# 13. Clifford Algebra Multilayer Perceptrons

**Sven Buchholz and Gerald Sommer**[*]

Institute of Computer Science and Applied Mathematics,
Christian-Albrechts-University of Kiel

## 13.1 Introduction and Preliminaries

Multilayer Perceptrons (MLPs) are one of the most common and popular neural architectures. They are widely used in many different areas like handwriting recognition, speech recognition, and time series prediction for instance. In this chapter, we will extend MLPs from the domain of real numbers to Clifford algebra domains.

MLPs consist of Perceptron–type neurons as processing units grouped together in layers. The computation in an MLP is feed forward only. The neurons processing the input to the net are grouped in the *input layer*. The output of the net is taken from the output neurons grouped in the output layer. Usually, there are also one or more layers between input and output layer called *hidden layers*, since they are not visible from the outside. Input neurons are just for making the data available to the net, they do not perform a computation. Any other single neuron computes as its so–called propagation function a weighted sum of its received inputs. Thus, the association of the weights and the inputs is linear. Nonlinearity is achieved by applying a so-called activation function $g$. The computation of such a neuron is therefore given by

[*] This work has been supported by DFG Grants So-320-2-1 and So-320-2-2.

$$y = g(\sum_{i=1}^{n} w_i x_i + \theta) \tag{13.1}$$

in the real case $(w, x \in \mathbb{R}^n, \theta \in \mathbb{R})$ and by either

$$\boldsymbol{y} = \boldsymbol{g}(w \otimes_{p,q} x + \theta) \tag{13.2}$$

or by

$$\boldsymbol{y} = \boldsymbol{g}(x \otimes_{p,q} w + \theta) \tag{13.3}$$

in the general case of a Clifford algebra $(w, x, \theta \in \mathcal{C}_{p,q})$ using the *geometric product* $\otimes$ as associator. The Clifford neuron in comparison with the real neuron was fully discussed in chapter 12. There we assumed $g$ to be the identity to discuss propagation functions and linear aspects exclusively.

A suitable nonlinear activation function $g$ allows to built powerful neural networks out of real neurons by using the superposition principle

$$y = \sum_i \lambda_i \, g_i(x) \, . \tag{13.4}$$

Thus, one hidden layer may be sufficient and no activation function in the output neurons is needed. This transforms directly in the MLP architecture shown below in Figure 13.1.

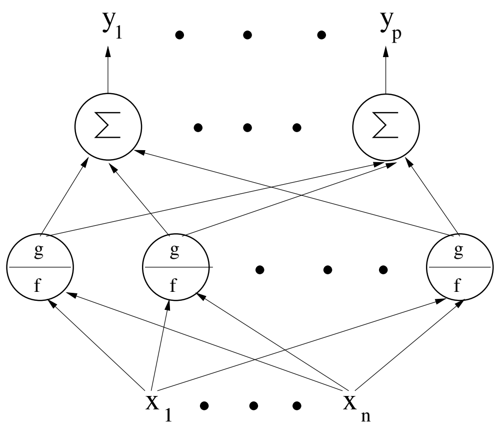

**Fig. 13.1.** *MLP with one hidden layer*

Cybenko proved in [53] that for so–called sigmoidal activation functions MLPs are universal approximators of continuous functions. In [121] these results were extended to the class of Borel measurable functions. Universal approximation in that sense means, that for any required approximation accuracy an MLP with one hidden layer with finite number of neurons is sufficient.

The graph representation (Figure 13.1) of a neural network is called its *topology*. Since an MLP is fully connected, its topology is therefore fully determined by the sequence of the number of nodes in any layer starting from the input layer.

Throughout this chapter let $N$ denote the number of neurons in the input layer, $M$ denote the number of neurons in the hidden layer, and $P$ denote the number of neurons in the output layer, respectively. Hence, we can speak of an $(N, M, P)$–MLP to denote the topology completely.

Let us denote the other parameters of the network according to Table 13.1.

**Table 13.1.** *Notations of MLP parameters*

| | |
|---|---|
| – $w^1_{nm}$ | weight connecting the $n$–th input neuron to the $m$–th hidden neuron |
| – $w^2_{mp}$ | weight connecting the $m$–th hidden neuron to the $p$–th output neuron |
| – $\theta^1_m$ | bias of the $m$–th hidden neuron |
| – $\theta^2_p$ | bias of the $p$–th output neuron |

The detailed structure of the chapter is as follows. Starting with a mathematically precise formulation of required notions of approximation theory we will derive in section 2 a sufficient criterion on activation functions to make Clifford MLPs universal approximators as well. In section 3, we will study activation functions of Clifford MLPs in detail. Reviewing the real, complex, and quaternionic special cases and proposed activation functions in the literature, we will get a systematic survey of the topic. We will prove therein that Clifford MLPs with sigmoid activation functions in every multivector component are universal approximators. After this we will develop in the subsequent section a backpropagation algorithm for such Clifford MLPs. In the final section we will report experimental results.

## 13.2 Universal Approximation by Clifford MLPs

In the introduction we gave already an informal characterization of the universal approximation property. Let us start with the formalization thereof by introducing the notion of "denseness". Thereby we will use $\mathcal{N}$ to indicate the class of functions realizable by a certain neural network and $\mathcal{F}$ the class of function that shall be approximated by $\mathcal{N}$. Then, the concept of denseness can be defined as follows.

**Definition 13.2.1.** *Let $F$ and $N$ be sets of functions of a normed space $(X, p)$. Let $d$ be the metric induced by $p$. That is for all $x \in \mathcal{N}, y \in \mathcal{F}$ the distance is defined by $d(x, y) = p(x - y)$. Then $N$ is dense under the norm $p$ in $F$ if, for any $f \in \mathcal{F}$ and any $\epsilon > 0$, there exists some $n \in \mathcal{N}$ with*

$$d(f, n) < \epsilon \,. \tag{13.5}$$

Thus, denseness is always measured with respect to some norm. Typical norms are the $L_p$ norms $(1 < p < \infty)$

$$\|f\|_p = \left( \int_X |f(x)|^p dx \right)^{1/p} . \tag{13.6}$$

However, the most relevant norm in our case will be the supremum norm $L_\infty$

$$\|f\|_\infty = \sup_{x \in X} |f(x)| \,. \tag{13.7}$$

A well known density theorem is the famous WEIERSTRASS theorem of real analysis. It states that polynomials of one real variable are dense in the set $C^0([a, b], \mathbb{R})$. A generalization of this, the STONE-WEIERSTRASS theorem, was used in [121] to prove the universal approximation capability of real valued MLPs. However, we cannot profit from these results in Clifford algebras. Moreover, these do not lead to a general density criterion of Clifford MLPs which is what we are looking for.

To reach this goal, we need functional analysis in Clifford algebras, especially an appropriate version of the HAHN-BANACH theorem. The real and complex HAHN-BANACH theorem has already been used in such a manner in [53] and accordingly in [6].

Let us first have an informal look at the HAHN-BANACH theorem before going into technical details. In its dominated extended version it states the following.

Let $M$ be a subspace of a linear space $X$ over $\mathbb{R}$, let $p$ be a sublinear functional defined on X and let $f$ be a linear form defined on $M$ dominated by $p$. The theorem asserts the existence of a linear extension $F$ of $f$ to X dominated by $p$ everywhere. The diagram below gives an illustration of the statement of the HAHN-BANACH theorem.

$$\begin{tikzcd}
F : X \arrow[d, dash] \arrow[dr] & & F \le p \\
f : M \arrow[r] & \mathbb{R} & f \le p
\end{tikzcd}$$

The basic idea now is the following. The neural architecture of $f$ above is $\mathcal{N}$ and $F$ is the class of functions it should be dense in. With this in mind we will get a nice criterion of denseness as a corollary from that theorem soon. To go ahead in this direction we have to formulate the theorem in terms of Clifford algebra. The functionality of a generic Clifford MLP is

$$\mathcal{C}_{p,q}^n \to \mathcal{C}_{p,q}^m \,. \tag{13.8}$$

In general, $\mathcal{C}_{p,q}^n$ cannot be a linear space, since $\mathcal{C}_{p,q}$ itself is not a skew field in general. Thus, we have to replace the concept of a linear space with the algebraic weaker one of a module.

**Definition 13.2.2.** *Let $R$ be a ring with 1. A left $R$-module $\mathcal{G}_l$ is an abelian group $G = (G, +)$ together with a mapping $R \times G_l \to G_l : (r, g) \mapsto rg$ in such a way, that*

$$\begin{array}{lll} (a) & \forall g_1, g_2 \in G \ \forall r \in R : & r(g_1 + g_2) = rg_1 + rg_2 \\ (b) & \forall g \in G \ \forall r_1, r_2 \in R : & (r_1 + r_2)g = r_1 g + r_2 g \\ (c) & \forall g \in G \ \forall r_1, r_2 \in R : & (r_1 r_2)g = r_1(r_2 g) \\ (d) & \forall g \in G : & 1g = g \end{array}$$

*are fulfilled.*

The corresponding definition of right modules is obvious. However, we only have to choose one version to be formally consistent without loss of generality. From now on we will always use left modules. To bound a function as required by the HAHN-BANACH theorem we next introduce the notion of a seminorm.

**Definition 13.2.3.** *Let $X$ be a $\mathcal{C}_{p,q}$-module. A function $p : X \to \mathbb{R}$ is called a seminorm on $X$ if it fulfills for all $f, g \in X, \lambda \in \mathcal{C}_{p,q}$ and $\kappa \in \mathbb{R}$*

$$\begin{array}{ll} (a) & p(f+g) \leq p(f) + p(g) \\ (b) & p(f) = 0 \Rightarrow f = 0 \\ (c) & p(\lambda f) \leq C\|\lambda\| p(f) \\ & p(\kappa f) = |\kappa| p(f)\,. \end{array}$$

The next definition gives then complete access to the needed concept of boundness.

**Definition 13.2.4.** *Let $X$ be a $\mathcal{C}_{p,q}$-module. A family $P$ of seminorms $p : X \to \mathbb{R}$ is called a proper system of seminorms on $X$ if for any finite sequence $p_1, p_2, \ldots, p_k \in P$ there exist $p \in P$ and $C > 0$ such that for all $f \in X$*

$$\sup_{j=1,\ldots,k} p_j(f) \leq C p(f)\,. \tag{13.9}$$

Hereafter, we will speak of a module equipped with a proper system of seminorms as a proper module for shortness. We are now in the position to formulate the HAHN-BANACH theorem of Clifford analysis.

**Theorem 13.2.1.** *Let $X$ be a proper $\mathcal{C}_{p,q}$-module, let $Y$ be a submodule of $X$ and let $T$ be a bounded left $\mathcal{C}_{p,q}$–functional on $Y$. Then there exists a bounded left $\mathcal{C}_{p,q}$–functional $T^*$ on $X$ such that*

$$T^*_{|Y} = T\,. \tag{13.10}$$

For a proof of this theorem and the following corollary see again [30]. This corollary will give us now the desired density criterion.

**Corollary 13.2.1.** *Let $X$ be a left proper $\mathcal{C}_{p,q}$-module and $Y$ a submodule of $X$. Then $Y$ is dense in $X$, iff for all $T \in X^*$ with $T_{|Y} = 0$ follows $T = 0$ on $X$.*

Let us now return to our neural architecture $\mathcal{N}$ that should be dense in the function class $\mathcal{F}$. If it is not, then the closure $\overline{\mathcal{N}}$ is not completely $\mathcal{F}$. By corollary 13.2.1 of the Clifford HAHN-BANACH theorem there exists a bounded linear functional $L : \mathcal{F} \to \mathcal{C}_{p,q}$, with $L(\overline{\mathcal{N}}) = L(\mathcal{N})$ and $L \neq 0$. Then furthermore, by the Clifford RIESZ theorem [30] there exists a unique Clifford measure $\mu$ on $X$ such that for all $g \in C^0(X, \mathcal{C}_{p,q})$

$$L(g) = \int_X g d\mu(x)\,. \tag{13.11}$$

Let us assume that the function $g$ has the special property to be *discriminatory.*

**Definition 13.2.5.** *A function $g : \mathcal{C}_{p,q} \to \mathcal{C}_{p,q}$ is said to be discriminatory if*

$$\int_{I^{2^{p+q}}} g(w \otimes x + \theta) d\mu(x) = 0 \tag{13.12}$$

*implies that $\mu(x) = 0$ for any finite regular Clifford measure $\mu$ with support $I^{2^{p+q}} := [0,1]^{2^{p+q}}$.*

If $g$ is discriminatory, then follows immediately by definition that $\mu(x) = 0$. But this is a contradiction to $L \neq 0$, which was a consequence of the assumption that $\mathcal{N}$ is dense in $\mathcal{F}$. Thus, we can conclude, that the use of a discriminatory activation function is sufficient to make Clifford MLPs universal approximators of $C^0(I^{2^{p+q}}, \mathcal{C}_{p,q})$ functions.

## 13.3 Activation Functions

With the discriminatory property of the preceding section we have already a criteria on hand regarding the approximation capabilities of activation functions of Clifford MLPs. We will now turn our attention to properties necessary from the algorithmic point of view. We will start with the real case and then proceed to the multi–dimensional Clifford algebra case. We can thereby make extensive use of previous work by other authors in the complex and quaternionic case.

### 13.3.1 Real Activation Functions

Let us start with a general property an activation function has to fulfill independent of the concrete training algorithm. It is simply due to implementation aspects. The property in mind is boundness to avoid overflows during simulation on computers.

In the real case, this property is easy to check and expressed by so–called *squashing functions*. A function $g : \mathbb{R} \to \mathbb{R}$ is called a squashing function if $\lim_{x\to-\infty} g(x) = a$ and $\lim_{x\to\infty} g(x) = b$ for $a, b \in \mathbb{R}, a < b$.

Since backpropagation is gradient descent in the weight space of the MLP, the activation function has to be differentiable. A class of squashing functions with this property are the *sigmoid* functions.

**Definition 13.3.1.** *The function*

$$\sigma_\beta(x) : \mathbb{R} \to \mathbb{R}; \quad x \mapsto \frac{1}{1+\exp(-\beta x)} \tag{13.13}$$

*is called a sigmoid function.*

Also the widely used hyperbolic tangent in real MLPs is only a slight modification of a sigmoid function, since

$$\tanh = 2\sigma_2 - 1\,. \tag{13.14}$$

In the following, we will only proceed with the most used activation function of real MLPs which is the so–called *logistic* function $\sigma := \sigma_1$. It has a very simple derivative $\dot{\sigma} = \sigma(1-\sigma)$. Figure 13.2 shows a plot of the logistic function.

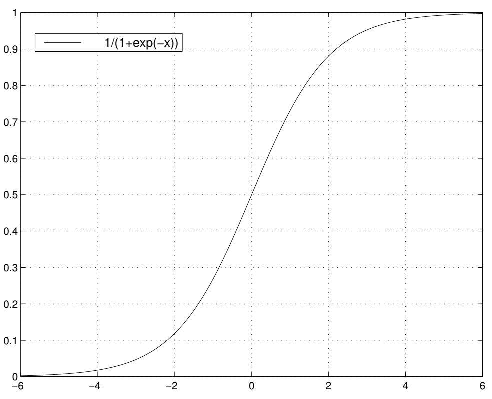

**Fig. 13.2.** *Logistic activation function*

The logistic function has approximately linear behaviour in $[-1, 1]$ and saturation is reached quickly outside of this region. To be complete at this point, we mention again that the universal approximator capability of real–valued MLPs with sigmoid activation function was first proven by Cybenko [53].

### 13.3.2 Activation Function of Clifford MLPs

There are two possible ways of generalization of real activation functions to Clifford algebra activation functions. One is to use the corresponding Clifford algebra formulation of such a function, the other is to use the real activation function in every multivector component separately. A formal characterization can be made in the following way.

**Definition 13.3.2.** *Let be $G : \mathcal{C}_{p,q} \to \mathcal{C}_{p,q}$ $(p + q > 1)$, $n := 2^{p+q}$.*
*$G$ is called a component–wise activation function if*

$$\forall i \in \{1, \ldots, n\}\ \exists g_i \in \mathbb{R} \to \mathbb{R}\ \forall x = (x_1, \ldots, x_n) \in \mathcal{C}_{p,q} : [G(x)]_i = g_i(x_i)\,,$$

*otherwise a multivector activation function.*

Thus, in the complex case $(G : \mathbb{C} \to \mathbb{C})$ a multivector activation function has the generic form

$$G(z) = u\,(x, y) + u\,(x, y)\,\mathrm{i}\,. \tag{13.15}$$

On the other hand, the generic form of a component–wise activation function is given by

$$G(z) = v\,(x) + v\,(y)\,\mathrm{i}\,. \tag{13.16}$$

The use of a multivector activation function seems to be more natural and quiet more sophisticated in comparison to component activation functions. Hence, as the complex MLP was introduced, the first proposed activation function (see e.g. [152]) was the extension of the real logistic function $\sigma$ to the complex domain

$$\sigma_{\mathbb{C}} :\ \mathbb{C} \to \mathbb{C};\ z \mapsto \frac{1}{1 + \exp(-z)}\,. \tag{13.17}$$

Later, it was pointed out by Georgiou and Koutsougeras [90] that $\sigma_{\mathbb{C}}$ is not bounded, since it has singularities with value $+\infty$ at $0 + \pi(2n + 1)\,\mathrm{i}$ $(n \in \mathbb{N})$. Due to that fact, these authors proposed as alternative the activation function

$$G_{c,r} :\ \mathbb{C} \to \mathbb{C};\ z \mapsto \frac{z}{c + \frac{1}{r}\,|z|} \qquad (c, r \in \mathbb{R})\,. \tag{13.18}$$

They also gave a complete list of necessary properties that complex activation functions of the form (13.15) have to fulfill. One of these is with respect to the backpropagation algorithm, that all partial derivatives have to exist together with further conditions on them. Clearly, these requirements are also valid in the general case of an arbitrary Clifford algebra.

The first Clifford MLP introduced by Pearson [187] used the straightforward Clifford algebra domain extension of the above function (13.18). However, he has not paid any attention to the fact that Clifford algebras are not division algebras in general. The consequence (impossibility) for the formulation of a correct backpropagation algorithm is the subject of the next section. Also, no universal approximation theorem for such networks could yet be proven.

Instead, we will use in our Clifford MLP the real logistic activation function in any component. Let us denote the function derived in this manner by $\boldsymbol{\sigma}$. This function was first introduced by Arena et al. [7] in their work on the quaternionic MLP (QMLP). For this QMLP they further derived a quaternionic version of the backpropagation algorithm and proved its universal approximation capability. To give a proof sketch for the universal approximation capability of Clifford MLPs with activation function $\boldsymbol{\sigma}$ is the final part of this section.

**Theorem 13.3.1.** *The function*

$$\boldsymbol{\sigma}(w;\theta) : \mathcal{C}_{p,q} \rightarrow \mathcal{C}_{p,q}; \quad x \mapsto \sum_{A \in \mathcal{A}} \sigma([w \otimes x + \theta])_A e_A \tag{13.19}$$

*is discriminatory.*

Proof (Sketch). Let $\mu(x)$ be a finite regular Clifford measure on the set $C^0(I^{2^{p+q}}, \mathcal{C}_{p,q})$ such that

$$\int_{I^{2^{p+q}}} \boldsymbol{\sigma}(w \otimes x + \theta) d\mu(x) = 0\,, \tag{13.20}$$

for all $w, x \in \mathcal{C}^n_{p,q}, \theta \in \mathcal{C}_{p,q}$. According to the definition of $\boldsymbol{\sigma}$, we have for all $i \in \ 1, \ldots, 2^{p+q}\}$

$$[\boldsymbol{\sigma}(w \otimes x + \theta)]_i = \sigma([w \otimes x + \theta]_i)\,. \tag{13.21}$$

Let us now consider the pointwise limit

$$\phi_i(w \otimes x + \theta) := \lim_{\lambda \rightarrow \infty} \sigma(\lambda [w \otimes x + \theta]_i)\,, \tag{13.22}$$

with $\lambda \in \mathbb{R}$. This limit evaluates to

$$\phi_i(w \otimes x + \theta) = \begin{cases} 1 & : \quad if \quad [w \otimes x + \theta]_i > 0 \\ 0 & : \quad if \quad [w \otimes x + \theta]_i \leq 0 \end{cases} \tag{13.23}$$

With the *Lesbesgue*-dominated convergence theorem of Clifford analysis follows

$$\begin{aligned}0 &= \int_{I^{2^{p+q}}} \boldsymbol{\sigma}(w \otimes x + \theta) d\mu(x) \\ &= \int_{I^{2^{p+q}}} (\sum_{A \in \mathcal{A}} \phi_A(w \otimes x + \theta) e_A)\, d\mu(x) \\ &= \lim_{\lambda \to \infty} \sigma(\lambda [w \otimes x + \theta]_i)\,.\end{aligned}$$

For all $j \in \{0,1\}^{2^{p+q}}$ define the following sets

$$H_j := \bigcap_{\substack{i \in \{1,\dots,2^{p+q}\} \\ j[i]=1}} \{[w \otimes x + \theta]_i > 0\} \cap \bigcap_{\substack{i \in \{1,\dots,2^{p+q}\} \\ j[i]=0}} \{[w \otimes x + \theta]_i \leq 0\}\,. \tag{13.24}$$

Thus, the $H_j$ sets give us a partition of $I^{2^{p+q}}$. Therefore, we have with (13.23), (13.24)

$$\mu(\cup H_j) = 0\,. \tag{13.25}$$

Unfortunately, no assumptions on $\mu$ can be made. Therefore, one has to prove that for all $j \in \{0,1\}^{2^{p+q}}$

$$\mu(H_j) = 0\,. \tag{13.26}$$

By the real theorem of Cybenko [53] we only know $\mu(H_{10\dots0}) = 0$. This can be extended with some effort to show that

$$\mu(\{H_j \mid \sum_{i=1}^{2^{p+q}} j[i] = 1\}) = 0 \tag{13.27}$$

However, the other cases remain open problems to be proved.

## 13.4 Clifford Back–Propagation Algorithm

In this section we will derive the Clifford back–propagation algorithm. For the sake of simplicity, we only deal with a Clifford MLP with one hidden layer, reminding the reader that this structure is already a universal approximator. Let $N$, $M$ and $P$ denote the number of input, hidden and output nodes, respectively. Furthermore, let be $w^1_{nm}$ the multivector weight connecting the $n$-th input node with the $m$-th hidden node, and $w^2_{mp}$ the one connecting the $m$-th hidden node with the $p$-th output node. Analogously, the bias nodes are denoted by $\theta^1_m$ and $\theta^2_p$, respectively.

Using the above nomenclature the feed-forward phase is given as follows:

- hidden node activation and output value

$$S^1_m := \sum_{n=1}^{N} w^1_{nm} \otimes x_n + \theta^1_m \tag{13.28}$$

$$h_m := \boldsymbol{\sigma}(S^1_m) \tag{13.29}$$

- output node activation and output value

$$S^2_p := \sum_{m=1}^{M} w^2_{mp} \otimes h_m + \theta^2_p \tag{13.30}$$

$$o_p := \boldsymbol{\sigma}(S^2_p)\,. \tag{13.31}$$

We will now apply gradient descent with respect to the weights to minimize the common sum–of–squared error function

$$E = \frac{1}{2}\sum_{p=1}^{P}(y_p - o_p)^2\,, \tag{13.32}$$

whereby $y = (y_1, \dots , y_p)$ stands for the expected output value. First, we have to compute the weights of the output layer according to

$$\nabla E_{w^2_{mp}} = \sum_{A\in\mathcal{A}} \frac{\partial E}{\partial [w^2_{mp}]_A}\, e_A\,. \tag{13.33}$$

Applying the chain rule to each term of (13.33) gives

$$\frac{\partial E}{\partial [w^2_{mp}]_A} = \sum_{B\in\mathcal{A}} \frac{\partial E}{\partial [S^2_p]_B}\frac{\partial [S^2_p]_B}{\partial [w^2_{mp}]_A}\,. \tag{13.34}$$

For the partial derivatives of the error function wrt. the output node activation $S^2_p$ we obtain

$$\frac{\partial E}{\partial [S^2_p]_B} = \frac{\partial E}{\partial [y_p]_B}\frac{\partial [y_p]_B}{\partial [S^2_p]_B} = ([y_p]_B - [o_p]_B)\,\dot{\boldsymbol{\sigma}}([S^2_p]_B)\,. \tag{13.35}$$

The computation of the partial derivatives of the output node activation wrt. the output layer weights is as easy to compute. However, some effort has to be made to get one single compact formula for it. Let us take a look at the case of 4–dimensional Clifford algebras. Table 13.2 shows exemplarily the partial derivatives $\frac{\partial [S^2_p]}{\partial [w^2_{mp}]_{e_1}}$.

**Table 13.2.** *Partial derivatives* $\frac{\partial[S_p^2]}{\partial[w_{mp}^2]_{e_1}}$

| | $\mathcal{C}_{0,2}$ | $\mathcal{C}_{1,1}$ | $\mathcal{C}_{2,0}$ |
|---|---|---|---|
| $\frac{\partial[S_p^2]_0}{\partial[w_{mp}^2]_{e_1}}$ | $-[h_m]_{e_1}$ | $+[h_m]_{e_1}$ | $+[h_m]_{e_1}$ |
| $\frac{\partial[S_p^2]_{e_1}}{\partial[w_{mp}^2]_{e_1}}$ | $+[h_m]_0$ | $+[h_m]_0$ | $+[h_m]_0$ |
| $\frac{\partial[S_p^2]_{e_2}}{\partial[w_{mp}^2]_{e_1}}$ | $+[h_m]_{e_{12}}$ | $-[h_m]_{e_{12}}$ | $+[h_m]_{e_{12}}$ |
| $\frac{\partial[S_p^2]_{e_{12}}}{\partial[w_{mp}^2]_{e_1}}$ | $-[h_m]_{e_2}$ | $-[h_m]_{e_2}$ | $+[h_m]_{e_2}$ |

It is easy to conclude from the above example (and also easy to verify directly), that

$$\frac{\partial[S_p^2]}{\partial w_{mp}^2} = h_m^* \tag{13.36}$$

for some involution $^*$ dependent on the underlying Clifford algebra. Clearly, this involution is already determined uniquely by any partial derivative $\frac{\partial[S_p^2]}{\partial[w_{mp}^2]_A}$. For Clifford algebras of the type $\mathcal{C}_{(0,q)}$ this involution is just conjugation, i.e. we have $h_m^* = \overline{h_m}$ in (13.36). Due to the fact that the geometric product of a multivector with a scalar is ordinary component–wise real multiplication we can get a very elegant description of the involution $^*$ via the scalar component. We can then use the fact

$$[x \otimes \bar{y}]_0 = xy^T \tag{13.37}$$

to describe $^*$ as the unique involution yielding

$$[x \otimes y^*]_0 = xy^T \,. \tag{13.38}$$

Putting now all the derived results together and using the symbol $\odot$ to denote component–wise multiplication, we get the following update rule for the weights of the output layer

$$\varDelta w_{mp}^2 = [\,\underbrace{(y_p - o_p) \odot \dot{\sigma}(S_p^2)}_{\delta_p^2}\,] \otimes \overline{h}_m \,. \tag{13.39}$$

The derivation of the updating rule for the weights of the hidden layer is analog, resulting in

$$\varDelta w_{nm}^1 = [\,\underbrace{(\sum_{p=1}^{P} \overline{w}_{mp}^2 \otimes \delta_p^2) \odot \dot{\sigma}(S_m^1)}_{\delta_m^1}\,] \,.\otimes \overline{x}_n \tag{13.40}$$

Finally, the update rule for the biases is then given by

$$\varDelta\,\theta_p^2 = \delta_p^2 \quad \text{and} \quad \varDelta\,\theta_m^1 = \delta_m^1\,. \tag{13.41}$$

Let us now verify briefly our claim made in section 3 regarding the impossibility of a general correctly Clifford back–propagation algorithm for not component–wise activation functions. This is simply due to the existence of divisors of zero in general Clifford algebras. Using non component–wise activation functions results in a geometric product $\otimes$ instead of a component–wise product $\odot$ in (13.39), respectively (13.40). Thus, $\delta_m^1$ and $\delta_p^2$ could then always be zero even if a non–zero error occurred. Due to the definition of $^*$ this can never be the case for the geometric products involving $^*$.

The above derived Clifford back-propagation rule therefore avoids problems with divisors of zero completely.

## 13.5 Experimental Results

Both real MLPs and Clifford MLPs (CMLPs) are universal approximators of continuous functions in several variables as we know from the previous sections. Thus, they have the same theoretical strength in principle. Moreover, they use the "same" activation function, since our Clifford MLP uses the logistic function $\sigma$ in every component. However, alternative activation functions are rare as argued before. Thus, a potential advantage of CMLPs versus MLPs seems to be based on the propagation function, i.e. on the involved geometric product. The propagation function was fully discussed in chapter 12, however only in the case of a single linear neuron. As we know from that chapter, in a Clifford MLP real vector data can be presented in many arbitrary different ways. But it is difficult to give general advises for Clifford MLPs for the optimal choice, especially due to the incorporated non–linearity.

However, this is only valid in a theoretically provable sense. In this section instead, we try to conclude from an experimental approach. Thereby, we will compare the space and time complexity of the real MLP and the CMLP with respect to their generalization performance. Time complexity is used in the loose sense of convergence time.

Space complexity is measured by the amount of real parameters. This is given for an MLP with one hidden layer by the formula

$$\#MLP := M \cdot (N+1) + P \cdot (M+1)\,. \tag{13.42}$$

The weights of an CMLP can be easily converted into real parameters by counting them component by component. Thus, one obtains for the number of real parameters of an CMLP with one hidden layer

$$\#CMLP_{(p,q)} := 2^{p+q} \cdot M \cdot (N+1) + 2^{p+q} \cdot P \cdot (M+1)\,. \tag{13.43}$$

An MLP with the same number of component activation functions as an CMLP would have $2^{p+q}$ times of real parameters, which follows easily from (13.42), (13.43). Clearly, the assumption that an MLP would require the same amount of activation functions as an CMLP to achieve the same performance is not realistic. However, about 20–25% fewer parameters of an CMLP in comparison to an MLP where reported by Arena et al. [6], [7] and this result was also obtained in earlier work of ours [15]. However, it is not easy to find another reason for this phenomenon than the more compact weight structure of CMLPs, especially in the case of processing real data. Thus, we will not make a simulation of the approximation of a real vector function, but one of a Clifford–valued function. The main goal of the simulations is to check, whether or not there are indications of algebraic model–based behavior of Clifford MLPs.

Let us study only one very simple example. The considered function is

$$f : \mathbb{C} \to \mathbb{C};\ (x + y\,\mathrm{i}) \mapsto (x^2 + y^2 + 2xy\,\mathrm{i})\,, \tag{13.44}$$

which is plotted in Figure 13.3.

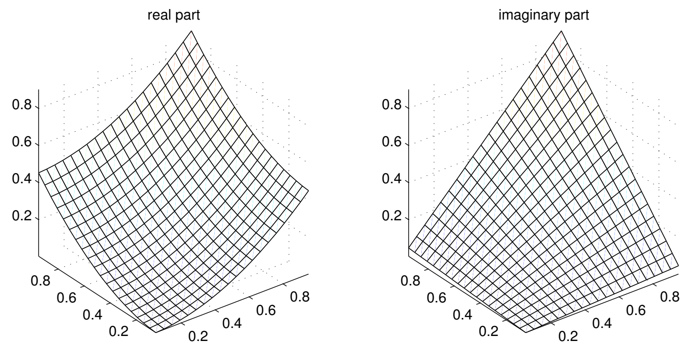

**Fig. 13.3.** *Plot of the real and the imaginary parts of f*

There is a very good reason to choose a low dimensional example. Namely, everything can be visualized. This is clearly helpful, to get a real evaluation of the performance of different networks. As outlined, we want to investigate whether or not there might be indications of algebraic interpretations of the approximation capability of the $\mathrm{CMLP}_{(0,1)}$. If on the other hand a $\mathrm{CMLP}_{(1,0)}$ achieved an equal or better performance then an algebraic reason would have to be rejected. Clearly, also if this were true for a real MLP. If both Clifford networks showed equivalent but better performance than the real MLP, this would then only be due to their more compact weight structure. As in the simulations of chapter 12 we will also have a closer look at the performance of the networks in the presence of noise.

Next, we report the obtained results in detail.

The training data consisted of 100 randomly drawn points from $[0, 1] \times [0, 1]$ with uniform distribution. For the test set we sampled this domain with a regular grid of size $20 \times 20$. Thus, we got 100 training points and 400 test points. This 20%/80% ratio of samples is well established and therefore often used in neural network simulations.

The number of hidden nodes of the Clifford networks was easy to determine. Two hidden nodes already gave results, which could not be improved significantly by the use of more hidden nodes. The performance of an MLP with two hidden nodes was not sufficient. The convergence of the training of the two CMLPs, a (2,3,2)-MLP and a (2,4,2)-MLP are shown in Figure 13.4.

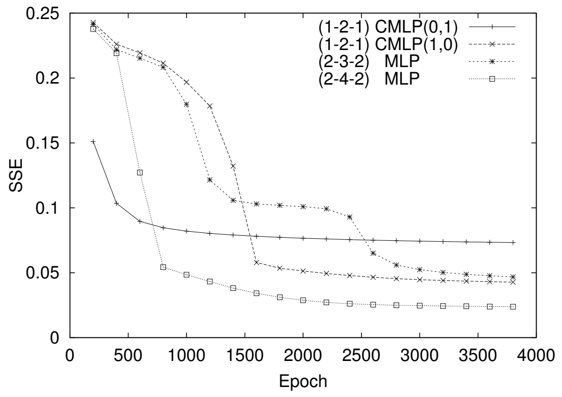

**Fig. 13.4.** *Convergence of the learning*

A first look at these results seems to be quite astonishing. The complex MLP shows the worst performance of all networks, followed by the CMLP$_{(1,0)}$. Taking into account that the number of parameters of the (2,3,2)-MLP is 17 ((2,4,2)-MLP: 22) and that of both CMLPs is 14, no advantage seems visible from the space complexity point of view either. We should remark, that all networks have reached a stable and optimal error plateau.

As already observed in the simulations on Clifford neurons in chapter 12, generalization is the measure that counts indeed. The obtained sum-of-squared errors (SSE) in training and testing are reported together in Table 13.3.

**Table 13.3.** *SSEs by noise free training*

| network | SSE Training | SSE Testing |
|---|---|---|
| (1,2,1)-CMLP$_{(0,1)}$ | 0.07302 | 0.00040 |
| (1,2,1)-CMLP$_{(1,0)}$ | 0.04175 | 0.00067 |
| (2,4,2)-MLP | 0.00971 | 0.03298 |
| (2,3,2)-MLP | 0.04315 | 0.15539 |

As we can see there, the complex MLP showed the best generalization performance of all nets, followed closely by the CMLP$_{(1,0)}$. Both generalization errors are very low, and remarkable orders smaller than the trainings errors. The MLPs on the other hand have both approximately 4-times higher generalization errors than training errors. With the generalization errors the situation has changed completely.

The complex MLP has reached the training error level corresponding to this superb generalization error very quickly within about 1000 epochs. Thus, we could say that it converges fastest. Another interesting fact observable from Figure 13.4 is that all other nets except the complex MLP show the same behavior during the beginning of the training. Hence, we could argue that the complex CMLP could match the underlying model of the data very early in the training, whereas both real MLPs have not observed the right model as can be concluded from their generalization error. The rapid descent of the MLP with 4 hidden nodes in comparison to that with 3 hidden nodes is clearly due to its greater amount of parameters (degrees of freedom).

Let us now have a closer look at the output of the networks shown in Figure 13.5. Especially compare the numerically nearly identical generalization of both CMLPs. Thereby, areas of high approximation errors are indicated by light shading in Figure 13.5.

Between them there are no great differences visible in fact. Both have learned the two component functions indeed, with less accuracy on the imaginary one. However, the learned real and imaginary functions of the MLPs are similar and (thus) far away from the expected shapes. Obviously, the MLPs have applied a global numerically concept to match the data, without notice to the structure of the data. The effect of the 4–th MLP hidden node is also easy to interpret. With only 3 hidden nodes the MLP learned a similar concept as with 4 hidden nodes. However, it decided to not descend down to zero height to approximate the range $[0, 0.3]^2$ accurately. Since the values of the function are low in this area an error is ”cheap” with respect to the other areas.

Thus, an MLP is not able to detect the algebraic structure of the data. But the CMLP$_{(1,0)}$ seems to be able to do so. We should remember that there is only one sign in the multiplication tables that makes both 2-dimensional Clifford algebras different.

In the following we discuss how things changed in the presence of noise. We therefore added 20% mean–free noise to the training data. The obtained errors are presented in Table 13.4.

**Table 13.4.** *SSEs by 20% noise in training data*

| network | SSE Training | SSE Testing |
|---|---|---|
| (1,2,1)-CMLP$_{(0,1)}$ | 1.32612 | 0.72041 |
| (1,2,1)-CMLP$_{(1,0)}$ | 1.57850 | 1.15853 |
| (2,4,2)-MLP | 1.24180 | 0.74325 |
| (2,3,2)-MLP | 1.51760 | 1.12471 |

The errors of the (2,4,2)-MLP and the (1,2,1)-CMLP$_{(0,1)}$ are nearly equal. The same can be said about the errors of the (2,3,2)-MLP and the (1,2,1)-CMLP$_{(1,0)}$. The (1,2,1)-CMLP$_{(0,1)}$ shows now a better performance than the (1,2,1)-CMLP$_{(1,0)}$. Clearly, that of the MLP with 4 hidden nodes is better than that with 3 hidden nodes again.

Let us study the outputs of the networks shown in Figure 13.6 beginning with that of the real MLPs. We can see a clear negative effect of the 4 hidden nodes there. In the presence of noise, this "additional" degree of freedom is just used to learn the noise. Actually, any simulation of an (2,4,2)-MLP with high noisy training data can produce an arbitrary output scheme. The applied concept leads then no longer to equally well learned component functions. This is still the case for the (2,3,2)-MLP. The real part in the case of the (2,4,2)-MLP shows an additional scaling error, while its imaginary part fits randomly the imaginary component function.

An important but well known conclusion from the simulations is that more degrees of freedom in an MLP (which cannot match the model of data) are only good for memorization. Clearly, things then get worse very quickly in the presence of noise.

The difference between both CMLPs does not seem too large again. However, it is significant as seen in Table 13.4. The imaginary part of the function in the range $[0, 0.2] \times [0, 0.8]$ is better approximated by the complex CMLP.

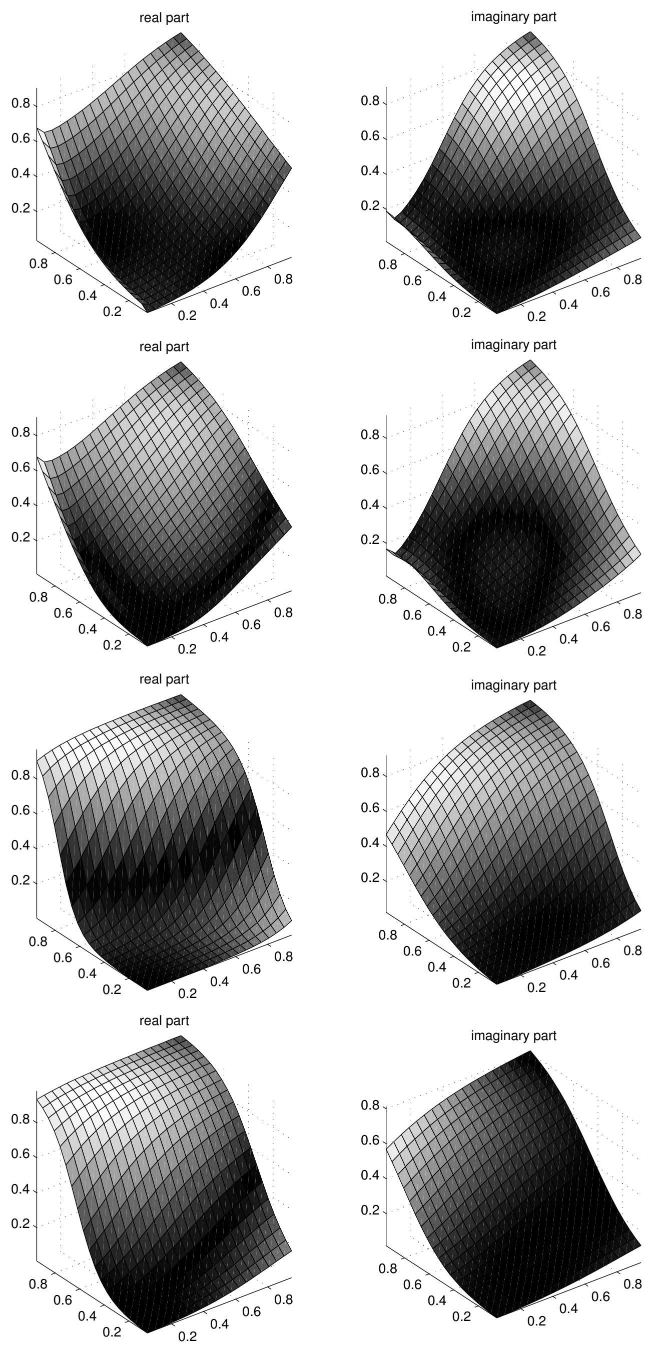

**Fig. 13.5.** *Approximation results (from top to bottom):* *(1,2,1)-CMLP*$_{(0,1)}$, *(1,2,1)-CMLP*$_{(1,0)}$, *(2,4,2)-MLP, (2,3,2)-MLP*

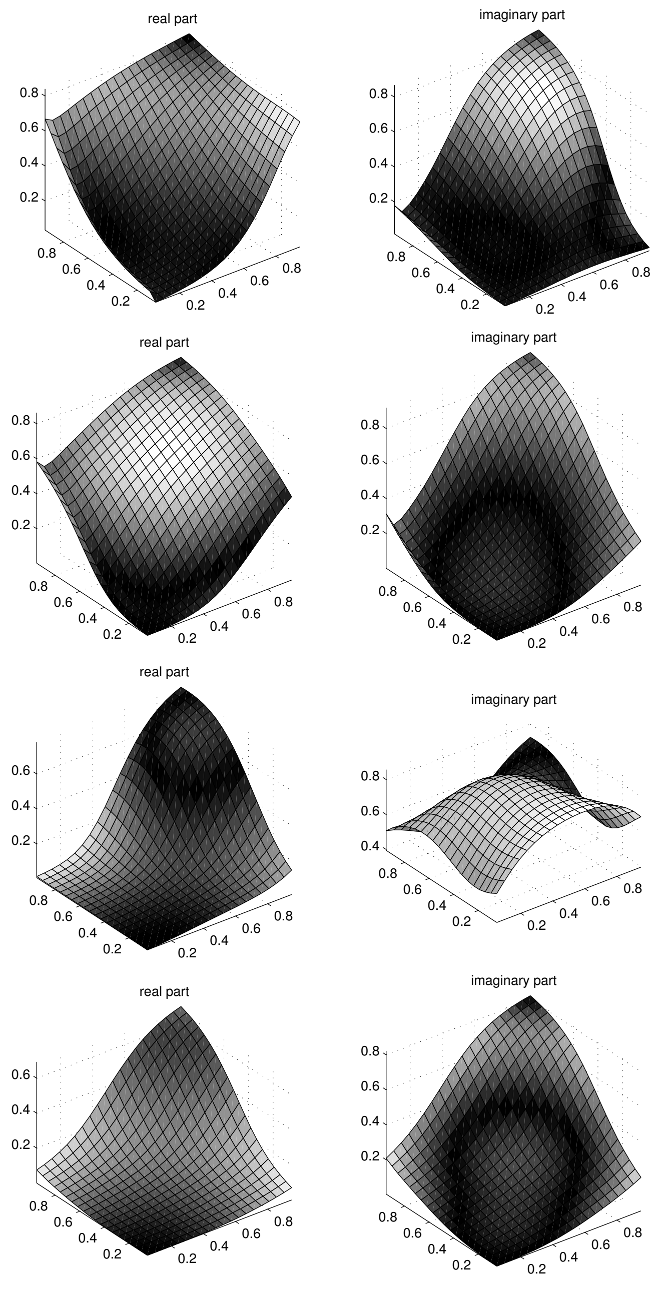

**Fig. 13.6.** *Approximation results by 20% noise (from top to bottom): (1,2,1)-CMLP$_{(0,1)}$, (1,2,1)-CMLP$_{(1,0)}$, (2,4,2)-MLP, (2,3,2)-MLP*

## 13.6 Conclusions and Outlook

In this chapter we introduced the Clifford Algebra Multilayer Perceptron (CMLP) as an extension of the well known real–valued MLP. Thereby, we applied mainly a theoretical point of view.

We discussed questions regarding the theory of function approximation in Clifford algebras in some detail. This led us to a criterion on the activation function that guarantees that CMLPs as MLPs are universal approximators too. We then reviewed basic facts on activation functions known from the literature in the complex and quaternionic special cases [90] [7]. We introduced the notion of component–wise activation functions and argued why this is a necessary property of activation functions for CMLPs.

A central part of this work was the derivation of the Clifford algebra back–propagation algorithm. We have found an elegant way to formulate the updating rules of the weights in terms of generally characterized involutions. The properties of these involutions guarantee the operativeness of the algorithm because excluding problems with zero divisors.

Although, concentrating on theoretical and technical aspects of computing with Clifford MLPs throughout this paper, we also made a simple simulation to compare the performance of Clifford MLPs with real–valued MLPs. Our interest was thereby to see if CMLPs are also model–based as we showed for single Clifford neurons in chapter 12. The obtained results were unfortunately weaker in that sense and partially showed up only in the presence of noise. However, the model–based property of CMLPs is not in general doubt. On the other hand it is also very clear, that non–linearity in any MLP architecture makes things less easy to interpret.

Thus, many more simulations have to be done in the future to get empirical confidence at this points. It is indicated, that such simulations should be biased in that way, that specifically geometric tasks are chosen for Clifford MLPs. However, such tasks might require a more suitable and flexible architecture. This could mean using inhomogeneous nodes in a layer, i.e. nodes of different Clifford algebra type. A step still further in this direction would be the use of nodes operating on single blades. All these steps would only require small modifications of the Clifford back–propagation algorithm as derived herein. From a conceptual point of view this would then invoke questions of self–organization.

It is our strong belief that a way based on this work towards more complex Clifford neural computation is worth being considered and would be fruitful.