C}_{n,t}$ mit ungleichen Signaturen s und t, sowie weiter für alle $x, y \in \mathbb{R}^n$ und alle $i \in \{1, \ldots, n\}$ gilt:

$$\begin{aligned} (i)\ \neg\exists\, f \in F^n_{+/-} \ &: \ [x \otimes_{n,s} \bar{y}] = [x \otimes_{n,t} f(y)] \Leftrightarrow \mathcal{C}_{n,s} \ncong \mathcal{C}_{n,t} \\ (ii)\ \exists_1 f \in F^n_{+/-} \ &: \ [x \otimes_{n,s} \bar{y}]_i = [x \otimes_{n,t} f(y)]_i \ . \end{aligned}$$

Der Beweis des Satzes ist technisch etwas aufwendig, obwohl die Aussagen selbst plausibel sind, da man ihn rein kombinatorisch führen muß. Wir werden daher nur den 4–dimensionalen Spezialfall explizit beweisen, wobei dann gleichwohl auch die gesamte Tragweite des Satzes plastisch deutlich werden wird.

Für den uns interessierenden 4–dimensionalen Spezialfall beweisen wir den Satz zunächst für $\mathcal{C}_{2,0}$ und $\mathcal{C}_{2,2}$. Dazu wählen wir die eindeutig bestimmte Funktion aus (3.22), für die eine Übereinstimmung der 1. Komponenten erreicht wird.

Versuchen wir also nun beispielhaft, den Übergang vom QMLP zum $\mathcal{C}_{2,0}$–MLP zu vollziehen. Die dazu gesuchte Ersetzung $\tilde{}$ der beim QMLP verwendeten Konjugation $\bar{}$ muß zwei notwendige Bedingungen erfüllen, genauer das Produkt zweier Multivektoren $x \otimes_{2,2} \tilde{y}$ an den entsprechenden Stellen (3.17) und (3.18) im Backpropagation–Algorithmus. Dieses Produkt muß für die Konvergenz des Algorithmus' genau dann Null sein, wenn die beiden Vektoren orthogonal sind. In dieser Aussage sind die beiden notwendigen Bedingungen zusammen enthalten, das „wenn" sichert nämlich die Konvergenz des Algorithmus', und das „genau dann" die Korrektheit trotz Nullteiler.

Damit ist die Abbildung $\tilde{}$ eindeutig bestimmt, nämlich zu

$$\tilde{} \ : \ \mathbb{R}^4 \to \mathbb{R}^4, \ \mathrm{x} \mapsto (\mathrm{x}_1, \mathrm{x}_2, \mathrm{x}_3, -\mathrm{x}_4) \tag{3.23}$$

für alle $x = (x_1, x_2, x_3, x_4) \in \mathbb{R}^4$. Allgemein wählen wir als Ersetzung der Konjugation die in Euklidischen Algebren beim Backpropagation–Algorithmus zum Einsatz kommt, diejenige Abbildung aus (3.22), die in der 1.Komponente das Skalarprodukt liefert. Deren Existenz sichert gerade Aussage (ii) von Satz 3.3 im Besonderen. Dessen 1. Teil gewährleistet darüber hinaus, daß bei nicht nicht isomorphen Algebren auch nicht isomorphe CA–Netze entstehen. Für $\mathcal{C}_{2,2}$ rechnen wir dies alles ausführlich vor.

Lemma 3.2: (Spezialfall von Satz 3.3)
Für alle $x, y \in \mathbb{R}^4$ und alle $i \in \{2, 3, 4\}$ gilt:

$$(i)\ [x \otimes_{2,0} \bar{y}]_1 \ = \ [x \otimes_{2,2} \tilde{y}]_1$$

$$(ii)\ [x \otimes_{2,0} \bar{y}]_i \neq [x \otimes_{2,2} \tilde{y}]_i \ .$$

Beweis: Seien also $x = (x_1, x_2, x_3, x_4)$, $y = (y_1, y_2, y_3, y_4) \in \mathbb{R}^4$ gegeben.
(i) Es gilt:

$$\begin{aligned}
[x \otimes_{2,0} \bar{y}]_1 &= [x \otimes_{2,0} (y_1, -y_2, -y_3, -y_4)]_1 \\
&= x_1y_1 + x_2y_2 + x_3y_3 + x_4y_4 \\
&= [x \otimes_{2,2} (y_1, y_2, y_3, -y_4)]_1 \\
&= [x \otimes_{2,2} \tilde{y}]_1
\end{aligned}$$

(ii) Es gilt:

$$\begin{aligned}
[x \otimes_{2,0} \bar{y}]_2 &= [x \otimes_{2,0} (y_1, -y_2, -y_3, -y_4)]_2 \\
&= -x_1y_2 + x_2y_1 - x_3y_4 + x_4y_3 \\
&\neq +x_1y_2 + x_2y_1 + x_3y_4 + x_4y_3 \\
&= [x \otimes_{2,2} (y_1, y_2, y_3, -y_4)]_2 \\
&= [x \otimes_{2,2} \tilde{y}]_2
\end{aligned}$$

$$\begin{aligned}
[x \otimes_{2,0} \bar{y}]_3 &= [x \otimes_{2,0} (y_1, -y_2, -y_3, -y_4)]_3 \\
&= -x_1y_3 + x_2y_4 + x_3y_1 - x_4y_2 \\
&\neq +x_1y_3 - x_2y_4 + x_3y_1 - x_4y_2 \\
&= [x \otimes_{2,2} (y_1, y_2, y_3, -y_4)]_3 \\
&= [x \otimes_{2,2} \tilde{y}]_3
\end{aligned}$$

$$\begin{aligned}
[x \otimes_{2,0} \bar{y}]_4 &= [x \otimes_{2,0} (y_1, -y_2, -y_3, -y_4)]_4 \\
&= -x_1y_4 - x_2y_3 + x_3y_2 + x_4y_1 \\
&\neq -x_1y_4 + x_2y_3 - x_3y_2 + x_4y_1 \\
&= [x \otimes_{2,2} (y_1, y_2, y_3, -y_4)]_4 \\
&= [x \otimes_{2,2} \tilde{y}]_4
\end{aligned}$$

□

Wie steht es nun mit isomorphen Clifford–Algebren?

Isomorphe Clifford–Algebren implizieren gleiche CA–Netze.

Das besagt genau die Rückrichtung der Negation von (i) von Satz 3.3. Den Beweis dieser Existenzaussage kann man wie folgt führen. Zu zwei isomorphen Clifford–Algebren $\mathcal{C}_{n,s}$

und $\mathcal{C}_{n,t}$ existieren nämlich insbesondere f und $g \in F^n_{+/-}$ mit

$$[x \otimes_{n,s} f(y)]_1 = [x \otimes_{n,t} g(y)]_1 \ , \tag{3.24}$$

woraus aber bereits

$$[x \otimes_{n,s} f(y)] = [x \otimes_{n,t} g(y)] \tag{3.25}$$

folgt.

Im 4–dimensionalen verifiziert man dies leicht für die isomorphen Clifford–Algebren $\mathcal{C}_{2,1}$ und $\mathcal{C}_{2,2}$, wobei man für $\mathcal{C}_{2,1}$

$$f \ : \ \mathbb{R}^4 \to \mathbb{R}^4, \ \mathrm{x} \mapsto (\mathrm{x}_1, \mathrm{x}_2, -\mathrm{x}_3, -\mathrm{x}_4) \tag{3.26}$$

betrachtet.

Es verbleibt noch der Nachweis, daß ein zum QMLP–Approximationstheorem analoger Satz auch für beliebige CA–Netze mit komponentenweiser Aktivierungsfunktion existiert, und diese Netze somit universelle Approximatoren sind. Der Beweis eines Satzes derartiger Allgemeinheit und Stärke liegt allerdings außerhalb der Grenzen dieser Arbeit, wir können hier nur sein Wesen andeuten. Ein derartiger Beweis basiert auf der im Kapitel 1 angegebenen Eigenschaft von Clifford–Algebren, isomorph zu einer Matrizenalgebra aus nicht gemischten Produkten über $\mathbb{R}, \mathbb{C}$ oder $\mathbb{H}$ zu sein. Man muß dann weiter die Clifford–Algebren nach diesen Isomorphietypen einteilen und für sie eine induktive Ordnung definieren, so daß man letztendlich für jeden der Isomorphietypen den Beweis per Induktion führen kann. Für den 4–dimensionalen Fall kann man aber auch einfach den Beweis für das QMLP–Approximationstheorem direkt umformen.

Lemma 3.3: (Lemma 3.1 für $\mathcal{C}_{2,2}$)
Sei $\phi : (\mathcal{C}_{2,2})^n \to \mathbb{R}$ eine lineare Funktion.
Dann existieren eindeutig bestimmte $y_1, y_2, y_3, y_4 \in (\mathcal{C}_{2,2})^n$ derart, daß für alle $x \in (\mathcal{C}_{2,2})^n$ gilt:

$$\phi(x) = [y_1 \cdot x]_1 = [y_2 \cdot x]_2 = [y_3 \cdot x]_3 = [y_4 \cdot x]_4 \ . \tag{3.27}$$

Beweis: Sei also $x \in (\mathcal{C}_{2,2})^n \setminus \{0\}$ gegeben.
Betrachte die Abbildung ψ, die aus ϕ durch Ersetzung von $\otimes_{2,2}$ durch $\otimes_{2,0}$ entsteht.
Dann gilt: $\psi : (\mathcal{C}_{2,0})^n \to \mathbb{R}$. Weiter ist ψ offenbar auch linear.
Also existieren nach (3.1.3) eindeutig bestimmte $y_r, y_i, y_j, y_k \in \mathbb{H}^n \ (= (\mathcal{C}_{2,0})^n)$ mit (3.11).
Wir zeigen hier explizit nur, daß daraus bereits die (eindeutige) Existenz eines $y_2 \in (\mathcal{C}_{2,2})^n$, wie in (3.27) behauptet, folgt.
Setze dazu zunächst $u := y_i$.
Weiter besitzt u ausführlich geschrieben die Form: $u = ((a_1, b_1, c_1, d_1), \ldots, (a_n, b_n, c_n, d_n))$.
Wir wenden jetzt Satz 3.3 (ii) an, indem wir $v := ((-d_1, a_1, -b_1, c_1), \ldots, (-d_n, a_n, -b_n, c_n))$ setzen. Dann gilt $\phi(x) = [v \cdot x]_2 = [y_2 \cdot x]_2$, und es folgt somit die Behauptung.

□

Damit ist letztendlich auch Satz 3.2 für $\mathcal{C}_{2,2}$ bewiesen, da der Beweis dieses Satzes in [Arena96b] komponentenweise geführt wird und wir dessen Situation insgesamt komponentenweise herstellen können.

Für $\mathcal{C}_{2,1}$ als isomorphe Clifford–Algebra zu $\mathcal{C}_{2,2}$ ist nichts mehr zu zeigen, so daß unsere theoretischen Überlegungen für CA–Netze mit komponentenweiser Aktivierungsfunktion somit ihren Abschluß gefunden haben.

Kapitel 4

Experimentelle Untersuchungen über CA–MLPs

Die theoretischen Grundlagen für den Einsatz von CA-MLPs wurden von uns im vorangegangenen Kapitel vollständig gelegt, insbesondere haben wir die Korrektheit der von uns entwickelten Netze im Gegensatz zu den in [Pearson94] vorgeschlagenen bewiesen.

Nun geht es also um nicht weniger, als eine Abschätzung des Potentials der CA–MLPs und somit um eine Beurteilung der Eignung unserer algebraischen Einbettungen auf der Ebene konkreter Neuronaler Netze und ihrer Berechnungsparadigmen. Im Mittelpunkt dieses Kapitels steht dabei folglich weniger die einzelne Simulation an sich, sondern das Verstehen der prinzipiellen Funktionsweise von CA–MLPs durch Simulationen, wobei wir natürlich hoffen, daß dabei konzeptionelle Vorteile von CA–MLPs gegenüber MLPs erkennbar sind.

Am Anfang steht zur Vorbereitung die Bestätigung und Illustration einzelner bedeutender theoretischer Resultate des letzten Kapitels.

4.1 Experimente zu theoretischen Fragestellungen

Eine der in Kapitel 3 bewiesenen theoretischen Aussagen besagte, daß isomorphe Clifford–Algebren isomorphe CA–MLPs (mit komponentenweisen Aktivierungsfunktionen) implizieren. Nun ist es aber durchaus noch möglich, daß dies auch bei nicht–isomorphen gleichdimensionalen Clifford–Algebren der Fall ist. Um sich allgemein ein Bild über die Zusammenhänge von isomorphen und nicht–isomorphen Clifford–Algebren und den resultierenden Netzen zu verschaffen, ist es hilfreich, ein möglichst kleines Problem zu betrachten, bei dem man den Zustand der zu untersuchenden Neuronalen Netze leicht überschauen kann. Konkret war also ein Problem gesucht, daß mit einem versteckten Knoten und einem Ausgabeknoten lösbar, aber nicht trivial ist, so daß unsere Wahl auf die XOR–Funktion

x	y	x XOR y
0	0	0
0	1	1
1	0	1
1	1	0

Tabelle 4.1: *Wahrheitstabelle der XOR–Funktion*

fiel, die in Tabelle 4.1 dargestellt ist.

Bei Betrachtung selbiger fällt allerdings sofort auf, daß wir ersteinmal ein praktisches Kodierungsproblem zu bewältigen haben, nämlich das der Formatierung der obigen Muster in 4–dimensionale Ein- und Ausgabevektoren.

Bei genauerer Betrachtung erkennt man dann aber schnell, daß verschiedene Kodierungen einfach Permutationen der Eingabe sind, insbesondere dann natürlich auch Isomorphismen. Demonstrieren wir nun die völlige (bereits bewiesene) Gleichwertigkeit isomorpher CA–MLPs, so haben wir als leichteren Spezialfall auch automatisch ersteres eingesehen, wobei wir uns als Denkhilfe verschiedene Kodierungen als Netze mit isomorphen Multiplikationstabellen vorstellen könnten.

Trotzdem müssen wir bei der Wahl der Kodierung des XOR–Problems in 4 Dimensionen nocheinmal Vorsicht walten lassen, was aber kein Widerspruch ist, sondern einfach daran liegt, daß z.B. die Kodierung

$$(x, y) \rightsquigarrow (x, y, 0, 0) \tag{4.1}$$

keine Beweiskraft hat, da sich die Multiplikationstabellen von $\mathcal{C}_{2,1}$ und $\mathcal{C}_{2,2}$ ja nur hinsichtlich der 3. und 4. Komponenten unterscheiden. Daher wurde für die Experimente die Kodierung

$$(x, y) \rightsquigarrow (0, 0, x, y) \tag{4.2}$$

benutzt.

Den Verlauf des Trainings zeigt Abbildung 4.1.

Die faktische Gleichheit im Verhalten des $\mathcal{C}_{2,1}$–MLPs und des $\mathcal{C}_{2,2}$–MLPs ist dabei bereits deutlich zu erkennen, ebenso der Unterschied beider Fehlerkurven zu der des $\mathcal{C}_{2,0}$–MLPs. Alle Netze erreichen aber dann doch schnell den gleichen kleinen Trainingsfehler. Die beobachtete Äquivalenz der beiden isomorphen CA–Netze währen des Trainings schlägt sich dann auch im Lernergebnis für das Gewicht zur Ausgabeschicht nieder, worüber Tabelle 4.2

Auskunft gibt, wobei auch ersichtlich wurde, daß unsere gewählte Kodierung völlig unproblematisch gleichsam mitgelernt wurde.

Genauso eindeutig, wenn auch etwas komplizierter, ist die Gleichwertigkeit der beiden

Abbildung 4.1: *Trainingsfehler für die XOR-Funktion mit verschiedenen CA–MLPs*

XOR	$[w]_1$	$[w]_2$	$[w]_3$	$[w]_4$
$\mathcal{C}_{2,0}$–MLP	-8.054	-0.081	-0.261	+0.190
$\mathcal{C}_{2,1}$–MLP	+0.036	-7.629	-0.111	+4.612
$\mathcal{C}_{2,2}$–MLP	-7.613	+0.041	-0.091	+4.697

Tabelle 4.2: *Gelerntes Gewicht zur Ausgabeschicht*

Netze anhand der Ausgabe der versteckten Schicht zu erkennen. In den Tabellen 4.3 und 4.4 wurden daher korrespondierende Einträge mit gleichen Indizes gekennzeichnet.

$\mathcal{C}_{2,1}$–MLP	[1]	[2]	[3]	[4]
0-0	$0.604_{\ 1}$	$0.076_{\ 3}$	$0.926_{\ 5}$	$0.943_{\ 7}$
0-1	$0.043_{\ 2}$	$0.767_{\ 4}$	$0.460_{\ 6}$	$0.685_{\ 8}$
1-0	$0.690_{\ =}$	$0.459_{\ =}$	$0.767_{\ 9}$	$0.043_{\ 11}$
1-1	$0.941_{\ =}$	$0.920_{\ =}$	$0.075_{\ 10}$	$0.607_{\ 12}$

Tabelle 4.3: *Gelernte Ausgabe der versteckten Schicht für das $\mathcal{C}_{2,1}$–MLP*

$\mathcal{C}_{2,2}$–MLP	[1]	[2]	[3]	[4]
0-0	$0.040_{\ 2}$	$0.757_{\ 4}$	$0.674_{\ 8}$	$0.393_{\ 6}$
0-1	$0.582_{\ 1}$	$0.077_{\ 3}$	$0.941_{\ 7}$	$0.913_{\ 5}$
1-0	$0.684_{\ =}$	$0.393_{\ =}$	$0.040_{\ 11}$	$0.758_{\ 9}$
1-1	$0.941_{\ =}$	$0.913_{\ =}$	$0.563_{\ 12}$	$0.077_{\ 10}$

Tabelle 4.4: *Gelernte Ausgabe der versteckten Schicht für das $\mathcal{C}_{2,2}$–MLP*

Auch hier zeigt ein Vergleich mit den Ergebnissen des $\mathcal{C}_{2,0}$–MLPs in Tabelle 4.5 die Unterschiede.

$\mathcal{C}_{2,0}$–MLP	[1]	[2]	[3]	[4]
0-0	0.049	0.849	0.886	0.775
1-0	0.807	0.040	0.868	0.716
0-1	0.845	0.743	0.039	0.860
1-1	0.889	0.780	0.769	0.042

Tabelle 4.5: *Gelernte Ausgabe der versteckten Schicht für das $\mathcal{C}_{2,0}$–MLP*

Weiter fällt bei näherer Betrachtung der obigen Tabelle sofort deren Symmetrie auf, d.h. das $\mathcal{C}_{2,0}$–MLP hat eine symmetrische Kodierung des symmetrischen XOR–Problems gelernt, obwohl seine Multiplikationstabelle „genauso“ unsymmetrisch ist wie die der anderen Netze (bzw. Clifford–Algebren). Wenngleich eine Begründung für diese Beobachtung somit schwer fällt, so ist damit zumindest erklärt, warum das $\mathcal{C}_{2,0}$–MLP die XOR–Funktion schneller gelernt hat als die beiden anderen Netze.

Als nächstes haben wir versucht, die XOR–Funktion auch mit den nichtkomponentenweisen Netzen aus [Pearson94] — die wir im folgenden einfach auch als PMLPs bezeichnen wollen — zu lernen. Die nachstehende Abbildung 4.2 zeigt ein dabei erhaltenes typisches Ergebnis im Vergleich mit den bereits bekannten Resultaten für CA–MLPs und einem herkömmlichen MLP mit 2 versteckten Knoten. Selbst bei dem Standardproblem der XOR–Funktion ist die in Kapitel 3 ausführlich besprochene Anfälligkeit des Lernverfahrens gegenüber Nullteilermultiplikationen offensichtlich. Die gezeigten Ergebnisse treten in verschiedenen Variationen bei unterschiedlichsten Lernparametern stets auf, so daß

Abbildung 4.2: *Trainingsfehler für das XOR–Problem*

das von uns theoretisch prognostizierte Versagen der nicht–komponentenweisen Netze in Clifford–Algebren mit Nullteilern sich vollauf bestätigt hat. Interessanterweise laufen die beiden isomorphen Netze auch hier zumindest zu Beginn eine kurze Zeit lang synchron; erst später trennen sich ihre Wege. Dies ist ein weiteres Indiz dafür, daß die gravierenden Probleme des Algorithmus' wirklich im Auftreten von Nullteilermultiplikationen bei der Gewichtsanpassung begründet liegen. Normalerweise sollten nämlich isomorphe Algebren auch bei nicht–komponentenweisen Netzen zu isomorphen Netzen führen, was wohl auch tatsächlich solange der Fall ist, bis eine Störung in Form einer Nullteilermultiplikation vorkommt, wie sie bei ungefähr 270 Zyklen auftrat und bewirkte, daß das $\mathcal{C}_{2,2}$–PMLP von da an nicht mehr lernte, sondern im fälschlicherweise angenommenen, aber überhaupt nicht vorhandenen, Minimum blieb. Eine ähnliche Ursache, wenn auch mit anderen Folgen, dürfte für das Verhalten des $\mathcal{C}_{2,1}$–PMLPs ab 430 Zyklen verantwortlich sein. Bei nur einem Knoten in der versteckten Schicht sind die Auswirkungen von Nullteilermultiplikationen natürlich sofort dramatisch, aber auch bei mehreren Knoten ist die Wahrscheinlichkeit, daß dann nacheinander eine relevante Anzahl von Knoten ausfällt extrem hoch.

Pearson–MLPs in Clifford–Algebren mit Nullteilern sind also nicht nur theoretisch sehr zweifelhaft, sondern auch praktisch nicht einsetzbar. Auch die Leistung des somit einzig verbleibenden $\mathcal{C}_{2,0}$–PMLPs ist wenig überzeugend. Die XOR–Funktion wird sogar langsamer als durch das MLP gelernt, was insbesondere bedeutet, daß keine effiziente Verwendung der zusätzlichen Komponenten der Gewichtsvektoren wie bei den komponentenwei-

sen CA–MLPs erfolgt. Ursächlich dafür ist höchstwahrscheinlich, daß die im Algorithmus angewandte globale Gewichtsänderung zu aufwendig bzw. deren Lokalisierung problematisch ist, so daß auch bei diesem Netz ein praktischer Einsatz kaum in Frage kommt und wir die Erörterung dieses Ansatzes somit gänzlich beenden können.

Die Leistung unserer CA–MLPs ist hingegen außerordentlich gut, das MLP wird von ihnen mit nur einem Knoten doch deutlich in der Schnelligkeit des Lernens übertroffen, so daß wir mit einigem Optimismus uns nun weiteren Experimenten zuwenden.

4.2 Beispiele zur Funktionsapproximation durch CA–MLPs

Als nächstes wollen wir untersuchen, wie sich die CA–MLPs untereinander und im Vergleich zum MLP bei der Approximation komplexerer Funktionen verhalten, also ob dabei zum Beispiel signifikante Unterschiede in der Qualität der einzelnen Approximationen auftreten, und wenn ja bei welchen Typen von Funktionen. Wegen den von uns bewiesenen und zuvor nocheinmal demonstrierten Isomorphieargumenten ist es ausreichend nur einen Vertreter pro Isomorphieklasse zu betrachten. Für den 4–dimensionalen Fall haben wir die Algebren $\mathcal{C}_{2,0}$ (Quaternionen) und $\mathcal{C}_{2,2}$ gewählt, die Repräsentanten für 8–dimensionale Algebren werden dann $\mathcal{C}_{3,0}$ und $\mathcal{C}_{3,3}$ sein, deren Multiplikationstabelle Anhang C entnommen werden können.

Bei den durchgeführten Test diente stets das MLP als Referenz, d.h. es wurden zuerst dessen Parameter (versteckte Knoten, Lernrate etc.) so bestimmt, daß eine nahezu optimale Leistung erzielt werden konnte. Die so erhaltene Knotenzahl für die versteckte Schicht wurde dann schrittweise verringert. Die Trainings– und Testmengen bestanden jeweils aus 150 zufälligen, in $[0,1]^4$ (bzw. $[0,1]^8$)uniform verteilten Punkten und den zugehörigen Funktionswerten. Am Anfang der Untersuchungen stand dabei die Funktion

$$f_1 : \mathbb{R}^4 \to \mathbb{R}^4,\ (a,b,c,d) \mapsto (ad,\ \tfrac{3}{4}b,\ sin(c),\ d - \tfrac{1}{10})$$

als Vertreter einer aus einfachen Komponentenfunktionen bestehenden nichtlinearen Funktion. Der Verlauf des Trainings mit 3 bzw. 4 versteckten Knoten ist in Abbildung 4.3 wiedergegeben.

Kann das MLP die Funktion mit 4 Knoten noch halbwegs adäquat zu den CA–MLPs zu lernen, so ist dies bei 3 Knoten nicht mehr möglich und verstärkt sich mit abnehmender Knotenzahl rapide, wie man der Auflistung aller Werte in Tabelle 4.6 entnehmen kann, bei der der Trainingsfehler dem Generalisierungsfehler kleiner gedruckt vorangestellt ist.

Die Leistung der CA–MLPs ist insgesamt als sehr gut zu bewerten, erst mit nur einem

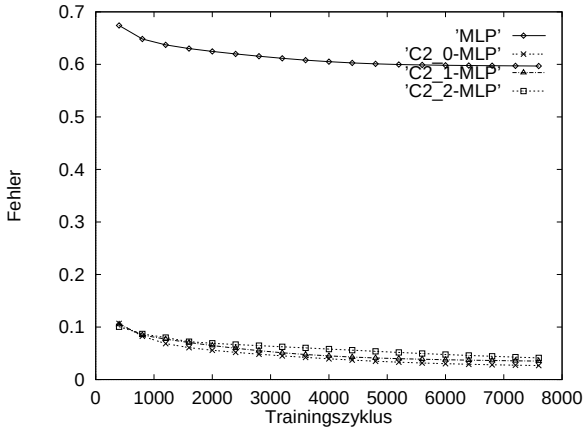


Abbildung 4.3: *Trainingsfehler für f_1 mit 3 (links) bzw. 4 (rechts) versteckten Knoten*

f_1	4	3	2	1
MLP	$_{0.114}$ 0.194	$_{0.597}$ 0.725	$_{1.773}$ 1.943	$_{3.438}$ 3.608
$\mathcal{C}_{2,0}$–MLP	$_{0.051}$ 0.057	$_{0.036}$ 0.051	$_{0.069}$ 0.077	$_{0.289}$ 0.395
$\mathcal{C}_{2,2}$–MLP	$_{0.029}$ 0.055	$_{0.060}$ 0.066	$_{0.068}$ 0.073	$_{0.311}$ 0.432

Tabelle 4.6: *Ergebnisse für f_1 nach 8000 Iterationen*

Knoten in der versteckten Schicht wurde die Funktion merklich schlechter gelernt, aber immer noch wesentlich besser als durch das MLP mit 3 Knoten. Weder beim Training noch beim Testen unbekannter Muster ist ein in keinsterweise nennenswerter Unterschied zwischen dem $\mathcal{C}_{2,0}$–MLP und dem $\mathcal{C}_{2,2}$–MLP festzustellen, da die Abweichungen in Anbetracht der absoluten Größe der Fehler vernachlässigbar sind.

Als nächste Funktion wurde der Sinus der Quaternionen

$$f_2 : \mathbb{H} \to \mathbb{H},\ \mathrm{q} \mapsto \tfrac{1}{2}\sin(\mathrm{q})$$

gewählt, was zum einem in Hinblick auf eine im Vergleich zu der vorangegangenen Funktion f_1 vermutlich für das MLP schwerer zu lernende Funktion geschah, und das $\mathcal{C}_{2,0}$–MLP

gegenüber dem $\mathcal{C}_{2,2}$–MLP bevorteilen sollte.

Wie die Ergebnisse (Abbildung 4.4 und Tabelle 4.7) zeigen, hat sich ersteres nicht bestätigt.

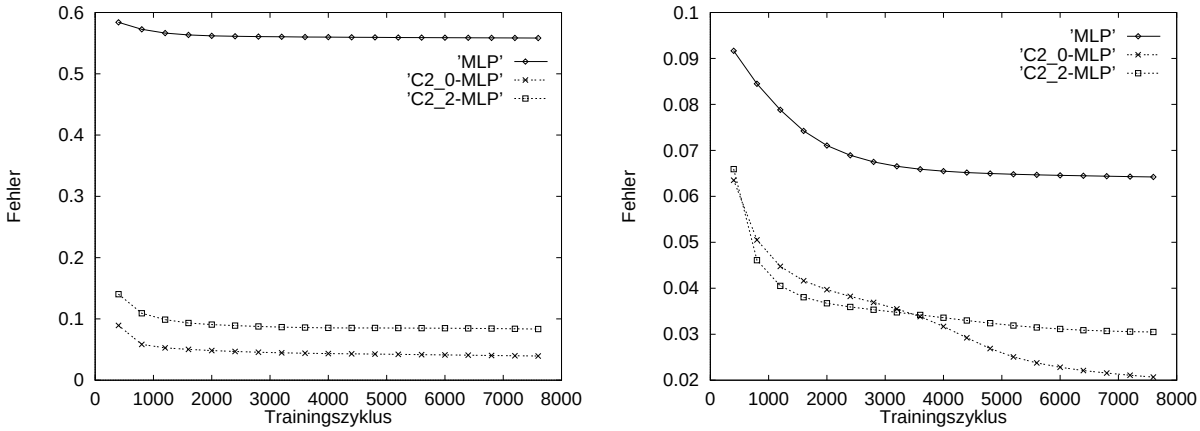


Abbildung 4.4: *Trainingsfehler für f_2 mit 3 (links) bzw. 4 (rechts) versteckten Knoten*

f_2	4	3	2	1
MLP	$_{0.064}$ 0.091	$_{0.558}$ 0.496	$_{0.948}$ 1.011	$_{1.958}$ 2.018
$\mathcal{C}_{2,0}$–MLP	$_{0.021}$ 0.026	$_{0.039}$ 0.044	$_{0.058}$ 0.061	$_{0.224}$ 0.229
$\mathcal{C}_{2,2}$–MLP	$_{0.031}$ 0.046	$_{0.075}$ 0.103	$_{0.083}$ 0.119	$_{0.225}$ 0.247

Tabelle 4.7: *Ergebnisse für f_2 nach 8000 Iterationen*

Das MLP hat die Funktion ähnlich wie zuvor die Funktion f_1 approximiert. Mit 4 Knoten sind die für f_2 erhaltenen Werte etwas kleiner als die von f_1, bei 3 Knoten ist der Trainingsfehler nahezu identisch. Das beide Funktionen mit 3 Knoten durch das MLP nur noch sehr bedingt gelernt wurden, ist auch an dem jeweils resultierenden Generalisierungsfehler ablesbar, der bei f_2 sogar kleiner als der Trainingsfehler war. Den Vergleich mit den beiden CA–MLPs konnte das MLP nur mit 4 Knoten bestehen. Das $\mathcal{C}_{2,0}$–MLP lieferte außer mit einem Knoten stets bessere Resultate als das $\mathcal{C}_{2,2}$–MLP, seine Leistung mit 3 Knoten entsprach sogar der des $\mathcal{C}_{2,2}$–MLPs mit 4 Knoten. Auch der Verlauf des Trainings läßt, insbesondere bei 4 Knoten, einen mutmaßlichen generellen Vorteil des $\mathcal{C}_{2,0}$–MLPs erkennen, der erst bei einem Knoten nicht mehr feststellbar ist, wobei die dabei erbrachte identische Leistung der CA–MLPs immer noch gut ist.

Mit den für den Sinus der Quaternionen erzielten Ergebnissen liegt es nahe, eine Begünstigung eines CA–Netzes für Funktionen aus „seiner“ Algebra gegenüber einem CA–Netz in einer anderen Algebra anzunehmen, weshalb wir nun die beiden Funktionen

$$f_3 : \mathbb{H} \to \mathbb{H},\ \mathrm{q} \mapsto \tfrac{1}{4}\,(\mathrm{q} \otimes_{2,0} \mathrm{q} + 2)$$

und

$f_4 : \mathbb{H} \to \mathbb{H},\ \mathrm{q} \mapsto \frac{1}{4}\,(\mathrm{q} \otimes_{2,2} \mathrm{q} + 2)$

untersucht haben.

Bei Betrachtung der Resultate für die Funktion f_3, die in Abbildung 4.5 und Tabelle 4.8 dargestellt sind,

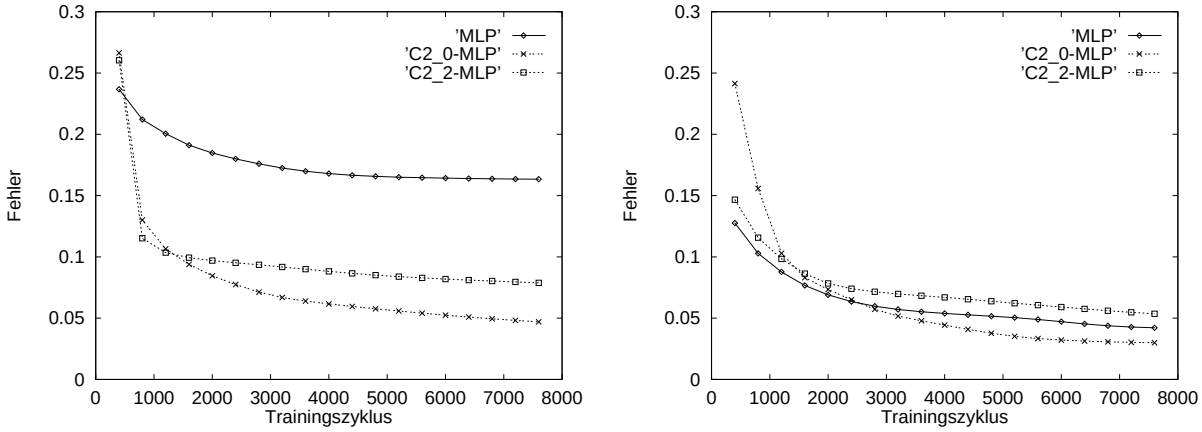


Abbildung 4.5: *Trainingsfehler für f_3 mit 3 (links) bzw. 4 (rechts) versteckten Knoten*

f_3	4	3	2	1
MLP	$_{0.042}$ 0.097	$_{0.163}$ 0.266	$_{0.342}$ 0.453	$_{0.717}$ 1.034
$\mathcal{C}_{2,0}$–MLP	$_{0.029}$ 0.074	$_{0.047}$ 0.136	$_{0.103}$ 0.183	$_{0.354}$ 0.484
$\mathcal{C}_{2,2}$–MLP	$_{0.054}$ 0.103	$_{0.079}$ 0.167	$_{0.111}$ 0.218	$_{0.496}$ 0.518

Tabelle 4.8: *Ergebnisse für f_3 nach 8000 Iterationen*

fällt zunächst auf, daß das MLP diese Funktion im Vergleich zu den beiden vorhergehenden am besten bewältigt hat. Ansonsten wollen wir unser Augenmerk gleich auf die CA–Netze richten. Bei 3 Knoten wird die Funktion durch sie gleichgut gelernt, der Vorteil des $\mathcal{C}_{2,0}$–MLPs gegenüber dem $\mathcal{C}_{2,2}$–MLP ist insgesamt nicht allzu stark ausgeprägt.

Ein ähnliches Bild zeigte sich dann umgekehrt auch bei der Approximation der Funktion f_4, wo wiederum bereits die Trainingsfehler (Abbildung 4.6) sehr dicht beieinanderlagen.

Den Diagrammen ist weiter zu entnehmen, daß das MLP sowohl bei 4 als auch noch bei 3 Knoten einen nur geringen Abstand zu den CA–Netzen aufweist. Die Tabelle 4.9 der Ergebnisse zeigt,

daß die Funktion für alle Netze leichter zu lernen war, als die Funktion f_3.

Der vermutete Vorteil eines CA–MLPs in der jeweils „richtigen“ Algebra hat sich zwar bestätigt, aber zum einen ist er nicht sonderlich groß, zum anderen ist er durch unsere Untersuchungen natürlich nicht als ursächlich nachgewiesen, sondern es traten nur die entsprechenden Beobachtungen auf.

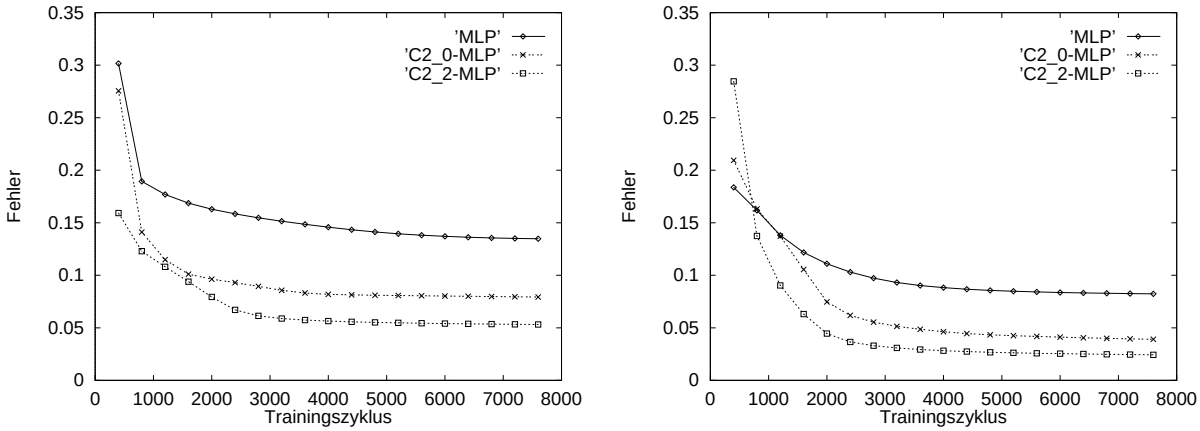


Abbildung 4.6: *Trainingsfehler für f_4 mit 3 (links) bzw. 4 (rechts) versteckten Knoten*

f_4	4	3	2	1
MLP	$_{0.082}$ 0.098	$_{0.135}$ 0.154	$_{0.244}$ 0.255	$_{0.565}$ 0.657
$\mathcal{C}_{2,0}$–MLP	$_{0.039}$ 0.086	$_{0.079}$ 0.147	$_{0.132}$ 0.175	$_{0.372}$ 0.474
$\mathcal{C}_{2,2}$–MLP	$_{0.024}$ 0.062	$_{0.053}$ 0.106	$_{0.104}$ 0.159	$_{0.360}$ 0.422

Tabelle 4.9: *Ergebnisse für f_4 nach 8000 Iterationen*

Andererseits können wir aber sehrwohl feststellen, daß das $\mathcal{C}_{2,0}$–MLP nicht a priori besser als das $\mathcal{C}_{2,2}$–MLP ist, obwohl die Quaternionen sogar nullteilerfrei sind. Um dies als Bestätigung dafür ansehen zu dürfen, daß es uns mit der Entwicklung eines entsprechend modifizierten Backpropagation–Algorithmus' im Kapitel 3 gelungen ist, dieses grundlegende Problem für die CA–MLPs vollständig und praxistauglich zu lösen, sollten wir auch noch ein Beispiel in 8-dimensionalen Clifford–Algebren untersuchen, was natürlich auch allein schon deshalb angebracht ist, um eine ungefähre Vorstellung über das dortige „Kräfteverhältnis“ zwischen dem MLP und unseren CA–MLPs zu bekommen, als deren (nicht isomorphe) Vertreter wir schon zu Anfang dieses Abschnittes das $\mathcal{C}_{3,0}$–MLP und das $\mathcal{C}_{3,3}$–MLP bestimmt hatten.

Die Funktion

$$f_5 : \mathbb{R}^8 \to \mathbb{R}^8,\ (a,b,c,d,e,f,g,h) \mapsto (ag-h, \tfrac{1}{2}c, \tfrac{1}{3}(d+e), b+\tfrac{1}{10}, \sin(f), c+h, df, \cos(g))$$

wurde gewählt, da das MLP im 4–dimensionalen Fall mit der Funktion f_1 gleichen Typus die größten Schwierigkeiten hatte, zum anderen gibt es bei ihr keine Anhaltspunkte für eine von vornherein gegebene Begünstigung eines der beiden CA–MLPs.

Tatsächlich fiel dann auch diese Funktion dem MLP relativ schwer, worüber die Abbildung 4.7 (rechts)

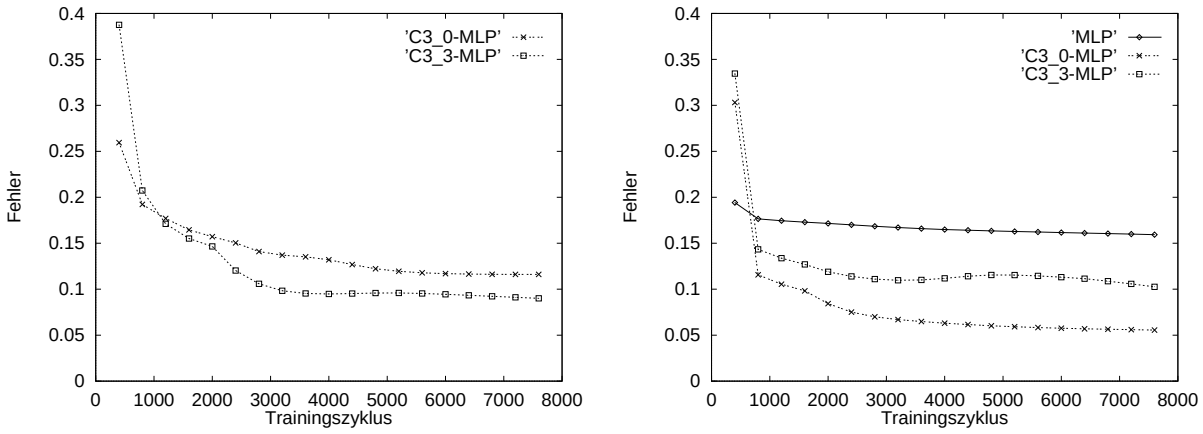


Abbildung 4.7: *Trainingsfehler für f_5 mit 4 (links) bzw. 6 (rechts) versteckten Knoten*

und die Tabelle

f_5	7	6	5	4	3	2
MLP	$_{0.077}$ 0.133	$_{0.158}$ 0.284	$_{0.365}$ 0.621	$_{0.604}$ 0.830	$_{0.918}$ 1.119	$_{2.778}$ 4.423
$\mathcal{C}_{3,0}$–MLP	$_{0.123}$ 0.151	$_{0.057}$ 0.063	$_{0.089}$ 0.132	$_{0.132}$ 0.146	$_{0.217}$ 0.237	$_{0.282}$ 0.327
$\mathcal{C}_{3,3}$–MLP	$_{0.035}$ 0.049	$_{0.082}$ 0.108	$_{0.086}$ 0.092	$_{0.082}$ 0.093	$_{0.177}$ 0.184	$_{0.249}$ 0.254

Tabelle 4.10: *Ergebnisse für f_5 nach 8000 Iterationen*

im einzelnen Auskunft geben. Das MLP benötigte immerhin 7 versteckte Knoten für ein gutes Ergebnis, die CA–MLPs konnten eine fast ähnlich gute Approximation noch mit 4 Knoten erzielen. Bei beiden CA–Netzen waren mehr als 4 Knoten schon zuviel des Guten, wie man auch den Abbildungen der Trainingsverläufe entnehmen kann. Besonders deutlich zu erkennen war ein derartiges Verhalten beim $\mathcal{C}_{3,0}$–MLP mit 7 Knoten in der versteckten Schicht. Vergleicht man die Leistung der beiden CA–MLPs miteinander, so konnte das $\mathcal{C}_{3,3}$–MLP im Regelfall stets bessere Werte erzielen, und zwar in einer Größenordnung, die ungefähr der bei den zuvor behandelten 4–dimensionalen Funktionen f_2, f_3 und f_4 beobachteten entspricht. Bemerkenswert ist, daß es dafür keinen „algebraischen“ Grund gibt.

Das unterschiedliche Verhalten von CA–MLPs weiter systematisch zu untersuchen ist sicher eine lohnende Aufgabe. Gelingt dabei eine gewisse (auch nur teilweise) Klassifizierung von Problemen, die für bestimmte Clifford–Netze besonders geeignet sind, so ergeben sich dann wiederum Möglichkeiten für entsprechende Spezialisierungen (Optimierungen) des Lernverfahrens. Bei der in Angriffnahme dieser Aufgabe würden zunehmend sicher noch mehr konzeptionelle Vorteile der Clifford–Netze zu Tage treten.

Die eben skizzierte Richtung können wir in dieser Grundlagenarbeit aber nicht mehr weiterverfolgen, zumal der Weg dahin vorher noch ein wenig mehr geebnet werden muß, vorallem durch die genauere Untersuchung der Effizienz unserer Clifford Netze gegenüber vergleichbaren MLPs, mit der es scheinbar nicht zum besten bestellt ist, denn unbestritten ist ein CA–MLP stets erheblich aufwendiger als ein MLP mit gleicher Zahl versteckter Knoten, außer auf der symbolischen Ebene, wo man ein Clifford Neuron so wie ein herkömmliches Neuron bewerten würde, die aber natürlich keine wirklichen Aussaugen zur Effizienz erlaubt, wie wir sie jetzt anstreben.

4.3 Effizienzbetrachtungen

Zunächst gilt es, den Begriff des Aufwands geeignet zu quantifizieren. Dazu betrachten wir wie üblich die Zeit- und Speicherplatzkomplexität der Neuronalen Netze in ihren natürlichen Maßen, der Anzahl der verwendeten Variablen bzw. der Anzahl der benötigten arithmetischen Operationen.

Beginnen wollen wir mit der Erörterung der Speicherplatzkomplexität, da die entsprechenden Formeln [1] sehr einfach sind, wenn man nur jeweils eine versteckte Schicht berücksichtigt, was für unsere Zwecke ja ausreichend ist. Die Gesamtzahl der Gewichte eines MLPs ist dann

$$Var_{MLP} := nh \cdot (ni + 1) + no \cdot (nh + 1)\,, \tag{4.3}$$

wobei der Schwellwert separat mit aufgeführt worden ist. Die analoge allgemeine Formel für die Anzahl reeller Parameter eines $\mathcal{C}_{n,s}$–MLPs lautet

$$Var_{\mathcal{C}_{n,s}-MLP} := 2^n \cdot nh \cdot (ni + 1) + 2^n \cdot no \cdot (nh + 1)\,. \tag{4.4}$$

Aus diesen beiden Formeln erhält man im speziellen für die uns interessierenden Fälle des $(1, nh, 1)$–$\mathcal{C}_{2,s}$–MLPs

$$Var_{\mathcal{C}_{2,s}-MLP} := 12 \cdot nh + 4 \tag{4.5}$$

und für das entsprechende MLP

$$Var_{(4,nh,4)-MLP} := 9 \cdot nh + 4\,. \tag{4.6}$$

[1] bei denen wieder unsere Standardnotationen zur Anwendung kommen und die Topologie eines Netzes in der Form (ni, nh, no) selbigen bei Bedarf vorangestellt wird

Im 8–dimensionalen Fall ist die Anzahl der Parameter

$$Var_{\mathcal{C}_{3,s}-MLP} := 24 \cdot nh + 8 \tag{4.7}$$

bzw.

$$Var_{(8,nh,8)-MLP} := 17 \cdot nh + 8\,. \tag{4.8}$$

Also benötigt in 4 Dimensionen ein CA–MLP mit 3 versteckten Knoten genausoviele Gewichte wie ein entsprechendes MLP mit 4 versteckten Knoten. Bei unseren Beispielfunktionen f_1 und f_2 reichte ein CA–MLP mit 2 Knoten um eine mindestens adäquate Leistung zum MLP mit 4 Knoten zu erbringen, so daß in diesen Fällen jeweils 8 reelle Gewichte (20 %) gegenüber dem MLP eingespart wurden. Bei Betrachtung der Ergebnisse der Approximation der 8-dimensionalen Funktion f_5 ergibt sich, bei Zugrundelegung gleicher Lernresultate, konkret kein derartiger Vorteil, aber wir können sicher die Existenz von Funktionen annehmen, bei denen man ein Knotenverhältnis von 2 (CA–MLP) zu 7 (MLP) erzielen kann, was dann schon eine sehr beachtliche Reduzierung der Anzahl der Gewichte um 56% bedeuten würde.

Die mit CA–MLPs erreichbaren Parameterreduzierungen sind durchaus als Ergebnis eines konzeptionellen Vorteils dieser Netze gegenüber dem herkömmlichen MLP anzusehen, da sie ja das Lernen einer effizienteren internen Kodierung des Problems im Rahmen einer Clifford–Algebra bedeuten. Für die Beantwortung theoretischer Fragen hat dies große Bedeutung, da es sicher einfacher ist, die Vorgänge innerhalb eines Netzes anhand von 20 statt von 50 Gewichten zu studieren, zumal wenn die Netze und somit insbesondere die Gewichte noch zusätzlich mit einer Struktur versehen sind.

Es gilt natürlich unter praktischen Gesichtspunkten abzuwägen, wie teuer diese Vorteile eventuell durch eine höhere Anzahl von arithmetischen Operationen erkauft werden.

Allein die Tatsache, daß die Berechnung eines geometrischen Produktes $\otimes_{n,s}$ 2^{2n} reelle Multiplikationen und ebensoviele Additionen erfordert, läßt erahnen, wie groß die Zeitkomplexität von Clifford–Netzen im Vergleich zu einem MLP ist. Wir wollen hier nicht bis auf die Ebene der Operationen selbst hinuntergehen, sondern die Laufzeit von Simulationen als Zeitmaß heranziehen, um zu einer Beurteilung zu gelangen, was für Fragen der praktischen Einsetzbarkeit von CA–MLPs völlig ausreichend ist.

Auf die Zeitkomplexität der Clifford–Netze wirken sich die möglichen Parameterreduzierungen natürlich auch schon positiv aus. Ebenso zeigt ein nochmaliger Blick zurück auf die Ergebnisse der Approximation unserer Beispielfunktionen, daß CA–MLPs schnell und meist auch schneller als ein MLP lernen. Dieses nicht zu unterschätzende Potential wird aber kaum ausreichen, wenn wir einen bisher noch nicht berücksichtigten aber im allgemeinen vorliegenden Faktor nicht kompensieren können. Solange wir Probleme betrachten, deren Dimension eine Zweierpotenz ist, haben wir stets eine „genau“ passende Clifford–Algebra zur Verfügung, in allen anderen Fällen aber erst die nächstgrößere Clifford–Algebra. Unserer Meinung nach stellt aber eine dann vorzunehmende Einbettung der Problemstellung in die nächstgrößere Clifford–Algebra nicht notwendigerweise ein unumgängliches kostenintensives Übel dar, sondern kann auch Chancen bieten, das Problem

letztendlich vielleicht sogar effektiver zu lösen. Als kleines — aber sehr überzeugendes — Beispiel können wir diesbezüglich bereits auf die Ergebnisse zum XOR-Problem aus Abschnitt 4.1 verweisen. Ob sich diese Behauptung auch mit einer größeren Anwendungsaufgabe untermauern läßt, können wir im folgenden gleich überprüfen, wenn wir unsere CA–MLPs einen letzten umfangreichen und praxisnahen Test unterziehen, der Vorhersage einer chaotischen Zeitreihe, welcher auch schon in der Originalarbeit zum QMLP [Arena96a] zur Anwendung kam.

Vorhersage des Lorenz–Attraktors

Der Lorenz–Attraktor wird durch das Differentialgleichungssystem

$$\begin{aligned} \dot{x} &= \sigma(x-y) \\ \dot{y} &= xz + rx - y \\ \dot{z} &= xy - bz \end{aligned}$$

beschrieben, wobei x, y, z zeitabhängige Funktionen und σ, r, b reelle Parameter sind. Das Verhalten des Systems ist chaotisch und führt auf den sogenannten Lorenz–Attraktor, der in Abbildung 4.8 mit einer Schrittweite von $\Delta t = 0.02$ im Zeitintervall $t = [600, 2500]$ diskretisiert für die Parameter [2] $\sigma = -3$, $r = 26.5$ und $b = 0$ und Anfangsbedingungen $x(0) = z(0) = 0$ und $y(0) = 1$ dargestellt ist.

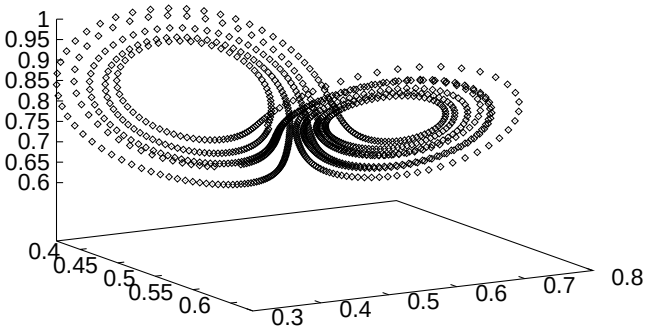


Abbildung 4.8: *Lorenz–Attraktor*

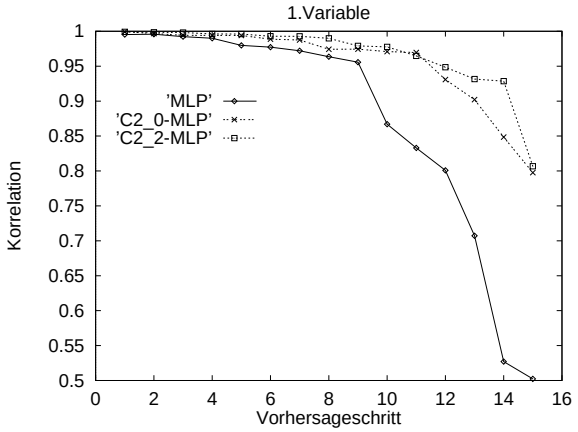
[2] in [Arena96a] wurde er für $\sigma = 10$, $r = \frac{8}{3}$ und $b = 28$ untersucht

Von diesen 1000 Abtastpunkten bildeten die ersten 250 für unsere Simulationen die Trainingsmenge und die restlichen 750 die Testmenge. Von den Netzen war also eine Extrapolationsaufgabe zu bewältigen, wodurch ihre Generalisierungsfähigkeiten verstärkt getestet werden sollten. Die Schwere dieser Aufgabe ist über die Vorhersageschrittweite τ beeinflußbar, wobei eine Schrittweite von $\tau = 1$ bedeutet, daß immer das jeweils nächstfolgende Muster trainiert/getestet wird, bei $\tau = 2$ das übernächste und so weiter, da die Natur der Zeitreihe chaotisch, also insbesondere auch nichtlinear ist. Eine Erhöhung der Vorhersageschrittweite τ bedeutet also eine Abnahme der Korrelation zwischen Eingabe- und Ausgabemustern und führt somit zu immer schwereren Aufgaben für die Netze. Die Korrelation zwischen erwarteter und erzeugter Ausgabe diente uns demzufolge auch zur Beurteilung der Güte der erhaltenen Vorhersagen und wird als Funktion von τ durch

$$\rho_s(\tau) = \frac{\sum_{t=1}^{M}(O_s(t) - O_{sm})(O'_s(t) - O'_{sm})}{\sqrt{\sum_{t=1}^{M}(O_s(t) - O_{sm})^2}\sqrt{\sum_{t=1}^{M}(O'_s(t) - O'_{sm})^2}} \tag{4.9}$$

beschrieben, wobei M die Anzahl der Testmuster, $O_s(t)$ die für die s–te Variable erwartete Ausgabe zum Zeitpunkt $t+\tau$ ist, O_{sm} der Mittelwert aller dieser Ausgaben ist, und $O'_s(t)$ bzw. O'_{sm} die korrespondierenden gemessenen Werte sind. Die Kodierung des Problems in 4 Dimensionen für die CA–MLPs erfolgte denkbar einfach, durch Einsetzen in die 3 letzten Komponenten und Nullsetzen der ersten.

Mit einem $(3, 12, 3)$–MLP und mit $(1, 5, 1)$–CA–MLPs wurden die in Abbildung 4.9 aufgeführten Ergebnisse erzielt. Die Anzahl der Knoten in der versteckten Schicht wurde jeweils so gewählt, daß bei einer mittelschweren Vorhersageschrittweite von $\tau = 7$ die Korrelationswerte für alle Variablen noch größer als 0.9 waren. Mit 4 versteckten Knoten wurde dieses Kriterium für die CA–MLPs einzig für die 2.Variable verletzt, die sich für alle Netze als deutlich am schwersten zu lernen herausstellte.



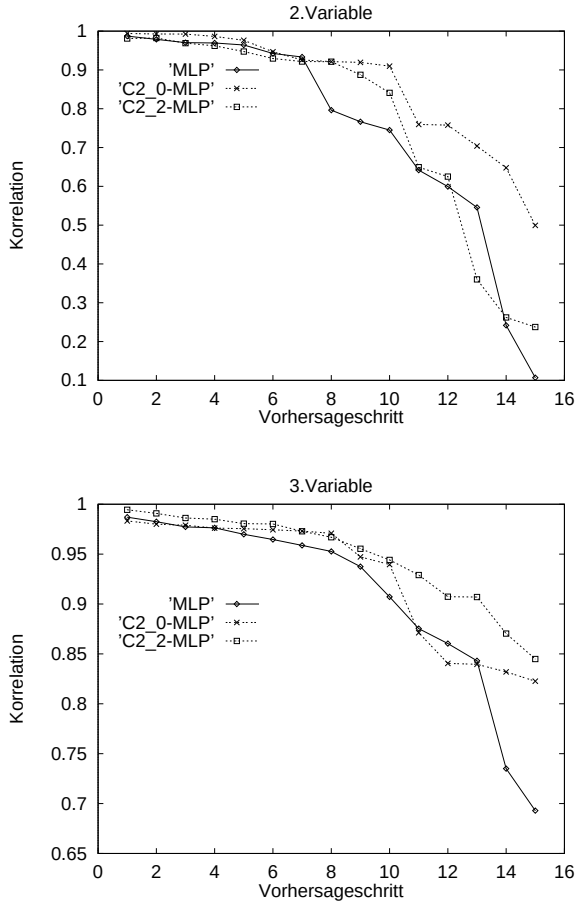


Abbildung 4.9: *Korrelationsverläufe der einzelnen Variablen bei der Vorhersage des Lorenz–Attraktors*

Die beim MLP verwendete Knotenzahl war keinesfalls zu groß bemessen, wie der weiteren Entwicklung der Korrelationen für dieses Netz entnehmbar ist. Bei der Auswertung der Ergebnisse ist auffällig, daß das $\mathcal{C}_{2,2}$–MLP zwar gegenüber dem $\mathcal{C}_{2,0}$–MLP bezüglich der 1. und 3.Variable besser, bezüglich der 2.Variable andererseits aber erheblich schlechter war und somit insgesamt beide CA–MLPs letztendlich vergleichbar gute Vorhersageergebnisse lieferten. Das MLP konnte, außer hinsichtlich der 1.Variable, bei der es erhebliche Probleme hatte, mit dem jeweils schwächeren CA–MLP bis zu einer Schrittweite von $\tau = 13$ mithalten.

Über die anschauliche Bedeutung der einzelnen erzielten Korrelationswerte gibt Abbildung 4.10 ein wenig Aufschluß.

Die von den CA–MLPs vorhergesagten Attraktoren sind tatsächlich nahezu völlig identisch, was man aus ihren gleichen numerischen Korrelationswerten noch nicht automatisch schließen durfte. Interessant ist, das alle Netze bei der Vorhersage die selben strukturellen Fehler machten, denn die schlechteren Ergebnisse des MLPs, resultierten größtenteils aus einem Skalierungsfehler bei der rechten Schleife des Attraktors.

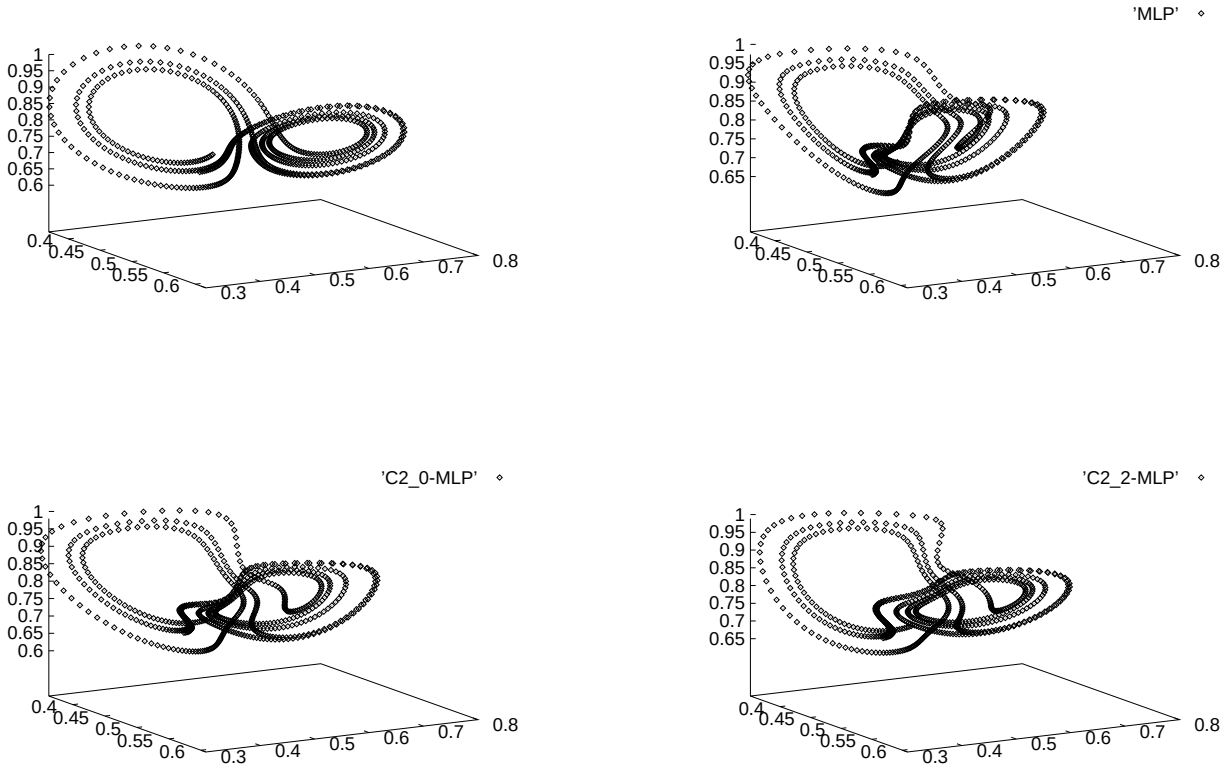


Abbildung 4.10: *Erwartete (links oben) und vorhergesagte Werte für den Lorenz–Attraktor bei* $\tau = 8$

Nachdem wir die Ergebnisse qualitativ ausgewertet haben, wollen wir nun den dabei aufgetretenen Verbrauch an Ressourcen betrachten. Da ein 3–dimensionales Problem vorlag, mußte zur Bestimmung der Parameteranzahl des MLPs die Formel

$$Var_{(3,nh,3)-MLP} := 7 \cdot nh + 3 \tag{4.10}$$

angewandt werden, was eine Reduzierung der Parameteranzahl um 2 pro versteckten Knoten im Vergleich zu einem $(4, nh, 4)$–MLP bedeutet (4.6), und genau den Nachteil angibt, den wir zusätzlich durch die notwendige Einbettung des Problems in 4 Dimensionen für die CA–MLPs in Kauf nehmen müssen. Im für das Beispiel des Lorenz–Attraktors relevanten Bereich haben wir die in Tabelle 4.11 aufgeführten Laufzeiten auf einem `Pentium/133` unter `Linux 1.2.8 (gcc 2.6.3 -O2)` gemessen, die sich auf 4000 Trainingszyklen beziehen.

Die Reduktion reeller Parameter bei den für den Lorenz-Attraktor zum Einsatz gekommenen $(1, 5, 1)$–CA–MLPs gegenüber dem $(3, 12, 3)$–MLP war natürlich nicht gänzlich ausreichend, um eine gleichschnelle Laufzeit zu bedingen, aber doch schon stark genug, um eine weit unter dem zu befürchtenden Ausmaß liegende Laufzeit zu bewirken, die bei 4 Knoten im CA–MLP sogar nahezu gleich wäre. Ebenso würde eine bei sehr großem τ indizierte Erhöhung der Knotenzahl des MLPs das Bild noch wesentlich verbessern.

	$\mathcal{C}_{2,s}$-MLP			MLP			
versteckte Knoten	4	**5**	6	11	**12**	13	14
Laufzeit (in s)	78	**94**	112	68	**73**	79	84
Parameter	52	**64**	76	80	**87**	94	101

Tabelle 4.11: *Speicher- und Zeitkomplexität von MLP und CA–MLPs*

Insgesamt dürfen wir also annehmen, daß die bestehenden Nachteile bezüglich der Zeitkomplexität der CA—MLPs gegenüber vergleichbaren MLPs, sich bei größeren Aufgaben nur noch schwach in der Laufzeit niederschlagen, d.h. kompensiert werden, und zwar bei mindestens gleichguten Ergebnissen und einer teils erheblichen Ersparnis benötigter Gewichte.

Fazit Wir haben in diesem Kapitel mehrere Simulationen durchgeführt, die hinsichtlich Schwere und Art einer Lernaufgabe eine relativ große Bandbreite abgedeckt haben. Die dabei erhaltenen Ergebnisse für CA–MLPs sind mehr als zufriedenstellend. Ihre Anwendung gegenüber einem vergleichbaren MLP bedeutet in nahezu allen Fällen eine Reduzierung der Parameteranzahl. In vielen Simulationen wurden die jeweilige Aufgabe nach weit weniger Trainingszyklen gelernt als durch das MLP bei vergleichbaren Fehlern. Besonders gut schnitten die CA–MLPs bei Aufgaben ab, deren Dimensionalität eine Einbettung des Problems erforderlich machten. Hier ist der von uns mit dem Übergang von einfachen reellen Vektorräumen zu höherdimensionalen zusätzlich algebraisch strukturierten Vektorräumen intendierte Vorteil erkennbar.

Kapitel 5

Analyse geometrischer Kodierungen

Unser bisheriges Vorgehen folgte im wesentlichen einem top–down–Konzept. Ausgehend von bekannten Bereichserweiterungen (Komplexe Zahlen und Quaternionen) für Neuronaler Netze, ist es uns im Kapitel 1 gelungen, diese Erweiterungen als konkrete algebraische Einbettungen reeller Vektorräume zu erkennen, und sie dann zu einem konsistenten umfassenden Modell zu verallgemeinern, nämlich das der Clifford–Algebren. Danach haben wir versucht bekannte Architekturen Neuronaler Netze in diesem Rahmen zu entwickeln. Für RBF–Netze haben wir schnell eingesehen, daß wir die reichhaltigen strukturellen Eigenschaften unserer Algebren dort garnicht ausnützen können. Für MLPs gelang uns dies dann sehrwohl, und zwar gerade deshalb, weil wir einen top–down–Ansatz zur Verfügung hatten und dadurch alle technischen Probleme beherrschbar waren, wobei bei allerdings häufig nur noch reine Mathematik zur Anwendung kam.

In diesem Kapitel wollen wir unabhängig von einer konkreten Architektur das geometrische Potential unseres Ansatzes veranschaulichen, indem wir untersuchen wollen, welche Merkmale sich durch das geometrische Produkt ausbilden. Damit stehen also nocheinmal die elementaren Grundlagen unseres Ansatzes im Blickpunkt. Zu Beginn befassen wir uns daher insbesondere nocheinmal mit der Interpretierbarkeit von Clifford–Algebren als Geometrische Algebren.

Eine Geometrische Algebra wird üblicherweise als assoziative Algebra definiert, in der das Quadrat jedes Vektors stets ein Skalar ist.

Demzufolge ist also jede Clifford–Algebra eine Geometrische Algebra wie wir dies auch in Kapitel 1 bereits festgestellt hatten. Angemerkt sei, daß die obige Definition Geometrischer Algebren zur Beantwortung systematischer Fragestellungen (etwa der nach der Anzahl Geometrischer Algebren bestimmter Dimension) wenig geeignet ist, sondern eher anschaulichen Charakter trägt, weshalb wir sie nicht zur Grundlegung unserer theoretischen Einbettungen benutzt haben und auch nicht wirklich konnten. Auf der anderen Seite mag man argumentieren, daß das Potential unseres Ansatzes dort womöglich besser zur Geltung kommt.

Haben wir also der geometrischen Interpretierbarkeit unseres algebraischen Modells bisher schon genug Aufmerksamkeit geschenkt, oder hätten wir mehr von ihr profitieren können und müssen?

Die Antwort ist, daß die geometrische Interpretierbarkeit letztendlich auf dem Prinzip der Einbettung beruht, deren explizite Anwendung beim XOR–Problem und bei der Vorhersage des Lorenz–Attraktors zu sehr guten Ergebnissen geführt hat. Wenn wir jetzt weiter fragen, ob dabei Geometrie gelernt oder benutzt wurde, müßten wir scheinbar die Antwort schuldig bleiben. Die Antwort lautet aber einfach, daß von den CA–MLPs neue gewinnbringende Merkmale gelernt wurden, die sicher auch eine geometrische Interpretation besitzen.

Unser Ziel in diesem Kapitel ist es nun zu illustrieren, daß durch Anwendung des geometrischen Produktes entstehende Kodierungen geometrischer Natur sind. Dazu ist es vorher zunächst hilfreich, den Bezug unseres Ansatzes zu sogenannten *Neuronalen Netzen höherer Ordnung* (HONNs) herzustellen.

5.1 Clifford–Netze und HONNs

Die grundlegende Idee von Neuronalen Netzen höherer Ordnung besteht darin, Eingabemuster ohne Hinzunahme weiterer Informationen in einen erweiterten Merkmalsraum abzubilden. Eine Möglichkeit dazu stellt das sogenannte *Außenprodukt-Modell* [Pao89] dar, bei dem jede Komponente des Eingabemusters mit dem gesamten Eingabemuster multipliziert wird. Die Eingabemuster erhalten durch diese Erweiterung dann auch nichtlineare Terme, die Ordnung des höchsten dabei vorkommenden Terms bestimmt dann gleichsam die Ordnung des Neuronalen Netzes. Es ergeben sich sofort gewisse Parallelen zu unseren Clifford–Netzen. So gestattet die verbesserte Repräsentation der Netzeingabe meist eine Reduktion der Netzgröße Neuronaler Netze höherer Ordnung im Vergleich zu herkömmlichen Netzen, wobei dies natürlich mit einer je nach Aufgabenstellung sehr schnell anwachsenden Anzahl der Eingabeterme einhergeht. Eine gängige Methode zur Verringerung dieses Zuwachses ist die Vernachlässigung von Termen, die über eine größere Anzahl von Eingabemustern keine oder nur geringe Korrelation aufweisen. HONNs werden meist auf diskreten Bereichen, z.B. bei der Merkmalsberechnung aus 2-D Bildern, und häufig nur als Vorverarbeitungsschritt angewandt.

Für den Fall, daß mit Clifford–Netzen Probleme bearbeitet werden, deren Dimension kleiner als die der Verwendung findenden Clifford–Algebra ist, arbeiten diese quasi in einem Modus, der den Prinzipien eines Neuronalen Netzes höherer Ordnung angelehnt ist, aber auch dann noch fundamentale Unterschiede und Erweiterungen aufweist, vornehmlich die, daß die Einbettung vom Clifford–Netz selbst erzeugt wird, diese eine algebraische Struktur aufweist und das Lernverfahren auch auf dieser beruht. Damit werden wesentliche Nachteile Neuronaler Netze höherer Ordnung umgangen bzw. aufwendige Anpassungsschritte vom Clifford–Netz selbstständig durchgeführt.

Bei Neuronalen Netzen höherer Ordnung wird ja nur die Eingabe erweitert, d.h. die Expansion findet vorher genau festgelegt statt und dieser Schritt (genauer die eindeutige Folge sukzessiver Teilschritte), d.h. die Selektion einer bestimmten Kodierung, will wohlüberlegt sein und erfordert im Praxisfall viel Vorkenntnis und Erfahrung. Bei Clifford–Netzen findet die Expansion während des Lernens dynamisch statt und wird somit auch ständig optimiert. Die Vorteile sind evident. Als einziger Einwand verbleibt höchstens, daß man die Kontrolle an das Netz abgibt, welches ja nur bestimmte Kodierungen zur Verfügung hat, von denen alle potentiell ungeeignet für ein bestimmtes Problem seien könnten, während man ein Neuronales Netz höherer Ordnung (insbesondere eine geeignete Problemkodierung) sehr leicht und unzweifelhaft auch optimal angeben kann. Damit ist aber auch klar, daß ein HONN auf ein Problem(typ) maßgeschneidert wird, wobei — wie bereits erwähnt — meist ein diskreter Wertebereich zugrunde liegt. Clifford–Netze konkurrieren also im allgemeinen nicht mit HONNs, und wenn dann müßten sie es auf der Ebene der Kodierungen, und zwar mit anderen Kodierungen geometrischer Merkmale.

Wir untersuchen daher nun zuerst, welche konkreten Möglichkeiten von Verknüpfungen sich in 4–dimensionalen Clifford–Algebren für 2 Vektoren (x_1, y_1) und (x_2, y_2) des $\mathbb{R}^2$ ergeben. Die Kodierungen dieser Vektoren als Multivektoren sind dann

$$mv_1 := (0, x_1, y_1, 0) \text{ und } mv_2 := (0, x_2, y_2, 0)\,. \tag{5.1}$$

Deren Verknüpfung über das geometrische Produkt ergibt dann einen Multivektor $mv := (z_1, 0, 0, z_2)$, der je nach verwendeter Algebra und eventuell noch zusätzlich angewendeter Konjugation die in Tabelle 5.1 angegebene Gestalt hat.

Operation	$mv[1]$	$mv[4]$
$mv_1 \otimes_{2,0} mv_2$	$-x_1x_2 - y_1y_2$	$x_1y_2 - y_1x_2$
$mv_1 \otimes_{2,1} mv_2$	$x_1x_2 - y_1y_2$	$x_1y_2 - y_1x_2$
$mv_1 \otimes_{2,2} mv_2$	$x_1x_2 + y_1y_2$	$x_1y_2 - y_1x_2$
$mv_1 \otimes_{2,0} \overline{mv_2}$	$x_1x_2 + y_1y_2$	$-x_1y_2 + y_1x_2$
$mv_1 \otimes_{2,1} \overline{mv_2}$	$-x_1x_2 + y_1y_2$	$-x_1y_2 + y_1x_2$
$mv_1 \otimes_{2,2} \overline{mv_2}$	$-x_1x_2 - y_1y_2$	$-x_1y_2 + y_1x_2$

Tabelle 5.1: *Resultate der Verknüpfung zweier Vektoren in 4–dimensionalen Clifford–Algebren*

Der jeweils enstehende Bivektor z_2 ist wie wir ebenfalls aus Kapitel 1 bereits wissen die (negative) Determinante und somit ein geometrisches Merkmal, genauso wie das in der Komponente z_1 vorhandene (bzw. herstellbare) Skalarprodukt. Somit haben alle 4-dimensionalen Clifford–Algebren dasselbe geometrische Potential, welches wir jetzt exemplarisch mit dem anderer natürlicher geometrischer Merkmale hinsichtlich der Nützlichkeit in der Vorverarbeitung vergleichen wollen.

5.2 Experimenteller Vergleich verschiedener Kodierungen

Das zu diesem Zwecke von uns durchgeführte Experiment ist eine sehr einfache Version der Untersuchungen in [Edelman92] zur Erkennung von Drahtgittermodellen aus verschiedenen 2-D Ansichten mit RBF–Netzen. Dieses Experiment wurde von uns wegen seiner einfachen geometrischen Interpretierbarkeit gewählt und auch um Kombinationsmöglichkeiten von geometrischer Vorverarbeitung und Netzarchitekturen, die dafür nicht ohne weiteres erweiterbar sind, zu demonstrieren.

Der Objektdatensatz bestand aus 8 3-D Drahtgittermodellen mit jeweils 6 Punkten, von denen das erste in Abbildung 5.1 dargestellt ist.

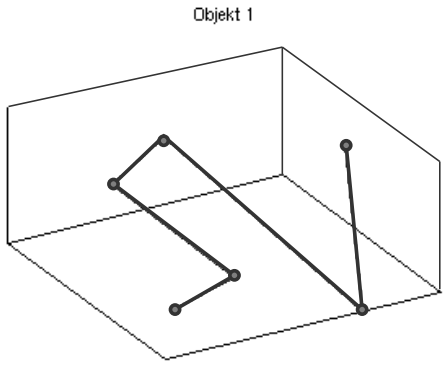


Abbildung 5.1: *3-dimensionales Drahtmodell von Objekt 1*

Zu jedem solchen Modell wurden 10 2-D Ansichten generiert, durch Drehung des Objektes um zufällige Rotationswinkel $\theta, \phi \in [0, 20]$ bezüglich im Objektzentrum zentrierter Kugelkoordinaten und anschließender orthographischer Projektion in die xy–Ebene, die als Trainingsansichten verwendet wurden. Die einfache orthographische Projektion der Objekte ohne Drehung lieferte dessen Standardansicht und wurde als Lernziel verwendet. In Abbildung 5.2 sind die Standardansichten der ersten vier Testobjekte dargestellt.

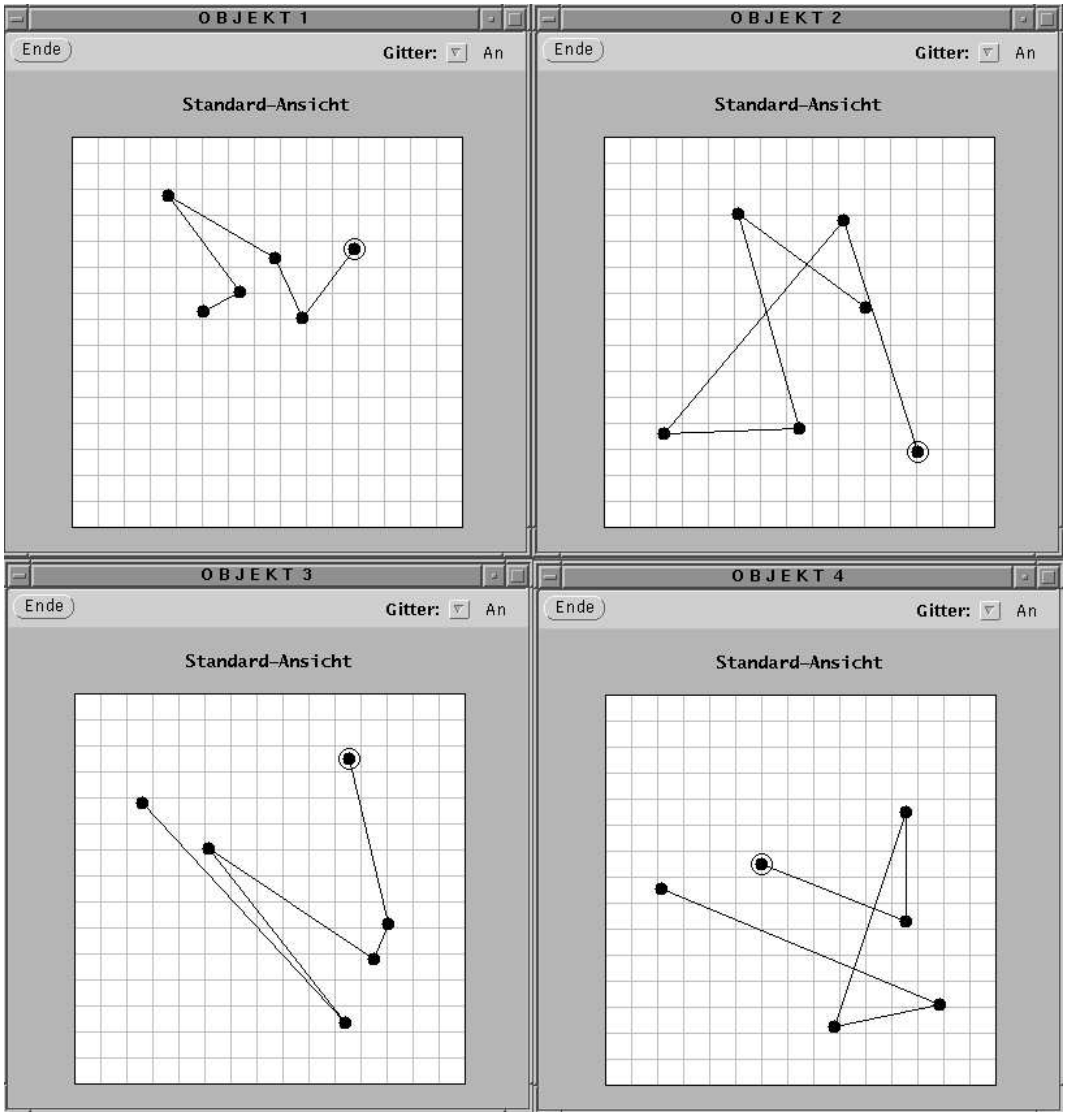


Abbildung 5.2: *Standardansichten der Objekte 1–4*

Für die Multivektorkodierungen wurde das Ergebnis der Anwendung des geometrischen Produktes der Clifford–Algebra $\mathcal{C}_{2,0}$ zwischen benachbarten Objektpunkten (mv), sowie gemäß Tabelle 5.1 als Einzelkodierungen die Projektionen in die Komponente (mv[1]) und in die 4.Komponente (mv[4]) benutzt. Diese Multivektorkodierungen mußten mit den geometrisch höchstplausiblen Kriterien der Länge und der Winkel zwischen benachbarten Punkten bzw Kanten konkurrieren, wobei selbstverständlich die selbe Reihenfolge der Punkte bei der Generierung dieser Merkmale zugrundegelegt wurde.

Tabelle 5.2 enthält die sich für die einzelnen Merkmale ergebenden Größen der Eingabe.

Kodierung	Größe der Eingabe
Winkel	4
Längen	5
Punkte	12
mv	10
mv[1]	5
mv[4]	5

Tabelle 5.2: *Kodierungen und resultierende Größe der Eingabe*

Zum Test der Netze wurden die Ansichten der Objekte bei gleichen ganzzahligen Rotationswinkeln zwischen 0^o und 20^o benutzt. In Tabelle 5.3 sind die minimalen, maximalen und durchschnittlichen Fehler aufgeführt. In den Abbildungen 5.3 und 5.4 sind für alle Objekte die Fehler im einzelnen dargestellt.

	Objekt 1			**Objekt 2**			**Objekt 3**			**Objekt 4**		
	min	max	Ø	min	max	Ø	min	max	Ø	min	max	Ø
Winkel	0.23	1.31	0.65	0.02	2.26	0.77	0.96	1.43	1.21	0.27	0.93	0.47
Längen	0.12	0.91	0.45	0.02	1.22	0.35	0.00	0.50	0.33	0.27	1.26	0.92
Punkte	0.00	0.20	0.06	0.02	0.87	0.24	0.00	0.33	0.07	0.21	0.54	0.32
mv	0.07	0.26	0.13	0.03	0.82	0.24	0.00	0.37	0.06	0.28	0.56	0.37
mv[1]	0.20	0.46	0.27	0.12	0.76	0.23	0.08	0.75	0.24	0.38	0.84	0.50
mv[4]	0.04	0.32	0.19	0.03	1.27	0.35	0.00	0.40	0.08	0.45	0.91	0.54

	Objekt 5			**Objekt 6**			**Objekt 7**			**Objekt 8**		
	min	max	Ø	min	max	Ø	min	max	Ø	min	max	Ø
Winkel	0.04	1.03	0.31	0.43	0.55	0.44	0.44	0.78	0.50	0.57	0.76	0.61
Längen	0.61	0.76	0.63	0.43	1.11	0.64	0.00	0.74	0.20	0.17	0.26	0.19
Punkte	0.18	0.22	0.18	0.23	0.76	0.37	0.10	0.35	0.16	0.06	0.31	0.12
mv	0.11	0.18	0.12	0.21	0.78	0.35	0.04	0.85	0.20	0.00	0.05	0.01
mv[1]	0.26	0.40	0.32	0.22	0.89	0.41	0.13	0.70	0.25	0.11	0.26	0.14
mv[4]	0.33	0.39	0.42	0.28	0.96	0.47	0.00	1.18	0.28	0.00	0.44	0.13

Tabelle 5.3: *Ergebnisse für die Erkennung der Drahtgittermodelle*

Der Unterschied zwischen der Multivektorkodierung und der direkten Verwendung der Punkte als Eingabe ist insgesamt gering. Die Objekte 1,4 und 7 wurden durch letztere besser erkannt, bei den Objekten 5,6 und 8 lieferte die Multivektorkodierung bessere Ergebnisse, die aber nur bei Objekt 8 deutlich besser waren. Der Vorteil der Punktkodierung beim Objekt 7 resultiert aus einer Verschlechterung der Multivektorkodierung für größere Winkel. Ansonsten bewegen sich die Fehler bei diesen Kodierungen, mit der abermaligen Ausnahme des Objektes 8, jeweils in den gleichen Bereichen. Auch sind die Kurvenverläufe der Fehler selbst meist ähnlich, mit Ausnahme der Objekte 1 und 8. Für die Anwendung des geometrischen Produktes spricht aber auch hier wieder die damit verbundene Reduk-

tion der Problemgröße von 12 auf 10 Komponenten, was ein schnelleres Lernen ermöglicht, bei nur konstanten Kosten für die Herstellung der Multivektorkodierung.

Bei den anderen Merkmalen waren die Kodierungen mv[1] und mv[4] meist etwas besser als ihre Konkurrenten und konnten grobe Ausreißer wie sie bei Winkeln oder Längen auftraten, gänzlich vermeiden. Auch kam bei ihnen kein linearer Fehlerverlauf auf hohen Niveau vor, was bei den Objekten 3 und 8 für Winkel und beim Objekt 5 für Längen der Fall war.

Der Abstand zwischen den einzelnen Komponenten der Multivektorkodierung und dieser selbst ist meist gering, ganz im Gegensatz zu den Verhältnissen bei den klassischen Merkmalen, wo dies eher die Ausnahme ist.

Dies erlaubt eine sehr positive Schlußfolgerung. Das geometrische Produkt verteilt die Information über größere Regionen so in den Komponenten des entstehenden Multivektors, daß möglichst wenig Information verloren geht, d.h. es teilt nicht einfach auf, sondern versucht zu komprimieren und das Ganze in jedem Teil bestmöglich zu erhalten.

Noch interessanter ist, daß dabei zwei grundsätzlich verschiedene Methoden zu beobachten sind, eine Gleichverteilung (Objekte 5 und 6) und eine komplementäre Verteilung (Objekte 1,2,4 und 7). Scheinbar gibt es Objekte, die die eine Kodierung bewirken und eben solche, die die andere erforderlich machen.

Insgesamt konnten wir mit unseren kleinen Experiment demonstrieren, daß durch das geometrische Produkt Merkmale gebildet werden, die geometrische Eigenschaften von Daten komprimiert zusammenfassen, und meist besser sind als herkömmliche Kodierungen.

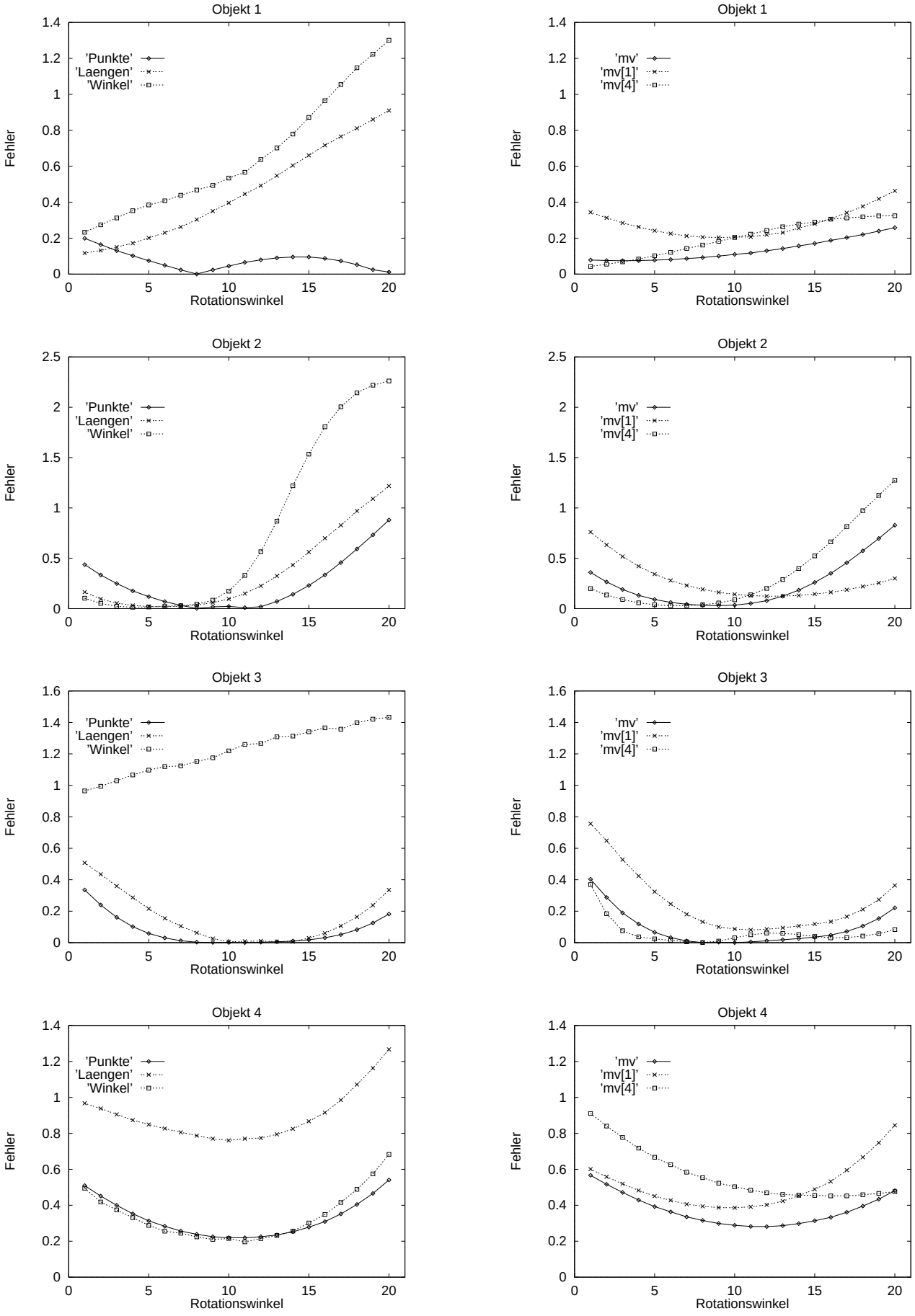


Abbildung 5.3: *Fehler bei Testansichten für die Objekte 1-4*

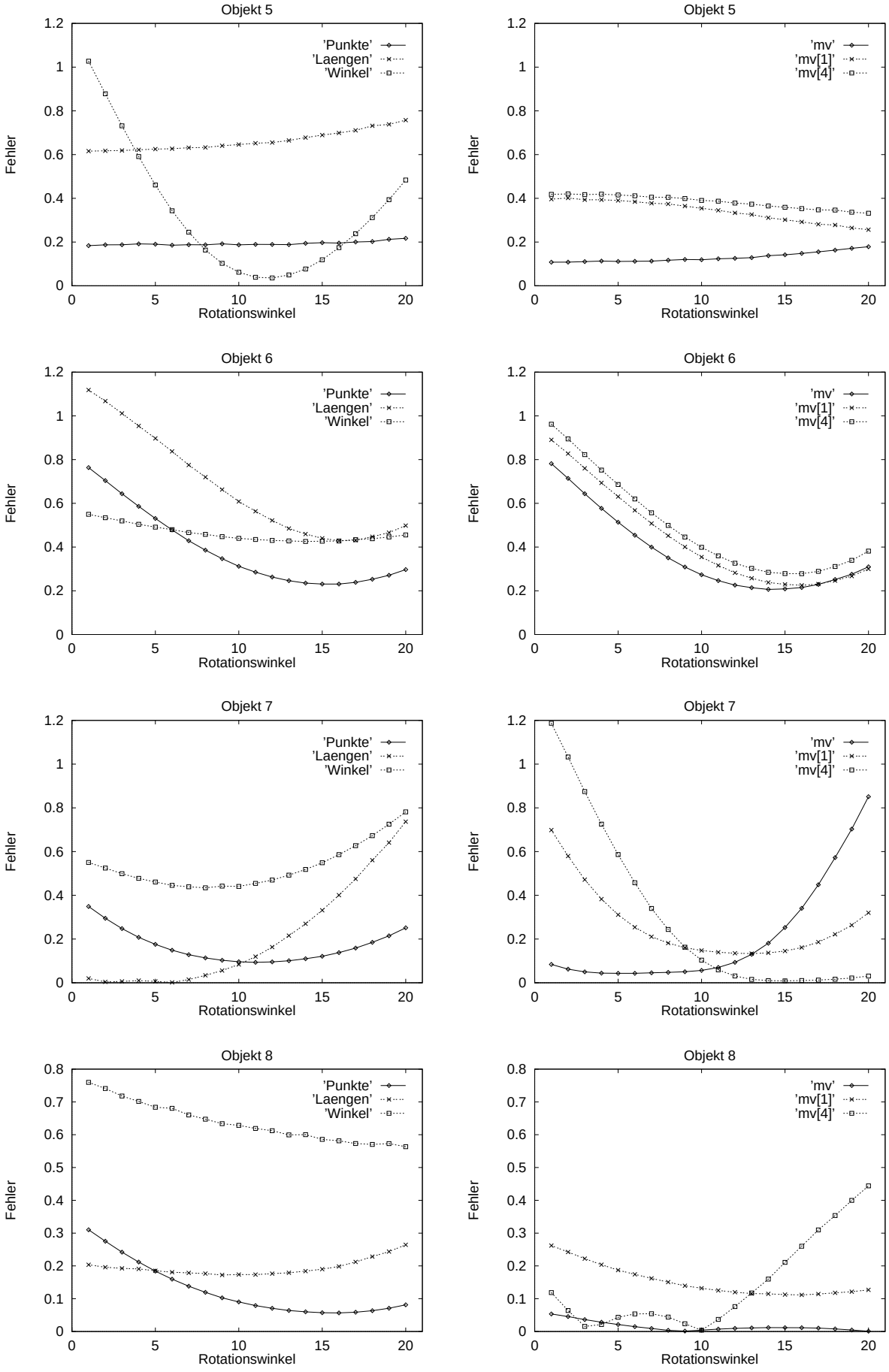


Abbildung 5.4: *Fehler bei Testansichten für die Objekte 5-8*

Kapitel 6

Resümee

In der Einleitung zu dieser Arbeit hatten wir mehrere generelle Probleme erkannt, die es bei der Entwicklung Neuronaler Netze in algebraischen Erweiterungen der reellen Zahlen zu bewältigen gilt. Begutachten wir nun abschließend, ob und wie wir sie gelöst haben, und ob wir unsere formulierten Ziele erreichen konnten.

Zunächst wollen wir nocheinmal festhalten, daß wir für unsere Untersuchungen kein algebraisches Modell gewählt haben, sondern dieses — die Clifford–Algebra — konstruktiv aus unserer Motivation der Erzeugung neuer Entitäten aus Vektoren hergeleitet haben und es durch rein mathematische Überlegungen auch als maximal nachweisen konnten. Daher sind die von uns entwickelten Clifford–Netze und zugehörigen Lernverfahren mehr als bloß eine spezielle Erweiterung reeller Netze, sie sind deren folgerichtige Verallgemeinerung.

Das geometrische Produkt einer Clifford–Algebra, welches das Potential der algebraischen Einbettungen als Operator verkörpert, genügt leider nur noch relativ schwachen algebraischen Gesetzen. Neben den damit verbundenen zu erwartenden technischen Schwierigkeiten konnten wir auch gleich am Beispiel der RBF–Netze das Auftreten des zweiten vorhergesagten Problems beobachten. Die speziellen Berechnungsparadigmen von RBF–Netzen erlauben schlichtweg keine einfache und sinnvolle Erweiterung mittels des geometrischen Produktes.

Die folgende Konzentration auf die Architektur des MLPs war daher angezeigt, und trägt auch dem bisherigen Stand der Forschung Rechnung. Die ausführliche Behandlung der CMLPs erlaubte uns erste hilfreiche Abstraktionen von konkreten Aktivierungsfunktionen hinzu Klassen solcher Funktionen, sowie erste prinzipielle Aussagen zu deren Handhabbarkeit.

Die in [Pearson94] entwickelten MLPs basieren auf nicht–komponentenweisen Funktionen. Wie wir nachweisen konnten, ist aber die gesamte Klasse von Funktionen in Clifford–Algebren mit Nullteilern nicht anwendbar, unabhängig von der konkreten Gestalt der

Verwendung findenden Aktivierungsfunktion. Für den Entwurf von Algorithmen stellt das Vorhandensein von Nullteilern allgemein ein großes Problem dar. Um so verwunderlicher ist es, daß diesem Umstand in der zitierten Arbeit keinerlei Aufmerksamkeit gewidmet worden ist, da der Ansatz aufgrund ihrer Existenz sofort fraglich und theoretisch sehr zweifelhaft ist. Wie wir experimentell zeigen konnten, sind daher derartige Netze auch in der Praxis unzuverlässig.

Es verblieben also einzig die komponentenweisen Aktivierungsfunktionen, um MLPs in Clifford–Algebren zu entwickeln. Mit dem QMLP aus [Arena96a] [Arena96b] stand dabei schon ein Grundgerüst zur Verfügung. Dessen Erweiterbarkeit für algebraische Strukturen mit Nullteilern wurde von den Entwicklern selbst in [Arena96b] allerdings bezweifelt. Durch die Verallgemeinerung der Konjugation im Backpropagation–Algorithmus des QMLPs mit Hilfe der Klasse der vorzeichenvertauschenden Abbildungen gelang uns die Herleitung korrekter Backpropagation–Lernverfahren für beliebige Clifford–Algebren. Interessanterweise gibt es für jede Clifford–Algebra genau eine derartige Abbildung, die zu einem korrekten Lernverfahren führt, wie wir ebenfalls bewiesen haben. Mit dem Nachweis der entsprechenden Approximationssätze, durch Zurückführung auf den Spezialfall der Quaternionen [Arena96a], gelangten wir schließlich zu vollständig korrekten Clifford–Algebra MLPs, womit die Barriere der Nullteiler von uns erstmalig erfolgreich durchbrochen wurde. Nichtsdestotrotz muß man Nullteilern beim Entwurf anderer Clifford–Netze als Clifford–Algebra MLPs noch Aufmerksamkeit schenken und wegen ihres Vorhandenseins mit Problemen rechnen.

Für die Clifford–Algebra MLPs können wir aufgrund unserer experimentellen Ergebnisse ein positives Fazit ziehen. Bei all unseren doch recht verschiedenartigen Untersuchungen zeigte sich im Mittel eine um 20% geringere Platzkomplexität der CA–MLPs gegenüber herkömmlichen MLPs bei gleicher Performance. Hinsichtlich der Laufzeiten ist daher der Vorteil des MLPs im Vergleich zu den CA–MLPs letztendlich eher gering, selbst im ungünstigen Fall nicht volldimensionaler Probleme. Auf der Ebene der Effizienz können unsere Netze also mit den MLPs bereits jetzt schon mithalten, obwohl ihr Optimierungspotential sicher noch nicht ausgeschöpft ist. Wichtiger ist aber natürlich, daß wir wohl auch die Frage nach konzeptionellen Vorteilen unserer Netze positiv beantworten können, was für die starke Plausibilität unseres Modells spricht.

Konzeptionelle Vorteile in Form von wesentlich besserer Leistung bezüglich vergleichbarer MLPs waren besonders beim XOR–Problem und beim Lorenz–Attraktor zu beobachten, traten also dann auf, wenn von den Netzen selbst eine Einbettung des Problems in einer höherdimensionalen Clifford–Algebra in der versteckten Schicht erzeugt wurden. Aber auch bei einigen Beispielen zur Funktionsapproximation waren konzeptionelle Vorteile im Ansatz zu erkennen. Dort hatten sich auch schnell weitere Fragen ergeben, wie die nach dem besten CA–MLP für einen bestimmten Datensatz zu deren Beantwortung die Anzahl unserer Experimente nicht ausreichend war, aber deren weitere Untersuchung angezeigt ist.

Mit den CA–MLPs haben wir ein qualitativ neues Netz entwickelt, weil durch diese Netze dynamisch Einbettungen in höherdimensionale Algebren erzeugt werden, was mit MLPs

natürlich nicht realisiert werden kann. Die Erzeugung der Einbettungen haben wir auch mit den Funktionsprinzipien Neuronaler Netze höherer Ordnung verglichen, und dabei ebenfalls die neue Qualität unserer Clifford–Algebra MLPs unter diesen Gesichtspunkten herausgearbeitet. Clifford–Netze adaptieren die Idee der Merkmalserweiterung bei Neuronaler Netzen höherer Ordnung, führen diese aber dynamisch über den Lernprozeß selbstständig durch.

Das die aus dem geometrischen Produkt resultierenden Kodierungen plausibel und mindestens gleichgute Ergebnisse wie andere geometrische Kriterien hervorbringen können, konnten wir mit dem Experiment zur Erkennung von Drahtgittermodellen zeigen. Die Resultate haben gleichsam nocheinmal das geometrische Potential von Clifford–Algebren unterstrichen. Daher ist auch die Interpretation der gelernten internen Kodierung eines CA–MLPs sinnvoll und möglich, ein sehr schönes Beispiel dafür lieferte das $\mathcal{C}_{2,0}$–MLP für die XOR–Funktion, wo eine symmetrische interne Kodierung beobachtet werden konnte. Man kann also von der Art wie das Netz eine Aufgabe lernt selbst wieder Rückschlüsse ziehen. Hierbei ist die geringere Parameterzahl von CA–MLPs natürlich ein zusätzlicher Vorteil.

Eine abschließende positive Bewertung des Potentials von Clifford–Netzen fällt aufgrund der in dieser Arbeit insgesamt erzielten Ergebnisse daher leicht. Für weiterführende Untersuchungen bietet sich eine Vielzahl von Ansatzpunkten. So haben wir uns in dieser Arbeit ja nahezu ausschließlich auf 4–dimensionale Clifford–Algebren beschränkt, um aufgrund ihrer Überschaubarkeit das Wesen unseres Ansatzes und die Prinzipien der von uns entwickelten Methoden möglichst klar herauszuarbeiten. Allein mit dem Übergang zu 8–dimensionalen Clifford–Algebren dürften sich eine Reihe interessanter Ergebnisse gewinnen lassen, insbesondere wenn man wieder Einbettungen in diese Algebren untersucht bzw. diese durch entsprechende Clifford–Netze dynamisch erzeugen läßt. Mit diesen Ergebnissen könnte man wohl bereits Antworten auf Fragen finden, die in dieser Arbeit, die hauptsächlich der Schaffung theoretischer Grundlagen und dem Gewinnen erster Einsichten gewidmet war, naturgemäß noch offen bleiben mußten.

So wäre zum Beispiel die Frage von Interesse, ob und unter welchen Bedingungen Clifford–Algebra MLPs zur Ausbildung symmetrischer interner Kodierungen neigen, und ob dies für Netze über verschiedenen Clifford–Algebren im gleichen Maße zutrifft. Von großem Interesse ist natürlich überhaupt die Gewinnung genereller Aussagen über die unterschiedliche Eignung gleichdimensionaler Clifford–Netze, die wir beobachten, aber noch nicht systematisieren konnten. Womöglich kann man die Entscheidung, welche Clifford–Algebra für ein Problem besonders gut geeignet ist, sogar von Netz selbst treffen lassen, indem man den Wechsel von Clifford–Algebren während des Lernens selbst erlaubt, was zuvor jedoch noch die Erkennung geeigneter Kriterien erfordert. Die experimentellen Möglichkeiten dürften hier sehr vielfältig seien. Das bessere Verstehen der Bedeutung der Anwendung des geometrischen Produktes ist natürlich auch noch eine verbleibende Aufgabe. Hier bietet das von uns durchgeführte Experiment zur Erkennung von Drahtgittermodellen noch einen sehr großen Spielraum, so könnte man versuchen die unterschiedlichen Arten der Kodierungen von Objekten durch das geometrische Produkt zur Klassifizierung der Objekte selbst zu nutzen. Auf dem Gebiete der Vorverarbeitung, aber auch damit sind die

Möglichkeiten der Vorverarbeitung mittels des geometrischen Produktes erst angedeutet.

Mit der zu erwartenden Etablierung von Clifford–Algebren bzw. Geometrischen Algebren in anderen Gebieten wie Computer Vision oder Robotik dürfte auch die Entwicklung Neuronaler Netze in diesen algebraischen Strukturen weitere Impulse erhalten und verstärktes Interesse finden. Umgekehrt werden natürlich auch Anstöße durch die Entwicklung von Clifford–Netzen auf diese und andere Gebiete ausgehen. Womit der große Vorteil des Modells der Clifford–Algebren bzw. der Geometrischen Algebren nocheinmal deutlich wird. Dieses Modell stellt einen universellen Beschreibungsrahmen für verschiedenartige Probleme zur Verfügung. Mit seiner Anwendung geht dann auch eine Vereinheitlichung der Methoden der unterschiedlichen Disziplinen einhergeht und auch zu einem verstärkten und fruchtbaren interdisziplinären Denken und Handeln bei der Entwicklung künstlicher autonomer Systeme, die natürlich und auch notwendig ist, aber zuweilen noch durch eine unterschiedliche Sprache erschwert wird.

Die Plausibilität, Realisierbarkeit und Stärke des Modells der Clifford–Algebra für Neuronale Netze zu demonstrieren war die Hauptzielstellung dieser Arbeit.

Anhang A

Algebraische Grundbegriffe

- Halbgruppe

  Sei $(G, \cdot)$ ein Verknüpfungsgebilde. G heißt *Halbgruppe* (bzw. $\cdot$ heißt *assoziativ*), wenn gilt:

  $$(a \cdot b) \cdot c = a \cdot (b \cdot c)$$

  für alle $a, b, c \in G$.

- Gruppe

  Sei $(G, \cdot)$ eine Halbgruppe. G heißt Gruppe, wenn gilt: (Existenz eines neutralen Elementes)

  $$\exists\, e \in G \,\forall\, x \in G : \; e \cdot x = x \cdot e = e$$

  und (Existenz von Inversen)

  $$\forall\, x \in G \,\exists\, y \in G : \; x \cdot y = y \cdot x = e\,.$$

  Ist zusätzlich $\cdot$ kommutativ, so heißt G *abelsche Gruppe*.

- Ring

  Ein Tripel $(R, +, \cdot)$ heißt *Ring*, wenn $(R, +)$ eine abelsche Gruppe, $(R, \cdot)$ eine Halbgruppe ist, und wenn für alle $a, b, c \in R$ die Distributivgesetze gelten:

  $$(a + b) \cdot c = a \cdot c + b \cdot c \;\text{ und }\; c \cdot (a + b) = c \cdot a + c \cdot b\,.$$

- Nullteiler

  Sei $(R, +, \cdot)$ ein Ring. Ein Element $a \in R$ heißt *Links–Nullteiler* (bzw. *Rechts–Nullteiler*), falls ein $b \in R\backslash\{0\}$ existiert mit $a \cdot b = 0$ (bzw. $b \cdot a = 0$).
  In kommutativen Ringen sind beide Begriffe identisch, man spricht daher dann nur von *Nullteilern*.

- Schiefkörper

  Sei $(R, +, \cdot)$ ein Ring. $(R, +, \cdot)$ heißt *Schiefkörper*, wenn $(R\backslash\{0\}, \cdot)$ eine Gruppe ist.

- Körper

  Sei $(R, +, \cdot)$ ein Ring. $(R, +, \cdot)$ heißt *Körper*, wenn $(R \backslash \{0\}, \cdot)$ eine abelsche Gruppe ist.

Anhang B

Quaternionen

Die Quaternionen sind hyperkomplexe Zahlen der Form

$$\boxed{\alpha = a + b\,\mathrm{i} + c\,\mathrm{j} + d\,\mathrm{k}}$$

mit $a, b, c, d \in \mathbb{R}$ und imaginären Einheiten i, j, k.

Wichtige Bezeichnungen und weitere Eigenschaften:

- Skalarteil
$$a$$
- Vektorteil
$$\vec{\alpha} := b\,\mathrm{i} + c\,\mathrm{j} + d\,\mathrm{k}$$
- Konjugiertes Quaternion
$$\bar{\alpha} := a - b\,\mathrm{i} - c\,\mathrm{j} - d\,\mathrm{k}$$
- Absoluter Betrag
$$|\alpha| := \sqrt{\alpha\bar{\alpha}}$$
- Inverses Quaternion
$$\alpha^{-1} := \frac{\bar{\alpha}}{|\alpha|^2}$$
- Addition
$$\alpha_1 + \alpha_2 := (a_1 + a_2) + (b_1 + b_2)\,\mathrm{i} + (c_1 + c_2)\,\mathrm{j} + (d_1 + d_2)\,\mathrm{k}$$
- Multiplikation
$$\alpha_1 \otimes \alpha_2 := a_1 b_1 - \langle \vec{\alpha_2}, \vec{\alpha_2} \rangle + a_2\vec{\alpha_1} + a_1\vec{\alpha_2} + \vec{\alpha_1} \wedge \vec{\alpha_2}$$

Anhang C

Ergänzendes Material zur Clifford–Algebra

In diesem Anhang ist ergänzendes Material zu Clifford–Algebren zusammengestellt, auf das im Text verwiesen worden ist.

- Multiplikationstabelle zu $\mathcal{C}_{3,0}$

$\otimes_{3,0}$	1	e_1	e_2	e_3	e_{12}	e_{13}	e_{23}	I
1	1	e_1	e_2	e_3	e_{12}	e_{13}	e_{23}	I
e_1	$-e_1$	1	$-e_{12}$	$-e_{13}$	e_2	e_3	$-\mathrm{I}$	e_{23}
e_2	$-e_2$	e_{12}	1	$-e_{23}$	$-e_1$	I	e_3	$-e_{13}$
e_3	$-e_3$	e_{13}	e_{23}	1	$-\mathrm{I}$	$-e_1$	$-e_2$	e_{12}
e_{12}	$-e_{12}$	$-e_2$	e_1	$-\mathrm{I}$	1	$-e_{23}$	e_{13}	e_3
e_{13}	$-e_{13}$	$-e_3$	I	e_1	e_{23}	1	e_{12}	$-e_2$
e_{23}	$-e_{23}$	$-\mathrm{I}$	$-e_3$	e_2	$-e_{13}$	$-e_{12}$	1	e_1
I	I	$-e_{23}$	e_{13}	$-e_{12}$	$-e_3$	e_2	$-e_1$	1

- Multiplikationstabelle zu $\mathcal{C}_{3,3}$

$\otimes_{3,3}$	1	e_1	e_2	e_3	e_{12}	e_{13}	e_{23}	I
1	1	e_1	e_2	e_3	e_{12}	$-e_{13}$	$-e_{23}$	$-\mathrm{I}$
e_1	e_1	1	$-e_{12}$	$-e_{13}$	e_2	e_3	$-\mathrm{I}$	$-e_{23}$
e_2	e_2	e_{12}	1	$-e_{23}$	$-e_1$	I	e_3	e_{13}
e_3	e_3	e_{13}	e_{23}	1	$-\mathrm{I}$	$-e_1$	$-e_2$	$-e_{12}$
e_{12}	e_{12}	e_2	$-e_1$	I	1	$-e_{23}$	e_{13}	e_3
e_{13}	e_{13}	e_3	$-\mathrm{I}$	$-e_1$	e_{23}	1	$-e_{12}$	$-e_2$
e_{23}	e_{23}	I	e_3	$-e_2$	$-e_{13}$	e_{12}	1	e_1
I	I	e_{23}	$-e_{13}$	e_{12}	e_3	$-e_2$	e_1	1

- Tabelle aller Clifford–Algebren bis zur Dimension 16 [Porteous95]

$\mathcal{C}_{p,q}$	0	1	2	3	4
0	$\mathbb{R}$	$\mathbb{C}$	$\mathbb{H}$	${}^2\mathcal{H}$	$\mathcal{H}(2)$
1	${}^2\mathcal{R}$	$\mathcal{R}(2)$	$\mathcal{C}(2)$	$\mathcal{H}(2)$	${}^2\mathcal{H}(2)$
2	$\mathcal{R}(2)$	${}^2\mathcal{R}(2)$	$\mathcal{R}(4)$	$\mathcal{C}(4)$	$\mathcal{H}(4)$
3	$\mathcal{C}(2)$	$\mathcal{R}(4)$	${}^2\mathcal{R}(4)$	$\mathcal{R}(8)$	$\mathcal{C}(8)$
4	$\mathcal{H}(2)$	$\mathcal{C}(4)$	$\mathcal{R}(8)$	${}^2\mathcal{R}(8)$	$\mathcal{R}(16)$

Literaturverzeichnis

[Arena96a] Arena P, Fortuna L, Muscato G, Xibilia MG: *Neural networks in multidimensional domains.* Journal of System Engineering. 6(1), 1996, 1–11.

[Arena96b] Arena P, Fortuna L, Muscato G, Xibilia MG: *Multilayer Perceptrons to Approximate Quaternion Valued Functios.* Neural Networks. 9(6), 1996, 1–8.

[Bayro96] Bayro–Corrochano E, Buchholz S, Sommer G: *Selforganizing Clifford neural network.* ICNN '96, Washington. IEEE Comp. Soc. Press, Washington, 1996, 120–125.

[Bayro97] Bayro–Corrochano E, Buchholz S: *Geometric neural networks.* In: Proc. AFPAC'97, Kiel. Sommer G, Koenderink JJ (Eds.), LNCS Vol. 1315, Springer Verlag, Heidelberg, 1997.

[Brackx82] Brackx F, Delanghe R, Sommen F: *Clifford Analysis.* Pitman Advanced Publishing Program, London, 1982.

[Buchholz96] Buchholz S: *Neuronale Netze im Rahmen der Clifford–Algebra.* Studienarbeit, Institut für Informatik der Christian–Albrechts–Universität Kiel, 1996.

[Cybenko89] Cybenko G: *Approximation by superposition of a sigmoidal function.* Mathematics of Control, Signals and Systems. 2(4), 1989, 303–314.

[Edelman92] Edelman S, Poggio T: *Bringing the Grandmother back into the Picture: A Memory–Based View of Object Recognition.* International Journal of Pattern Recognition and Artifical Intelligence. 6(1), 1992, 37–61.

[Georgiou92] Georgiou G, Koutsougeras C: *Complex domain backpropagation.* IEEE Transactions on Circuits and Systems–II: Analog and Digital Signal Processing. 39(5), 1992, 330–334.

[Hesteneś84] Hestenes D, Sobczyk G: *Clifford Algebra to Geometric Calculus: A unified language for mathematics and physics.* D. Reidel, Dordrecht, 1984.

[Pao89] Pao YH: *Adaptive Pattern Recognition and Neural Networks.* Addison–Wesley, Redwood City, 1989.

[Pearson94] Pearson JK: *Clifford Networks*. Ph.D. Thesis, University of Kent, 1994.

[Porteous95] Porteous IR: *Clifford Algebras and the Classical Groups*. Cambridge University Press, Cambridge, 1995.

Erklärung

Hiermit versichere ich, daß ich die vorliegende Arbeit selbständig verfaßt und ausschließlich die angegebenen Hilfsmittel und Quellen verwendet habe.