Optimal Learning Rates for Clifford Neurons

Sven Buchholz[1], Kanta Tachibana[2], and Eckhard M.S. Hitzer[3]

[1] Institute of Computer Science, University of Kiel, Germany
[2] Graduate School of Engineering, Nagoya University, Japan
[3] Department of Applied Physics, University of Fukui, Japan

Abstract. Neural computation in Clifford algebras, which include familiar complex numbers and quaternions as special cases, has recently become an active research field. As always, neurons are the atoms of computation. The paper provides a general notion for the Hessian matrix of Clifford neurons of an arbitrary algebra. This new result on the dynamics of Clifford neurons then allows the computation of optimal learning rates. A thorough discussion of error surfaces together with simulation results for different neurons is also provided. The presented contents should give rise to very efficient second–order training methods for Clifford Multilayer perceptrons in the future.

1 Introduction

Neural computation in Clifford algebras, which include familiar complex numbers and quaternions, has recently become an active research field [6,5,11]. Particularly, Clifford-valued Multi-layer perceptrons (CMLPs) have been studied. To our knowledge, no second–order methods to train CMLPs are available yet. This paper aims to provide therefore the needed facts on Hessians and optimal learning rates by studying the dynamics on the level of Clifford neurons. After an introduction to Clifford algebra, we review Clifford neurons as introduced in [4]. Section 4 provides as a general and explicit notion for Hessian matrix of these neurons, which enables easy and accurate computation of optimal learning rates. This is demonstrated by simulation in section 5, after which the paper concludes with a summary.

2 Basics of Clifford Algebra

An algebra is a real linear space that is equipped with a bilinear product. Clifford algebras are of dimension $2^n (n \in \mathbb{N}_0)$ and are generated by so-called quadratic spaces, from which they inherit a metric structure. Every quadratic space has an orthonormal basis. Here we are interested in spaces $\mathbb{R}^{p+q}$ $(p, q \in \mathbb{N}_0)$ with canonical basis $\mathcal{O} := (e_1, ..., e_{p+q})$ that are endowed with a quadratic form Q, such that for all $i, j \in \{1, \ldots, p+q\}$

$$Q(e_i) = \begin{cases} 1, & i \leq p \\ -1, & p+1 \leq i \leq p+q, \end{cases} \tag{1}$$

$$Q(e_i + e_j) - Q(e_i) - Q(e_j) = 0\,, \tag{2}$$

J. Marques de Sá et al. (Eds.): ICANN, Part I, LNCS 4668, pp. 864–873, 2007.

which, in turn, renders $\mathcal{O}$ to be orthonormal. These quadratic spaces, abbreviated as $\mathbb{R}^{p,q}$ hereafter, give now rise to the following definition.

Definition 1 (Clifford Algebra [12]). *The Clifford algebra $\mathcal{C}_{p,q}$ of $\mathbb{R}^{p,q}$ is the 2^{p+q} dimensional associative algebra with unity $1_{\mathbb{R}}$ given by*

(i) *$\mathcal{C}_{p,q}$ is generated by its distinct subspaces $\mathbb{R}$ and $\mathbb{R}^{p,q}$*
(ii) *$xx = Q(x)$ for any $x \in \mathbb{R}^{p,q}$.*

Note the use of juxtaposition for the so–called geometric product of $\mathcal{C}_{p,q}$. Since nothing new is generated in case of $p = q = 0$, $\mathcal{C}_{0,0}$ is isomorphic to the real numbers $\mathbb{R}$. The Clifford algebras $\mathcal{C}_{0,1}$ and $\mathcal{C}_{0,2}$ are isomorphic to complex numbers and quaternions, respectively.

The set of all geometric products of elements of $\mathcal{O}$

$$\mathcal{E} := \{e_{j_1} e_{j_2} \cdots e_{j_r},\ 1 \leq j_1 < \ldots < j_r \leq p+q\} \tag{3}$$

constitutes a basis of $\mathcal{C}_{p,q}$, which follows directly from condition (ii) in Definition 1. This basis can be ordered by applying the canonical order of the power set $\mathcal{P}$ of $\{1, \ldots, p+q\}$, i.e. by using the index set

$$\mathcal{I} := \{\{i_1, \ldots, i_s\} \in \mathcal{P} \mid 1 \leq i_1 < \ldots < i_s \leq p+q\} \tag{4}$$

and then define $e_I := e_{i_1} \ldots e_{i_s}$ for all $I \in \mathcal{I}$. Set $e_\emptyset := 1_{\mathbb{R}}$. A general element x of $\mathcal{C}_{p,q}$

$$x = \sum_I e_I\, x_I := \sum_{I \in \mathcal{I}} e_I\, x_I, \quad (x_I \in \mathbb{R}) \tag{5}$$

is termed multivector. The name stems from the fact that for $k \in \{0, \cdots, p+q\}$

$$\mathcal{C}^k_{p,q} := \{e_I \mid I \in \mathcal{I}, |I| = k\} \tag{6}$$

spans a linear subspace of $\mathcal{C}_{p,q}$ ($\mathcal{C}^0_{p,q} = \mathbb{R}$, $\mathcal{C}^1_{p,q} = \mathbb{R}^{p,q}$ and $\sum_k \mathcal{C}^k_{p,q} = \mathcal{C}_{p,q}$). Projection onto subspaces is defined via the grade operator

$$\langle\cdot\rangle_k : \mathcal{C}_{p,q} \to \mathcal{C}^k_{p,q}, x \mapsto \sum_{\substack{I \in \mathcal{I}, \\ |I|=k}} x_I e_I\,, \tag{7}$$

with convention $\langle\cdot\rangle := \langle\cdot\rangle_0$ for the so-called scalar part. Projection onto the I-th component (w.r.t. $\mathcal{E}$) of a multivector, on the other hand, is written as $[\cdot]_I$.

Involutions on $\mathcal{C}_{p,q}$ are linear mappings that when applied twice yield the identity again. Examples are inversion $inv(x) := \sum_{k=0}^{p+q} (-1)^k \langle x\rangle_k$ and reversion $rev(x) := \sum_{k=0}^{p+q} (-1)^{\frac{k(k-1)}{2}} \langle x\rangle_k$. The principal involution defined by $\widetilde{e_I} = rev(e_I)\, e_0^{[I_q]}$ is of main importance for us, whereby $[I_q]$ stands for the number of negative signature vector factors in e_I. Note that $\widetilde{xy} = \widetilde{y}\widetilde{x}$. Any non–null vector of $\mathcal{C}_{p,q}$ has an inverse (w.r.t. the geometric product) and is an element of the Clifford group.

Definition 2 (Clifford group [10]). *The Clifford group associated with the Clifford algebra $\mathcal{C}_{p,q}$ is defined by*

$$\Gamma_{p,q} = \{s \in \mathcal{C}_{p,q} \mid \forall x \in \mathbb{R}^{p,q}, s\, x\, inv(s)^{-1} \in \mathbb{R}^{p,q}\}\,. \tag{8}$$

The action of $\Gamma_{p,q}$ on $\mathbb{R}^{p,q}$ is an orthogonal automorphism [12] that extends to multivectors due to the bilinearity of the geometric product. Since $\mathcal{C}_{0,1}$ is commutative, the fact that complex multiplication can be viewed geometrically as dilation–rotation [14] follows as special case from (8).

3 Clifford Neurons and the Linear Associator

Formally, a Clifford neuron can be easily introduced as follows. Consider the most common real neuron (of perceptron–type with identity as activation function) that computes a weighted sum of its inputs x_i according to

$$y = \sum_i w_i x_i + \theta\,. \tag{9}$$

Using instead multivector weights $w_i \in \mathcal{C}_{p,q}$, a multivector threshold $\theta \in \mathcal{C}_{p,q}$ and replacing real multiplication by the geometric product readily turns (9) into a Clifford neuron (for $\mathcal{C}_{0,0}$, of course, one gets the real neuron back again). Although complex-valued and quaternionic–valued Multi-layer perceptrons (MLPs) are known for a long time (see [9] and [2], respectively), not much research has been carried out on the neuron level itself in the beginning. Only recently, the following two types of Clifford neurons have been studied in greater detail in the literature.

Definition 3 (Basic Clifford Neuron (BCN) [4]). *A Basic Clifford Neuron,* $\mathrm{BCN}_{p,q}$*, computes the following function from $\mathcal{C}_{p,q}$ to $\mathcal{C}_{p,q}$*

$$y = wx + \theta. \tag{10}$$

A BCN has only one multivector input and therefore also only multivector weight. Contrary to a real neuron (9) it makes sense to study the one input only case, since this will be in general a multidimensional entity. The general update rule using gradient descent for a BCN with simplified, for notation purposes, error function

$$E = \|error\|^2 := \|d - y\|^2\,, \tag{11}$$

where d stands for desired output, reads

$$\Delta w = -\frac{\partial E}{\partial w} = 2error\,\widetilde{x}\,. \tag{12}$$

Equation (12) generalizes the various specific update rules given in [4] by using the principal involution of Section 2. The second type of Clifford neuron proposed in [4] mimics the action of the Clifford group.

Definition 4 (Spinor Clifford Neuron (SCN) [4]). *For $p, q \in \mathbb{N}_0,\ p+q>1$ a Spinor Clifford Neuron, $\mathrm{SCN}_{p,q}$, computes the following function from $\mathcal{C}_{p,q}$ to $\mathcal{C}_{p,q}$*

$$y = w\, x\, \phi(w) + \theta\,, \tag{13}$$

where ϕ denotes a particular involution (e.g. inversion, reversion,...).

Like a BCN, a SCN has also only one independent weight though a two–sided multiplication takes place. For a SCN with (adapted) error function (11) gradient descent therefore results in a two-stage update procedure. First compute the update of the "right weight" in (13) as

$$w_R := \widetilde{(w\, x)}\, error \tag{14}$$

which then yields $\Delta w = \phi(w_R)$. Again, this is a powerful generalization of the few specific rules known from the literature.

BCNs are the atoms of Clifford MLPs while SCNs are the atoms of Spinor MLPs. Clifford MLPs have already found numerous applications (see e.g. [6,1] and the references therein). Recent applications of Spinor MLPs can be found in [11,5]. All these works, however, rely on a simple modified Back–propagation algorithm. To our knowledge, there are no known general second–order based training algorithms for Clifford MLPs and Spinor MLPs yet. Given the increasing interest in such architectures a high demand for second order methods can be stated. This paper aims to establish the foundations of such forthcoming algorithms by providing general and explicit expressions for describing the dynamics of Clifford neurons in terms of their Hessian matrices.

Hessians (of the error functions) of Clifford neurons are not a complete *terra incognita* though. Both BCNs and SCNs can be viewed, to some extent, as particular Linear Associators (LAs) (see e.g. [3,13] for an introduction to this most simple feed–forward neural network). In the case of BCNs this readily follows from the well–known theorem that every associative algebra of finite dimension is isomorphic to some matrix algebra. For SCNs, however, this additionally, relies on the fact (mentioned in Section 2) that the action of the Clifford group is always an orthogonal automorphism.

Dynamics of the Linear Associator are well understood and [8] is perhaps the first complete survey on the topic. Note that for a LA gradient descent, with an appropriate learning rate η, converges to the global minimum for any convex error function like the sum-of-squares error (SSE). The Hessian $\mathrm{H_{LA}}$ of a LA with error function of SSE-type (of which (11) is a simplified version) is always a symmetric matrix. For batch–learning (see e.g. the textbook [13]) the optimal learning rate η_{opt} is known to be $\frac{1}{\lambda_{max}}$ where λ_{max} is the largest eigenvalue of $\mathrm{H_{LA}}$. Also, $\mathrm{H_{LA}}$ is known to be the auto–correlation matrix of the inputs.

In principle it would be possible to compute Hessians of BCNs from the above facts. There are however several drawbacks that limit this procedure to only low–dimensional algebras and render it impossible for all other cases. First of all, note that in order to proceed in this way one has to know and apply the appropriate matrix isomorphism for every single algebra. There is no general

solution and the isomorphisms can get rather complicated. Also, by applying an isomorphism numerical problems may occur for the eigenvalue computation. Even if one succeeds, no explicit structural insights are easily obtained from this procedure since everything is expressed in terms of a particular input. For SCNs the situation will be even more complicated, and, by missing the big picture again, no conclusions of truly general value are obtained.

The next section therefore now offers a new general and explicit solution for computing Hessian matrices of Clifford neurons without the above drawbacks. This is done by invoking the principal involution.

4 Hessian Matrices of Clifford Neurons

Hessian matrices w.r.t. the parameters of a neural architecture allow to compute optimal learning rates. Decoupling of parameters and using an individual optimal learning rate for every single one yields fastest possible convergence [7]. For the two types of Clifford neurons introduced in the previous section it is very natural to decouple the multivector weight w from the multivector threshold θ. The threshold θ of a BCN or SCN is a generic parameter in the sense, that it does not depend on the signature (p, q) of the underlying Clifford algebra. In fact, it can be handled in exactly the same way as the threshold vector of a LA of dimension $p + q$. In what follows we therefore only study Hessians w.r.t. the multivector weight w. To simplify notations, inputs to the neurons are written as x without indexing over the training set.

Theorem 1 (Hessian of a BCN). *The elements of the Hessian matrix of a* BCN *are given by*

$$\partial_{IJ}E = \partial_{JI}E = \frac{\partial E}{\partial w_J \partial w_I} = 2\langle \widetilde{e_I} e_J x \widetilde{x} \rangle \tag{15}$$

where $\widetilde{\cdot}$ *denotes principal involution.*

Proof. Rewriting the error function as

$$\begin{aligned} E = \|d - (wx + \theta)\|^2 &= \sum_I \left(d_I - ([wx]_I + \theta_I)\right)^2 \underbrace{e_I \widetilde{e_I}}_{1} \\ &= \langle (d - wx - \theta)(d - wx - \theta)\widetilde{\ } \rangle \\ &= \left\langle wx\widetilde{(wx)} \right\rangle - \langle wx(d-\theta)\widetilde{\ } \rangle - \left\langle (d-\theta)\widetilde{(wx)} \right\rangle \\ &= \langle wx\widetilde{x}\widetilde{w} \rangle - 2\,\langle (d-\theta)\widetilde{x}\widetilde{w} \rangle \end{aligned} \tag{16}$$

yields

$$\begin{aligned} \frac{\partial E}{\partial w_I} &= \underbrace{\langle e_I x \widetilde{x}\widetilde{w} \rangle}_{\langle (x\widetilde{x}\widetilde{w})\widetilde{\ } \widetilde{e_I} \rangle} + \langle wx\widetilde{x}\widetilde{e_I} \rangle - 2\,\langle (d-\theta)\widetilde{x}\widetilde{e_I} \rangle \\ &= 2\,\langle wx\widetilde{x}\widetilde{e_I} \rangle - 2\,\langle (d-\theta)\widetilde{x}\widetilde{e_I} \rangle \\ &= -2\,\langle (d - wx - \theta)\widetilde{x}\widetilde{e_I} \rangle \end{aligned}$$

and finally

$$\begin{aligned}\frac{\partial E}{\partial w_J \partial w_I} &= 2\,\langle e_J x \widetilde{x} \widetilde{e_I} \rangle \\ &= 2\langle \widetilde{e_I} e_J x \widetilde{x} \rangle\,. \end{aligned} \tag{17}$$

For every I, J, (17) "picks out" the scalar component of the product resulting from multiplying $\widetilde{e_I} e_J$ with the multivector $x\widetilde{x}$. Consequently, one gets easily the following result for the neurons $\mathrm{BCN}_{0,q}$.

Corollary 1. *The Hessian of* $\mathrm{BCN}_{0,q}$ *is a scalar matrix* aE*, where* E *denotes the identity matrix.*

Of course, every diagonal entry of the Hessian of a $\mathrm{BCN}_{p,q}$ equals the scalar a above. The next corollary gives a complete overview for BCNs of dimension four.

Corollary 2. *Let* $\{x^s = c_0^s e_0 + c_1^s e_1 + c_2^s e_2 + c_{12}^s e_{12}\}_{s=1\ldots S}$ *denote the input set to a* $\mathrm{BCN}_{p,q}$*,* $p+q=2$*. Set* $a := \sum_{s=1}^{S}(c_0^s c_0^s + c_1^s c_1^s + c_2^s c_2^s + c_{12}^s c_{12}^s)$*,* $b := 2\sum_{s=1}^{S}(c_0^s c_1^s + c_2^s c_{12}^s)$*,* $c := 2\sum_{s=1}^{S}(c_0^s c_2^s - c_1^s c_{12}^s)$*,* $d := 2\sum_{s=1}^{S}(c_0^s c_{12}^s - c_1^s c_2^s)$*. Then the Hessian w.r.t. the SSE-function of the* $\mathrm{BCN}_{0,2}$*, the* $\mathrm{BCN}_{1,1}$*, and the* $\mathrm{BCN}_{2,0}$ *is given by*

$$\mathbf{H}_{\mathrm{BCN}_{0,2}} = \begin{pmatrix} a & 0 & 0 & 0 \\ 0 & a & 0 & 0 \\ 0 & 0 & a & 0 \\ 0 & 0 & 0 & a \end{pmatrix}, \tag{18}$$

$$\mathbf{H}_{\mathrm{BCN}_{1,1}} = \begin{pmatrix} a & b & 0 & d \\ b & a & d & 0 \\ 0 & d & a & -b \\ d & 0 & -b & a \end{pmatrix}, \text{ and} \tag{19}$$

$$\mathbf{H}_{\mathrm{BCN}_{2,0}} = \begin{pmatrix} a & b & c & 0 \\ b & a & 0 & c \\ c & 0 & a & -b \\ 0 & c & -b & a \end{pmatrix}, \tag{20}$$

respectively.

Unfortunately, the Hessian of a SCN does have a more comlicated form and does not depend alone on the received input.

Theorem 2 (Hessian of a SCN). *The elements of the Hessian matrix of a* SCN *are given by*

$$\begin{aligned}\frac{\partial E}{\partial w_J \partial w_I} = \;& 2\,\langle (e_J x \overline{w} + w x \overline{e_J})\,(e_I x \overline{w} + w x \overline{e_I})\tilde{}\,\rangle \\ & -2\,\langle (d - w x \overline{w})\,(e_I x \overline{e_J} + e_J x \overline{e_I})\tilde{}\,\rangle \end{aligned} \tag{21}$$

where $\overline{(\cdot)}$ *stands for the involution of the* SCN*.*

Proof. From

$$\begin{aligned} E &= \|d - wx\overline{w}\|^2 = \sum_I \left(d_I - (wx\overline{w})_I\right)^2 e_I \widetilde{e_I} \\ &= \langle (d - wx\overline{w})(d - wx\overline{w})\tilde{}\rangle \\ &= \left\langle d\widetilde{d} + wx\overline{w}(wx\overline{w})\tilde{}\right\rangle - 2\left\langle d(wx\overline{w})\tilde{}\right\rangle \end{aligned} \tag{22}$$

follows

$$\begin{aligned} \frac{\partial E}{\partial w_I} &= \langle (e_I x\overline{w} + wx\overline{e_I})\,(wx\overline{w})\tilde{}\rangle \\ &\quad + \langle (wx\overline{w})(e_I x\overline{w} + wx\overline{e_I})\tilde{}\rangle \\ &\quad - 2\left\langle d(e_I x\overline{w} + wx\overline{e_I})\tilde{}\right\rangle \\ &= -2\left\langle (d - wx\overline{w})(e_I x\overline{w} + wx\overline{e_I})\tilde{}\right\rangle \end{aligned} \tag{23}$$

and by computing $\partial_{w_J} \frac{\partial E}{\partial w_I}$ follows the assumption.

Hence, if w is of small absolute value the approximation $\mathbf{H}_{\mathrm{SCN}_{\mathrm{p,q}}} \approx 2 \cdot \mathbf{H}_{\mathrm{BCN}_{\mathrm{p,q}}}$ holds. Due to space limitations, this concludes or study of Hessians of Clifford neurons.

5 Simulations on Optimal Learning Rates

From Theorem 1 direct structural insights on the dynamics of BCNs can be obtained. According to Corollary 1, the error function of a $\mathrm{BCN}_{0,q}$ is always isotropic, its Hessian is a scalar matrix aE, and therefore $\eta = \frac{1}{a}$ is its optimal learning rate. This section now reports simulation results for all four–dimensional BCNs (theoretically covered by Corollary 2) in order to give some illustrations and to get further insights on the dynamics of these neurons. For the quaternionic $\mathrm{BCN}_{0,2}$ 100 points have been generated from a uniform distribution of $[0, 1]^4$. Each element of the that way obtained input training set was left multiplied in $\mathcal{C}_{0,2}$ by $1e_0 + 2e_1 + 3e_2 + 4e_{12}$ in order to get the output training set. Using the optimal learning rate $\eta = \eta_{opt}$ (computed as $1/\lambda_{max}$ from (19) and

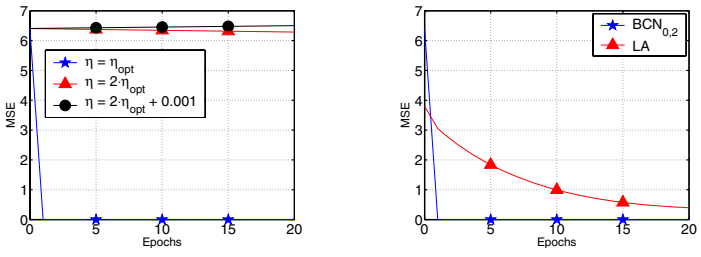

Fig. 1. Left panel shows learning curves of the $\mathrm{BCN}_{0,2}$ for different learning rates η. Right panel shows comparison of the optimal learning curves for the $\mathrm{BCN}_{2,0}$ and the LA. Both architectures are trained with their respective optimal learning rate.

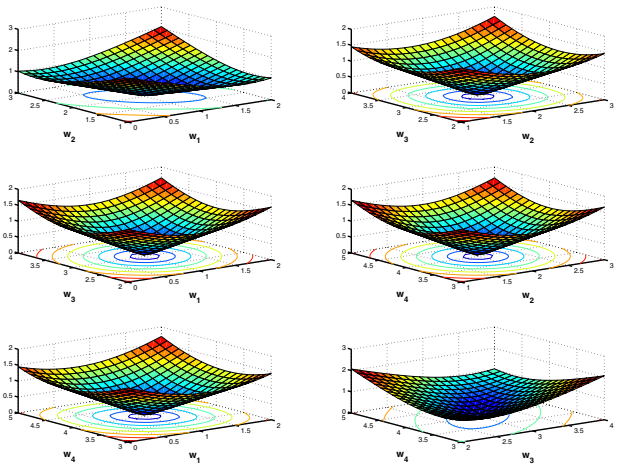

Fig. 2. Plot of error function for the $\mathrm{BCN}_{1,1}$. Projections on the six 2D planes that coincide with the coordinate axes of weight space.

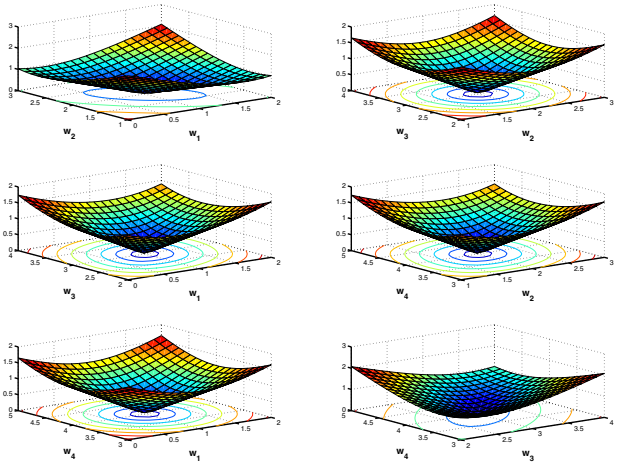

Fig. 3. Plot of error function for the $\mathrm{BCN}_{2,0}$. Projections on the six 2D planes that coincide with the coordinate axes of weight space.

(20), respectively) batch-learning converged indeed in one epoch to the global solution. This can be seen from the left panel of figure 1, which also shows obtained learnig curves for $\eta = 2\eta_{opt}$ (slowest convergence) and $\eta = 2\eta_{opt} + 0.001$ (beginning of divergence). Right panel of figure 1 shows optimal learning curve of the BCN $_{0,2}$ versus optimal learning curve of the LA for illustration.

Similar simulations have been carried out for the remaining two BCNs of dimension four. In both cases input set of the $\mathrm{BCN}_{0,2}$ has been used again. For

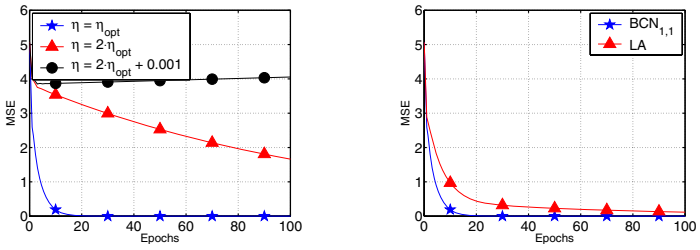

Fig. 4. Left panel shows learning curves of the $\mathrm{BCN}_{1,1}$ for different learning rates η. Right panel shows comparison of the optimal learning curves for the $\mathrm{BCN}_{2,0}$ and the LA. Both architectures are trained with their respective optimal learning rate.

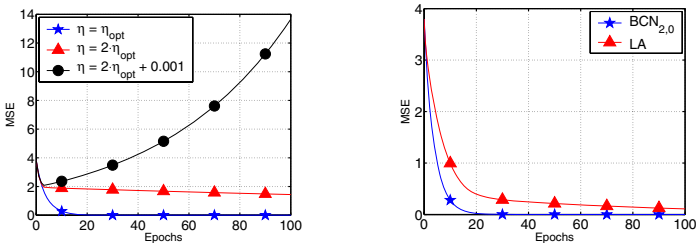

Fig. 5. Left panel shows learning curves of the $\mathrm{BCN}_{2,0}$ for different learning rates η. Right panel shows comparison of the optimal learning curves for the $\mathrm{BCN}_{2,0}$ and the LA. Both architectures are trained with their respective optimal learning rate.

obtaining the output set, it was left multilpied in $\mathcal{C}_{1,1}$ and $\mathcal{C}_{0,2}$ by $1e_0+2e_1+3e_2+4e_{12}$, respectively. Error surfaces obtained for these data are plotted in figure 2 and figure 3, respectively. Both surfaces do have a partial isotropy since isotropic projections do exist. Moreover, the following can be proven from (19),(20) for the projection planes $E_{w_I,w_J}(\mathrm{BCN}_{p,q})$: $E_{w_1,w_4}(\mathrm{BCN}_{1,1}) = E_{w_2,w_3}(\mathrm{BCN}_{1,1}) = E_{w_1,w_3}(\mathrm{BCN}_{2,0}) = E_{w_2,w_4}(\mathrm{BCN}_{2,0})$ and $E_{w_1,w_3}(\mathrm{BCN}_{1,1}) = E_{w_2,w_4}(\mathrm{BCN}_{1,1}) = E_{w_1,w_4}(\mathrm{BCN}_{2,0}) = E_{w_2,w_3}(\mathrm{BCN}_{2,0})$. Note how this nicely resembles the fact that $\mathcal{C}_{1,1}$ and $\mathcal{C}_{0,2}$ are isomorphic algebras. For both neurons a further decoupling of weights might be of interest due to the existence of isotropic planes. Finally, plots of learnig curves are provided by figure 4 and figure 5. Inclusion of an LA in all simulations was done to show that and how Clifford neurons do differ significantly from an LA.

6 Summary

The dynamics of Clifford neurons have been studied in this paper. General and explicit expressions of the Hessian of a BCN and of a SCN have been derived. From the general BCN result structural insights on the dynamics of specific BNCs can be easily obtained. The Hessian of a $\mathrm{BCN}_{0,q}$ has been identified to be a scalar matrix aE, which readily gives $1/a$ as optimal learning rate for such a

neuron. Simulations have been carried out to show the stability and accuracy of computation of optimal learning rates from the derived Hessians. We hope that the presented material will give rise to fast second–order training methods for Clifford and Spinor MLPs in the future.

References

1. Arena, P., Fortuna, L., Muscato, G., Xibilia, M.G.: Neural Networks in Multidimensional Domains. LNCIS, vol. 234. Springer, Heidelberg (1998)
2. Arena, P., Fortuna, L., Occhipinti, L., Xibilia, M.G.: Neural networks for quaternion valued function approximation. In: IEEE Int. Symp. on Circuits and Systems, London, vol. 6, pp. 307–310. IEEE Computer Society Press, Los Alamitos (1994)
3. Baldi, P.F., Hornik, K.: Learning in linear neural networks: A survey. IEEE Transactions on Neural Networks 6(4), 837–858 (1995)
4. Buchholz, S.: A theory of neural computation with Clifford algebra. Technical Report Number 0504, Universität Kiel, Institut für Informatik (2005)
5. Buchholz, S., Le Bihan, N.: Optimal separation of polarized signals by quaternionic neural networks. In: EUSIPCO 2006. 14th European Signal Processing Conference, Florence, Italy (September 4-8, 2006)
6. Hirose, A.: Complex–Valued Neural Networks. Studies in Computational Intelligence, vol. 32. Springer, Heidelberg (2006)
7. LeCun, Y., Bottou, L., Orr, G.B., Müller, K.–R.: Efficient BackProp. In: Orr, G.B., Müller, K.-R. (eds.) Neural Networks: Tricks of the Trade. LNCS, vol. 1524, Springer, Heidelberg (1998)
8. LeCun, Y., Kanter, I., Solla, S.A.: Eigenvalues of Covariance Matrices: Application to Neural–Network Learning. Physical Review Letters 66(18), 2396–2399 (1991)
9. Leung, H., Haykin, S.: The Complex Backpropagation Algorithm. IEEE Transactions on Signal Processing 3(9), 2101–2104 (1991)
10. Lounesto, P.: Clifford Algebras and Spinors. Cambridge University Press, Cambridge (1997)
11. Matsui, N., Isokawa, T., Kusamichi, H., Peper, F., Nishimura, H.: Quaternion neural network with geometrical operators. Journal of Intelligent & Fuzzy Systems 15(3–4), 149–164 (2004)
12. Porteous, I.R.: Clifford Algebras and the Classical Groups. Cambridge University Press, Cambridge (1995)
13. Rojas, R.: Neural Networks. Springer, Heidelberg (1996)
14. Yaglom, I.M.: Complex Numbers in Geometry. Academic Press, New York (1968)