CLIFFORD NETWORKS

A THESIS SUBMITTED TO

THE UNIVERSITY OF KENT AT CANTERBURY

IN THE SUBJECT OF ELECTRONIC ENGINEERING

FOR THE DEGREE

OF DOCTOR OF PHILOSOPHY.

By

Justin K. Pearson

October 1995

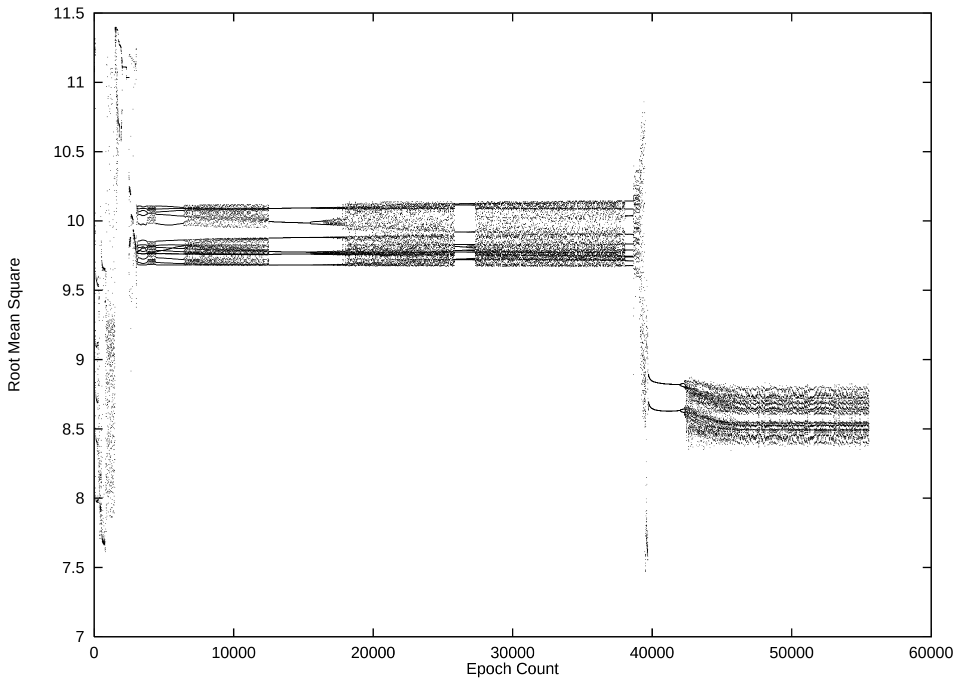
Abstract

This thesis is concerned with an extension of feed-forward networks, Clifford networks, which use multi-dimensional values from Clifford algebras as weight and activation values. Clifford algebras are a natural extension of both the complex and quaternion algebras to many dimensions. An extended back-propagation algorithm is derived and results are proved which show that Clifford networks can approximate any continuous function. Clifford networks are then applied to a real world problem, that of Q^2PSK demodulation which is a four dimensional extension of QPSK. Finally an extension of the real valued Perceptron to the Clifford-valued Perceptron is presented and a generalized convergence theorem is presented together with a discussion of the pattern recognition power of these new networks.

Acknowledgments

In writing this thesis, many people have helped. First I would like to thank my supervisor Dr. Dave Bisset and Dr. M. Gell for originally arranging a BT CASE award without which I would not have been able carry out this work, Gail Bye for coping with my late monthly reports, and ploughing through pages of often obscure mathematics just to work out what I am talking about. My interest in Clifford Algebras would not have been started, if it was not for the enthusiasm of Prof Roy Chisholm in the Mathematics Department at Kent for the subject; also I have had many useful conversations with Prof Alan Common on Clifford algebras. I would also like to thank the Staff and Students at the Academy of Science in Prague especially Dr Vera Kůrková and Dr Katka Hlavockova who on visits to Prague were both interesting and useful to talk to about my work.

I could not have just done research in Neural Networks while writing this thesis and at Kent there have been many academic and non-academic distractions. The Theoretical Computer Science group run by Dr Simon Thompson has provided numerous excuses to give talks on subjects I know little about, and consequently I have learnt much about other aspects of computing. The University's Philosophy society has been a constant source of enjoyment, in both philosophical discussion, papers and less philosophical discussion at the Wednesday night meals with visiting speakers after papers. Many people have had the pleasure of Adrian Weddell's wonderfully extravagant meals, which always seem to start in conception as only about 5 courses and always end up with about 8 or 9. I also would like to thank the programmers who wrote Purify[1], without which my back-prop simulator would still be outputting results like this:



Finally, but not in the means least, I would like to thank Elinor Williams, for being the loveliest person in the world and not jumping in the Vltava to escape from me.

[1]Pure Software Inc.

Contents

List of Tables

List of Figures

Key of Mathematical symbols used in the Text

$\mathcal{R}$ The Real numbers

$\mathcal{C}$ The Complex numbers

$\mathcal{H}$ The Quaternions

$\mathcal{N}$ The natural numbers including zero.

$\mathcal{R}_{p,q}$ A Clifford algebra with signature p, q.

$\mathcal{R}(n)$ The n by n real square matrices, with analogous notation for complex and Quaternions square matrices.

e_A An arbitrary basis element of some Clifford algebra. In general summations such as:

$$\sum_A x_A e_A$$

Sum over elements of the basis set of the algebra in question.

$\Box$ Denotes the end of a proof.

$\mathcal{P}$ denotes the power set operation

$X \setminus Y$ denotes the set of elements which are in X but are not in Y.

Chapter 1

Introduction and Background

1.1 Introduction

This thesis is mainly concerned with extensions to multi-layer feed-forward networks and an extension of Rosenblatt's Perceptron.

The thesis takes as its starting point the work of George M.Georgiou [Georgiou, 1993; Georgiou and Koutsougeras, 1992] where weight and activation values in multi-layer networks are replaced by complex numbers and hence allow the networks to represent two dimensional signals in a more compact way. Recently there has been interest from other authors on the use of complex networks applied to certain problems which map naturally to the complex domain [Hirose, 1992b; Hirose, 1992a; Hirose, 1993] and some advantages over traditional networks have been seen, but as yet it is to early for conclusive results to be obtained.

The present work extends the complex numbers to multi-dimensional Clifford Numbers [Crumeyrolle, 1990; Gilbert and Murry, 1991; Chisholm and Common, 1986; Brackx *et al.*, 1982; Porteous, 1981; Blaine Lawson Jr. and Michelsohn, 1989] to derive a generalized back propagation algorithm [Pearson and Bisset, 1992; Pearson and Bisset, 1994] (Chapter 3), proves approximation results analogous to [Hornick *et al.*, 1989] (Chapter 4). The multi-layer networks are applied to signal processing applications (Chapter 5) and finally a multi-dimensional extension of Rosenblatt's Perceptron is presented (Chapter 6).

The author's justification for going to higher dimensional algebras is twofold. The first is a mathematician's answer: it is natural to extend the complex numbers to Clifford algebras so why not extend multi-layer feed-forward networks to Clifford valued multi-layer networks and see what happens. Second is an engineering answer, to see if extending neural networks to multi-dimensional algebras does give a more compact representation of certain signal spaces and hence a more powerful model.

The rest of this introductory chapter surveys the material from neural networks required for this thesis (introductory material on Clifford algebras is in Chapter 2). A word of warning about the author's preoccupations about neural networks: there are many classes of networks not discussed here, such as Hopfield networks, Boolean networks, Hebbian learning networks etc. The author, although aware of other types of networks, is only concerned with feed-forward networks and perceptrons, hence any historical omission (and bias) in this introduction can be explained by this fact.

1.2 From Neurons to Networks

Neural network research was originally biologically inspired. Part of the driving force was to find simple models of real neurons [McCulloch and Pitts, 1943]. An artificial neuron or Perceptron (Figure 1) consists of a number of inputs $x_1 \ldots x_n$ weighted by $w_1 \ldots w_n$ and a single threshold value. The weighted sum of the inputs is added to

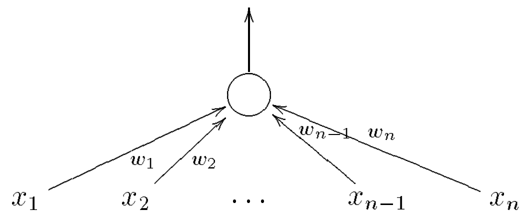


Figure 1: A Single Perceptron

the threshold value:

$$net = \sum_i w_i x_i + \theta \tag{1}$$

and passed through an activation function $f(net)$ to produce an output. In the first approximation neurons can be seen to have an all or nothing response with respect to the net input, and thus the following activation can be used:

$$f(net) = \begin{cases} 1 & \text{if } net > 0 \\ 0 & \text{if } net \leq 0 \end{cases} \tag{2}$$

It was discovered that single Perceptrons could perform simple pattern recognition tasks, and it was later discovered [Rosenblatt, 1962], that there existed a learning algorithm, such that anything that a Perceptron could do was learnable from an initial random weight set[1]. This result was initially thought to be quite powerful, and it seemed that single Perceptrons could perform any number of tasks.

Unfortunately it was later shown by Minsky and Papert in [Minsky and Papert, 1969] that there are non-trivial pattern recognition tasks which single Perceptrons cannot do. These tasks include the so called "parity problem", that is, for an arbitrary finite set of points decide if that set is even or odd in cardinality, or the connectivity problem, which is to decide if a figure is connected or not.

[1]An extension of this algorithm is explained in Chapter 6

Minsky and Papert introduced powerful mathematical tools to deduce what problems are solvable by Perceptrons. Perhaps the most important tool is the Group Invariance Theorem ([Minsky and Papert, 1969] page 48) which gives conditions on the values of the weight set with respect to the group of transforms under which the patterns recognised remain invariant. Minsky and Papert's book is seen as a rather negative book by many people, and it was seen be some as a call to shift the emphasis from biologically inspired artificial intelligence research to more symbolic A.I. But the book has many positive aspects many of which are still not noticed, such as the need for mathematical tools in neural networks (although the situation is better now than it was 5 years ago), and the group theoretical view of the pattern set, rather than concentrating on the supposed vector space properties of the patterns.[2] In the 1988 edition of the book Minsky and Papert look back and assess the significance of the book in he light of the 80's explosion of interest in neural networks. They still see the need for a more detailed mathematical analysis to be carried rather than a naive experimental optimism, a view that inspired the author when starting his research (but unfortunately this goal was not always achieved). In particular the problem of scaling in neural networks, that is the relationship between problem size and the representation of the problem in a neural network, Minskey and Papert saw still needed to be addressed. Many of the criticisms from the book still hold seven years after the second edition, but it should be seen and is a very positive contribution to the mathematical study of neural networks.

Since it is known that a single Perceptron can not perform all pattern recognition tasks ([Minsky and Papert, 1969]), it is natural to ask if multi-layer networks can do better? A multi-layer network consists of an input layer, with one or more hidden

[2]Perhaps the most important lesson that Minsky and Papert's book can teach us, can be summed up in the slogan: *"Pattern sets do not form vector spaces"*.

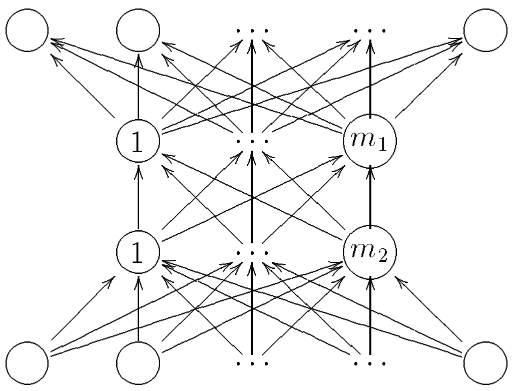


Figure 2: A Feedforward Network

layers and an output layer; each layer contains one or more Perceptron like processing units, for example Figure 2. The outputs of the previous layer are fed forward to the inputs of the next layer.

In [Werbos, 1974; Parker, 1985; Rumelhart and McClelland, 1986] an algorithm often called Back Error Propagation (BEP) is derived which trains feed-forward networks to classify certain pattern classes, but this algorithm only works with activation functions which are continuous in the first derivative. One of the most popular activation functions is the so called sigmoid function (see Figure 3):

$$f(net) = \frac{1}{1+e^{-net}} \tag{3}$$

which tends to $+1$ as net tends to $+\infty$ and 0 as x tends to $-\infty$[3].

Essentially the BEP algorithm calculates the partial derivatives of an error measure with respect to the individual weight values and uses these equations to minimize the error. Because the networks are multi-layered, the chain rule has to be used and hence when the error is calculated for the output layer, this is fed back to the previous layers. This algorithm is explained in more detail in Chapter 3.

[3]Other popular activation functions include $\tanh(x)$; also $x/(1+|x|)$ can be quite useful.

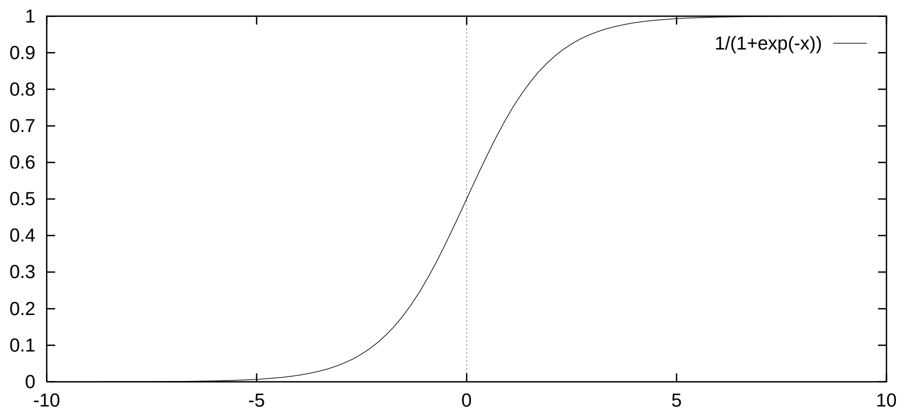


Figure 3: The sigmoid function

As was stated earlier the starting point of this thesis is to replace the activation and weight values of feed-forward networks by complex numbers and then by Clifford algebras, before going on to examine what classes of networks result.

Chapter 2

Clifford algebras: An Introduction.

This chapter is provided to serve as an introduction to Clifford algebras. The reader with knowledge of Clifford algebras can safely skip this chapter. Where possible the notation has been kept consistent with [Porteous, 1981]. Throughout the thesis $\mathcal{R}$ is used to denote the real numbers, $\mathcal{C}$ is used to denote the complex numbers and $\mathcal{H}$ is used to denote the quaternions.

Clifford algebras are geometric in nature and are constructed from real (only real vector spaces are considered in this thesis) or complex vector spaces with a quadratic form. Rather than constructing Clifford algebras from quadratic forms a more elementary approach is used first, where the algebra is given an explicit representation. This representation will be used throughout the thesis. Section 2.2 shows how Clifford algebras arise naturally from quadratic form theory.

2.1 A Direct Construction

A $p+q$ dimensional real vector space will be denoted as $\mathcal{R}^{p+q}$ (the reason for the notation $p+q$ will become clear later on). A Clifford algebra $\mathcal{R}_{p,q}$ is two things, a 2^{p+q} vector space constructed from $\mathcal{R}^{p+q}$ and a set of algebraic rules defining multiplication and addition of vectors (when constructing Clifford algebras from quadratic form theory, these rules come out naturally).

A $\mathcal{R}^{p+q}$ will have a basis of the form:

$$e_1, e_2, e_3, \ldots e_{p+q}$$

From this construct a $2^{n=p+q}$ dimensional vector space with basis elements:

$$\{e_A = e_{(h_1 \ldots h_r)} | A = (h_1, \ldots, h_r) \in \mathcal{P}(\mathcal{N}), 1 \leq h_1 < \ldots < h_r \leq n\}.$$

For example the vector space over $\mathcal{R}^{1,1}$ would have the basis,

$$e_\emptyset,\ e_{(1)},\ e_{(2)}\ e_{(1,2)}$$

For notational convenience when no confusion can arise, $e_{(h_1,\ldots h_r)}$ will be denoted as $e_{h_1 h_2 \ldots h_r}$ and $e_\emptyset = e_0$.

An element of the Clifford algebra can be written as a formal sum,

$$x = \sum_A x_A e_A \tag{4}$$

with each $x_A \in \mathcal{R}$. In what follows a summation with a capital letter near the begining of the alphabet denotes a sum over the basis elements of a Clifford algebra.

Thus an element in $\mathcal{R}_{1,1}$ can be written as,

$$x = x_0 e_\emptyset + x_1 e_1 + x_2 e_2 + x_{12} e_{12} \tag{5}$$

It is useful to drop the $e_\emptyset$ when writing out formal sums, because $e_\emptyset$ acts as the unit of the algebra, hence (5) would be written as:

$$x = x_0 + x_1 e_1 + x_2 e_2 + x_{12} e_{12} \tag{6}$$

Addition of two elements of the algebra is defined as for vectors:

$$x + y = \sum_A (x_A + y_A) e_A \tag{7}$$

Multiplication is slightly more complicated. It is done formally element by element as in expanding brackets subject to the following algebraic rules,

$$e_i^2 = 1 \quad , \quad i = 1, \ldots, p \tag{8}$$

$$e_i^2 = -1 \quad , \quad i = p+1, \ldots, p+q \tag{9}$$

$$e_i e_j = -e_j e_i \quad , \quad i \neq j \tag{10}$$

with $1 \leq h_1 < \ldots h_r \leq n$, $e_{h_1} \cdot e_{h_2} \cdots e_{h_r} = e_{h_1 \ldots h_r}$. This can be expressed more compactly in the following way,

$$e_A e_B = \kappa_{A,B} e_{A \Delta B}, \tag{11}$$

where:

$$\kappa_{A,B} = (-1)^{\#((A \cap B) \setminus P)} (-1)^{p(A,B)} \tag{12}$$

P stands for the set $1, \ldots p$, and $\#X$ represents the number of elements in the set X,

$$p(A,B) = \sum_{j \in B} p'(A,j), \quad p'(A,j) = \#\{i \in A | i > j\}, \tag{13}$$

and the sets A, B and $A \Delta B$(the set difference of A and B) are ordered in the prescribed way.

Thus for example if in $\mathcal{R}_{1,1}$

$$
\begin{aligned}
x &= 3 + 4e_1 + e_2 \\
y &= e_2 + 2e_{12} \\
xy &= (3 + 4e_1 + e_2)(e_2 + 2e_{12}) \\
&= 3e_2 + 6e_{12} + 4e_1e_2 + 8e_1e_{12} + e_2^2 + 2e_2e_{12}
\end{aligned}
$$

By using the reduction rules $e_1e_{12} = e_1e_1e_2 = e_1^2e_2 = e_2$ and $e_2e_{12} = -e_2e_{21} = -e_2^2e_1 = e_1$. So

$$xy = -1 + 2e_1 + 11e_2 + 10e_{12} \tag{14}$$

In general a Clifford algebra is associative but non-commutative.

2.2 From Quadratic Forms to Clifford Algebras

This section is intended to show that Clifford algebras arise naturally as mathematical structures and in particular how the rules (8 - 10) arise. The reader with no interest in the mathematical origins of Clifford algebras can safely skip this section.

An orthogonal space is a real linear vector space X together with a symmetric inner product $(\cdot|\cdot)$ from X^2 to $\mathcal{R}$ which is linear in each component. That is, the following equations are satisfied:

$$(x, y) = (y, x) \qquad \text{For all } x \text{ and } y \text{ in } X \tag{15}$$

$$(\alpha x + \beta y, z) = \alpha(x, z) + \beta(y, z) \quad \text{For all } \alpha, \beta \in \mathcal{R} \text{ and all } x, y, z \in X \tag{16}$$

$$(x, \alpha y + \beta z) = \alpha(x, y) + \beta(x, z) \quad \text{For all } \alpha, \beta \in \mathcal{R} \text{ and all } x, y, z \in X \tag{17}$$

These equations abstract the standard inner product on eucledian vector spaces. A quadratic form Q on X, can be constructed from an inner product by defining:

$$Q(x) = (x|x) \tag{18}$$

The inner product is recoverable from the quadratic form by the following equation:

$$(x|y) = \frac{Q(x) + Q(y) - Q(x-y)}{2} \tag{19}$$

Thus the quadratic form uniquely determines the inner product and vice versa.

A Clifford algebra for a vector space X with respect to a quadratic form Q is a real associative algebra A, such that for each element x of X, the following is true:

$$x^2 = -Q(x) \tag{20}$$

(where x^2 is carried out in the algebra A).

There is a theorem in quadratic form theory due to Sylvester (for a proof see [Lam, 1973] or [Bromwich, 1906]) that given a real vector space and a quadratic form, it is possible by change of basis to represent the quadratic form as:

$$Q(x) = -x_1^2 - x_2^2 - \ldots x_p^2 + x_{p+1}^2 + \ldots x_{p+q}^2 \tag{21}$$

where p, q is called the signature of the Quadratic form, and $p + q$ is the dimension of X.

Thus to get the equations (8 - 10), we apply the equation (20) to the basis elements of X:

$$e_i^2 = -Q(e_i) = +1 \qquad \text{For } 0 < i \leq q \tag{22}$$

$$e_i^2 = -Q(e_i) = -1 \quad \text{For } q < i < p + q \tag{23}$$

$$\tag{24}$$

The anti-commutative law (equation (10)) can be derived from the following proposition.

Proposition 1 *For all $x, y \in X$ then in A,*

$$(x|y) = \frac{-(xy + yx)}{2} \tag{25}$$

Proof: From the formula (19):

$$2(x|y) = Q(x) + Q(y) + Q(x-y) = -x^2 - y^2 + (x-y)^2 = -xy - yx \tag{26}$$

$\square$

In particular for distinct basis elements e_i and e_j we have:

$(e_i\|e_j) = 0$	Because e_i and e_j are orthogonal
$-2(e_i\|e_j) = e_ie_j + e_je_i$	From the equation (25)

which implies that:

$$e_ie_j + e_je_i = 0 \tag{27}$$

2.3 Some familiar Clifford algebras

This section is to show that some Clifford algebras are certain familiar algebras in disguise. The real numbers $\mathcal{R}$ are trivially isomorphic to the Clifford algebras $\mathcal{R}_{0,0}$.

2.3.1 Complex numbers

The complex numbers are isomorphic to the Clifford algebra $\mathcal{R}_{0,1}$. To see this an element of $\mathcal{R}_{0,1}$ is written as the formal sum,

$$x = x_0 + x_1e_1 \tag{28}$$

with the multiplication rule $e_1^2 = -1$. This is simply the complex numbers with e_1 representing the imaginary unit i.

2.3.2 The Quaternions.

The quaternion algebra is generated from the basis elements $1, i, j, k$ with the relations,

$$\begin{aligned} i^2 = j^2 = k^2 &= -1, \\ ij = k &= -ji, \\ jk - i &= -kj, \\ ki = j &= -ik, \end{aligned}$$

The quaternions can be seen as isomorphic to $\mathcal{R}_{0,2}$ with the following isomorphisms,

$$\begin{aligned} e_0 &\longleftrightarrow 1 \\ e_1 &\longleftrightarrow i \\ e_2 &\longleftrightarrow j \\ e_{12} &\longleftrightarrow k \end{aligned}$$

Some perhaps less familiar examples include the Dirac algebra $\mathcal{R}_{4,1}$ and the Pauli algebra $\mathcal{R}_{3,3}$.

2.3.3 The algebra ${}^2\mathcal{R}$

The algebra ${}^2\mathcal{R}$ (often denoted as $\mathcal{R} \oplus \mathcal{R}$) is defined over ordered pairs (x_1, x_2) with addition and multiplication defined as:

$$\begin{aligned} (x_1, x_2) + (y_1, y_2) &= (x_1 + y_1, x_2 + y_2) \\ (x_1, x_2) * (y_1, y_2) &= (x_1 * y_1, x_2 * y_2) \end{aligned}$$

This algebra is isomorphic to the algebra $\mathcal{R}_{1,0}$ under the isomorphism ϕ given by:

$$\phi(\alpha + \beta e_1) \mapsto (\alpha + \beta, \alpha - \beta)$$

with

$$\phi^{-1}(a, b) \mapsto \frac{a+b}{2} + \frac{a-b}{2} e_1$$

As with all Clifford algebra isomorphisms **both** the following equations have to be checked, for all x, y:

$$\phi(x + y) = \phi(x) + \phi(y) \tag{29}$$

$$\phi(xy) = \phi(x)\phi(y) \tag{30}$$

The first is easy to check:

$$\phi(x_1 + x_2e_1 + y_1 + y_2e_1) = (x_1 + y_1 + x_2 + y_2, x_1 + y_1 - x_2 - y_2) =$$
$$\phi(x_1 + x_2e_1) + \phi(y_2 + y_2e_1)$$

The second equation is more complicated:

$$\phi((x_1 + x_2e_1)(y_1 + y_2e_1)) = \phi(x_1y_1 + x_1y_2e_1 + x_2y_1e_1 + x_2y_2)$$
$$= (x_1y_1 + x_2x_2 + x_1y_2 + x_2y_1, x_1y_1 + x_2y_2 - x_1y_2 - x_2y_1) = \phi(x_1 + x_2e_1) * \phi(y_2 + y_2e_1)$$

The algebra $\mathcal{R}_{1,0}$ is called the algebra of hyperbolic complex numbers and can be used in relativity calculations in physics.

2.3.4 2 by 2 matrices $R(2)$

$R_{1,1}$ is isomorphic to the set of 2 by 2 real valued matrices. This can seen by the following identification,

$$e_1 \longleftrightarrow \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad e_2 \longleftrightarrow \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$
$$e_{12} \longleftrightarrow \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad e_0 \longleftrightarrow \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

This is a basis for $\mathcal{R}(2)$ and defines the isomorphism of algebras. This can be generalized to show that $\mathcal{R}_{n,n} \cong \mathcal{R}(2^n) \cong \mathrm{End}(\mathcal{R}^n)$.

2.4 Some relationships between Clifford algebras.

The general idea of a Clifford network is to be able to process multi-dimensional signals, with fewer Clifford nodes instead of many real valued nodes. For some problems the choice of which Clifford algebra to use might be obvious, but in many cases such an obvious choice might not present itself. The engineer then has to experiment with different Clifford algebras. This section is a short guide to the relationships between algebras. Knowing the relationships between each algebra saves the engineer from needless duplication.

Every Clifford algebra is either isomorphic to a matrix algebra of $\mathcal{R}$,$\mathcal{C}$, $\mathcal{H}$ or a direct product of such matrix algebras. Not all Clifford algebras are distinct, and there is the so called periodicity theorem which relates higher dimensional Clifford algebras to low dimensional algebras. This section is a short guide to the relationships between Clifford algebras.

All proofs in this section are omitted and can be found in Chapter 13 of [Porteous, 1981]. So far the following relationships have been demonstrated:

$$\begin{aligned}
\mathcal{R}_{0,0} &\cong \mathcal{R} \\
\mathcal{R}_{0,1} &\cong \mathbf{C} \\
\mathcal{R}_{0,2} &\cong \mathcal{H} \\
\mathcal{R}_{n,n} &\cong \mathcal{R}(2^n) \\
\mathcal{R}_{1,1} &\cong {}^2\mathcal{R}
\end{aligned}$$

In fact all Clifford algebras defined over the real numbers can be constructed in some

Table 1: A table of Clifford algebras up to dimension 256

$\mathcal{R}_{p,q}$	$p=0$	1	2	3	4	5	6	7	8
$q=0$	$\mathcal{R}$	$\mathcal{C}$	$\mathcal{H}$	$^2\mathcal{H}$	$\mathcal{H}(2)$	$\mathcal{C}(4)$	$\mathcal{R}(8)$	$^2\mathcal{R}(8)$	$\mathcal{R}(16)$
1	$^2\mathcal{R}$	$\mathcal{R}(2)$	$\mathcal{C}(2)$	$\mathcal{H}(2)$	$^2\mathcal{H}(2)$	$\mathcal{H}(4)$	$\mathcal{C}(8)$	$\mathcal{R}(16)$	$^2\mathcal{R}(16)$
2	$\mathcal{R}(2)$	$^2\mathcal{R}(2)$	$\mathcal{R}(4)$	$\mathcal{C}(4)$	$\mathcal{H}(4)$	$^2\mathcal{H}(4)$	$\mathcal{H}(8)$	$\mathcal{C}(16)$	$\mathcal{R}(32)$
3	$\mathcal{C}(2)$	$\mathcal{R}(4)$	$^2\mathcal{R}(4)$	$\mathcal{R}(8)$	$\mathcal{C}(8)$	$\mathcal{H}(8)$	$^2\mathcal{H}(8)$	$\mathcal{H}(16)$	$\mathcal{C}(32)$
4	$\mathcal{H}(2)$	$\mathcal{C}(4)$	$\mathcal{R}(8)$	$^2\mathcal{R}(8)$	$\mathcal{R}(16)$	$\mathcal{C}(16)$	$\mathcal{H}(16)$	$^2\mathcal{H}(16)$	$\mathcal{H}(32)$
5	$^2\mathcal{H}(2)$	$\mathcal{H}(4)$	$\mathcal{C}(8)$	$\mathcal{R}(16)$	$^2\mathcal{R}(16)$	$\mathcal{R}(32)$	$\mathcal{C}(32)$	$\mathcal{H}(32)$	$^2\mathcal{H}(32)$
6	$\mathcal{H}(4)$	$^2\mathcal{H}(4)$	$\mathcal{H}(8)$	$\mathcal{C}(16)$	$\mathcal{R}(32)$	$^2\mathcal{R}(32)$	$\mathcal{R}(64)$	$\mathcal{C}(64)$	$\mathcal{H}(64)$
7	$\mathcal{C}(8)$	$\mathcal{H}(8)$	$^2\mathcal{H}(8)$	$\mathcal{H}(16)$	$\mathcal{C}(32)$	$\mathcal{R}(64)$	$^2\mathcal{R}(64)$	$\mathcal{R}(128)$	$\mathcal{C}(128)$
8	$\mathcal{R}(16)$	$\mathcal{C}(16)$	$\mathcal{H}(16)$	$^2\mathcal{H}(16)$	$\mathcal{H}(32)$	$\mathcal{C}(64)$	$\mathcal{R}(128)$	$^2\mathcal{R}(128)$	$\mathcal{R}(256)$

way from $\mathcal{R}$,$\mathcal{C}$ or $\mathcal{H}$. Table 1 extends this information. The reader interested in the explicit construction of the table should again consult [Porteous, 1981] or [Blaine Lawson Jr. and Michelsohn, 1989].

The most useful fact (again for a proof see [Porteous, 1981]) is perhaps,

$$\mathcal{R}_{p+1,q} \cong \mathcal{R}_{q+1,p} \tag{31}$$

.

To complete the table to arbitrary algebras it is enough to know that,

$$\mathcal{R}_{p,q+8} \cong \mathcal{R}_{p,q} \otimes_{\mathcal{R}} \mathcal{R}(16) \cong \mathcal{R}_{p,q}(16). \tag{32}$$

where $\otimes_{\mathcal{R}}$ denotes the real tensor product of two algebras. This is the so called periodicity theorem a proof can be found in [Porteous, 1981] or [Blaine Lawson Jr. and Michelsohn, 1989]

2.5 Multiplication as a linear transform

If as a vector space the algebra $\mathcal{R}_{p,q}$ is identified as the vector space $\mathcal{R}^{2^{p+q}}$, then given an element z it is possible to define a transformation:

$$\tau_z(x) = zx \tag{33}$$

This is a linear transformation since,

$$\tau_z(\alpha x + y) = \alpha\tau_z(x) + \tau_z(y) \tag{34}$$

(with α real). For the quaternions, if we restrict z to be an element of unit length, these transforms are in fact rotations; this fact will be useful in Chapter 5.

Clifford algebras have rich applications in mathematical physics (see [Chisholm and Common, 1986]) and can be used to construct the so called spin-groups used in gauge theory (see [Crumeyrolle, 1990]). They also bring together many constructions in quadratic form theory and differential geometry (see [Blaine Lawson Jr. and Michelsohn, 1989]). The author's main motivation for using Clifford algebras is that they provide a natural extension to both the complex and quaternion numbers and it seems fruitful to investigate the relevant extensions of neural networks.

Chapter 3

The Back Propagation algorithm.

This chapter deals with back error propagation (BEP) for Multi-Layer Feed-Forward networks defined over Clifford algebras. The case for real networks has been presented in [Bryson and Ho, 1969; Werbos, 1974; Parker, 1985; Rumelhart and McClelland, 1986]. The complex case has been treated by many people, see for instance [Little *et al.*, 1990; Henseler and Braspenning, 1990; Leung and Haykin, 1991; Benvenuto and Piazza, 1992]. The account here of Clifford back propagation is a generalization of Georgiou and Koutsougeras's work [Georgiou and Koutsougeras, 1992]. The Clifford case was first presented in [Pearson and Bisset, 1992; Pearson and Bisset, 1994]. Specialising this to the complex domain yields the algorithm desribed in [Georgiou and Koutsougeras, 1992]. Because error metrics are dealt with differently from most accounts, the case for a single real valued neuron is presented; extensions to multi-layer and Clifford valued networks are then more easy to perform.

3.1 Feed-forward Networks

This chapter, in fact most of this thesis, is concerned with feed-forward networks (often called Multi-Layer-Perceptrons or MLPs). A feed-forward network is built up from a number of layers each consisting of single neurons. There is no restriction on the activation functions of the neurons, and each neuron could have a different activation function[1], but in the networks considered in this chapter, all neurons will have the same activation function.

A single neuron (see figure 1) with, n inputs, a weight set $\omega_1, \ldots \omega_n$, with threshold value θ and an activation function can be seen to be computing the function:

$$\tau(\mathbf{x}) = f(\sum_{i=1}^{n} \omega_i x_i + \theta) \tag{35}$$

with $\mathbf{x} = (x_i)_{i=1}^{n}$. This function will in general be referred to as the transfer function of a neuron. A multi-layer network is made of a number of layers of neurons. Each layer of k neurons each with n inputs[2] can be seen as computing a function from $\mathcal{R}^n$ to $\mathcal{R}^k$, by taking the vector of transfer equations of each neuron. Then multi-layer transfer functions can be calculated by composing the transfer functions for each layer. In general the value ω_{ij} will refer to the value of the weight between a node labeled i and a node labeled j[3]. It does not matter how the neurons are labeled as long as each has its own unique label.

[1]This is not strictly true, as, for the Back Error Propagation algorithm to make sense, the activation functions must be at least continuous in the first derivative, hence networks composed of neurons with step functions as activation functions giving an all or nothing response, cannot be trained by Back Error Propagation.

[2]If each neuron has a different number of inputs, then define n to be the number of inputs for the neuron with the largest number of inputs and add extra 'imaginary' inputs to the other neurons permanently weighted at zero

[3]Most people denote this by w_{ji}, but since this thesis does not make any direct use of matrix mathematics, this notation is more convenient.

3.2 Choice of Error Metric

Generally, a feed-forward network is trained on finite pattern set of size p,$\{X_i\}_{i=1}^p$ and is expected to produce a certain output Y_i for each input pattern. In [Rumelhart and McClelland, 1986], the following error measure is defined, assuming the network computes the function Ψ_ω (ω the weight set), the error E is given as:

$$E = \frac{1}{2}\sum_{i=0}^{p} |\Psi_\omega(X_i) - Y_i|^2 \tag{36}$$

where $|\cdot|$ is the Euclidean metric. The metric used here is defined as:

$$E = \frac{1}{2}\int_{x\in X} |\Psi_\omega(x) - \Phi(x)|^2 \tag{37}$$

where Φ is the target function the network is required to learn. (37) can be seen as a generalization of the metric (36) because the metric (37) can deal with continuous functions as well as point data sets. A point data set can be seen as a function and hence can be used in equation (37) by taking the set of patterns $(X_i) \subset \mathcal{P}(\mathcal{R}^n)$ and using suitable translated and multiplied 'bump' functions, such as Gaussians, to model the data set. Thus for example in one dimension, given the pattern set $\{-0.8, -0.2, 0.6\}$ and the expected output set $\{1, 2, 2\}$ the following function (see Figure 4):

$$\Phi(x) = \sigma(x, -0.8) + 2 * \sigma(x, -0.2) + 2 * \sigma(x, 0.6) \tag{38}$$

with

$$\sigma(x, c) = e^{-(x-c)^2/0.01} \tag{39}$$

could be used in equation (37) to model the pattern set. Unfortunately bump functions are needed to represent point data sets in equation (37) otherwise it will not be defined.. The metric (37) brings the treatment of back-propagation inline with the approximation theorems treated in the next chapter. Further it emphasises the

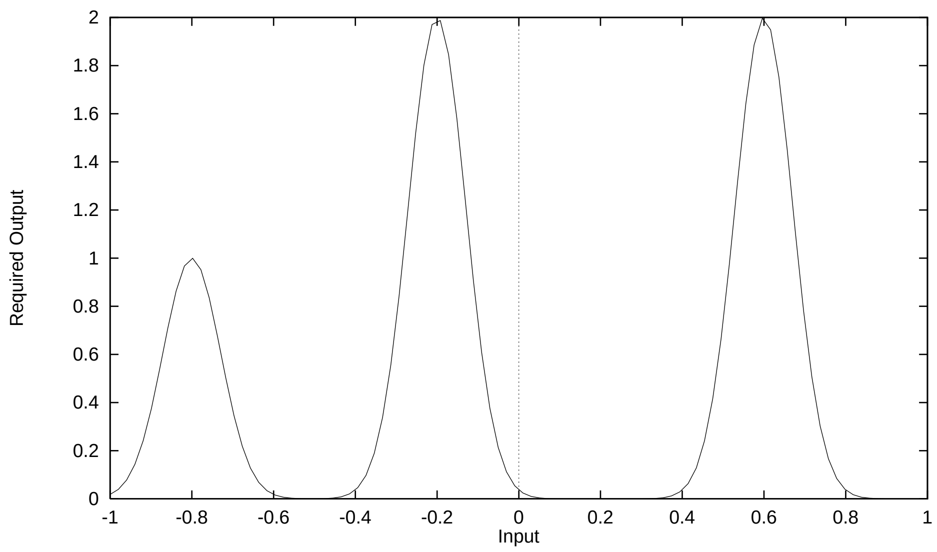


Figure 4: Modeling a point data set by translated bump functions

the minimisation of the error should take into account the whole output of the network, considerations such as this togther with thinking about how points could be interpolated in a problem independent way leads us to the theory of statistical model selection for neural networks, a subject which is not treated in this thesis.

3.3 Gradient Descent

Once the error metric E has been defined, the problem is then, for a network, to find the weight set that yields the minimum value of E. The most popular optimization technique used in the neural network community is gradient descent. Gradient descent can either be seen as using the first term of the Taylor expansion of E in the variables ω_i to approximate a solution to the equation $E = 0$, or as a dynamic system where:

$$\frac{\partial E}{\partial t} \tag{40}$$

is required to be negative so that E will reduce with time (the variable t), so since:

$$\frac{\partial E}{\partial t} = \sum_i \frac{\partial E}{\partial \omega_i}\frac{\partial \omega_i}{\partial t} \tag{41}$$

setting:

$$\frac{\partial \omega_i}{\partial t} = -\frac{\partial E}{\partial \omega_i} \tag{42}$$

will result in E decreasing in time. This system will obviously need some sort of approximation on a digital system, and setting:

$$\omega_i^{t+\Delta t} = \omega_i^t - \eta \frac{\partial E}{\partial \omega_i^t} \tag{43}$$

(where ω_i^t represents the value of ω_i at time t) for sufficiently small values of η will usually result in a stable system where E converges to a minimal value.

3.4 The Single Real Valued Neuron.

Consider a single real valued neuron with n-inputs and one output. The transfer function $\Psi : \mathcal{R}^n \to \mathcal{R}$ is defined by:

$$\Psi = f(\mathbf{x} \cdot \omega) \tag{44}$$

with $\mathbf{x}$ and ω vectors in $\mathcal{R}^n$ being the input and weight vectors respectively.

It is required to calculate the partial derivatives:

$$\frac{\partial E}{\partial \omega_i} \tag{45}$$

where ω_i represents the i'th part of the vector ω.

Then,

$$\frac{\partial E}{\partial \omega_i} = \frac{1}{2}\int_{\mathbf{x} \in X} \frac{\partial}{\partial \omega_i} |\Psi - \Phi|^2 \tag{46}$$

Define,

$$net = \mathbf{x} \cdot \omega = \sum_i x_i w_i \tag{47}$$

and,

$$\lambda = |\Psi - \Phi|^2 \tag{48}$$

Thus,

$$\frac{\partial E}{\partial \omega_i} = \int_{\mathbf{x} \in X} \frac{\partial \lambda}{\partial \omega_i}$$

Then using the chain rule,

$$\frac{\partial \lambda}{\partial w_i} = \frac{\partial \lambda}{\partial net} \frac{\partial net}{\partial w_i} \tag{49}$$

Since $net = \sum x_i \omega_i$

$$\frac{\partial net}{\partial \omega_i} = x_i \tag{50}$$

Then again using the chain rule,

$$\frac{\partial \lambda}{\partial net} = \frac{\partial \lambda}{\partial \Psi} \frac{\partial \Psi}{\partial net} \tag{51}$$

$$\frac{\partial \Psi}{\partial net} = f'(net), \tag{52}$$

$$\frac{\partial \lambda}{\partial \Psi} = \frac{\partial}{\partial \Psi} |\Psi - \Phi|^2 \tag{53}$$

Rewriting $|\Psi - \Phi|^2$ as $(\Psi - \Phi)^2$,

$$\frac{\partial \lambda}{\partial \Psi} = (\Psi - \Phi) \tag{54}$$

So bringing all this together,

$$\frac{\partial E}{\partial \omega_i} = \int_{x \in X} ((\Psi - \Phi) f'(net) x_i) \tag{55}$$

Neurons with n input values and a single threshold value calculate the function:

$$f(\omega \cdot \mathbf{x} + \theta) \tag{56}$$

and can be seen as neurons with $n+1$ inputs with weight vector $(\omega_1, \omega_2, \ldots \omega_n, \theta)$ with the extra input clamped to 1.

3.5 BEP for a multi-layer feed-forward network.

Consider a multi-layer network with a transfer function,

$$\Psi : \mathcal{R}^n \to \mathcal{R}^m$$

Each node will be assigned a unique number j and ω_{ij} represents the weight on the connection from node i to node j. Again it is required to find,

$$\frac{\partial E}{\partial \omega_{ij}} = \frac{1}{2} \int_{\mathbf{x} \in X} \frac{\partial}{\partial \omega_{ij}} |\Psi - \Phi|^2$$

Define $\lambda = |\Psi - \Phi|^2$ as before.

The first steps are the same as for the previous case,

$$\frac{\partial \lambda}{\partial \omega_{ij}} = \frac{\partial \lambda}{\partial net_j} \frac{\partial net_j}{\partial \omega_{ij}} \tag{57}$$

$$\frac{\partial net_j}{\partial \omega_{ij}} = o_i \tag{58}$$

where o_i represents the output of the i'th neuron.

$$\frac{\partial \lambda}{\partial net_j} = \frac{\partial \lambda}{\partial o_j} \frac{\partial o_j}{\partial net_j} \tag{59}$$

again,

$$\frac{\partial o_j}{\partial net_j} = f'(net_j) \tag{60}$$

If o_j is an output neuron then,

$$\frac{\partial \lambda}{\partial o_j} = (\Psi_j - \Phi_j) \tag{61}$$

where Ψ_j represents the j'th component of Ψ.

When the unit is hidden a further application of the chain rule is required,

$$\begin{aligned}\frac{\partial \lambda}{\partial o_j} &= \sum_k \frac{\partial \lambda}{\partial net_k}\frac{\partial net_k}{\partial o_j} \\ &= \sum_k \frac{\partial \lambda}{\partial net_k}\omega_{jk}\end{aligned} \tag{62}$$

where k is summing over all the nodes in the next layer in the feed-forward network.

So define for hidden units j,

$$\delta_j = f'(net_j)\sum_k \delta_k \omega_{jk} \tag{63}$$

and for output units j,

$$\delta_j = (\Psi_j - \Phi_j)f'(net_j) \tag{64}$$

Thus,

$$\frac{\partial E}{\partial \omega_{ij}} = \int_{x\in X} \delta_j o_i \tag{65}$$

3.6 Clifford Back error Propagation

Some care has to be taken when choosing a norm for a Clifford algebra. In what follows the norm $|\cdot|$ will be used, where,

$$|x| = \left(\sum_A [x]_A^2\right)^{\frac{1}{2}} \tag{66}$$

where $[x]_A$ represents the A'th part of the Clifford number x.

A feed-forward Clifford network with n inputs and m outputs will have a transfer function,

$$\Psi : (\mathcal{R}_{p,q})^n \rightarrow (\mathcal{R}_{p,q})^m \tag{67}$$

Where $(\mathcal{R}_{p,q})^n$ is the n-dimensional left module over the Clifford algebra $\mathcal{R}_{p,q}$[4].

Again some sort of error metric has to be defined. The basic form is the same,

$$E = \frac{1}{2} \int_{\mathbf{x} \in X} \|\Psi - \Phi\|^2 \tag{68}$$

where X is some compact subset of the Clifford module $(\mathcal{R}_{p,q})^n$ with the product topology derived from the norm (66).

It is convenient from the point of view of the derivation of the BEP equations to define $\| \cdot \|$ as,

$$\|\mathbf{x}\|^2 = \sum_{i=1}^{k} |(x)_i|^2 \tag{69}$$

where $(x)_i$ is a Clifford number representing the i'th part of $\mathbf{x}$ in the m-dimensional Clifford module over $\mathcal{R}_{p,q}$.

Assume that each node in the network has the same Clifford valued activation function $f : \mathcal{R}_{p,q} \to \mathcal{R}_{p,q}$.

The output o_j of the j'th neuron can be written as,

$$o_j = f(net_j) = \sum_A u_A^j e_A \tag{70}$$

With u_A^j a function from $\mathcal{R}_{p,q}$ to $\mathcal{R}$ and

$$net_j = \sum_l \omega_{lj} o_l \tag{71}$$

where l sums over all the inputs to neuron j.

It is important to notice since $\mathcal{R}_{p,q}$ is in general non-commutative the order of multiplication in the above equation is important, although it will be shown later (in Chapter 4) that networks with left weight multiplication are equivalent to networks with right multiplication.

[4]If the reader is not familiar with the concept of a module, it is enough to view these Clifford modules as weaker forms of n dimensional vectors with Clifford valued scalars instead of real valued scalars.

In the real case E depends on the number of weights in the network. In the Clifford case E depends not only on all the weights but on the components of each of the weights. Again define $\lambda = \|\Psi - \Phi\|^2$. Then:

$$\frac{\partial E}{\partial[\omega_{ij}]_A} = \frac{1}{2}\int_{\mathbf{x}\in X} \frac{\partial \lambda}{\partial[\omega_{ij}]_A} \tag{72}$$

Using the chain rule,

$$\frac{\partial \lambda}{\partial[\omega_{ij}]_A} = \sum_B \left(\frac{\partial \lambda}{\partial u_B^j} \left(\sum_C \frac{\partial u_B^j}{\partial[net_j]_C} \frac{\partial[net_j]_C}{\partial[\omega_{ij}]_A} \right) \right) \tag{73}$$

The partial derivative

$$\frac{\partial[net_j]_C}{\partial[\omega_{ij}]_A} \tag{74}$$

needs a bit of care. Using equation 71:

$$\frac{\partial[net_j]_C}{\partial[\omega_{ij}]_A} = \sum_l \frac{\partial[\omega_{lj}x_l]_C}{\partial[\omega_{ij}]_A} = \frac{\partial[\omega_{ij}o_i]_C}{\partial[\omega_{ij}]_A} \tag{75}$$

Then using the fact that:

$$\omega_{ij}o_i = \sum_{D,E} [\omega_{ij}]_D [o_i]_E e_D e_E \tag{76}$$

$$\frac{\partial[\omega_{ij}o_i]_C}{\partial[\omega_{ij}]_A} = \frac{\partial\left(\sum_{D,E}[\omega_{ij}]_D[o_i]_E \kappa_{A,E}\right)}{\partial[\omega_{ij}]_A}$$

with κ defined as in (12) and D, E summing over all the elements such that $e_D e_E = \pm e_C$. Since the denominator of the partial derivative only refers to $[\omega_{ij}]_A$ the partial derivative will equal:

$$\frac{\partial[\omega_{ij}o_i]_C}{\partial[\omega_{ij}]_A} = \frac{\partial[\omega_{ij}]_A[o_i]_E \kappa_{A,E}}{\partial[\omega_{ij}]_A} = \kappa_{A,E}[o_i]_E \quad \text{with} \quad e_A e_E = \pm e_C \tag{77}$$

For example in the algebra $\mathcal{R}_{2,0}$ the table of derivatives would look like,

$\frac{\partial[x]_B}{\partial[\omega_{jl}]_A}$	$B=0$	1	2	12
$A=0$	$[x_{jl}]_0$	$[x_{jl}]_1$	$[x_{jl}]_2$	$[x_{jl}]_{12}$
1	$[x_{jl}]_1$	$[x_{jl}]_0$	$[x_{jl}]_{12}$	$[x_{jl}]_2$
2	$[x_{jl}]_2$	$-[x_{jl}]_{12}$	$[x_{jl}]_0$	$-[x_{jl}]_1$
12	$-[x_{jl}]_{12}$	$[x_{jl}]_2$	$-[x_{jl}]_1$	$[x_{jl}]_0$

(78)

The error derivative is now quite easy to calculate. If j is an output neuron then,

$$\frac{\partial\lambda}{\partial u^j_A} = \frac{\partial}{\partial u^j_A}\|\Psi - \Phi\|$$

$$\frac{\partial}{\partial u^j_A}|o_j - \Phi_j^2|^2 = 2[o_j - \Phi_j]_A$$

If j is not an output unit then the chain rule has to be used again.

$$\frac{\partial\lambda}{\partial u^j_A} = \sum_k \frac{\partial\lambda}{\partial u^k_A}\left(\sum_{B,C}\frac{\partial u^k_B}{\partial[net_k]^C}\frac{\partial[net_k]^C}{\partial u^j_A}\right) \tag{79}$$

with k running over the neurons that receive input from neuron j.

The term

$$\frac{\partial[net_k]_C}{\partial[u_j]_A}$$

is calculated in a similar manner to (77),

$$\frac{\partial[net_k]_C}{\partial u^j_A} = \kappa_{A,E}[\omega_{jk}]_D \tag{80}$$

where $\kappa_{A,E}e_Ae_E = e_C$.

The derivatives:

$$\frac{\partial u^k_B}{\partial[x_k]_C}$$

play the same rôle as $f'(net_j)$ does in the real-valued case and depends on the activation function used; this will be discussed in the next section.

Bringing this all together we have,

$$\frac{\partial E}{\partial[\omega_{ij}]_A} = \frac{1}{2}\int_{x\in X}\sum_B \lambda_j^B \left(\sum_C \frac{\partial u_B^j}{\partial[net_j]_C}\kappa_{A,E}[o_k]_E\right) \tag{81}$$

with $e_Ae_E = \pm e_C$ and

$$\lambda_j^B = \frac{\partial\|\Psi - \Phi\|^2}{\partial u_B^j} = 2[o_j - \Phi_j]_B \tag{82}$$

if j is an output neuron.

If j is not an output unit then the chain rule has to be used again.

$$\lambda_j^B = \sum_k \lambda_k^B \left(\sum_{B,C} \frac{\partial u_B^k}{\partial[net_k]^C}\kappa_{A,D}[\omega_{jk}]_D\right) \tag{83}$$

with k running over the neurons that receive input from neuron j and $e_Ae_D = e_C$.

3.7 Choice of activation function

So far nothing has been said about the choice of activation functions used in networks. The most widely used class of activation functions are the so called semi-linear functions. A semi-linear function $f(x)$ is a continuous function that satisfies the following condition:

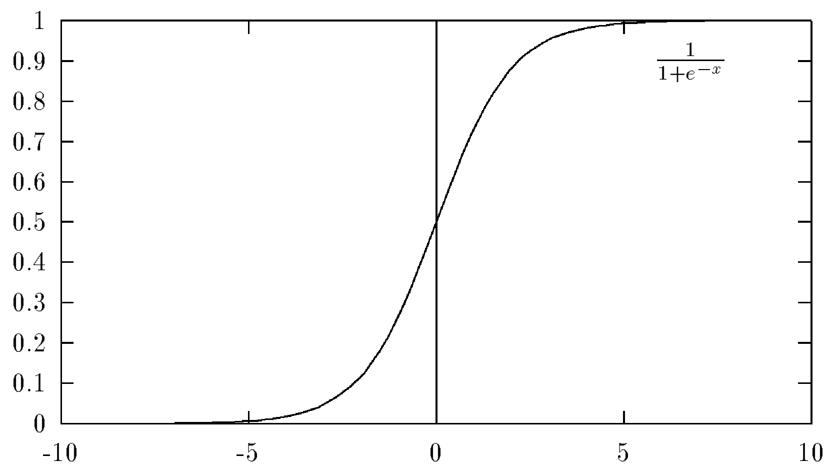
$$\lim_{x\to\infty} f(x) = 1 \quad \lim_{x\to-\infty} f(x) = -1 \tag{84}$$

Typically the second condition is replaced by the condition:

$$\lim_{x\to-\infty} f(x) = 0$$

Typical examples include the sigmoid function:

$$f(x) = \frac{1}{1+e^{-x}} \tag{85}$$



and the tanh function:

$$f(x) = \tanh(x) \tag{86}$$

The reason for this choice of functions comes from two observations. First, linear functions are not satisfactory since: any multi-layer network composed of linear functions is equivalent to a single layer network; and further, many decision problems are not linearly separable. The second reason is that traditionally the first neural networks (see for instance McCulloch and Pitts [McCulloch and Pitts, 1943]) had an all or nothing response; typically neurons had step valued activation function:

$$f(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} \tag{87}$$

These functions do indeed provide multi-layer networks with more power than a single layer network. Indeed it was shown by Minsky and Papert [Minsky and Papert, 1969] that single layer networks with step functions cannot solve some non-trivial decision problems. Also because the step functions are non-linear and are integer valued this turns training into an integer programming problem, which is known to be NP-hard (see ([Garey and Johnson, 1979]) (although there might exist individual cases solvable in polynomial time). Further it limits the networks to functions defined on binary data.

Semi-linear functions provide a compromise between linear functions and step functions. From the observation that $f(\alpha x)$ for large values of α acts like a step functions or more correctly the function:

$$f(x) = \begin{cases} 1 & \text{if } x > 0 \\ \frac{1}{2} & \text{if } x = 0 \\ 0 & \text{if } x < 0 \end{cases} \tag{88}$$

and for values of α close to zero, $f(\alpha x)$ acts like a linear function. Indeed Sontag [Sontag, 1992b] has shown that networks composed of sigmoid functions are better at binary decision problems than networks with only step functions, for exactly the reasons stated above (that sigmoids can approximate either linear or step valued activation functions).

Further, it has been shown by many authors (see for instance [Cybenko, 1989; Hornick *et al.*, 1989]) that a network with one hidden layer with sigmoid activation function, is able to approximate any continuous function defined on a compact subset (providing a sufficient number of neurons are present).

3.7.1 Complex activation functions.

It might be naively assumed that the rich field of complex analysis is going to provide a suitable class of activation functions. There exists a complex extension of the sigmoid function:

$$f(z) = \frac{1}{1 + e^{-z}}$$

where e^{-z} is the complex exponential function. This function is analytic (in the sense of complex analysis) but it is not bounded. Any function that is complex analytic and bounded is necessarily constant by Liouville's theorem (see any standard text on complex analysis, such as [Priestley, 1990], for a proof).

The most important characteristic of a complex activation function is that it should be bounded and nonlinear in its components, and the partial derivatives should exist and be continuous. The partial derivatives must also be such that learning always takes place in the presence of non-zero error. For a fuller discussion see [Georgiou and Koutsougeras, 1992].

In [Georgiou and Koutsougeras, 1992] a simple complex activation is proposed:

$$f(z) = \frac{z}{c + \frac{1}{r}|z|} \tag{89}$$

This activation function has been used successfully in some simple applications, for instance the complex encoder decoder problem.

3.7.2 The Clifford case

The simple activation function proposed above extends to the Clifford case. The partial derivatives are easy to work out (using the notation of the previous section):

$$\frac{\partial u_A}{\partial [x]_B} = \begin{cases} -\frac{r[x]_A[x]_B}{(c+\frac{1}{r}|x|)^2 r|x|} & \text{if } |x| > 0 \\ 0 & \text{if } |x| = 0 \end{cases} \tag{90}$$

if $A \neq B$ and if $A = B$ then,

$$\frac{\partial u_A}{\partial [x]_B} = \begin{cases} \frac{r(|x|^2 - [x]_A^2 + cr|x|)}{|x|(cr+|x|)^2} & \text{if } |x| \neq 0 \\ \frac{1}{c} & \text{if } |x| = 0. \end{cases} \tag{91}$$

the norm being the Clifford norm defined in the previous section.

The activation function in both the Clifford and the complex case can be seen to be mapping a vector in $\mathcal{R}^{2^n}$ (the vector space which carries the Clifford algebra) inside the unit sphere keeping the direction constant, but mapping the norm of the vectors $|\mathbf{x}|$ to $|\mathbf{x}|/(1+|\mathbf{x}|)$.

3.8 Experimental results

The encoder-decoder problem is often used to test new techniques in back error propagation. While it is not a formal benchmark it is useful to get a feel of how new algorithms can perform. Essentially for a network to solve the encoder-decoder problem a training set is presented to the network which forces the hidden units to encode the training set in some way. For instance in the real case with a 3-2-3 network if the network is trained on the set of vectors $(1,0,0)$, $(0,1,0)$, $(0,0,1)$ then then two hidden units will learn a binary coding of the input signals.

The results presented in this section show how a Clifford network is able to solve this encoder-decoder problem for multidimensional patterns.

Results are given for different algebras and different network configurations. In each case the system successfully encodes and decodes the input patterns providing a well separated response. It is also interesting to note that the epoch count is approximately the same as would be expected for a similar problem on a conventional BEP network (wall clock time is obviously increased due to the extra complexity in the individual arithmetic operations). This preliminary test shows that Clifford networks are able to learn simple tasks and confirms the convergent operation of the learning algorithm.

Table 2: Table of outputs from a trained 3-2-3 encoder-decoder $\mathcal{R}_{0,2}$.

Pattern 0	(1.00000,0.00000,0.00000,0.00000) (0.00000,0.00000,0.00000,0.00000) (0.00000,0.00000,0.00000,0.00000)
Output	(0.82741,0.00180,-0.00034,0.00075) (-0.00896,-0.01346,0.03117,-0.00653) (-0.02300,0.00890,-0.00371,-0.00493)
Pattern 1	(0.00000,0.00000,0.00000,0.00000) (1.00000,0.00000,0.00000,0.00000) (0.00000,0.00000,0.00000,0.00000)
Output	(0.01103,0.02384,-0.00875,-0.00106) (0.82393,-0.00412,-0.00112,-0.00050) (-0.02084,0.00501,-0.00617,0.00553)
Pattern 2	(0.00000,0.00000,0.00000,0.00000) (0.00000,0.00000,0.00000,0.00000) (0.00000,0.00000,0.00000,-1.00000)
Output	(0.01646,0.03726,-0.02079,-0.00340) (-0.01691,-0.02265,0.03517,-0.00731) (-0.00301,0.00194,-0.00061,-0.82819)

Table 3: Table of input-output for a trained 4-2-4 encoder-decoder over the algebra $\mathcal{R}_{0,2}$.

Pattern 0	(1.00000,0.00000,0.00000,0.00000) (0.00000,0.00000,0.00000,0.00000) (0.00000,0.00000,0.00000,0.00000) (0.00000,0.0 0000,0.00000,0.00000)
Output	(0.89258,-0.00044,0.00001,0.00025) (0.00020,-0.00002,-0.00002,-0.00004) (0.00018,-0.00004,0.00012,-0.00015) (-0 .00000,-0.00000,-0.00000,0.00000)
Pattern 1	(0.00000,0.00000,0.00000,0.00000) (1.00000,0.00000,0.00000,0.00000) (0.00000,0.00000,0.00000,0.00000) (0.00000,0.0 0000,0.00000,0.00000)
Output	(0.00020,0.00006,-0.00006,-0.00005) (0.89238,0.00023,0.00064,0.00041) (0.00018,0.00001,-0.00019,-0.00002) (-0.0 0000,-0.00000,0.00000,0.00000)
Pattern 2	(0.00000,0.00000,0.00000,0.00000) (0.00000,0.00000,0.00000,0.00000) (1.00000,0.00000,0.00000,0.00000) (0.00000,0.0 0000,0.00000,0.00000)
Output	(0.00019,0.00007,0.00005,0.00001) (0.00019,-0.00006,-0.00005,-0.00002) (0.89278,0.00014,0.00065,0.00152) (0.000 00,0.00000,0.00000,-0.00000)

Table 4: Table of input-ouput relationships for a trained 4-2-4 encoder-decoder the algebra $\mathcal{R}_{1,1}$.

Pattern 0	(1.00000,0.00000,0.00000,0.00000)
	(0.00000,0.00000,0.00000,0.00000)
	(0.00000,0.00000,0.00000,0.00000)
	(0.00000,0.0 0000,0.00000,0.00000)
Output	(0.79505,0.00213,-0.03843,0.00260)
	(0.60009,0.00017,-0.07628,0.00026)
	(-0.51896,-0.00026,0.17408,-0.00081)
	(0.0 0000,0.00000,-0.00000,-0.00000)
Pattern 1	(0.00000,0.00000,0.00000,0.00000)
	(1.00000,0.00000,0.00000,0.00000)
	(0.00000,0.00000,0.00000,0.00000)
	(0.00000,0.0 0000,0.00000,0.00000)
Output	(0.49496,0.00049,0.28122,-0.00068)
	(-0.46598,-0.05589,0.07024,0.00565)
	(0.78560,0.00071,-0.15706,0.00044)
	(0.00 000,0.00000,-0.00000,0.00000)
Pattern 2	(0.00000,0.00000,0.00000,0.00000)
	(0.00000,0.00000,0.00000,0.00000)
	(1.00000,0.00000,0.00000,0.00000)
	(0.00000,0.0 0000,0.00000,0.00000)
Output	(-0.42091,0.00011,-0.31815,0.00012)
	(0.78305,-0.00003,0.13566,-0.00022)
	(0.57565,0.00087,0.00391,0.00224)
	(-0.0 0000,-0.00000,0.00000,-0.00000)

Table 5: Table of input and output of a 3-2-3 encoder over the algebra $\mathcal{R}_{0,3}$ with graph of rms error.

Pattern 0	(1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00) (0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00) (0.00,0.00,0.00,0.00,0.00,0.00,0 .00,0.00)
Output	(0.81,-0.00,-0.00,-0.00,0.00,-0.00,0.00,0.00) (0.00,0.00,0.00,-0.00,-0.00,0.00,0.00,0.00) (-0.00,0.00,-0.00,-0.00, -0.00,-0.00,0.00,0.00)
Pattern 1	(0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00) (1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00) (0.00,0.00,0.00,0.00,0.00,0.00,0 .00,0.00)
Output	(0.00,0.00,0.00,0.00,-0.00,0.00,0.00,0.00) (0.81,0.00,-0.00,0.00,0.00,-0.00,-0.00,-0.00) (-0.00,-0.00,0.00,-0.00,0 .00,0.00,0.00,0.00)
Pattern 2	(0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00) (0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00) (0.00,0.00,0.00,0.00,0.00,0.00,0 .00,1.00)
Output	(0.00,0.00,-0.00,0.00,0.00,0.00,-0.00,-0.00) (0.00,-0.00,-0.00,0.00,0.00,0.00,-0.00,-0.00) (0.00,-0.00,-0.00,0.00, -0.00,0.00,-0.00,0.81)

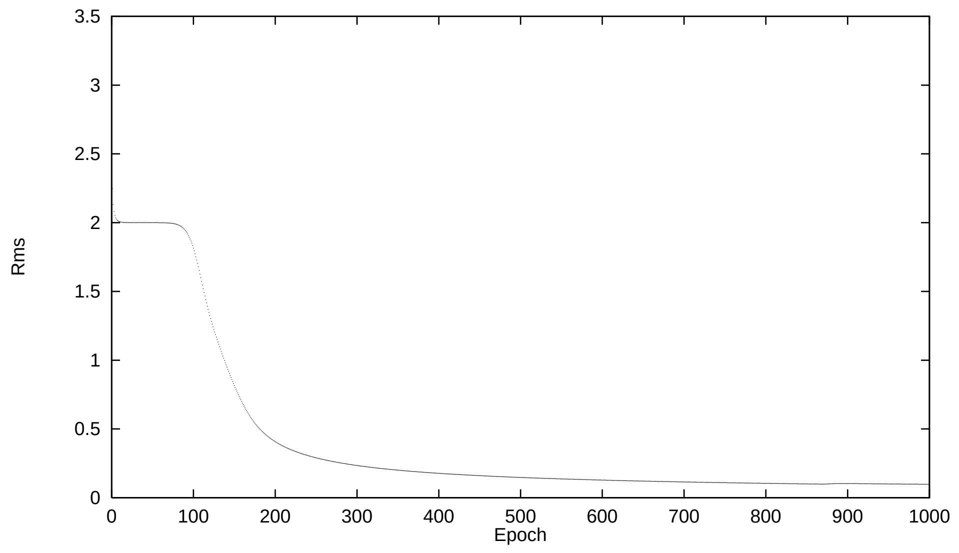


Figure 5: Graph of RMS error for a 3-2-3 encoder-decoder problem over the algebra $\mathcal{R}_{0,2}$

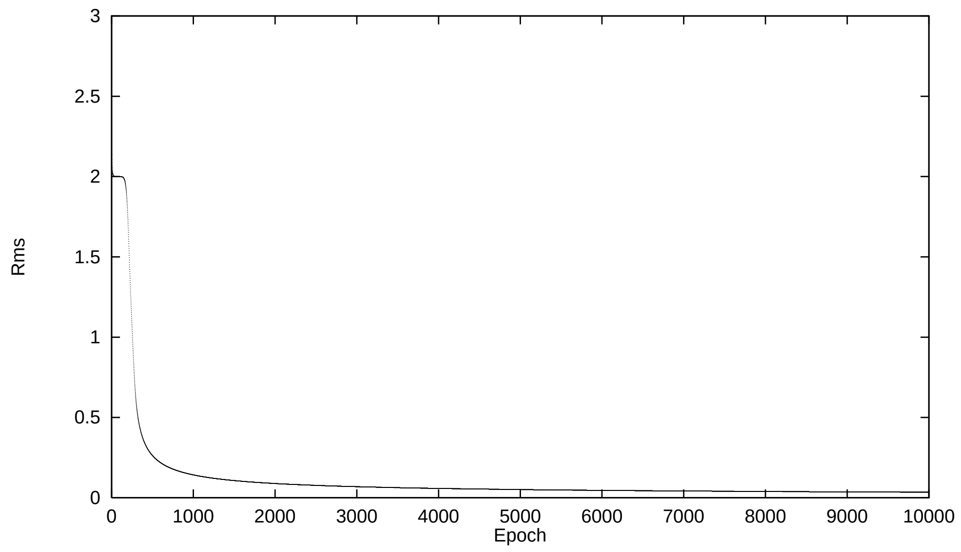


Figure 6: Graph of RMS error during training the 4-2-4 encoder-decoder over the algebra $\mathcal{R}_{0,2}$

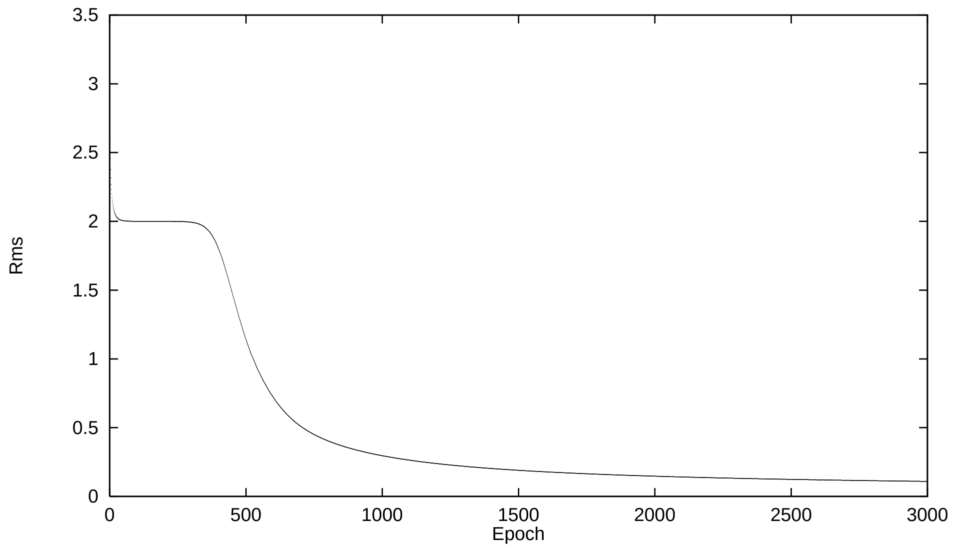


Figure 7: Graph of RMS for 4-2-4 encoder-decoder of the algebra $\mathcal{R}_{1,1}$

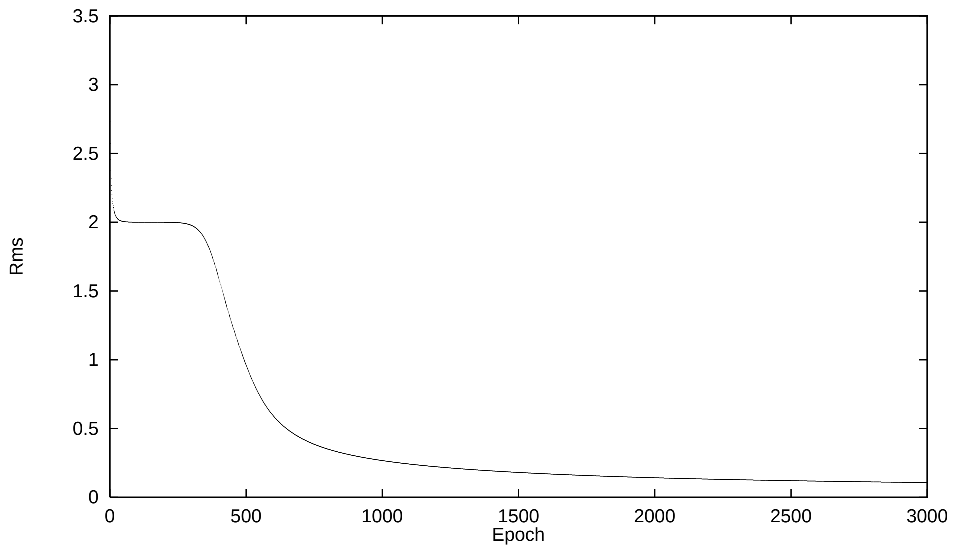


Figure 8: RMS during training for the encoder-decoder problem over $\mathcal{R}_{0,3}$

Chapter 4

Some Theoretical Results

4.1 Introduction

There are essentially two ways of analyzing feed-forward networks. The first views a feed-forward network as a pattern classifier and uses statistical techniques to assess the performance of a network; see [MacKay, 1992]. The second treats a feed-forward network as essentially a function approximator, that is, given a network with n inputs and m outputs and a set of weight values ω_{ij} the network can be seen to be computing a function:

$$\Phi_\omega : \mathcal{R}^n \rightarrow \mathcal{R}^m \tag{92}$$

In the Clifford case the real numbers $\mathcal{R}$ are replaced by an arbitrary Clifford algebra $\mathcal{R}_{p,q}$. The sort of question then asked is how well can a given class of networks approximate classes of functions? Various theorems have been proved [Cybenko, 1989; Hornick *et al.*, 1989; Ito, 1991; Kůrková, 1991] which show that feed-forward networks with one hidden layer are sufficient to approximate continuous functions. Further results by Sontag [Sontag, 1992a] show that in certain problems two hidden layers are required; this is because the function trying to be approximated is too discontinuous to be approximated by a single hidden layer network.

This chapter first extends Cybenko's [Cybenko, 1989] proof, that real valued networks with a single hidden layer can approximate any bounded continuous function with compact support, to networks over an arbitrary Euclidean Clifford algebra (that is algebras with signature $0, q$). The second part of the chapter looks at how Clifford networks represent functions: first it is shown that although Clifford networks represent the same class of functions as real valued networks, they do it in a different way and secondly it is shown that as for real networks, Clifford networks with identical input output behaviour are identical in structure modulo certain symmetry relations on the hidden units.

4.2 Talking about Approximations: A Crash course in metric space theory

Mathematically some care has to be taken with the notion of approximation. What is required is some function E which given two functions ϕ,ψ gives:

$$E(\phi, \psi) = 0 \text{ if and only if } \phi = \psi \tag{93}$$

and $E(\phi, \psi)$ be close to zero if ϕ is 'close' to ψ. Formalizing these requirements leads to the notion of a metric space (see [Copson, 1988] for a good introduction), which is defined as a set X together with a distance metric $d : X \to \mathcal{R}$ satisfying the following axioms:

$$d(x, x) = 0 \qquad \text{for all } x \in X \tag{94}$$

$$d(x, y) = d(y, x) \qquad \text{for all } x, y \in X \tag{95}$$

$$d(x,z) \leq d(x,y) + d(y,z) \quad \text{for all } x,y,z \in X \tag{96}$$

.

Various choices of metrics are available for function spaces, first the metric of uniform convergence:

$$E(\phi,\psi) = \sup\{|\phi(x) - \psi(x)| \,\| x \in X\} \tag{97}$$

where X is a subset on which both ϕ and ψ are defined, more formally $X \subseteq \mathrm{dom}(\phi) \cap \mathrm{dom}(\psi)$. To guarantee this value is defined extra conditions either have to be imposed on X or on ϕ and ψ. If X is stipulated to be compact (i.e. X is finite) or extra conditions must be imposed on the asymptotic behavior of the functions ϕ and ψ as both tend to infinity then in both cases the equation (97) will be well defined. In this thesis only compactness assumptions will be needed. Uniform convergence metrics are useful where networks are required to perform equally well over the whole of the input space.

Second there are the L_p metrics which are much easier to deal with mathematically and are defined as:

$$\rho_{p,\mu}(\phi,\psi) = \left(\int |\phi - \psi|^p d\mu\right)^{\frac{1}{p}} \tag{98}$$

Again extra conditions have to be stipulated to make the metric always well defined; similar to the conditions for the metric (97) above, see [Hornick, 1991] for details.

In order to ask the question whether a class of networks can approximate a class of continuous functions, the concept of density is needed. Given a class of functions C and a function norm $|\cdot|$ a subset S is said to be dense in C if the closure of S is the whole of C. What this means in practical terms for neural networks is S is dense in C if given a function $f \in C$ and an arbitrary $\epsilon > 0$ there exists a $g \in S$ such that $|f - g| < \epsilon$. Many theorems prove that Neural networks are universal approximators by showing the the function space of Neural networks is dense in an appropriate function space.

4.3 Clifford modules

This section deals with a generalization of vector spaces, the theory of Modules over rings: specifically Clifford modules. Various theorems are stated which are generalizations of traditional theorems such as the Hahn-Banach theorem and the Riesz representation theorem [Rudin, 1966; Rudin, 1973]; all proofs are omitted, but these can be found in [Brackx *et al.*, 1982].

From now on the convention adopted in [Brackx *et al.*, 1982] is used, where a Euclidean Clifford algebra is referred as an $\mathcal{A}$ algebra. A module is a generalization of a vector space, where the set of coefficients come from some ring instead of a field, thus modules have a different geometrical structure from vector spaces.

Definition 1 *A unitary left* $\mathcal{A}$-module $X_{(l)}$ *is an Abelian group* $X_{(l)}, +$ *and an operation* $(\lambda, f) \rightarrow \lambda f$ *from* $\mathcal{A} \times X_{(l)}$ *into* $X_{(l)}$ *s.t. for all* $\lambda, \mu \in \mathcal{A}$ *and* $f, g \in X_{(l)}$ *the following hold:*

$$(\lambda + \mu) f = \lambda f + \mu f \tag{99}$$

$$(\lambda \mu) f = \lambda (\mu f) \tag{100}$$

$$\lambda (f + g) = \lambda f + \lambda g \tag{101}$$

$$e_0 f = f \tag{102}$$

We have already met an example of a Clifford Module in Chapter 3, the space $\mathcal{R}^n_{p,q}$.

Definition 2 *Let $X_{(l)}$ be a unitary left $\mathcal{A}$-module, then a function $p : X_{(l)} \rightarrow \mathcal{R}$ is said to be a* proper semi-norm *if there exists a constant $C_0 \geq 0$ s.t. for all $\lambda \in \mathcal{A}$ and $f, g \in X_{(l)}$ the following conditions are satisfied:*

$$p(f+g) \leq p(f) + p(g) \tag{103}$$

$$p(\lambda f) \leq C_0 |\lambda| p(f) \tag{104}$$

$$p(\lambda f) = |\lambda| p(f) \quad \textit{if } \lambda \in \mathcal{R} \tag{105}$$

$$\textit{If } p(f) = 0 \textit{ then } \; f = 0 \tag{106}$$

Definition 3 *Given a module $X_{(l)}$ the* algebraic dual $X_{(l)}^{*alg}$ *is defined to be the set of left $\mathcal{A}$-linear functionals from $X_{(l)}$ into $\mathcal{A}$.*

That is the set of functionals $T : X_{(l)} \rightarrow \mathcal{A}$ s.t.

$$T(\lambda f + g) = \lambda T(f) + T(g) \tag{107}$$

$f, g \in X_{(l)}$ and $\lambda \in \mathcal{A}$.

Definition 4 *The set of bounded T functionals with respect to a semi-norm p is denoted $X_{(l)}^{*} \subset X_{(l)}^{*alg}$. Explicitly for all functionals T and for all $f \in X_{(l)}$:*

$$|T(f)| \leq C p(f) \tag{108}$$

for some real constant C.

The following theorem is a a corollary to a Hahn-Banach type theorem for Clifford modules for details and proof see sections 2.10-2.11 in [Brackx *et al.*, 1982].

Theorem 1 *Let $X_{(l)}$ be a unitary left $\mathcal{A}$-module provided with a semi norm p and let $Y_{(l)}$ be a submodule of $X_{(l)}$. Then $Y_{(l)}$ is dense in $X_{(l)}$ if and only if for each $T \in X_{(l)}^{*}$ such that $T|Y_{(l)} = 0$*[1] *we have $T = 0$ on $X_{(l)}$.*

[1] T restricted to $Y_{(l)}$ equal to zero

Now a useful class of function spaces is introduced.

Definition 5 *The space $C^0(\mathcal{K};\mathcal{A})$. Let $\mathcal{K}$ be a compact subset of $\mathcal{R}^r$ $(r \geq 1)$. Then $C^0(\mathcal{K};\mathcal{A})$ stands for the unitary bi-$\mathcal{A}$-module of $\mathcal{A}$-valued continuous functions on $\mathcal{K}$.*

This can be thought of as a product of classical real valued functions i.e.:

$$C^0(\mathcal{K};\mathcal{A}) = \Pi_A C^0(\mathcal{A};\mathcal{R})e_A \tag{109}$$

where A runs over all the basis elements in the Clifford algebra in question. A norm can be defined for each $f \in C^0(\mathcal{K};\mathcal{A})$:

$$||f|| = \sup_{x\in\mathcal{K}} |f(x)| \tag{110}$$

This norm is equivalent to the product norm taken from (109).

Definition 6 *Given an open set $\Omega \subset \mathcal{R}^n$ and a sequence $(\mu_B)_B$ of real valued measures on Ω. Then for any open set in Ω an $\mathcal{A}$ valued measure can be defined:*

$$\mu(I) = \sum_B \mu_B(I)e_B \tag{111}$$

Definition 7 *An $\mathcal{A}$-valued function:*

$$f = \sum_A f_a e_A \tag{112}$$

is said to be μ-integrable in Ω if for all A and B ranging over the basis elements of $\mathcal{A}$ each f_A is μ_B integrable.

Definition 8 *For any μ-integrable function f define*[2]*:*

$$\int_\Omega f(x)d\mu = \sum_{A,B} e_A e_B \int_\Omega f_A(x)d\mu_B \tag{113}$$

[2]These are Clifford valued integrals and hence are different from the integrals used, in Chapter 3, to define the error metrics

A Riesz representation type theorem can be obtained.

Theorem 2 *Let T be a bounded $\mathcal{A}$ valued function in $C^0_{(l)}(\mathcal{K};\mathcal{A})$. Then there exists a unique $\mathcal{A}$ valued measure μ with support contained in $\mathcal{K}$ such that for all $f \in C^0_{(l)}(\mathcal{K};\mathcal{A})$:*

$$T(f) = \int_{\mathcal{K}} f(x)d\mu \tag{114}$$

For a proof again see [Brackx *et al.*, 1982].

4.4 The Approximation result

A feed-forward network with one output neuron and N inputs units and K hidden units computes a function:

$$\Phi(\mathbf{x}) = \sum_{j=1}^{K} \alpha_j f(\sum_{i=1}^{N} y_{ij}x_i + \theta_j) \tag{115}$$

with f the activation function x_i the i'th input, y_{ij} weight values for the connection between the input layer and the hidden layer and α_j the weights from the hidden layer to the output node.

$\Phi(\mathbf{x})$ can be seen as a function from $\mathcal{R}^{N2^n}$(where 2^n is the dimension of $\mathcal{A}$) to $\mathcal{A}$ and hence a member of $C^0_{(l)}(\mathcal{R}^{N2^n};\mathcal{A})$. This is why the material of the last section was relevant. The next definition is important. What is shown is that all activation functions satisfying the definition, when used in feed-forward networks, are universal approximators. Then to complete the proof all that is needed to show is that the activation functions considered in Chapter 3 satisfy the definition.

Definition 9 *An activation function f (considered as a function from $\mathcal{R}^{N2^n}$ to $\mathcal{A}$) is said to be* discriminating *if for any given Clifford valued measure μ with support*

I^{N2^n} if:

$$\int_{I^{N2^n}} f(\sum_{i=1}^{N} y_i x_i + \theta) d\mu(x) = 0 \tag{116}$$

for all $y_i, \theta \in \mathcal{R}_{0,n}$ implies $\mu(x) = 0$.

Theorem 3 *Let f be any continuous discriminating functions. Then finite sums of the form:*

$$\Phi(x) = \sum_{j=1}^{K} \alpha_j f(\sum_{i=1}^{N} y_{ij} x_i + \theta_j) \tag{117}$$

are dense in $C^0_{(l)}(I^{N2^n}; \mathcal{A})$

Proof: This proof is essentially a modification of Cybenko's Theorem 1 in [Cybenko, 1989] using the theory of Clifford modules in the last section.

Let S be the function space generated by sums of the form (117). Assume that the closure of S is not all of $C^0_{(l)}(I^{N2^n}; \mathcal{A})$; denote the closure of S by R. By the Hahn-Banach type theorem 1 there is a bounded linear functional T on $C^0_{(l)}(I^{N2^n}; \mathcal{A})$, with $T \neq 0$ but $T(R) = T(S) = 0$. By Theorem 2 this bounded linear functional is of the form:

$$T(h) = \int_{I^{N2^n}} h(x)\mu(x) \tag{118}$$

for some measure μ and $h \in C^0_{(l)}(I^{k2^n}, \mathcal{A})$. In particular since $f \in C^0_{(l)}(I^{k2^n}, \mathcal{A})$ is in R, for any y_i:

$$T(f) = \int_{k2^n} f(\sum_{i=1}^{N} y_i x_i + \theta) d\mu(x) = 0 \tag{119}$$

Since f is discriminating this implies $\mu = 0$ contradicting our assumption hence S must be dense in $C^0_{(l)}(I^{N2^n}; \mathcal{A})$. $\square$

So to prove that the class of feed-forward networks considered in Chapter 3 are universal approximators, we have to show that functions of the form:

$$f(x) = \frac{x}{1 + |x|} \tag{120}$$

are discriminating.

Theorem 4

$$f(x) = \frac{x}{1+|x|} \tag{121}$$

is discriminatory.

Proof: A function $f(x)$ is discriminatory , if:

$$\int_{N2^n} f(\sum_{i=1}^{N} y_i x_i + \theta) d\mu(x) = 0 \tag{122}$$

for all y_i implies that $\mu(x) = 0$. This is equivalent to saying that:

$$\int_{N2^n} f(\sum_{i=1}^{N} y_i x_i + \theta) d\mu(x) = \sum_{A,B} e_A e_B \int_{N2^n} f_A(\sum_{i=1}^{N} y_i x_i + \theta) d\mu_B(x) = 0 \tag{123}$$

for all y_i.

Define $\gamma_A(x) : I^{N2^n} \to \mathcal{R}$ to be the limit of:

$$\gamma_A(x) = \lim_{\lambda\to\infty} f_A(\lambda x) \tag{124}$$

(where λx is a Clifford multiplication, with λ a real number). So

$$f_A(\lambda x) = \frac{[\lambda z]_A}{1+|\lambda z|} = \frac{\lambda [z]_A}{1+\lambda |z|} \tag{125}$$

So

$$\gamma_A(z) = \begin{cases} 1 & \text{if} \quad [z]_A > 0 \\ 0 & \text{if} \quad [z]_A = 0 \\ -1 & \text{if} \quad [z]_A < 0 \end{cases} \tag{126}$$

In our case:

$$\gamma_A(\sum_{i=1}^{N} y_i x_i + \theta) = \begin{cases} 1 & \text{if} \quad [\sum_{i=1}^{N} y_i x_i + \theta]_A > 0 \\ 0 & \text{if} \quad [\sum_{i=1}^{N} y_i x_i + \theta]_A = 0 \\ -1 & \text{if} \quad [\sum_{i=1}^{N} y_i x_i + \theta]_A < 0 \end{cases} \tag{127}$$

The sets defined by $[\sum_{i=1}^{N} y_i x_i + \theta]_A = 0$ are hyper-planes, since $[\sum_{i=1}^{N} y_i x_i + \theta]_A$ is just a set of linear equations in the components of x_i.

The rest of the proof is almost verbatim from Lemma 1 of Cybenko [Cybenko, 1989]. So let $\Pi^A_{y,\theta} \subset I^{2^n}$ be the hyper-plane defined by:

$$\left\{ x | \left[\sum_{i=1}^{N} y_i x_i + \theta \right]_A = 0 \right\} \tag{128}$$

and let $H^A_{y,\theta}$ be the half space defined by:

$$\{x | [\sum_{i=1}^{N} y_i x_i + \theta]_A > 0\} \tag{129}$$

Then by the Lebesgue bounded convergence theorem we have:

$$0 = \int_{I^{N2^n}} f_A(\lambda x) d\mu_B(x) = \int_{I^{N2^n}} \gamma_A(x) d\mu_B(x) = \mu(H^A_{y,\theta}) \tag{130}$$

Now if μ_B were always a positive measure the result would be trivial, but since μ_B is an arbitrary measure the result is harder (since positive bits of μ might cancel out negative bits of μ_B).

Fix the y_i's and define:

$$F_A(h) = \int_{I^{N2^n}} h([\sum_{i=1}^{K} y_i x_i]_A) \tag{131}$$

for some bounded μ_B measurable function $h : \mathcal{R} \to \mathcal{R}$. F_A is a bounded functional on $L^\infty(\mathcal{R})$.

Let h be the indicator function on the interval $[\theta_A, \infty)$, then:

$$F(h) = \int_{k2^n} h([\sum_{i=1}^{K} y_i x_i]_A) = \mu_B(\Pi^A_{y,\theta}) + \mu(H^A_{y,\theta}) \tag{132}$$

Similarly $F(h) = 0$. If h is the indicator of any open interval, by linearity $F(h) = 0$ and hence for any simple function. Since the simple functions are dense in $L^\infty(\mathcal{R})$, $F = 0$.

In particular given the two functions $s(x) = \sin(x)$, $c(x) = \cos(x)$:

$$F_A(s(x) + ic(x)) = \int_{I^{k2^n}} s([\sum_{k=1}^{K} y_k x_k]_A) + ic([\sum_{k=1}^{K} y_k x_k]_A) d\mu_B =$$
$$\int_{I^{k2^n}} \exp(i[\sum_{k=1}^{K} y_k x_k]_A) d\mu_B = 0 \qquad (133)$$

for all y_k. Therefore the Fourier transform of μ_B is zero, hence μ_B must be zero and hence f is discriminatory. $\square$

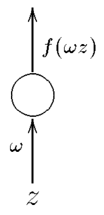
One important thing to point out with this proof is that the order of weight multiplication is irrelevant; the whole proof could be repeated with networks where multiplication was done on the right. Thus it does not matter theoretically which sort of nets (left or right weight multiplication) is used for a particular problem. Practically not much is known, but in all the examples the author has tried, the performance of the net does not seem to be affected by the order of weight multiplication.

4.5 Cliffords into Reals won't go

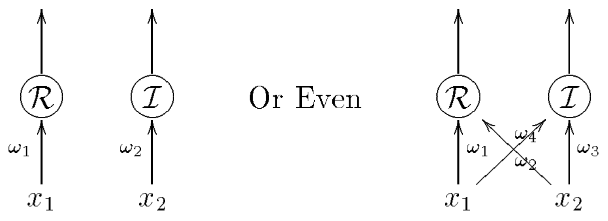
People are often confused by Clifford networks. Once they see that the weight and activation values are multi-dimensional, they ask: well, what's the difference between a network with x inputs, y hidden neurons and z outputs and a network with $x2^n$ inputs $y2^n$ hidden units and $z2^n$ outputs (where 2^n is the dimension of the algebra in question)?

Although real and Clifford networks can represent the same class of functions, there is not a direct and simple mapping between them. To make things simpler the complex case is concentrated on; these observations scale up to any Clifford algebra without difficulty.

Consider a single complex neuron with one input and no threshold:



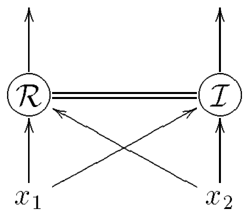
It might be thought that this is translatable into an ordinary real valued network like this:



This is not so, for writing out the equations for a single neuron we have:

$$\begin{aligned} f(\omega z) &= \frac{\omega z}{1+|\omega z|} = \frac{(w_1 + w_2 i)(z_1 + z_2 i)}{1+|wz|} \\ &= \frac{(\omega_1 z_1 - \omega_2 z_2)}{1+|\omega z|} + i\frac{(\omega_2 z_1 + \omega_1 z_2)}{1+|\omega z|} \end{aligned} \tag{134}$$

This indicates the second diagram is more appropriate, but each term in equation (134) has $1+|\omega z|$, which involves weight values from both neurons and hence implies that there is some form of cross linkage between neurons:



where the linkage terms are supplying the extra contributing factors to make up the denominators.

This shows that although Clifford networks have the same computational power as real-valued networks, they compute their results in different ways.

4.6 Function and form in a Clifford network

This section deals with a result which at first seems quite surprising. It turns out that given a feed-forward network with one hidden layer which implements a function Φ, modulo certain conditions and transforms of the weight space, that is the only network which will implement that function Φ.

The result for a single real valued neuron is rather easy to show. A neuron with no threshold value, one input weight ω and one output weight α will compute function:

$$\alpha f(\omega x) \tag{135}$$

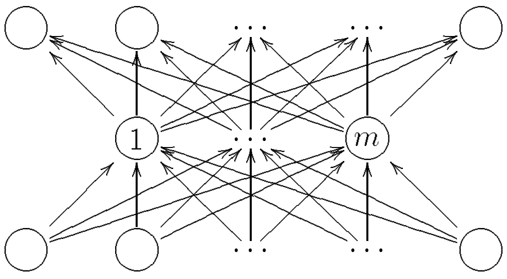
where f is the activation of the neuron. Given another neuron $\alpha' f(\omega' x)$ if f is assumed to be one to one and odd (for instance $f(x) = \tanh(x)$), and certain non-degeneracy conditions are assumed (for our purposes neither ω or α are zero) then for $\alpha f(\omega x)$ and $\alpha' f(\omega' x)$ to have the same i/o behaviour (equivalent output values for each input value x) only the following weight assignments for the second neuron will yield identical behaviour:

$$\omega = \omega' \quad \alpha = \alpha' \tag{136}$$

$$\omega = -\omega' \quad \alpha = -\alpha' \tag{137}$$

$$\tag{138}$$

Extending the result to a network with m hidden units:



there is another set of transforms on the weight space as well as the sign interchanges that will preserve the i/o behaviour the so called *exchange transforms*. Effectively an exchange transform takes two hidden units i and j and exchanges all the input and output weights of unit i to j and vice versa (this can be visualized by imagining picking up the two hidden units and swapping them around taking the relevant connections with them). The result which will be proved in this section for Clifford networks shows that the sign transforms and exchange transforms are the only behaviour preserving weight transforms. This result was first proved for the real case by [Kůrková and Kainen, 1994; Albertinie and Sontag, 1993; Sussmann, 1992] although [Albertinie and Sontag, 1993] is a much more general result for dynamical systems. The proofs in this section are not much different from [Sussmann, 1992], only they are transformed to the Clifford domain instead of being restricted to the real domain.

An extra condition has to be put on the networks to state the result. The networks have to be irreducible with respect to the following notion of reducibility. A Clifford network with m input nodes and n hidden nodes and one output unit, can be seen as a function over the Clifford domain:

$$\Phi(\mathbf{x}) = c_0 + \sum_{j=1}^{n} c_j y_j(x) \tag{139}$$

where $y_j(x) = f(v_j(x))$ (f the activation function studied in Chapter 3) and

$$v_j(\mathbf{x}) = \theta_j + \sum_{i=1}^{n} \omega_{ij} x_i \tag{140}$$

. where $\mathbf{x} = (x_1, x_2 \ldots, x_m)$ is the input to the network. A network is said to be reducible if any of the following conditions hold:

1. One of the c_js is zero

2. There exist two j_1 and j_2 s.t. $v_{j_1}(x) = v_{j_2}(x)$ or $v_{j_1}(x) = -v_{j_2}(x)$. In this case the hidden units j_1 and j_2 are said to be sign equivalent.

3. One of the v_js is constant.

Clearly if condition (1) holds, then that hidden node can be deleted without affecting the i/o-behaviour of the network. If condition (2) holds, in the first case the node j_2 can be deleted and the output weight of j_1 changed from c_{j_1} to $c' = c_{j_1} + c_{j_2}$, or in the second case the output weigh set to $c' = c_{j_1} - c_{j_2}$; in both cases the input weights are simply v_{j_1} (or v_{j_2}). If condition (3) holds, then that node can be deleted and a suitable adjustment made to c_0.

A network is called irreducible if it is not reducible by steps (1) to (3) above. A network is minimal if it is not i/o-equivalent to a network with fewer hidden units. It will be shown that a network is irreducible if, and only if, it is minimal.

Theorem 5 *Given two networks N_1 and N_2 with n_1 and n_2 hidden units, which are i/o-equivalent and both N_1 and N_2 are irreducible, then $n_1 = n_2$ and N_1 and N_2 are related by sign changes and interchanges.*

Proof: It is sufficient to show for an arbitrary basis element A if $[\Phi_1]_A = [\Phi_2]_A$ the N_1 and N_2 are related by interchanges and sign changes. Since $\Phi_1 = \Phi_2$ for all x and the activation function f is one to one, the following will be true:

$$\left[c_0^1 + \sum_{j=1}^{n_1} c_j^1 f(v_j^1(x))\right]_A = \left[c_0^2 + \sum_{j=1}^{n_1} c_j^2 f(v_j^2(x))\right]_A \tag{141}$$

Define a new set of labels $a_0 \ldots a_{n_1} \ldots a_{n_1+n_2}$ and linear functional $\nu_1 \ldots \nu_{n_1+n_2}$. Let

$$a_0 = c_0^1 - c_0^2 \tag{142}$$

$$a_j = c_j^1 \qquad 1 \le j \le n_1 \tag{143}$$

$$a_j = -c_{j-n}^2 \quad n_1 + 1 \le j \le n_1 + n_2 \tag{144}$$

Also let:

$$\nu_j = v_j \qquad 1 \le j \le n_1 \tag{145}$$

$$\nu_j = v_{j-n_1} \quad n_1 + 1 \le j \le n_2 \tag{146}$$

Then from (141) we have

$$\left[a_0 + \sum_{i=1}^{n_1+n_2} a_j f(\nu_j(x))\right]_A = 0 \tag{147}$$

for all x and A. If all the a_js is zero then the two nets are exactly equivalent. If one of the a_j's is non zero then by the linear independence lemma 1 there must exist j_1 and j_2 such that ν_{j_1} and ν_{j_2} are sign equivalent. So we have:

$$\overbrace{[a_{j_1} f(\nu_{j_1}(x)) + a_{j_2} f(\nu_{j_2}(x))]_A}^{1} + \underbrace{\left[a_0 + \sum_{i \in \{1 \ldots n_1+n_2\} \setminus \{j_1, j_2\}} a_j f(\nu_j(x))\right]}_{2} = 0 \tag{148}$$

Now both parts 1 and 2 of the above equation will equal zero, and this means that either:

$$v^1_{j_1} = v^2_{j_2-n_1} \quad c^1_{j_1} = c^2_{j_2-n_1} \tag{149}$$

or,

$$v^1_{j_1} = -v^2_{j_2-n_1} \quad c^1_{j_1} = -c^2_{j_2-n_1} \tag{150}$$

Rewriting (141) removing the units j_1 and j_2 results in another equation that equals zero; this process can be continued until either both nets are identical or $a_0 = c^1_0 = c^2_0$ is left which again makes the nets equal. Because this process terminates and uses up all the nodes then $n_1 = n_2$, so the two nets are related by sign changes and interchange weight transformations only. □

Lemma 1 *Let J be a finite set and let $\{\nu_j\}_{j \in J}$ be a finite set of non constant linear affine functions on $\mathcal{R}^m_{p,q}$ no two of which are sign equivalent. Then the functions of the form $[f(\nu_j)]_A$, $j \in J$ and the constant function* 1 *are linearly independent.*

Proof: Sussmann [Sussmann, 1992] proves this result for linear functionals on $\mathcal{R}^n$ with activation function $f(x) = \tanh(x)$, but it is clear that for the results in [Kůrková and Kainen, 1994][3] to hold, the linear independence result must be true for any sigmoid activation function. Now the Clifford functional above can be thought of as a set of functionals on $\mathcal{R}^{n2^{p+q}}$ and as remarked in the proof of the approximation result $[x/(1+|x|)]_A$ is a sigmoid like function and so the result must hold. □

Corollary 1 *An irreducible net is minimal.*

Proof: Let N be irreducible, if N were not minimal then it would be i/o-equivalent to a net with fewer hidden nodes, which is itself irreducible (if it is not, reduce it until it becomes irreducible). But the original net and the smaller net are both irreducible and i/o-equivalent hence by the previous theorem they are related by interchanges and sign changes only, hence they have the same number of hidden units. Hence a irreducible net is minimal.

4.7 Function and Form and its implication to weight space

What exactly does the Function and Form result (section 4.6) tell us, and what is its implication for learning algorithms? Suppose a network was able to learn a required

[3][Kůrková and Kainen, 1994] proves a general function and form result for non-minimal real valued networks, with arbitary sigmoid like functions.

pattern set or training function with zero error (unrealistic, but a starting point). Then the Function and Form result tells us that up to exchange transforms the weight set is unique, assuming the network is minimal (non-minimal networks can be minimized by pruning using the criteria for minimality from the previous section). So modulo the exchange transforms there is only one solution to the particular problem (if there was more than one solution not related by the exchange transforms then this would contradict the Function and Form result). This tells us that the error surface has certain symmetries imposed by the exchange transforms and there is only one essentially unique global minimum[4].

In the more realistic case where the network only learns with a certain error ϵ close to zero, the situation is not changed, because that function which the network has learned is still unique up to transformations of the weight space imposed by the exchange transforms[5]. This means learning algorithms could be restricted to certain subsets of the weight space and still ensure learning. The author conjectures (but is not in a position to prove) that during the BEP algorithm when, in the initial stages of training, the error measure seems to be at a plateau for a certain number of iterations, that in weight space the network is simply trying to settle in one isomorphic subset of the weight space. This is because of the fact that at the origin of the weight space, the regions will be connected and on the whole networks are started with weight values close to zero.

[4]Actually, if the activation functions are even there will be $n!$ solutions (n the number of hidden units) and if the activation functions are odd there will be $2n!$ solutions.

[5]Another way of looking at this is to look at topologies on the weight set that are imposed by the inverse image of the error metric. That is two weights w and w' would be close in the new topolgy if E_w and $E_{w'}$ are close on the real line. This has the effect of identifying essentially equivalent soloutions (equivalent by exchange transforms), so there would be only one global soloution in this new space. But if non-minimal networks are considered the topology is more interesting, because pruning will make weights close, which are not close in euclidean topology.

Chapter 5

Q^2PSK

5.1 Introduction

This chapter describes how Clifford valued networks can be used to demodulate signals. Hopfield networks have already been applied to the same modulation scheme [Saied and Soliman, 1989], and when the results in Figure 18, are compared with the results reported in [Saied and Soliman, 1989] Clifford networks perform better. Clifford networks are then compared with real valued feed-forward networks, and it is found that in this task Clifford networks perform worse, and an analysis is then offered as to why this is the case.

5.2 Modulation and Demodulation

In any communication environment where more than one signal has to be transmitted at any time, or where high power signals are required for long distance transmission, some form of modulation is required.

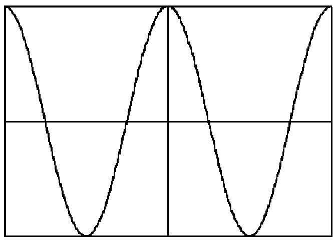


Figure 9: Modulating signal

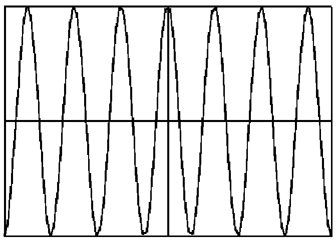


Figure 10: Carrier signal

A modulation scheme takes some information signal $f_m(t)$ and uses this to modulate various parameters of a carrier signal:

$$f_c(t) = E_c \cos(\omega_c t + \phi) \tag{151}$$

There are three basic modulation schemes: amplitude modulation, which varies the parameter E_c, frequency modulation which varies ω_c and phase modulation which varies ϕ. Figures 11, 12, 13 show the effects of modulating the signal in Figure 9.

These schemes are designed for analogue signals. When digital signals are modulated more efficient schemes can be implemented, such as MSK and $BPSK$ (see [Pearson, 1992; Tomasi, 1987] for details). This chapter uses neural networks to demodulate Quadrature-Quadrature phase modulation (Q^{2}PSK) signals which is an

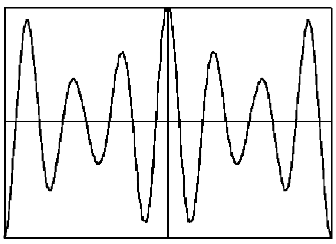


Figure 11: Amplitude modulation

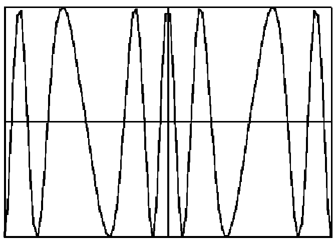


Figure 12: Frequency modulation

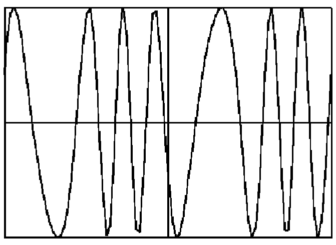


Figure 13: Phase modulation

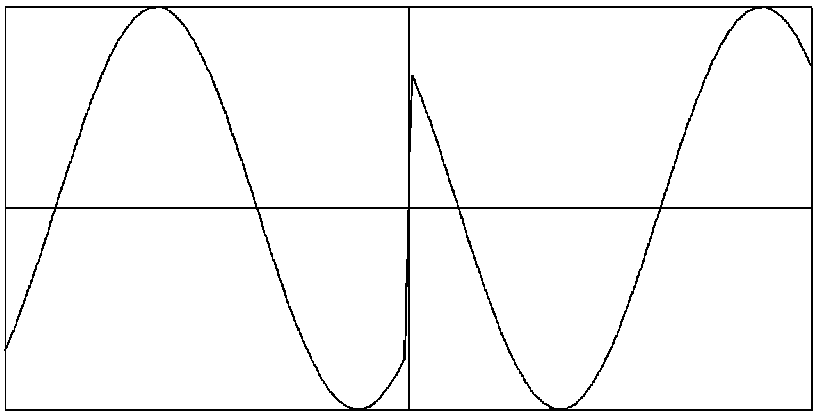


Figure 14: Modulated signal for input 1.0, -1.0 , -1.0 , +1.0

extension to Quadrature phase modulation.

5.3 Q^{2}PSK

Quadrature phase modulation (QPSK) is a two dimensional signal modulation scheme. A binary signal $\{\alpha_k\}$ with $a_k \in \{-1, 1\}$ is split into two parallel signals, a_1 and a_2 each taking alternate bits of the signal, there will be 2 bits per a cycle in QPSK. These are multiplied by two orthogonal carrier signals

$$s_1(t) = \sin(\omega_c t) \tag{152}$$

$$s_2(t) = \cos(\omega_c t) \tag{153}$$

and summed to produce the modulated signal.

Figure 15 shows the modulation scheme and Figure 14 shows two cycles of a typical modulated signal. At reception two components y_1 and y_2 can be extracted

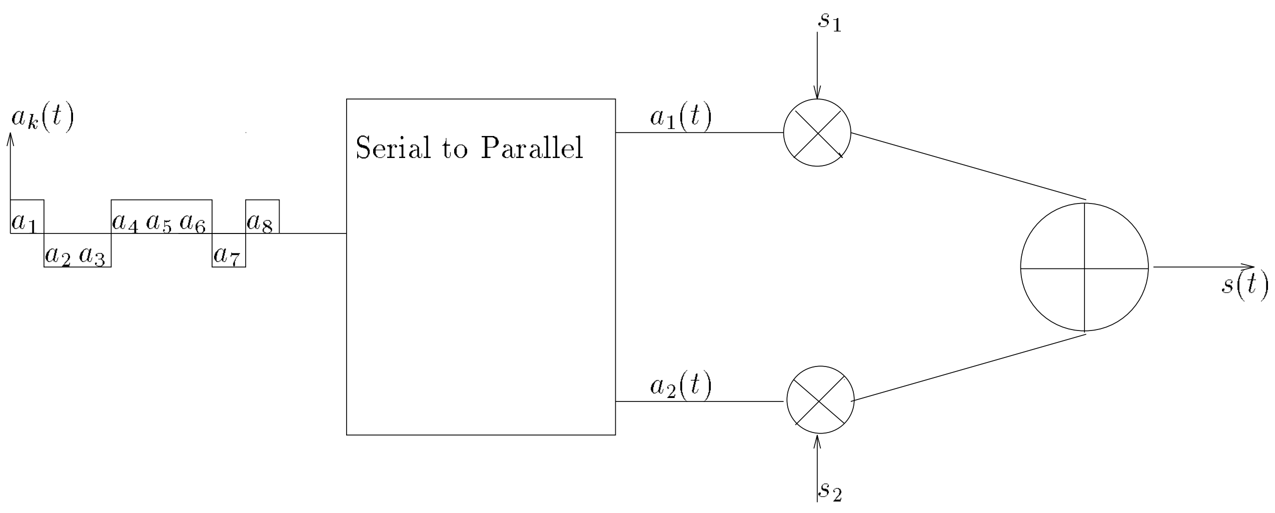


Figure 15: QPSK modulation scheme

from the received signal $y(t)$ by setting:

$$y_1(t) = y(t)s_1(t) \tag{154}$$

$$y_2(t) = y(t)s_2(t) \tag{155}$$

The task of the demodulator is to estimate the values of a_1 and a_2 from y_1 and y_2.

In [Saha, 1983; Saha and Birdsall, 1986] an extension of QPSK, is proposed Q^{2}PSK , which utilizes four dimensions of the signal space. In [Saha, 1983] he shows that a channel with a bandwidth occupancy of $1/T$ has a four dimensional signal space. Q^{2}PSK takes the original signal a_k and splits it into four parallel signals; these are then multiplied by four orthogonal signals:

$$s_1(t) = \cos(\frac{\pi t}{2T}) \cos(2\pi f_c t) \tag{156}$$

$$s_2(t) = \sin(\frac{\pi t}{2T}) \cos(2\pi f_c t) \tag{157}$$

$$s_3(t) = \cos(\frac{\pi t}{2T}) \sin(2\pi f_c t) \tag{158}$$

$$s_4(t) = \sin(\frac{\pi t}{2T}) \sin(2\pi f_c t) \tag{159}$$

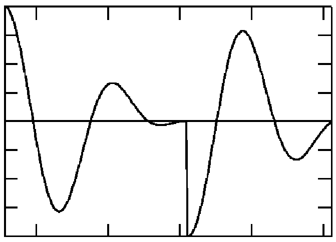


Figure 16: Modulated signal for input 1.0, -1.0 , -1.0 , -1.0 , -1.0 , 1.0 , -1.0 , -1.0

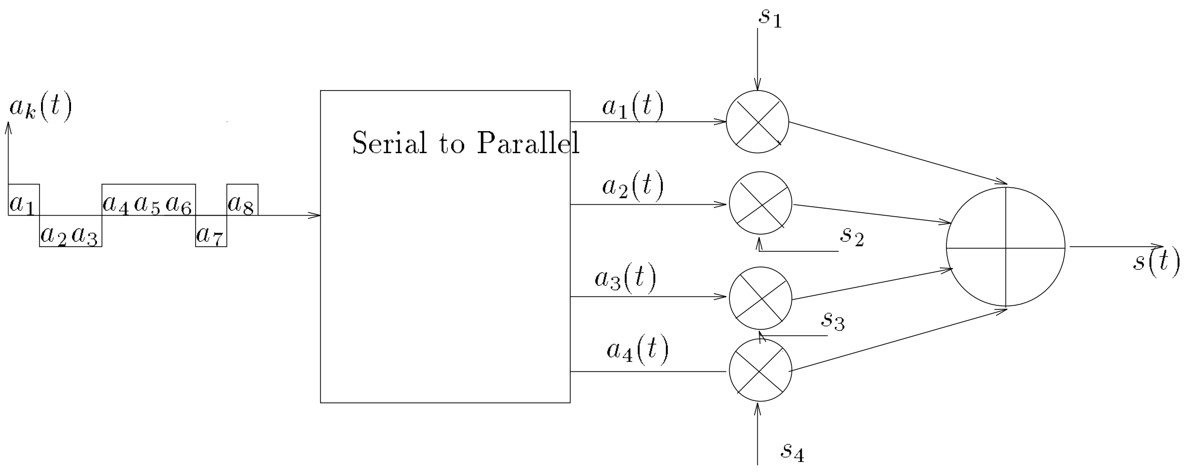


Figure 17: QPSK modulation scheme

and are summed to produce a signal s_{qqpsk}.

Figure 17 shows the modulation scheme and Figure 16 shows two cycles of a typical modulated signal.

5.4 Demodulation by maximum likelihood detection

During transmission the signal $s(t)$ will be degraded by noise and be affected by the medium's transmission characteristics. Throughout the work a linear channel model has been assumed and that the noise has a Gaussian characteristic. On receiving a signal $y(t)$ four components $y_1, \ldots y_4$ can be extracted by a similar scheme to QPSK:

$$y_1(t) = y(t)s_1(t) = \sum_n a_{1_n} \cos^2(\frac{\pi t}{T_0}) + \frac{\phi(t)}{4} \tag{160}$$

$$y_2(t) = y(t)s_2(t) = \sum_n a_{2_n} \cos^2(\frac{\pi(t-T)}{T_0}) + \frac{\phi(t)}{4} \tag{161}$$

$$y_3(t) = y(t)s_3(t) = \sum_n a_{3_n} \sin^2(\frac{\pi t}{T_0}) + \frac{\phi(t)}{4} \tag{162}$$

$$y_4(t) = y(t)s_4(t) = \sum_n a_{4_n} \sin^2(\frac{\pi(t-T)}{T_0}) + \frac{\phi(t)}{4} \tag{163}$$

where $\phi(t)$ is the noise term and T_0 is the basis length of the $\cos^2$ and $\sin^2$ terms.

Again the demodulator has to be able to find the most likely values (in a statistical sense) of $a_1 \ldots a_4$. Various schemes exist for demodulating modulated signals such as the modified Viterbi algorithm [Ungerboeck, 1966] for instance. These schemes are basically implementations of the maximum likelihood analysis. Unfortunately, these schemes are tedious to implement and have to be rederived for each different modulation scheme. This chapter uses neural networks to implement demodulators, by treating the problem as a pattern recognition problem.

5.5 Clifford Networks as maximum likelihood detectors

The task of a Clifford network, given various values of $y_1, \ldots y_4$, is to estimate the most likely values of $a_1, \ldots a_4$. This can be seen as a pattern recognition problem, certain values of $y_1, \ldots y_4$ are identified by values of $a_1, \ldots a_4$. Once trained, the network should be able to recognize noisy versions of $y_1 \ldots y_4$.

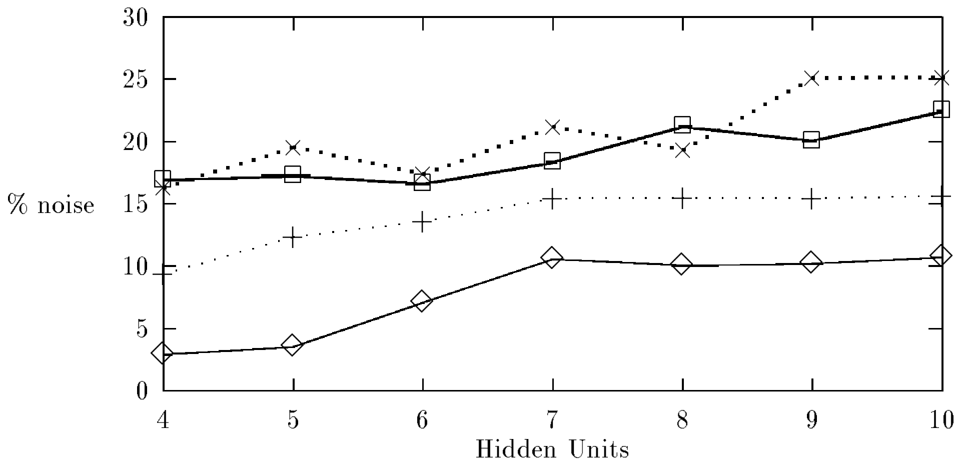
A Quaternion valued network is used. The inputs are sampled values of $y_1, \ldots y_4$ over a time interval, the hidden layer contains a number of hidden units, and there are four output units representing the values of $a_1, \ldots a_4$. Table 6 and 7 shows typical training data.

After training the networks are tested on random 4000 bit length sequences, and a program determines the maximum amount of noise for which the network can still decode perfectly the 1000 element sequence. Figure 18 shows the performance of the network for various numbers of hidden units and input nodes. As might be expected increasing the number of inputs increases the performance of the network and also increasing the number of hidden nodes increases the performance.

5.6 Comparison with Real valued networks

The above experiments were tried with real valued networks. For comparison the real networks had four times the number of hidden units and four times the number of input units in the Clifford networks (because the Clifford signals are four dimensional in this case).

The results are summarized below in Figure 19. As can be seen, the real networks



Key: 2 Inputs $\diamond$, 3 Inputs $+$, 4 Inputs $\diamond$, 5 Inputs $\times$

Figure 18: Noise tolerance levels

perform consistently better than the Clifford valued networks. There are two reasons for this. First, the four signal values are modulated with the four carrier waves and then they are demodulated in a linear way, which means that there appears to be no advantage in keeping the four demodulated components together as one Clifford value. Secondly, a different process is going on inside the Clifford valued network to the real valued network: a real valued network during learning is essentially finding separating hyper-planes to classify the input data. A Clifford network is performing a very different task. If the quaternions are identified with the vector space $\mathcal{R}^4$ the operation:

$$\tau(x) = \omega x \tag{164}$$

can be seen as a linear transformation, since

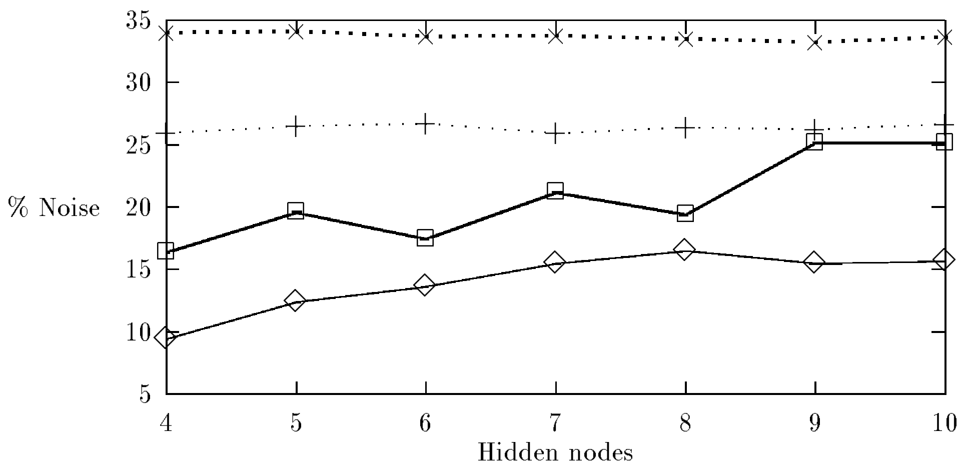
$$\tau(\alpha x + y) = \alpha\tau(x) + \tau(y) \text{ for } \quad \alpha \in \mathcal{R} \tag{165}$$

and the set of quaternions of length 1 are the set of rotations in $\mathcal{R}^4$. If during training a certain quaternion-valued neuron is required to produce a certain output for a certain input, then the training process is finding a set of weight vectors that will rotate the input patterns to produce the required output, as in Figure 20. Thus it seems in this case that once a set of weight vectors is found that will rotate the input to produce the correct output, if the components of the input patterns are independent then the networks are less tolerant to noise.

5.7 Conclusion

It seems that for Q^{2}PSK demodulation Clifford Networks are inappropriate. When the four components of the Q^{2}PSK signal are modulated and then demodulated any degradation caused from interference during transmission, is linear in each component (see (160 - 163)); the author conjectures that this is the reason why real networks outperform Clifford Networks. One of the motivations for studying Clifford Networks is that when related components of multi-dimensional signal are brought together as a single Clifford Number, a more compact solution to the problem should be found using Clifford Networks. In the case of Q^{2}PSK it seems that the four components are not related enough for any benefit to be seen with Clifford Networks. It is however possible that a more complicated non-linear modulation scheme might yield better results using Clifford Networks.

The question is then, what class of problem will benefit from the application of Clifford Networks? Clifford algebras have many applications in mathematical physics, such are motion modeling using the quaternions. By applying Clifford Networks to these types of problems which can be analyzed using Clifford Algebras (see [Chisholm and Common, 1986]) the author hopes that Clifford Networks will provide a significant benefit.



Key: 3 Input Clifford $\diamond$, 12 Input Real $+$, 5 Input Clifford $\square$, 20 Input Real $\times$

Figure 19: Comparison between Real and Clifford valued networks

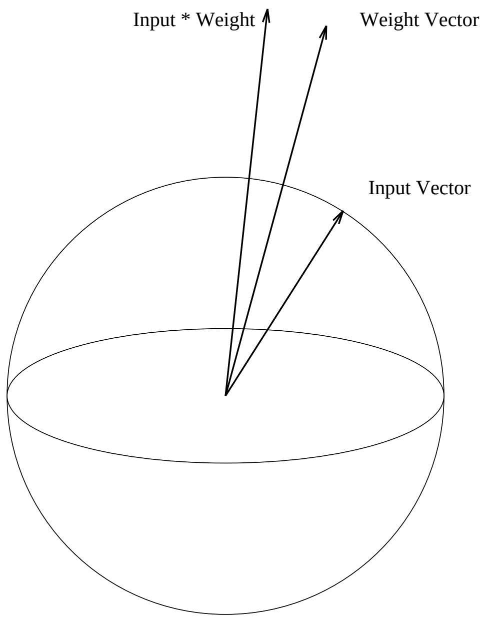


Figure 20: Weight multiplication inside a Clifford neuron

Inputs for output pattern -1 -1 -1 -1
(-1, 1, -0, 0) (-0.853553, 0.853553, -0.146447, 0.146447) (-0.5, 0.5, -0.5, 0.5) (-0.146447, 0.146447, -0.853553, 0.853553)
Inputs for output pattern 1 -1 -1 -1
(1, 1, -0, 0) (0.853553, 0.853553, -0.146447, 0.146447) (0.5, 0.5, -0.5, 0.5) (0.146447, 0.146447, -0.853553, 0.853553)
Inputs for output pattern -1 1 -1 -1
(-1, -1, -0, 0) (-0.853553, -0.853553, -0.146447, 0.146447) (-0.5, -0.5, -0.5, 0.5) (-0.146447, -0.146447, -0.853553, 0.853553)
Inputs for output pattern 1 1 -1 -1
(1, -1, -0, 0) (0.853553, -0.853553, -0.146447, 0.146447) (0.5, -0.5, -0.5, 0.5) (0.146447, -0.146447, -0.853553, 0.853553)
Inputs for output pattern -1 -1 1 -1
(-1, 1, 0, 0) (-0.853553, 0.853553, 0.146447, 0.146447) (-0.5, 0.5, 0.5, 0.5) (-0.146447, 0.146447, 0.853553, 0.853553)
Inputs for output pattern 1 -1 1 -1
(1, 1, 0, 0) (0.853553, 0.853553, 0.146447, 0.146447) (0.5, 0.5, 0.5, 0.5) (0.146447, 0.146447, 0.853553, 0.853553)
Inputs for output pattern -1 1 1 -1
(-1, -1, 0, 0) (-0.853553, -0.853553, 0.146447, 0.146447) (-0.5, -0.5, 0.5, 0.5) (-0.146447, -0.146447, 0.853553, 0.853553)
Input for pattern 1 1 1 -1
(1, -1, 0, 0) (0.853553, -0.853553, 0.146447, 0.146447) (0.5, -0.5, 0.5, 0.5) (0.146447, -0.146447, 0.853553, 0.853553)
Input for pattern -1 -1 -1 1
(-1, 1, -0, -0) (-0.853553, 0.853553, -0.146447, -0.146447) (-0.5, 0.5, -0.5, -0.5) (-0.146447, 0.146447, -0.853553, -0.853553)
Input for pattern 1 -1 -1 1
(1, 1, -0, -0) (0.853553, 0.853553, -0.146447, -0.146447) (0.5, 0.5, -0.5, -0.5) (0.146447, 0.146447, -0.853553, -0.853553)

Table 6: Training data: For the modulated signal samped at 4 time intervals

Input for pattern -1 1 -1 1
(-1, -1, -0, -0) (-0.853553, -0.853553, -0.146447, -0.146447)
(-0.5, -0.5, -0.5, -0.5) (-0.146447, -0.146447, -0.853553, -0.853553)
Input for pattern 1 1 -1 1
(1, -1, -0, -0) (0.853553, -0.853553, -0.146447, -0.146447)
(0.5, -0.5, -0.5, -0.5) (0.146447, -0.146447, -0.853553, -0.853553)
Input for pattern -1 -1 1 1
(-1, 1, 0, -0) (-0.853553, 0.853553, 0.146447, -0.146447)
(-0.5, 0.5, 0.5, -0.5) (-0.146447, 0.146447, 0.853553, -0.853553)
Inputs for output pattern 1 -1 1 1
(1, 1, 0, -0) (0.853553, 0.853553, 0.146447, -0.146447)
(0.5, 0.5, 0.5, -0.5) (0.146447, 0.146447, 0.853553, -0.853553)
Inputs for output pattern -1 1 1 1
(-1, -1, 0, -0) (-0.853553, -0.853553, 0.146447, -0.146447)
(-0.5, -0.5, 0.5, -0.5) (-0.146447, -0.146447, 0.853553, -0.853553)
Inputs for output pattern 1 1 1 1
(1, -1, 0, -0) (0.853553, -0.853553, 0.146447, -0.146447)
(0.5, -0.5, 0.5, -0.5) (0.146447, -0.146447, 0.853553, -0.853553)

Table 7: Training data continued

Chapter 6

The Generalized Perceptron and its Properties

6.1 Introduction

This chapter introduces the generalized Clifford valued perceptron, which is an extension of the multi-valued Perceptron in [Georgiou, 1993]. The basic idea is to extend the ordinary real valued Perceptron, which has the output values $+1$ and -1, to a Perceptron which has output values taken from the unit sphere in various dimensions. The complex valued Perceptron was first discussed in [Georgiou, 1993], together with a proof of the convergence theorem.

A slightly different proof of the convergence theorem is presented in this chapter, which emphasises, how the inner product on the complex vector space interacts with the group structure of the output values of the Perceptron. With this new viewpoint extension to the multi-dimensional Clifford case is then possible. The chapter then ends with a discussion of the classification powers of complex and Clifford Perceptrons.

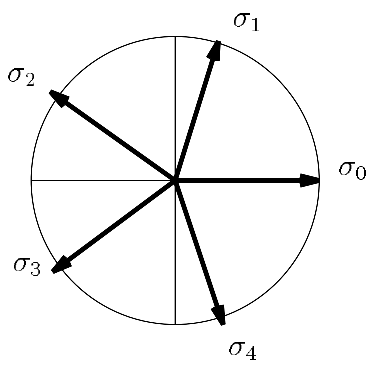


Figure 21: Roots of unity

6.2 The Complex Perceptron

In [Georgiou, 1993] two Perceptrons are defined: the continuous and the multi-valued Perceptron. In both cases each Perceptron has n inputs $(x_1, \ldots, x_n)$ n weights $(w_1 \ldots w_n)$ and one threshold value w_{n+1}. First the weighted sum of the inputs and the threshold is computed:

$$net = \sum_{k=0}^{n} w_k x_k + w_{n+1} \tag{166}$$

The output of the q-state multi-valued Perceptron is taken from the set:

$$\{\sigma_k : \sigma_k = e^{\frac{2\pi ik}{q}}, k = 1 \ldots q-1\} \tag{167}$$

These are the q roots of unity and are distributed around the unit circle as in Figure 21.

The output for a certain weighted input net is the value σ_j such that:

$$\sigma_j = |\arg(net) - j\frac{2\pi}{q}| < \frac{\pi}{q} \tag{168}$$

($\arg(z)$ is the complex argument of z). The output is essentially the closest σ_j to the unit normalized value of net.

The continuous-valued Perceptron just outputs the normalized value $net/|net|$.

6.3 The Group Structure of the Outputs

The outputs of the q-state multi-valued Perceptron and the continuous valued Perceptron have a group theoretical structure. Roots of unity can be multiplied to produce again a root of unity, that is for any σ_k and σ_j the product:

$$\sigma_k \sigma_j = e^{\frac{2\pi i k}{q}} e^{\frac{2\pi i j}{q}} = e^{\frac{2\pi i (j+k)}{q}} = \sigma_{k+j} \tag{169}$$

The inverse of an element σ_k is simply σ_{-k}. This group under multiplication is isomorphic to $(Z_q, +)$ (the integers modulo q under addition). What will become important in the proof of the convergence theorem is that:

$$(\sigma_k)^{-1} = \overline{\sigma_k} = \sigma_{-k} \tag{170}$$

where $\overline{z}$ is the complex conjugate of z.

The outputs of the continuous valued Perceptron form the group $U(1)$, since each output can be represented as the complex number $e^{i\theta}$, with θ real. The product of two elements $e^{i\theta_1}$ and $e^{i\theta_2}$ is simply $e^{i(\theta_1+\theta_2)}$ and the inverse of an element $e^{i\theta}$ is simply $e^{-i\theta}$.

6.4 Real Hilbert Spaces on Complex Vector Spaces

Given the complex numbers $\mathcal{C}$ a real Hilbert space structure can be put on them. That is a real inner product $(\cdot|\cdot)$ satisfying the axioms of a Hilbert space. Given $x, y \in \mathcal{C}$ define:

$$(x|y) = \Re(x\overline{y}) \tag{171}$$

w here $\overline{y}$ is the complex conjugate of y and $\Re(z)$ is the real part of the complex number x. In particular the norm of a complex number can be defined which corresponds to the standard norm on $\mathcal{C}$:

$$|x| = \sqrt{(x|x)} \tag{172}$$

The Cauchy-Schwartz inequality is then satisfied:

$$|(x|y)| \leq |x||y| \tag{173}$$

This construction can be extended from $\mathcal{C}$ to $\mathcal{C}^{n+1}$, given $X = (x_j)_{i=0}^{n}$ and $Y = (y_j)_{i=0}^{n}$ define the inner product as:

$$(X|Y) = \Re(\sum_{j=0}^{n} x_j \overline{y}_j) \tag{174}$$

A norm can be defined in the same way:

$$\|x\| = \sqrt{(x|x)} \tag{175}$$

Again the Cauchy-Schwartz is satisfied.

It is possible to put a Complex Hilbert space structure on $\mathcal{C}^n$ as in [Georgiou, 1993] but the author found it more convenient to work with a Real Hilbert Space in the proof of the Perceptron Convergence Theorem. What is important for the Perceptron convergence theorem which follows, is how the inner product structure interacts with with group structure of the outputs of the Complex Perceptron.

6.5 The Complex Convergence Theorem

In what follows a n element pattern will be represented as an element of $\mathcal{C}^{n+1}$ as:

$$X = (x_0, x_2, \ldots x_{n-1}, 1) \tag{176}$$

A weight and threshold value is represented as:

$$W = (w_0, w_2, \ldots w_{n-1}, \theta = w_n) \tag{177}$$

Thus the net input of the network can be calculated as:

$$net = X \cdot \overline{W} = \sum_{k=0}^{n} x_i w_i \tag{178}$$

where for two complex numbers X and Y, $X \cdot Y$ is the complex inner product $\sum_{k=0}^{n} x_k \overline{y}_k$.

Before stating and proving the convergence theorem, an error measure has to be defined. For a finite pattern set $(X_k)_{k=0}^{m}$ and an expected output set $(Y_k)_{k=0}^{m}$ (which will later be called a classification of the pattern set) where each Y_k is a possible output value of the type of Perceptron in question. The error for a particular pattern presentation j is defined as:

$$\epsilon_j = Y_j - \Theta(W_j \cdot \overline{X}_j) \tag{179}$$

where W_j is the current weight set and Θ is the output of the Perceptron for the net input $W_j \cdot \overline{X}_j$.

Theorem 6 *For the Continuous Valued Complex Perceptron, starting from a random initialized weight W_1, if each pattern is repeatedly presented to the Perceptron from a finite pattern set and there exists a weight vector W' which correctly classifies the pattern set, then at stage k of the procedure if the weight vector W_k is modified by:*

$$W_{k+1} = W_k + \epsilon_k \overline{X}_k \tag{180}$$

then the procedure will terminate in a finite number of steps yielding a weight vector which correctly classifies the pattern set.

Proof: At stage k the weight vector will be:

$$W_{k+1} = W_1 + \epsilon_1 \overline{X}_1 + \epsilon_2 \overline{X}_2 + \ldots \epsilon_k \overline{X}_k \tag{181}$$

Forming the inner product $(W'|W_{k+1})$ the above equation gives:

$$(W'|W_{k+1}) = (W'|W_1) + (W'|\epsilon_1\overline{X}_1) + \ldots (W'|\epsilon_k\overline{X}_k) \tag{182}$$

Without loss of generality it can be assumed that at each stage ϵ_j is non zero. Define:

$$\delta = \min_{1\leq j\leq k} (W'|\epsilon_j\overline{X}_j) \tag{183}$$

The δ can be positive or negative and is essentially the same δ as in section 11.2 in [Minsky and Papert, 1969]. This means that:

$$(W'|W_{k+1}) \geq (W'|W_1) + k\delta \tag{184}$$

Another inequality is needed to complete the proof.

$$\|W_{j+1}\|^2 = (W_j + \epsilon_j\overline{X}_j|W_j + \epsilon_j\overline{X}_j) = \|W_j\|^2 + \|\epsilon_j\overline{X}_j\|^2 + 2(W_j|\epsilon_j\overline{X}_j) \tag{185}$$

Now lemma 2 implies that $(W_j|\epsilon_j\overline{X}_j)$ is negative, so defining:

$$Q = \max_{1\leq j\leq k} \|\epsilon_j\overline{X}_j\|^2 \tag{186}$$

gives:

$$\|W_{k+1}\|^2 \leq \|W_1\|^2 + Qk \tag{187}$$

From the Cauchy-Schwarz inequality:

$$\frac{(W'|W_{k+1})^2}{\|W'\|^2\|W_{k+1}\|^2} \leq 1 \tag{188}$$

Now combining the inequality 188 with 187 and 184 the following is obtained:

$$\frac{((W'|W_1) + k\delta)^2}{\|W'\|^2(\|W_1\|^2 + Qk)} \leq 1 \tag{189}$$

If k was unbounded the right hand side of the above inequation would grow with bound contradicting the inequation (189) hence k must be finite. $\square$

Lemma 2 *For the Continuous Complex Valued Perceptron, if W_j mis-classifies X_j then:*

$$(\epsilon_j \overline{X}_j | W_j) < 0 \tag{190}$$

Proof: From the definition of ϵ:

$$\epsilon_j = \sigma_k - \sigma_j \tag{191}$$

Where σ_k is the required output for the current pattern and σ_j is the actual output. The inner product is:

$$(\epsilon_j \overline{X}_j | W_j) = \Re(\epsilon \overline{X_j W_j}) \tag{192}$$

The product $W_j X_j$ can be rewritten as $a\sigma_j$ for some positive real constant a. So the inner product expression becomes:

$$\Re(\epsilon \overline{W_j X_j}) = \Re((\sigma_k - \sigma_j)(a\overline{\sigma_j})) \tag{193}$$

Since $z\overline{z} = 1$ for all complex z, the above equation reduces to:

$$a(\Re(\sigma_k \overline{\sigma_j}) - 1) \tag{194}$$

Because both σ_k and $\overline{\sigma_j}$ are unit vectors, each component will have a value less than or equal to one. In the case where the real components are less than one then (194) will be negative and hence the lemma proved. Otherwise both real components can not be equal to one, since this would make e_j zero and the lemma would not apply, therefore (194) will negative. □

In examining the proof of the convergence theorem the only step that actually requires the Perceptron to be continuous is Lemma (2). So to prove the convergence theorem for the discrete q-state multi-valued Perceptron a new lemma is required.

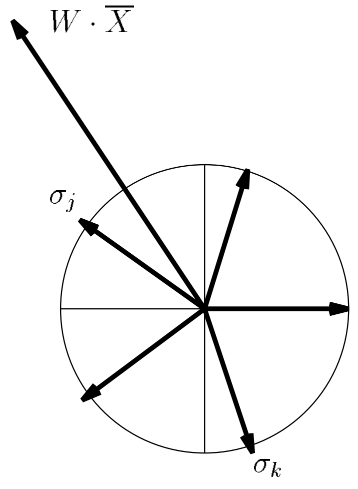


Figure 22: Net input $W \cdot \overline{X}$, actual output of Perceptron σ_j and expected output σ_k

Lemma 3 *For the q-state multi-valued complex Perceptron* $(\epsilon_j \overline{X}_j | W_j)$ *is negative whenever* W_j *mis-classifies* X_j*.*

Proof: As before $\epsilon_j = \sigma_k - \sigma_j$, but $W_j X_j$ is no longer equivalent to $a\sigma_j$. Since σ_j is the actual output, we know that $(W_j X_j)/\|W_j X_j\|$ must be within a certain range of σ_j see Figure 22. In fact:

$$\frac{W_j X_j}{\|W_j X_j\|} = e^{i\delta}\sigma_j \tag{195}$$

Where:

$$|\delta| \leq \frac{\pi}{q} \tag{196}$$

Expanding the inner product gives:

$$(\epsilon_j \overline{X}_j | W_j) = \Re((\sigma_k - \sigma_j)(a\overline{e^{i\delta}\sigma_j})) = (\sigma_k | e^{i\delta}\sigma_j) - (e^{i\delta}|1) \tag{197}$$

The inner product gives the cosine of the angle between two complex numbers considered as vectors in $\mathcal{R}^2$. So the above equation becomes:

$$(\epsilon_j \overline{X}_j | W_j) = \cos(2\pi \frac{(k-j)}{q} + \delta) - \cos(\delta) \tag{198}$$

Since j and k are integers and $|\delta| < \pi/q$ the above expression will be negative. $\square$

6.6 The Clifford Case

It is possible for a Euclidean Clifford algebra (an algebra with the signature $0, n$) to define both continuous and multi-valued Perceptrons. Before they can be defined some extra definitions are needed; both can be found in Sections 1.8 - 1.10 of [Brackx *et al.*, 1982].

In any Clifford algebra it is possible to define two involutions (see [Porteous, 1981] page 252) which are independent of the basis chosen. The *main involution*, also called inversion, is defined for any:

$$x = \sum_A x_A e_A \tag{199}$$

as,

$$x^* = \sum_A x_a (-1)^{\#(A)} e_A \tag{200}$$

($\#(A)$ is the size of the set A). For example given:

$$x = x_1 e_1 + x_2 e_{12}$$

then,

$$x^* = x_2 e_{12} - x_1 e_1$$

The second involution called *reversion* given by:

$$x^\dagger = \sum_A (-1)^{\frac{(\#(A)-1)\#(A)}{2}} x_A e_A \tag{201}$$

which on a particular basis element $e_{h_1 h_2 \ldots h_m}$ gives:

$$e_{h_m h_{m-1} \ldots h_1} \tag{202}$$

Thus for example, if

$$x = x_1 e_{12} + x_2 e_{123}$$

then,

$$x^\dagger = -(x_1 e_{12} + x_2 e_{123})$$

These can be combined to produce the following involution:

$$\overline{x} = \sum_A x_A (e_A^*)^\dagger = \sum_A (-1)^{\frac{\#(A)(\#(A)+1)}{2}} x_A e_A \tag{203}$$

This is analogous to complex conjugation. An important property is:

$$\overline{xy} = \overline{y}\,\overline{x} \tag{204}$$

In a Euclidean algebra it is possible to use the above involution to define a real inner product which gives rise to a Hilbert space structure:

$$(x|y) = [x\overline{y}]_0 = \sum_A x_A y_A \tag{205}$$

This inner product has many useful properties including:

$$(x|y) = (y|x) = (\overline{x}|\overline{y}) = (\overline{y}|\overline{x}) \tag{206}$$

It gives rise to a norm:

$$|x| = \sqrt{(x|x)} \tag{207}$$

also

$$|xy| \leq |x||y| \tag{208}$$

Also because it forms a real Hilbert space the Cauchy-Schwarz inequality is satisfied:

$$(x|y)^2 \leq |x|^2|y|^2 \tag{209}$$

As before the inner product extends from a Euclidean algebra $\mathcal{A}$ to the Clifford Module $\mathcal{A}^n$. It is possible to put a Hilbert Module (see Chapter 15 in [Wegge-Olsen, 1993]) structure on $\mathcal{A}^n$ where the inner product has values in the algebra $\mathcal{A}$, but again it is easier to work with the Real Hilbert Space in the proof of the Convergence Theorem.

So given a Clifford algebra $\mathcal{A}$ it is possible to define the Clifford sphere as:

$$S_{\mathcal{A}} = \{x : |x| = 1\} \tag{210}$$

A Clifford valued Perceptron is then easy to define. The continuous case is as before, given the net input:

$$net = W \cdot \overline{X} \tag{211}$$

The output is:

$$\frac{W \cdot \overline{X}}{|W \cdot \overline{X}|} \tag{212}$$

The discrete multi-valued case involved picking q^{2^n} (2^n the dimension of the Clifford algebra) points evenly distributed around the Clifford sphere. The output for a particular net input $W \cdot \overline{X}$ is the closest output vector (in the topology defined by the norm) to the value $\frac{W \cdot \overline{X}}{|W \cdot \overline{X}|}$. To implement this various schemes could be used; probably the most efficient would be to convert the representation of the net input from Cartesian coordinates to spherical polar coordinates and find the closest discrete value in each polar coordinate separately.

Unfortunately in general the outputs of the continuous valued Clifford and the Discrete multi-valued Clifford Perceptron do not form a group structure. Luckily all is not lost. In the complex case, what was important was how the group structure interacted with the inner product. Fortunately as shall be seen in the proofs of the convergence theorem the output elements do interact with the inner product in a group-like way.

Theorem 7 *For the Continuous Clifford valued Perceptron given a finite pattern set (X_i) with a classification (Y_i), if there exists a weight vector W' which classifies the pattern set correctly then the following procedure will converge in a finite number of steps:*

$$W_{k+1} = W_k + \epsilon_k \overline{X}_k \tag{213}$$

Proof: Looking back at the complex case, it is found that the non-commutative nature of the Clifford algebras will not affect the proof, because this is absorbed in the first line of the proof (equation (181)). Again what is required is to replace (2) by the following lemma 4 and the proof will work. □

Lemma 4 *For the Continuous Clifford Valued Perceptron, if W_j mis-classifies X_j then:*

$$(\epsilon_j \overline{X}_j | W_j) < 0 \tag{214}$$

Proof: From the definition of the inner product the above equation can be rewritten as:

$$(\epsilon_j \overline{X}_j | W_j) = [\epsilon_j \overline{X}_j \overline{W}_j]_0 \tag{215}$$

Using the properties of the involution the above equation becomes:

$$[\epsilon_j \overline{W_j X_j}]_0 = [(\sigma_k - \sigma_j)(a\overline{\sigma_j})]_0 \tag{216}$$

With σ_j, σ_k and a as before in lemma 181. Expanding the brackets gives:

$$a((\sigma_k|\sigma_j) - (\sigma_j|\sigma_j)) \tag{217}$$

In the complex case the fact that $\sigma_j\overline{\sigma_j} = 1$ was used, but because the Clifford elements do not form a group, this fact can not be used. However since $|\sigma_j| = 1$, the equation becomes:

$$a((\sigma_k|\sigma_j) - 1) = a\left(\sum_A \sigma_{k_A}\sigma_{j_A} - 1\right) \tag{218}$$

But since each component of σ_j and σ_k is less than or equal to 1, for the same reasons as in Lemma 2, the above equation will always be negative, hence the lemma is proved. □

The following proof will come as no surprise to anybody who has followed this chapter so far.

Theorem 8 *Theorem 7 is valid for the discrete valued Perceptron.*

Proof: To prove the theorem for the discrete multi-valued Perceptron again lemma 4 needs to be replaced by a new lemma 5 □

Lemma 5 *For the Discrete Clifford Valued Perceptron, if W_j mis-classifies X_j then:*

$$(\epsilon_j\overline{X}_j|W_j) < 0 \tag{219}$$

Proof: As in the discrete case $W_j \cdot X_j$ is not equal to $a\sigma_j$, but will be equal to $a\tilde{\sigma_j}$ where $\tilde{\sigma_j}$ is a unit vector and a is a positive constant. So expanding out the inner product the following is obtained:

$$(\epsilon_j\overline{X}_j|W_j) = a[(\sigma_k - \sigma_j)\overline{\tilde{\sigma_j}} = (\sigma_k|\tilde{\sigma_j}) - (\sigma_j|\tilde{\sigma_j}) \tag{220}$$

We know that $(\sigma_j|\tilde{\sigma_j}) < \pi/q$, thus for the same reasons as in the complex case the equation will be negative and hence the lemma proved. □

It seems, if we look at these proofs, that the minimal mathematical structure required for convergence is a C^* algebra[1] and a suitable Perceptron activation function. This area could be explored to find a more general Class of Perceptrons and perhaps there would be a relation between various structure theorems of C^*-Algebras (such as K-Theory [Wegge-Olsen, 1993]) and pattern recognition capabilities of Perceptrons [Minsky and Papert, 1969].

6.7 What are Complex Perceptrons Doing?

The Classical Real Valued Perceptron is essentially separating hyper-planes. That is given a n input Perceptron with weight values $w_1 \ldots w_n, \theta$ and inputs $x_1 \ldots x_n$, the output of the Perceptron is either $+1$ or -1 depending if $w \cdot x + \theta$ is greater than zero or less. So the set of points classified by $+1$ is separated by the points that are classified by -1 by a hyper-plane. Then the convergence algorithm for real Perceptron can be restated as: if the the two classes are linearly separable then the algorithm will find a weight set which separates the two classes[2].

The complex valued Perceptron is more complicated. For a single input q-valued Perceptron with no threshold value, the net input to the activation function is simply:

$$net = wx \tag{222}$$

(where w is the weight and x the input). Using polar representation of the complex

[1]A C^* algebra is a Banach algebra with an operation $*$ such that:

$$||x|| = ||xx^*|| = ||x^*x|| \tag{221}$$

See [Sakai, 1971; Wegge-Olsen, 1993] for details.

[2]But, it can do more, it will converge faster than a simpleminded approach such as Homeostats, see [Minsky and Papert, 1969] page 180

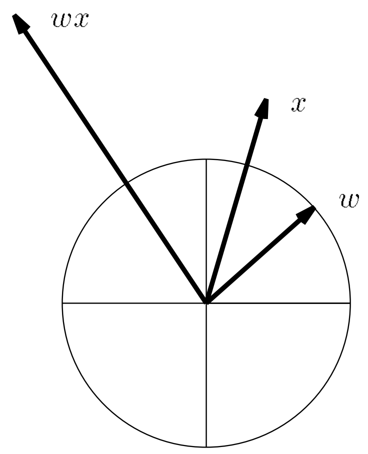


Figure 23: Complex Multiplication

number ($w = |w|e^{i\theta_w}$, $x = |x|e^{i\theta_x}$) the net input becomes:

$$net = |w|e^{i\theta_w}|x|e^{i\theta_x} = |x||w|e^{i(\theta_w+\theta_x)} \tag{223}$$

So multiplication of two complex numbers results in a product, whose length is the product of the two multiplicands and argument which is the sum of the arguments (see Figure 23). When passed through the activation function , the argument $\theta_x + \theta_w$ is set to the closest value in the output set[3].

For a given weight w, *activation regions* in the input space can be defined as the sets of points which produce different outputs $\sigma_1 \ldots \sigma_{q-1}$. Figure 24 shows typical activation regions. Formally the activation region with respect to a weight set $w_1 \ldots w_{n+1}$, for an output value σ_j is defined as the set of points in the input space such that:

$$h(\sum_i w_i x_i) = \sigma_j \tag{224}$$

[3]This means that the modulus of both the weight and input vectors is unimportant; only the direction is significant.

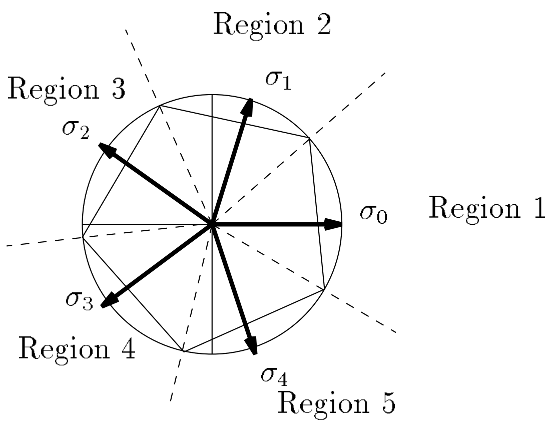


Figure 24: Activation regions for $q = 5$

where h is the output of the continuous or multi-valued Perceptron. This definition could be adapted to any class of neural networks, but in general, because of the non-linear nature of the networks, the regions would be too complicated to calculate or reason about.

The effect of changing w simply rotates the activation regions around the axis and adding a threshold translates the axis in the complex plane.

So when training a single input complex network, the learning procedure is finding a weight value which will rotate the input around to produce the required output.

The question now to ask is, what classifications are possible? Given a q-state neuron and a n element pattern set, is it possible for the neuron to represent all possible classifications.[4] For certain sets of points, it is possible to represent a large number of classifications. For example in Figure 25, by rotating the axis (changing the weight value) different classifications can be achieved by moving the activation

[4]A generalized VC dimension could be defined, as the largest possible arbitrary set for which ${}^{n}C_{q}$ classifications are possible (q = the order of the output group) , but the author conjectures this will equal the ordinary VC dimension of the neuron.

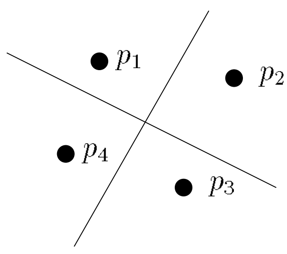


Figure 25: A possible classification

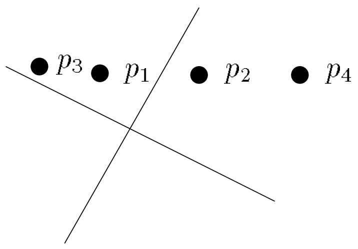


Figure 26: Co-linear points

regions.

But if for instance the points are colinear as Figure 26 then fewer classifications are possible. This implies that not all classifications are possible (even in Figure 25 the classification $\{p_1, p_2\} \rightarrow \sigma_0$, $p_3 \rightarrow \sigma_1$ and $p_4 \rightarrow \sigma_3$ is not in general achievable by rotation of the axis).

The reason for this lack of flexibility is that the activation regions are coupled together, so just moving one activation region independently of the others is not possible.

For the multiple input q-state Perceptron the activation regions are more complicated. Consider the two input case, with net input to the activation function:

$$net = w_1 x_1 + w_2 x_2 \tag{225}$$

For a fixed w_2 and x_2 the activation regions will be the same as in Figure 24. The effect of varying both x_2 and w_1 results in activation regions in $\mathcal{C}^2$, and in higher dimensions, making a multi-dimensional open 'pyramid'. Again the Classification is limited, because the activation regions cannot be manipulated independently.

The higher dimensional Clifford case has similar activation regions, but the single input neuron no longer takes sectors out of the unit circle, but out of the unit hypersphere.

6.8 Conclusion

A new class of neurons has been defined in this chapter, together with a learning algorithm which is guaranteed to converge for all learnable patterns. An analysis of the class of patterns learnable by a single generalized perceptron is given, and as with the real valued case, there are restrictions on the class of patterns learnable. The work of Hirose in [Hirose, 1992b] points to the possibility of a generalized multi-dimensional Hopfield type network, where the multi-dimensional nature of the computing unit would give a different attractor structure and possibly higher pattern storage capabilities.

The other value of Generalized Perceptrons is that, in the limit, the neurons considered in the previous chapters act like Generalized Perceptrons. Consider the activation function previously defined:

$$f(x) = \frac{x}{1 + |x|} \tag{226}$$

This maps a Clifford number considered as a vector in a 2^{p+q} dimensional space into the unit sphere preserving the orientation of the vector. Adding a real parameter λ

to the above equation,

$$f(\lambda x) = \frac{\lambda x}{1 + |\lambda||x|} \tag{227}$$

and letting λ tend to $+\infty$ an activation function resembling the Continuous Generalized Perceptron is obtained. In the same way if λ tends to $+\infty$ in the function:

$$\sigma(\lambda x) = \frac{1}{1 + exp(-\lambda x)} \tag{228}$$

used in many real valued feed-forward networks yields an activation function resembling the traditional Perceptron activation function. This chapter has defined a generalization of the Real Valued Perceptron and proved a general convergence result, which sheds new light on the real valued case, by the interaction between the inner product structure of the weight and pattern space and the group structure of the output of the neuron.

Chapter 7

Conclusion

This thesis has presented an extension to Complex Valued Networks [Georgiou and Koutsougeras, 1992] and Complex Valued Perceptrons [Georgiou, 1993]. It has been shown that it is possible to derive and implement training algorithms and networks that can be used on non trivial problems. The extension to Complex Perceptrons has been analyzed together with a proof of the convergence of the learning algorithm. This proof (in the complex case) is a different proof to the one found in [Georgiou, 1993] and even when restricted to the real case, throws new light on the Perceptron Convergence theorem, because of the emphasis on the group theoretical structure of the outputs of the neuron and in particular, how the group structure interacts with the inner product on the input space.

A version of the BEP algorithm has been successfully derived (Chapter 3) and shown to converge on test problems. The number of learning epochs for a multi-dimensional problem has been shown to be the same order as for the equivalent real valued problem, which seems to indicate that during learning comparable encoding strategies are being used for the hidden units.

The theorems proved in Chapter 4 show that Clifford-valued networks can represent non-trivial classes of functions. The final importance of Clifford networks is still unclear, although they can represent the same classes of functions as real valued networks, they will do it differently[1]. The author still believes despite the poor results in Chapter 5 that the more compact representation of multi-dimensional signals, will, in the end give real gains and that the identification of appropriate problems is a critical issue in the application of Clifford Networks. An avenue of exploration which could be investigated on the theoretical side would be look at pruning of weights in Clifford networks. It is reasonable to assume that if Clifford networks represent multi-dimensional signals in a more compact way, then pruning would be more effective in the Clifford case over the real case, because pruning a single Clifford weight is equivalent to pruning 2^n real weights.

The development of the Clifford valued Perceptron in Chapter 6 points to extensions of Complex valued Hopfield type networks in [Hirose, 1992b].

To show Clifford networks do give gains over real valued networks means that new problem domains need to be found where there is a real advantage in using more compact representations of multi-dimensional signals. This is highlighted by the reason for the poor results in Chapter 5 in that it is because the four components of the signal were too independent, that the Clifford Network failed to outperform the Real valued Network.

The author believes that a natural extension of the real and complex numbers should give an advantage when using them in neural networks. If anything this thesis has shown that such an extension is possible and that it is a fruitful area of research.

[1]If we were only interested in the class of functions a network could represent then why not use polynomial approximation theory instead?

Bibliography

[Albertinie and Sontag, 1993] F. Albertinie and E.D. Sontag. For neural networks function determines form. *Neural Networks*, pages 975–990, 1993.

[Benvenuto and Piazza, 1992] N Benvenuto and F Piazza. On the complex backpropagation algorithm. *IEEE Transactions on Signal Processing*, 40(4):967–969, April 1992.

[Blaine Lawson Jr. and Michelsohn, 1989] H Blaine Lawson Jr. and Marie-Louise Michelsohn. *Spin Geometry.* Princeton University Press, Princeton, New Jersey, 1989.

[Brackx *et al.*, 1982] F. Brackx, R. Delenghe, and F. Sommen. *Clifford Analysis.* Research notes in mathematics; 76. Pitman Advanced Publishing Program, 1982.

[Bromwich, 1906] T.J. I'A Bromwich. *Quadratic Forms and Their Classification by Means of Invariant-Factors*, volume 3 of *Cambridge Tracts in Mathematics and Mathematical Physics.* Cambridge University Press, 1906.

[Bryson and Ho, 1969] A.E. Bryson and Y.-C. Ho. *Applied Optimal Control.* New York: Blaisdell, 1969.

[Chisholm and Common, 1986] J.S.R Chisholm and A.K. Common, editors. *Clifford algebras and their applications in mathematical physics*, volume 183 of *NATO ASI series, C:Mathematical and physical sciences*, 1986.

[Copson, 1988] E.T. Copson. *Metric Spaces.* Cambridge University Press, first paper back edition, 1988.

[Crumeyrolle, 1990] Albert Crumeyrolle. *Orthogonal and Symplectic Clifford Algebras: Spinor Structures*, volume 57 of *Mathematics and its applications.* Kluwer Academic Publishers, Dordercht and London, 1990.

[Cybenko, 1989] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control Signals and Systems*, pages 303–314, 1989.

[Garey and Johnson, 1979] Micheal R. Garey and David S. Johnson. *Computers and intractability: a guide to the theory of NP-completeness.* 1979.

[Georgiou and Koutsougeras, 1992] George M. Georgiou and Cris Koutsougeras. Complex domain backpropagation. *IEEE Transactions on Circuits and Systems*, pages 330–334, 1992.

[Georgiou, 1993] George M. Georgiou. The multivalued and continuous perceptrons. In *World Congress on Neural Networks*, volume IV, pages 679–683, Portland, OR, 1993.

[Gilbert and Murry, 1991] John E Gilbert and Margaret A.M. Murry. *Clifford algebras and Dirac operators in harmonic analysis*, volume 26 of *Cambridge studies in advanced mathematics.* Cambridge University Press, 1991.

[Henseler and Braspenning, 1990] J Henseler and P.J. Braspenning. Training complex multi-layer neural networks. Technical Report CS 90-02, University of Limburg, Department of Computer Science, P.O. Box 616, 6200 MD Maastricht, The Netherlands, 1990.

[Hirose, 1992a] A. Hirose. Continuous complex-valued back-propagation learning. *Electronics Letters*, 20(20):1854–1855, September 1992.

[Hirose, 1992b] A. Hirose. Dynamics of fully complex-valued neural networks. *Electronics Letters*, 28(16):1492–1493, July 1992.

[Hirose, 1993] Aikra Hirose. Motion controls using complex-valued neural networks with feedback loops. In *IEEE International conference on neural networks*, 1993.

[Hornick *et al.*, 1989] K Hornick, Stinchcombe, and White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[Hornick, 1991] K Hornick. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4:251–257, 1991.

[Ito, 1991] Yoshifua Ito. Representation of functions by superpositions of step or sigmoid functions and their applications to neural network theory. *Neural Networks*, 4:385–394, 1991.

[Kůrková and Kainen, 1994] Věra Kůrková and Paul C. Kainen. Semigroups of function preserving weight transformations. *Neural Computation*, 6, May 1994.

[Kůrková, 1991] V. Kůrková. Kolmogorov's theorem is relevant. *Neural Computation*, 3(4):617–622, 1991.

[Lam, 1973] T.Y. Lam. *The algebraic theory of quadratic forms.* Mathematics lecture note series. W.A. Benjamin, Reading, Mass., 1973.

[Leung and Haykin, 1991] H. Leung and S. Haykin. The complex backpropagation algorithm. *IEEE Transactions on Signal Processing*, 39:2101–2104, September 1991.

[Little *et al.*, 1990] Gordon R. Little, Steven C. Gustafson, and Robert A. Senn. Generalization of the backpropagation neural network learning algorithms to permit complex weights. *Applied Optics*, 29(11):1591–1592, April 1990.

[MacKay, 1992] David J.C MacKay. *Bayesian Methods for Adaptive Models.* PhD thesis, California Institute of Technolgy, Pasadena, California, 1992.

[McCulloch and Pitts, 1943] W.S. McCulloch and Walter Pitts. A logical calculus of the idea immanent in neural nets. *Bulletin of Mathematical Biophysics*, pages 115–137, 1943.

[Minsky and Papert, 1969] Marvin L Minsky and Seymour A Papert. *Perceptrons, Expanded edition.* MIT Press, Cambridge M.A., 1969. Second edition 1988.

[Parker, 1985] D.B. Parker. Learning logic. Technical Report TR-47, Center for Computational Research in Economics and Management Science, Massachusetts Institute of Technology, Cambridge, M.A., 1985.

[Pearson and Bisset, 1992] J.K. Pearson and D.L. Bisset. Back Propagation in a Clifford Algebra. In *ICANN Brighton*, September 1992.

[Pearson and Bisset, 1994] J.K. Pearson and D.L. Bisset. Neural Networks in the Clifford Domian. In *IEEE94 symposium on Neural Networks Orlando*, June 1994.

[Pearson, 1992] John. E. Pearson. *Basic communication theory: a teacher's eye view of the way non-mathematicians see the mathematics and use of the basic ideas of modulation.* Prentice Hall, 1992.

[Porteous, 1981] I.R. Porteous. *Topological Geometry.* Cambridge University Press, 1981.

[Priestley, 1990] Hilary A Priestley. *Introduction to complex analysis.* Oxford: Clarendon Press, 1990.

[Rosenblatt, 1962] Frank Rosenblatt. *Principles of neurodynamics: perceptrons and the theory of brain mechanisms.* Spartan, Washington, 1962.

[Rudin, 1966] Walter Rudin. *Real and complex analysis.* McGraw-Hill series in higher mathematics. McGraw-Hill, 1966.

[Rudin, 1973] Walter Rudin. *Functional analysis.* McGraw-Hill series in higher mathematics. McGraw-Hill, 1973.

[Rumelhart and McClelland, 1986] D.E. Rumelhart and J.L McClelland. Learning internal representations by error propagation. In *Parallel Distributed Processing*, volume 1, chapter 8. M.I.T. Press, Cambridge, MA, 1986.

[Saha and Birdsall, 1986] Debabrata Saha and Theodore G. Birdsall. Quadrature-quadrature phase shift keying: a constant envelope modulation scheme. In *Proc. Conf. on Information Sciences and Systems*, March 1986.

[Saha, 1983] D. Saha. Q^2PSK - a new spectrally efficient modulation scheme. In *Proc. Twenty-first Annual Allerton Conference on Communication, Control and Computing*, pages 954–970, October 1983.

[Saied and Soliman, 1989] Feiz Saied and Samir S Soliman. Adaptive ML neural network based receiver for Q^2PSK modualted data-transmission systems. In *IEEE GLOBECOM 89 CH.2379*, pages 263 – 269, 1989.

[Sakai, 1971] Shoichiro Sakai. *C^*-algebras and W^*-algebras*, volume 60 of *Ergebnisse der Mathematik und ihrer Grenzgebiete.* Springer-Verlag, Berlin, 1971.

[Sontag, 1992a] E.D Sontag. Feedback stablisation using two hidden layer nets. *IEEE Transactions on Neural Networks*, pages 981–990, 1992.

[Sontag, 1992b] E.D Sontag. Feedforwards nets for interpolation and classication. *Journal of Computer and System Sciences*, 4:299–322, 1992.

[Sussmann, 1992] Hector J. Sussmann. Uniqueness of the weights for minimal feed-forward nets with a given input-output map. *Neural Networks*, pages 589–593, 1992.

[Tomasi, 1987] Wayne Tomasi. *Advanced electronic communication systems.* Prentice-Hall international editions, 1987.

[Ungerboeck, 1966] G Ungerboeck. Adaptive maximum-likelihood receiver for carrier-modulated data-transmission systems. *IEEE Trans. Inform. Theory*, pages 327–336, July 1966.

[Wegge-Olsen, 1993] N.E. Wegge-Olsen. *K-Theory and C^* -Algebras.* Oxford Univeersity Press, 1993.

[Werbos, 1974] P.J. Werbos. *Beyond Regression: New Tools for Predicition and Analysis in the Behavioral Sciences.* PhD thesis, Harvard University, 1974.