Geometric Neural Networks

Eduardo Bayro-Corrochano and Sven Buchholz
Computer Science Institute, Cognitive Systems Group
Christian Albrechts University, Kiel, Germany
email: edb,sbh@informatik.uni-kiel.de

Abstract. The representation of the external world in biological creatures appears to be defined in terms of geometry. This suggests that researchers should look for suitable mathematical systems with powerful geometric and algebraic characteristics. In such mathematical context the design and implementation of neural networks will be certainly more advantageous. This paper presents the generalization of feedforward neural networks in the Clifford or geometric algebra framework. The efficiency of the geometric neural nets indicate a step forward in the design of algorithms for multidimensional artificial learning.

Categories: Clifford algebra; geometric algebra; feedforward neural networks; hyper-complex neural networks; RBF geometric neural networks.

1 Introduction

Biological creatures interact with their environment in order to survive. This activity is triggered by different needs which should be satisfied. The most important ones are nutrition and conservation. As soon a creature moves secure its internal activity may switch on higher cognition levels to satisfy other sophisticated needs, e.g. the joy during playing. If we are interested to build artificial intelligent systems which should autonomously perceive and act in their surroundings we should first of all ask how the machine should build its internal representation of the world. Nowadays there is the believe that the brain might be seen as a geometric engine [14, 16]. A general hypothesis of the geometric interpretation of the brain may relay on the assumption that the mapping between the external world and the brain is certainly a result of the perception and action activities within a time cycle. These activities controlled by the central nervous system might be seen in the context of learning by experience as a basic way to build the internal geometric representation.

In mathematical terms we can formalize the relations of the physical signals of world objects with the creature ones by using extrinsic vectors coming from the world and intrinsic vectors which depict the internal representation. We can also assume that the external world and the internal world have different reference coordinate systems. If we see the acquisition of knowledge as a distributed process it is imaginable that there exist various domains of representation with different vectorial basis and obeying different metrics. How is it possible that nature through the evolution has acquired such tremendous representation power for dealing with such geometric vector representations. In a stimulating series of articles Pellionisz and Llinàs [16, 17] claim that the formalization of the geometrical representation seems to be the dual expression of extrinsic physical cues by intrinsic central nervous system vectors. They quoted that these vectorial representations related to reference frames intrinsic to the creature are co-variant for perception analysis and contra-variant for action synthesis. These authors explain that the geometric mapping between these two vectorial spaces is implemented by a neural network which performs as a metric tensor [17]. The Clifford algebra in the geometric interpretation of Hestenes [10] appears to be an alternative to the tensor analysis employed since 1980 by Pellionisz and Llinàs for the perception action cycle theory. Since tensor calculus is co-variant, in other words it requires of transformation laws for getting coordinate-independent relations, Clifford or geometric algebra appears more attractive as it is not only essentially a coordinate free or invariant system but also includes spinors which tensor theory does not. The computational efficiency of geometric algebra has been shown in various challenging areas of mathematical physics [6]. Preliminary attempts for applying the geometric algebra in neural geometry have been already done by Hestenes and in the fields of computer vision, robotics and neural nets can be found in [11, 12, 3, 4, 2]. Analyzing other approaches for neural computation we see that the mostly used is the matrix algebra. Geometric algebra and matrix algebra both are associative algebras, yet geometric algebra captures the geometric characteristics of the problem better independent of a coordinate reference system and offers also other computational mechanisms that matrix algebra has not, e.g. the geometric product using hypercomplex, double and dual entities.

In this paper we will specify a geometric algebra $\mathcal{G}_n$ of the n-dimensional space by $\mathcal{G}_{p,q}$, where p and q stand for the number of basis vectors which squares to 1 and -1 respectively and fulfill n=p+q. See [10, 2, 3] for a more complete introduction in geometric algebra . The next section reviews the computing principles of feedforward neural networks underlining their most important characteristics. Section three deals with the extension of the multilayer perceptron (MLP) to complex and quaternionic MLPs. Section four presents the generalization of the feedforward neural networks in the geometric algebra system. Section five describes the generalized learning rule across different geometric algebras. Section six presents various comparative experiments of geometric neural networks with real valued MLPs. The last section discusses the suitability of the geometric algebra system for neural computing.

2 Real Valued Neural Networks

The approximation of nonlinear mappings using neural networks is useful in various areas of signal processing like pattern classification, prediction, system modelling and identification. This section reviews the fundamentals of standard real valued feedforward architectures.

Cybenko [5] used for the approximation of a given continuous function $g(\mathbf{x})$ the superposition of weighted functions:

$$y(\mathbf{x}) = \sum_{j=1}^{N} w_j \sigma_j(\mathbf{w}_j^T \mathbf{x} + \theta_j), \tag{1}$$

where $\sigma(.)$ is a continuous discriminatory function like a sigmoid, $w_j \in \mathcal{R}$ and $\mathbf{x}, \theta_j, \mathbf{w}_j \in \mathcal{R}^n$. The finite sums of the form of Eq. (1) are dense in $C^0(I_n)$ if $|g(\mathbf{x}) - y(\mathbf{x})| < \varepsilon$ for a given $\varepsilon > 0$ and all $\mathbf{x} \in [0,1]^n$. This is called a *density theorem* and is a fundamental concept in approximation theory and nonlinear system modelling [5, 13].

A structure with k outputs y_k having several layers using logistic functions is known as the Multilayer Perceptron (MLP) [22]. The output of any neuron of a hidden layer or of the output layer are represented in similar way,

$$o_j = f_j(\sum_{i=1}^{N_i} w_{ji} x_{ji} + \theta_j) \qquad\qquad y_k = f_k(\sum_{j=1}^{N_j} w_{kj} o_{kj} + \theta_k), \tag{2}$$

where $f_j(\cdot)$ is logistic and $f_k(\cdot)$ is logistic or linear. Linear functions at the outputs are often used for pattern classification. In some tasks of pattern classification suffices one hidden layer whereas in some tasks of automatic control it may be required two hidden layers. Hornik [13] showed that standard multilayer feedforward networks are able to approximate accurately any measurable function to a desired degree. Thus they can be seen as *universal approximators*. In case of a training failure we should rather attribute to an inadequate learning, incorrect number of hidden neurons or a poor deterministic relation between input and output patterns.

Poggio and Girosi [19] developed the Radial Basis Function (RBF) network which consists of a superposition of weighted Gaussian functions as follows

$$y_j(\mathbf{x}) = \sum_{i=1}^{N} w_{ji} G_i(\mathtt{D}_i(\mathbf{x} - \mathbf{t}_i)) \tag{3}$$

where y_j is the j-output, $w_{ji} \in \mathcal{R}$, G_i is a Gaussian function, $\mathtt{D}_i$ a $N \times N$ dilatation diagonal matrix and $\mathbf{x}, \mathbf{t}_i \in \mathcal{R}^n$. The vector $\mathbf{t}_i$ is a translation vector. This architecture is supported by the regularization theory.

3 Complex MLP and Quaternionic MLP

A MLP is extended in the complex domain when its weights, activation function and outputs are complex valued. Yet, the selection of the activation function is a non-trivial matter. For example, the extension of the sigmoid function from $\mathcal{R}$ to $\mathcal{C}$, i.e.

$$\boldsymbol{f}(\boldsymbol{z}) = \frac{1}{(1+e^{-\boldsymbol{z}})} \tag{4}$$

where $\boldsymbol{z} \in \mathcal{C}$, is not allowed as this function is analytic and unbounded [7]. Similar is the case of $\tanh(\boldsymbol{z})$ and $e^{-\boldsymbol{z}^2}$. This kind of activation functions troubles the convergence during training due to its singularities. The necessary conditions that a complex activation $\boldsymbol{f}(\boldsymbol{z}) = a(x,y) + ib(x,y)$ has to fulfill are: $\boldsymbol{f}(\boldsymbol{z})$ non-linear in x and y, partial derivatives a_x, a_y, b_x and b_y exist (where $a_x b_y \not\equiv b_x a_y$) and $\boldsymbol{f}(\boldsymbol{z})$ is not entire. Accordingly Georgiou and Koutsougeras [7] proposed

$$\boldsymbol{f}(\boldsymbol{z}) = \frac{\boldsymbol{z}}{c + \frac{1}{r}|\boldsymbol{z}|} \tag{5}$$

where $c, r \in \mathcal{R}^+$. These authors extended the usual real back-propagation learning rule for the Complex MLP (CMLP).

Arena et al.[1] introduced the Quaternionic MLP (QMLP) which is an extension of the CMLP. The weights, activations functions and outputs of this net are represented in terms of quaternions [8]. They choose the following non-analytic bounded function

$$\begin{aligned}\boldsymbol{f}(\boldsymbol{q}) &= \boldsymbol{f}(q_0 + q_1 i + q_2 j + q_3 k)\\ &= (\frac{1}{1+e^{-q_0}}) + (\frac{1}{1+e^{-q_1}})i + (\frac{1}{1+e^{-q_2}})j + (\frac{1}{1+e^{-q_3}})k,\end{aligned} \tag{6}$$

where $\boldsymbol{f}(\cdot)$ is now the function for quaternions. These authors proved that superpositions of such functions approximate accurately any continuous quaternionic function defined in the unit polydisc of $\mathcal{C}^n$. The extension of the training rule along the lines of the CMLP was done straightforwardly [1].

4 Geometric Algebra Neural Networks

Real, complex and quaternionic neural networks can be further generalized in the Clifford or geometric algebra framework. The weights, the activation functions and the outputs will be now represented using multivectors. In the real valued neural networks of section 3, the vectors are multiplied with the weights using the scalar product. For geometric neural networks the scalar product will be substituted by the Clifford or geometric product.

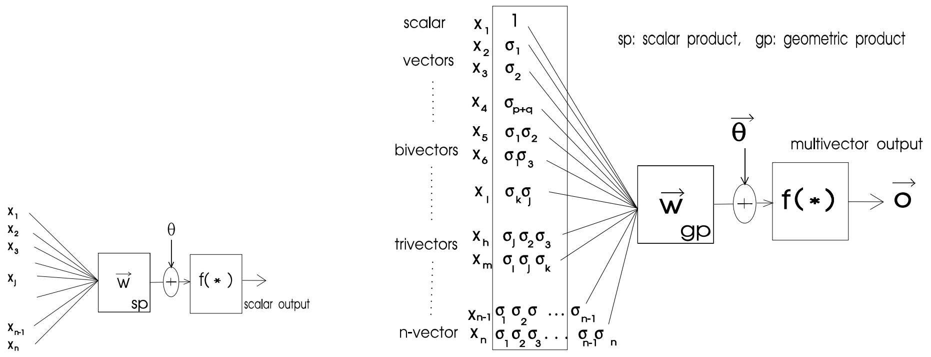

Fig. 1. *McCulloch-Pitts Neuron and Geometric Neuron*

4.1 The activation function

The activation function of Eq. (5) used for the CMLP was extended by Pearson and Bisset [15] for a type of Clifford MLP for different Clifford algebras including the quaternion algebra. In this paper we propose an activation function which affects each multivector basis element. This function was introduced independently by the authors [2] and is in fact a generalization of the function of Arena et al [1]. The function for a n-dimensional multivector $\boldsymbol{m}$ reads

$$\begin{aligned}\boldsymbol{f}(\boldsymbol{m}) &= \boldsymbol{f}(m_0 + m_i\sigma_i + m_j\sigma_j + m_k\sigma_k + \ldots + m_{ij}\sigma_i\wedge\sigma_j + \ldots + \\ &\qquad m_{ijk}\sigma_i\wedge\sigma_j\wedge\sigma_k + \ldots + m_n\sigma_1\wedge\sigma_2\wedge\ldots\wedge\sigma_n) \\ &= f(m_0) + f(m_i)\sigma_i + f(m_j)\sigma_j + f(m_k)\sigma_k + .. + f(m_{ij})\sigma_i\wedge\sigma_j + \ldots + \\ &\qquad f(m_{ijk})\sigma_i\wedge\sigma_j\wedge\sigma_k + \ldots + f(m_n)\sigma_1\wedge\sigma_2\wedge\ldots\wedge\sigma_n, \end{aligned} \tag{7}$$

where $\boldsymbol{f}(\cdot)$ is written in bold to be distinguished from the one used for a single argument $f(\cdot)$. The values of $f(\cdot)$ can be of the type sigmoid or Gaussian.

4.2 The geometric neuron

The McCulloch-Pitts neuron uses the scalar product of the input vector and its weight vector [22]. The extension of this model to the geometric neuron requires the substitution of the scalar product with the Clifford or geometric product, i.e.

$$\mathbf{w}^T\mathbf{x} + \theta \qquad \Rightarrow \qquad \boldsymbol{w}\boldsymbol{x} + \boldsymbol{\theta} = \boldsymbol{w}\cdot\boldsymbol{x} + \boldsymbol{w}\wedge\boldsymbol{x} + \boldsymbol{\theta} \tag{8}$$

Figure 1 shows in detail the McCulloch-Pitts neuron and the geometric neuron. This figure also depicts the way how the input pattern is formated in a specific geometric algebra. The geometric neuron outputs a more rich kind of pattern, let us illustrate this with an example in $\mathcal{G}_{3,0}$

$$\begin{aligned}\boldsymbol{o} &= \boldsymbol{f}(\boldsymbol{w}\boldsymbol{x}+\boldsymbol{\theta})\\ &= \boldsymbol{f}(s_0+s_1\sigma_1+s_2\sigma_2+s_3\sigma_3+s_4\sigma_1\sigma_2+s_5\sigma_1\sigma_3+s_6\sigma_2\sigma_3+s_7\sigma_1\sigma_2\sigma_3)\\ &= f(s_0)+f(s_1)\sigma_1+f(s_2)\sigma_2+f(s_3)\sigma_3+f(s_4)\sigma_1\sigma_2+...+\\ &\qquad f(s_5)\sigma_1\sigma_3+f(s_6)\sigma_2\sigma_3+f(s_7)\sigma_1\sigma_2\sigma_3, \end{aligned} \tag{9}$$

where $\boldsymbol{f}$ is the activation function defined in Eq. (7) and $s_i \in \mathcal{R}$. Using the McCulloch-Pitts neuron in the real valued neural networks the output is simply a scalar given by

$$o = f(\sum_i^N w_i x_i + \theta). \tag{10}$$

The geometric neuron outputs a signal with more geometric information

$$\boldsymbol{o} = \boldsymbol{f}(\boldsymbol{w}\boldsymbol{x}+\boldsymbol{\theta}) = \boldsymbol{f}(\boldsymbol{w}\cdot\boldsymbol{x}+\boldsymbol{w}\wedge\boldsymbol{x}+\boldsymbol{\theta}) \tag{11}$$

which on the one hand has the scalar product like the McCulloch-Pitts neuron, i.e.

$$\boldsymbol{f}(\boldsymbol{w}\cdot\boldsymbol{x}+\theta) = f(s_0) \equiv f(\sum_i^N w_i x_i + \theta) \tag{12}$$

and on the other hand the outer product expressed by

$$\begin{aligned}\boldsymbol{f}(\boldsymbol{w}\wedge\boldsymbol{x}+\boldsymbol{\theta}-\theta) &= f(s_1)\sigma_1+f(s_2)\sigma_2+f(s_3)\sigma_3+f(s_4)\sigma_1\sigma_2+...+\\ &\qquad f(s_5)\sigma_1\sigma_3+f(s_6)\sigma_2\sigma_3+f(s_7)\sigma_1\sigma_2\sigma_3. \end{aligned} \tag{13}$$

Note that the outer product supplies the scalar cross-products between the individual components of the vector which are nothing else as the multivector components of higher grade like point or lines (vectors), planes (bivectors) and volumes (trivectors). This characteristic will be used in section 7.2 for the implementation of the embedded geometric processing in the extended geometric neural networks. These kind of neural networks resemble to the higher order neural networks, however the extended geometric neural networks use not only scalar products of higher order but all the necessary scalar cross-products for carrying out a geometric cross-correlation. That is why a geometric neuron can be seen as a sort of geometric correlator which for the interpolation offers in contrast to the McCulloch-Pitts neuron not only points but also higher grade multivector components like planes, volumes,...,hyper-volumes.

4.3 Feedforward geometric neural networks

Figure (2) depicts the standard neural and network structures for function approximation in the geometric algebra framework. Here the inner vector product

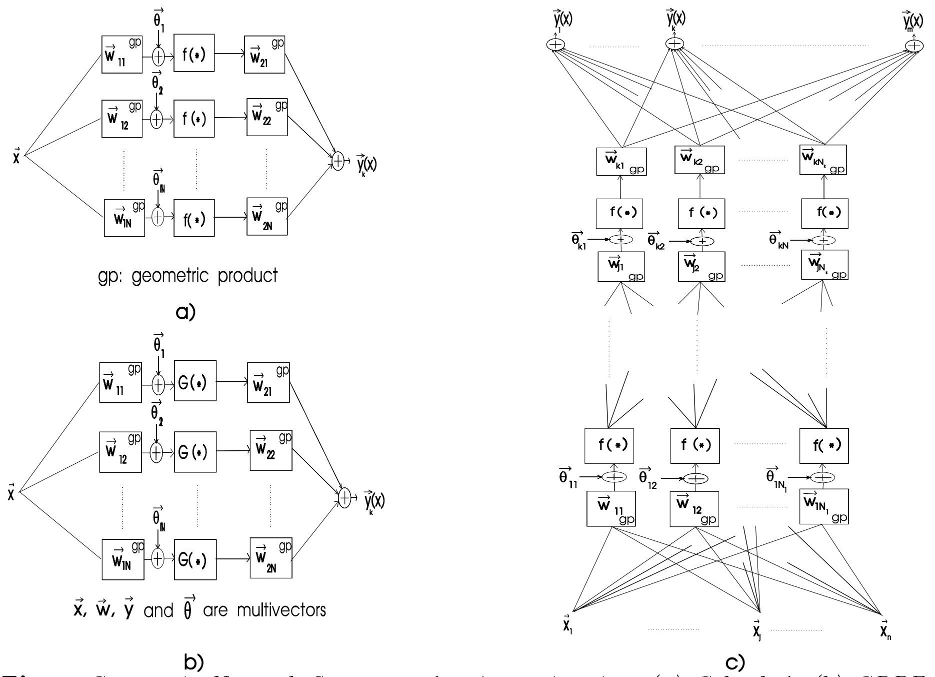
gp: geometric product
a)
$\vec{x}$, $\vec{w}$, $\vec{y}$ and $\vec{\theta}$ are multivectors
b)
c)

Fig. 2. *Geometric Network Structures for Approximation: (a) Cybenko's (b) GRBF network (c) GMLP$_{p,q}$*

has been extended to the geometric product and the activation functions are according (7).

The equation (1) of the Cybenko's model in geometric algebra reads

$$\boldsymbol{y}(\boldsymbol{x}) = \sum_{j=1}^{N} \boldsymbol{w}_j \boldsymbol{f}(\boldsymbol{w}_j \cdot \boldsymbol{x} + \boldsymbol{w}_j \wedge \boldsymbol{x} + \boldsymbol{\theta}_j). \tag{14}$$

The extension of the MLP is straightforward. The equations using the geometric product of the outputs of hidden and output layers read

$$\begin{aligned} \boldsymbol{o}_j &= \boldsymbol{f}_j(\sum_{i=1}^{N_i} \boldsymbol{w}_{ji} \cdot \boldsymbol{x}_{ji} + \boldsymbol{w}_{ji} \wedge \boldsymbol{x}_{ji} + \theta_j) \\ \boldsymbol{y}_k &= \boldsymbol{f}_k(\sum_{j=1}^{N_j} \boldsymbol{w}_{kj} \cdot \boldsymbol{o}_{kj} + \boldsymbol{w}_{kj} \wedge \boldsymbol{o}_{kj} + \theta_k) \end{aligned} \tag{15}$$

In the radial basis function networks, the dilatation operation (via the diagonal matrix $\mathtt{D_i}$) can be implemented by means of a geometric product with a dilation $\boldsymbol{D}_i = e^{\alpha \frac{\boldsymbol{i}\bar{i}}{2}}$ [10], i.e.

$$\mathtt{D}_i(\mathbf{x} - \mathbf{t}_i) \Rightarrow \boldsymbol{D}_i(\boldsymbol{x} - \boldsymbol{t}_i)\tilde{\boldsymbol{D}}_i \tag{16}$$

$$\boldsymbol{y}_k(\boldsymbol{x}) = \sum_{j=1}^{N} \boldsymbol{w}_{kj} \boldsymbol{G}_j(\boldsymbol{D}_j(\boldsymbol{x}_{ji} - \boldsymbol{t}_j)\tilde{\boldsymbol{D}}_j) \tag{17}$$

Note that in the case of the geometric RBF we are also using an activation function according the Eq. (7).

5 Learning Rule

This section presents the multidimensional generalization of the gradient descent learning rule in the geometric algebra framework. This rule can be used for the Geometric MLP (GMLP) and for tuning the weights of the Geometric RBF (GRBF). Previous learning rules for the real valued MLP, complex MLP [7] and the quaternionic MLP [1] are simply especial cases of this extended rule.

5.1 Generalized multi-dimensional back-propagation training rule

The norm of a multivector $\boldsymbol{x}$ for the learning rule reads

$$|\boldsymbol{x}| = (\boldsymbol{x}|\boldsymbol{x})^{\frac{1}{2}} = \big(\sum_{A} [\boldsymbol{x}]_A^2\big)^{\frac{1}{2}}. \tag{18}$$

The geometric neural network with n inputs and m outputs is supposed to approximate the target mapping function

$$\mathcal{Y}_t : (\mathcal{G}_{p,q})^n \rightarrow (\mathcal{G}_{p,q})^m, \tag{19}$$

where $(\mathcal{G}_{p,q})^n$ is the n-dimensional module over the geometric algebra $\mathcal{G}_{p,q}$ [15]. The error at the output of the net is measured according the metric

$$E = \frac{1}{2} \int_{\boldsymbol{x} \in \boldsymbol{X}} ||\mathcal{Y}_w - \mathcal{Y}_t||^2, \tag{20}$$

where $\boldsymbol{X}$ is some compact subset of the Clifford module $(\mathcal{G}_{p,q})^n$ involving the product topology derived from equation (18) for the norm and $\mathcal{Y}_w$ and $\mathcal{Y}_t$ are the learned and target mapping functions respectively. The back-propagation algorithm [22] is a procedure for updating the weights and biases. This algorithm is a function of the negative derivative of the error function (Eq. (20)) with respect to the weights and biases themselves. The computing of this procedure is straightforward and here we will only give the main results. The updating equation for the multivector weights of any hidden $j-$layer is

$$\boldsymbol{w}_{ij}(t+1) = \eta\big[(\sum_{k}^{N_k} \delta_{kj} \otimes \overline{\boldsymbol{w}_{kj}}) \odot \boldsymbol{F}'(\boldsymbol{net}_{ij})\big] \otimes \overline{\boldsymbol{o}_i} + \alpha \boldsymbol{w}_{ij}(t), \tag{21}$$

for any $k-$output with a non-linear activation function

$$\boldsymbol{w}_{jk}(t+1) = \eta\big[(\boldsymbol{y}_{k_t} - \boldsymbol{y}_{k_a}) \odot \boldsymbol{F}'(\boldsymbol{net}_{jk})\big] \otimes \overline{\boldsymbol{o}_j} + \alpha \boldsymbol{w}_{jk}(t), \tag{22}$$

and for any $k-$output with a linear activation function,

$$\boldsymbol{w}_{jk}(t+1) = \eta(\boldsymbol{y}_{k_t} - \boldsymbol{y}_{k_a}) \otimes \overline{\boldsymbol{o}_j} + \alpha \boldsymbol{w}_{jk}(t), \tag{23}$$

where $\boldsymbol{F}$ is the activation function defined in equation (7), t is the update step, η and α are the learning rate and the momentum respectively, $\otimes$ defined for clearness is the Clifford or geometric product, $\odot$ is a scalar component by component product and $\overline{(\cdot)}$ is a multivector antiinvolution (reversion or conjugation).

In the case of the non-Euclidean geometric algebra $\mathcal{G}_{0,3}$ $\overline{(\cdot)}$ corresponds to the simple conjugation. Each neuron now consist of p+q units, each for a multivector component. The biases are also multivectors and are absorbed as usual in the sum of the activation signal called here $\boldsymbol{net}_{ij}$. In the learning rules, Eqs. (21)-(23), the way how the geometric product and the antiinvolution are computed varies according the geometric algebra being used [20]. As illustration we give the conjugation required in the learning rule for the quaternion algebra or $\mathcal{G}_{0,2}$, where the index indicates the number of the basis vector, i.e. 0 for 1, 1 for σ_1, 2 for σ_2 and 3 for$\sigma_1\sigma_2$. The conjugation reads: $\bar{\boldsymbol{x}} = x_0 - x_1\sigma_1 - x_2\sigma_2 - x_3\sigma_1\sigma_2$, where $\boldsymbol{x} \in \mathcal{G}_{0,2}$.
The reversion in case of non-Euclidean $\mathcal{G}_{0,3}$ is given by $\bar{\boldsymbol{x}} = x_0 + x_1\sigma_1 + x_2\sigma_2 + x_3\sigma_3 - x_4\sigma_1\sigma_2 - x_5\sigma_2\sigma_3 - x_6\sigma_3\sigma_1 - x_7 i$.

5.2 Simplification of the learning rule using the density theorem

Given $\boldsymbol{X}$ and $\boldsymbol{Y}$ as compact subsets belonging to $(\mathcal{G}_{p,q})^n$ and to $(\mathcal{G}_{p,q})^m$ respectively and considering $\mathcal{Y}_t : \boldsymbol{X} \to \boldsymbol{Y}$ a continuous function, there are some coefficients $w_1, w_2, w_3, ..., w_{N_j} \in \mathcal{R}$ and some multivectors $\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{y}_3, ..., \boldsymbol{y}_{N_j} \in \mathcal{G}_{p,q}$ and $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3, ..., \boldsymbol{\theta}_{N_j} \in \mathcal{G}_{p,q}$ so that the following inequality $\forall \epsilon > 0$ is valid

$$E(\mathcal{Y}_t, \mathcal{Y}_w) = sup\big[|\mathcal{Y}_t(\boldsymbol{x}) - \sum_{j=1}^{N_j} w_j \boldsymbol{f}_j(\sum_{i=1}^{N_i} \boldsymbol{w}_i\boldsymbol{x} + \boldsymbol{\theta}_i)|\boldsymbol{x} \in \boldsymbol{X}\big] < \epsilon, \tag{24}$$

where $\boldsymbol{f}_j$ is the multivector activation function of Eq. (7). Here the approximation given by

$$S = \sum_{j=1}^{N_j} w_j \boldsymbol{f}_j(\sum_{i=1}^{N_i} \boldsymbol{w}_i\boldsymbol{x} + \boldsymbol{\theta}_i) \tag{25}$$

is the subset of the class of functions $C^0(\mathcal{G}_{p,q})$ with the norm $|\mathcal{Y}_t|$=$sup_{\boldsymbol{x}\in\boldsymbol{X}}|\mathcal{Y}_t(\boldsymbol{x})|$. Since equation (24) is true we can finally say that S is dense in $C^0(\mathcal{G}_{p,q})$. The density theorem presented here is the generalization of the one used for the quaternionic MLP by Arena et al. [1]. Its complete prove will be published elsewhere.

The density theorem shows that for the training of geometric feedforward networks the weights of the output layer could be real values. Therefore the training of the output layer can be simplified, i.e. the output weight multivectors could be scalars of k-grade. This k-grade element of the multivector is selected by convenience.

5.3 Learning using the appropriate geometric algebras

The main motivation to process signals in the geometric algebra framework is to have access to representations with a strong geometric character and to take advantage of the geometric product. An important question arises regarding the type of geometric algebra we should use for a specific problem. For some application the modelling of the suitable space would be straightforward. However in other cases it would be somehow difficult unless some a priori knowledge of the problem is available. In case we do not have any clue we should then explore various network topologies in different geometric algebras. This will require some orientation about the different geometric algebras we could use. Since each geometric algebra is either isomorphic to a matrix algebra of $\mathcal{R}$, $\mathcal{C}$ or $\mathcal{H}$ or simply a product of these algebras a great care has to be taken for choosing the algebras. Porteous [20] showed the isomorphisms

$$\mathcal{G}_{p+1,q} = \mathcal{R}_{p+1,q} \cong \mathcal{G}_{q+1,p} = \mathcal{R}_{q+1,p} \tag{26}$$

and presented the following expressions for completing the universal table of geometric algebras

$$\begin{aligned} \mathcal{G}_{p,q+4} = \mathcal{R}_{p,q+4} \cong \mathcal{R}_{p,q} \otimes \mathcal{R}_{0,4} \cong \mathcal{R}_{0,4} \cong \mathcal{H}(2) \\ \mathcal{G}_{p,q+8} = \mathcal{R}_{p,q+8} \cong \mathcal{R}_{p,q} \otimes \mathcal{R}_{0,8} \cong \mathcal{R}_{0,8} \cong \mathcal{R}(16), \end{aligned} \tag{27}$$

where $\otimes$ stands for the real tensor product of two algebras. The last equation is known as the periodicity theorem [20]. Examples of this table are R$\cong \mathcal{G}_{0,0}$, $\mathcal{R}_{0,1} \cong \mathcal{C} \cong \mathcal{G}_{0,1}$, $\mathcal{H} \cong \mathcal{G}_{0,2}$ and $\mathcal{R}_{1,1} \cong {}^2\mathcal{R} \cong \mathcal{G}_{1,1}$, $\mathcal{C}(2) \cong \mathcal{C} \otimes \mathcal{R}(2) \cong \mathcal{G}_{3,0} \cong \mathcal{G}_{1,2}$ for the 3D space and $\mathcal{H}(2) \cong \mathcal{G}_{1,3}$ for the 4D space. The two later examples correspond to the geometric algebras mentioned in section 2.

6 Experiments

In this section the GMLP using the geometric algebra $\mathcal{G}_{p,q}$ will be denoted as GMPL$_{p,q}$. Firstly we test the component-wise activation function and then investigate new ways of geometric preprocessing embedded in the first network layer. Finally we identify the key role of the geometric learning in 3D object recognition and prediction in chaotic processes.

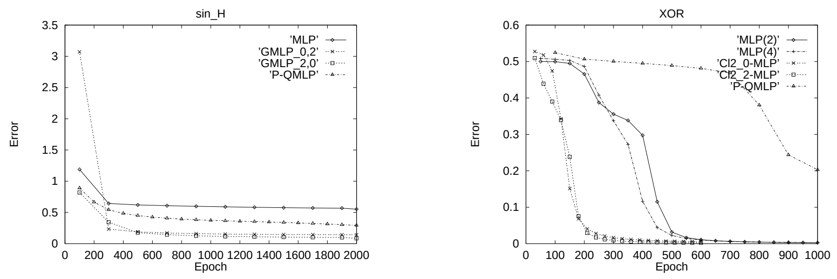

Fig. 3. *a) Learning* sin_H *(left) b) XOR (right) using the MLP(2 and 4),* $GMLP_{0,2}$ *,* $GMLP_{2,0}$ *and P-QMLP.*

6.1 Inputs without any preprocessing

First the performance of the geometric net was tested using the sigmoid multivector activation function (7) against the real valued one and the one of Person (P-QMLP) [15]. The function to be approximated is $\sin_H = \sin(\boldsymbol{q})$, where $\boldsymbol{q} \in \mathcal{H}$. All nets had the same 4-D input and 4-D output. The MLP uses 4 inputs and the GMLPs one 4-D input. The MLP had 3 hidden nodes and the GMLPs 3 geometric hidden nodes. Figure 3.a shows the performance of MLP, $\text{GMLP}_{0,2}$, $\text{GMLP}_{2,0}$ and the P-QMLP. In this figure the vertical axis indicates the total error per epoch. We can see clearly that the $\text{GMLP}_{0,2}$ and the $\text{GMLP}_{2,0}$ have a better performance than the P-QMLP. The reason is probably that during learning the activation function of the P-QMLP suffers of the consequence a zero division which slows down the convergence rate. In order to verify the generalization capability using different amount of hidden neurons after the training the nets MLP, $\text{GMLP}_{0,2}$ and $\text{GMLP}_{2,0}$ were tested using 50 before unseen patterns. For this evaluation the mean error was considered. Table 1 shows that the GMLPs have almost a similar generalization capability using 4 hidden nodes than the MLP, better using 3 and much better using just 2. The P-QMLP is not considered due to its poor performance. These experiments show so far that the complex neuron of the GMLPs process compactly the information better than the real neuron of the MLP. That is because the GMLPs have more weights due to their multivector valued nodes and presumably in this example the GMLPs profit of the geometric product based processing.

	4	3	2
MLP	0.0827	0.5014	1.0145
$\text{GMLP}_{0,2}$	0.0971	0.1592	0.2243
$\text{GMLP}_{2,0}$	0.0882	0.1259	0.1974

*T*able 1. Mean error by different amount of hidden nodes.

A dramatic confirmation that the component-wise activation function (7) works much better than the one of Pearson (P-QMLP) is observed when testing the XOR problem. Figure 3.b shows that geometric nets $\mathrm{GMLP}_{0,2}$ and $\mathrm{GMLP}_{2,0}$ have a faster convergence rate than the MLP with 2- and 4- dimensional inputs and by far than the P-QMLP. Since the MLP(4) working even in 4D can not beat the GMLP, it can be claimed that the better performance of the geometric neural network is not only due to the higher dimensional quaternionic inputs but rather due to the algebraic advantages of the geometric neurons of the net.

6.2 Embedded geometric preprocessing

This subsection shows experiments with geometric neural networks with an embedded geometric preprocessing in the first layer. This has been done in order to capture the geometric cross-products of the input data much better. This proved to improve the convergence. In this paper these kind of nets will be called $\mathrm{EGMLP}_{p,q}$. Figure 2 shows a geometric network with its extended first layer. The function sin_H is again employed to test the performance of the MLP,

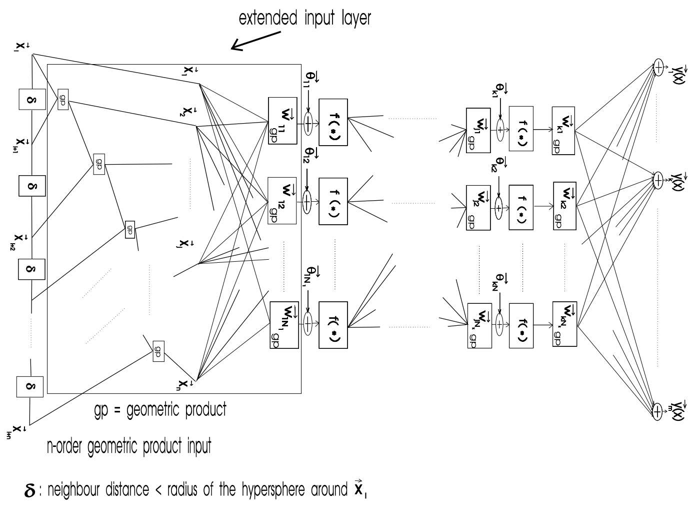
extended input layer
gp = geometric product
n-order geometric product input
δ : neighbour distance < radius of the hypersphere around x

Fig. 4. *Geometric neural network with extended input layer*

$\mathrm{GMLP}_{0,2}$ and $\mathrm{GMLP}_{2,0}$. Here we use as first input $\boldsymbol{x}_i$ and as second one the geometric product $\boldsymbol{x}_i\boldsymbol{x}_{i+1}$ (second order) or $\boldsymbol{x}_i\boldsymbol{x}_{i+1}\boldsymbol{x}_{i+2}\boldsymbol{x}_{i+3}$ (fourth order). Figure 3 shows that the performance of the three networks improves. It is clear that the extension of the first layer of the geometric network helps. Since the MLP uses the same type of inputs, its improvement relays also on this kind of geometric preprocessing.

6.3 Geometric RBF networks for 3D object recognition

In this section we explore the potential of the geometric RBF (GRBF) net to deal with the geometric nature of the task in question. The net is supposed to

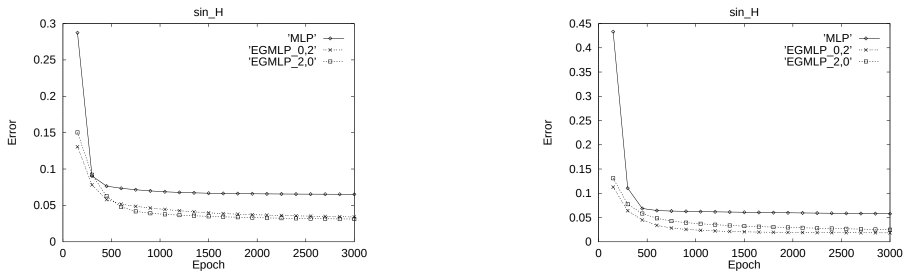

Fig. 5. *Learning* sin_H *using MLP ,* $EGMLP_{0,2}$ *and* $EGMLP_{2,0}$ *with second (left) and fourth order geometric product inputs (right).*

recognize a wire–frame 3D object from any of its perspective views. The object attributes are its N feature points, N-1 lengths and N-2 angles, see Figure 4a. The nets are trained on several random views and should map any view of the same object into a standard view. We trained real valued RBF neural networks and GRBF nets using the embedded geometric processing.

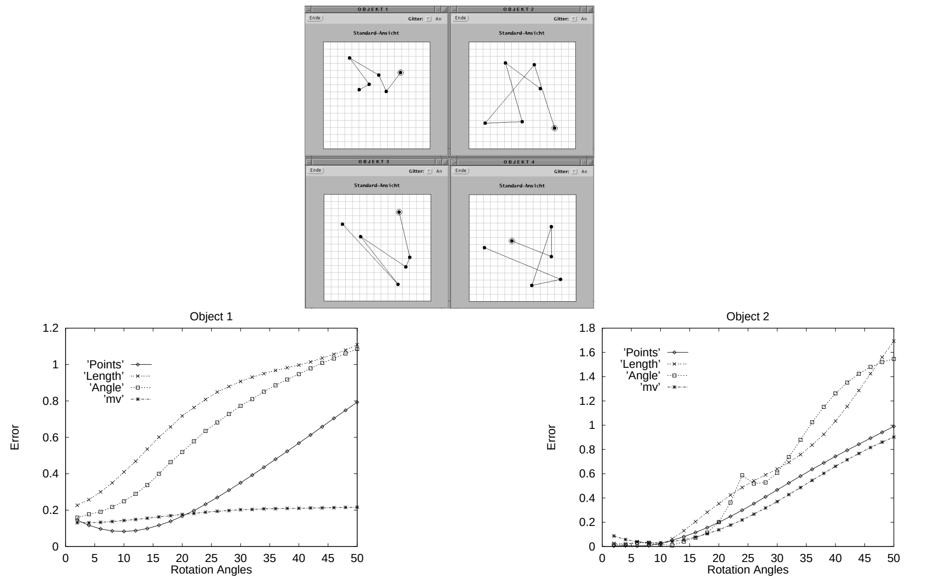

Fig. 6. *a) Wire frames b-c) Recognition error by RBF net using points or lines or angles and by* $GRBF_{0,2}$ *using* $\boldsymbol{x}_i\boldsymbol{x}_{i+1}$ *(mv). The two rotation angles are equal.*

In Figure 4b-c we can see for two of the objects that the GRBF$_{0,2}$ with $\boldsymbol{x}_i\boldsymbol{x}_{i+1}$ (second order embedded preprocessing depicted as *mv*) in a range from 0 to 50 degrees performs the generalization better than a real valued RBF using points or lengths or angles. This results should encourage researchers to apply this kind of geometric RBF networks with higher order of geometric products for various tasks like recognition of objects and the recover of pose.

6.4 Recognition of geometry in chaotic processes

This experiments shows that the geometric neural networks are well suited to distinguish the geometric information in a chaotic process.

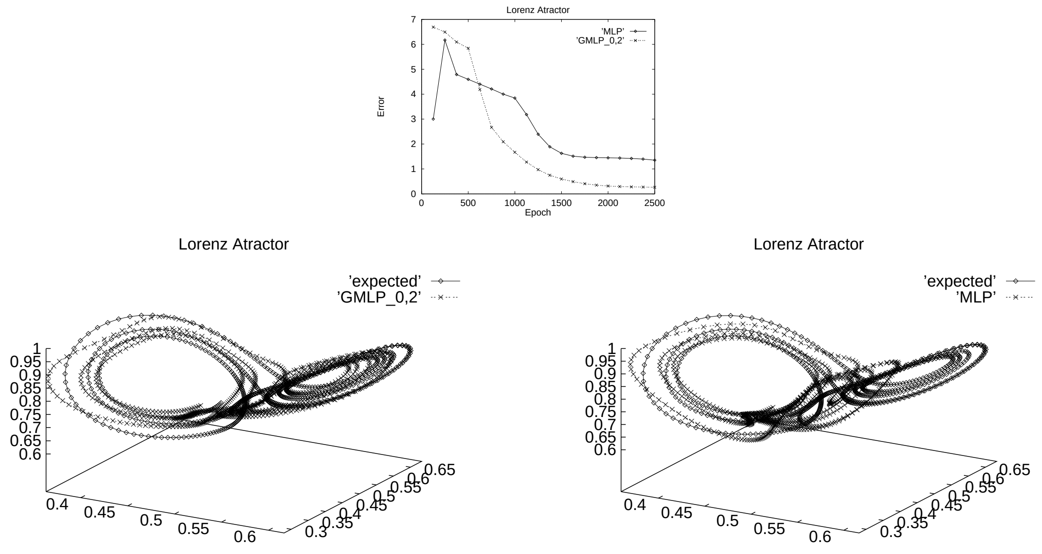

Fig. 7. *a) Training error b) Prediction by $GMLP_{0,2}$ and expected trend c) Prediction by MLP and expected trend.*

For that we used the well known Lorenz attractor (σ=3, r=26.5 and b=1) with the initial conditions [0,1,0] and sample rate 0.02 sec. A 3-12-3 MLP and a 1-4-1 $\mathrm{GMLP}_{0,2}$ were trained in the interval [12, 17] sec. to perform a 8 τ step ahead prediction. The next 750 samples unseen during training were used for the test. The Figure 7.a show the error during training, note that the $\mathrm{GMLP}_{0,2}$ converges faster than the MLP. Interesting is how they behave by the prediction. The figures 7b-c shows that the $\mathrm{GMLP}_{0,2}$ predicts better than the MLP. Analyzing the covariance parameters of the MLP [0.96815, 0.67420,0.95675] and of the $\mathrm{GMLP}_{0,2}$ [0.9727, 0.93588, 0.95797] we can see that the MLP requires longer to get the geometry involved in the second variable, that is why the convergence is slower. As a result of that the MLP loses control to predict well in the other side of the looping. On contrast the geometric net from early stage captures the geometric characteristics of the attractor so it can not fail if it has to predict in the other side of the looping.

7 Conclusions

This paper started with basic reflections regarding geometry in biological creatures. The complexity of the mapping between the external and the internal world demands that the representation and calculus has to be carried out in a coordinate-free mathematical system with a strong algebraic and geometric characteristic. According the literature there are basically two mathematical systems used in neural computing: the tensor algebra and the matrix algebra. This paper chooses the coordinate-free system of Clifford or geometric algebra. The authors use this general and powerful mathematical system for the analysis and design of multidimensional feedforward neural networks. The reader can see that real-, complex- and quaternionic–valued neural networks are simple particular cases of the geometric algebra multidimensional neural networks. This work shows that the component-wise activation function defeats the activation function used in the complex neural nets [7] and in the hypercomplex nets by Pearson [15]. In case of the XOR problem the MLP using a 2-D or 4-D coded inputs can not perform as well as the GMLPs. The authors show also how the embedded geometric processing in the first layer helps to capture the geometric correlation of the data. The algebraic character of the nets is due to the activation function of the geometric neurons and the operations through the layers. The GA algebra is a coordinate free approach for neural nets. This can be seen by the experiment with GRBF net where the geometric products capture the geometric relations of the lines using the bivector between points liberating in this way the coordinate dependency existing in the point manifold. The ability of the geometric neural networks to recognize the geometric characteristics during the dynamic evolution of a chaotic process exemplifies the power of geometric neural learning for prediction.

Acknowledgment

This work was financed by the Deutsche Forschungsgemeinschaft project SO 320-2-1 Geometrische Algebra ein Repräsentationsrahmen für den Wahrnehmungs-Handlungs-Zyklus.

References

1. Arena P., Caponetto R., Fortuna L., Muscato G. and Xibilia M.G. 1996. Quaternionic multilayer perceptrons for chaotic time series prediction. IEICE Trans. Fundamentals. Vol. E79-A. No. 10 October, pp. 1-6.
2. Bayro-Corrochano E., Buchholz S., Sommer G. 1996. Selforganizing Clifford neural network *IEEE ICNN'96 Washington, DC*, June, pp. 120-125.

3. Bayro-Corrochano E., Lasenby J., Sommer G. Geometric Algebra: A framework for computing point and line correspondences and projective structure using n-uncalibrated cameras *IEEE Proceedings of ICPR'96 Viena, Austria*, Vol. I, pages 334-338, August, 1996.
4. Bayro-Corrochano E., Daniilidis K. and Sommer G. 1997. Hand-eye calibration in terms of motion of lines using Geometric Algebra. To appear in SCIA'97, June 9-11,Lappeenranta, Finland.
5. Cybenko G. Approximation by superposition of a sigmoidal function. *Mathematics of control, signals and systems*, Vol. 2, 303:314, 1989.
6. Doran C.J.L. 1994. Geometric algebra and its applications to mathematical physics. *Ph.D. Thesis*, University of Cambridge.
7. Georgiou G. M. and Koutsougeras C. Complex domain backpropagation. *IEEE Trans. on Circuits and Systems*, 330:334, 1992.
8. Hamilton W.R. 1853. Lectures on Quaternions. Hodges and Smith, Dublin.
9. Hestenes D. 1966. Space-Time Algebra. *Gordon and Breach.*
10. Hestenes D. and Sobczyk G. 1984. Clifford Algebra to Geometric Calculus: A unified language for mathematics and physics. *D. Reidel*, Dordrecht.
11. Hestenes D. 1993. Invariant body kinematics I: Saccadic and compensatory eye movements. Neural Networks, Vol. 7, 65-77.
12. Hestenes D. 1993. Invariant body kinematics II: Reaching and neurogeometry. Neural Networks, Vol. 7, 79-88.
13. Hornik K. 1989. Multilayer feedforward networks are universal approximators. Neural Networks, Vol. 2, pp. 359-366.
14. Koenderink J. J. 1990. The brain a geometry engine. Psychological Research, Vol. 52, pp. 122-127.
15. Pearson J. K. and Bisset D.L. . 1992. Back Propagation in a Clifford Algebra. *Artificial Neural Networks, 2, I. Aleksander and J. Taylor (Ed.)*, 413:416.
16. Pellionisz A. and Llinás R. 1980. Tensorial approach to the geometry of brain function: cerebellar coordination via a metric tensor. Neuroscience Vol. 5, pp. 1125-1136
17. Pellionisz A. and Llinás R. 1985. Tensor network theory of the metaorganization of functional geometries in th central nervous system. Neuroscience Vol. 16, No. 2, pp. 245-273.
18. Perantonis S. J. and Lisboa P. J. G. 1992. Translation, rotation, and scale invariant pattern recognition by high-order neural networks and moment classifiers. IEEE Trans. on Neural Networks, Vol. 3, No. 2, March, pp 241-251.
19. Poggio T. and Girosi F. Networks for approximation and learning. *IEEE Proc.*, Vol. 78, No. 9, 1481:1497, Sept. 1990.
20. Porteous I.R. 1995. Clifford Algebras and the Classical Groups. *Cambridge University Press*, Cambridge.
21. Sommer G., Bayro-Corrochano E. and Bülow T. 1997. Geometric algebra as a framework for the perception–action cycle. Workshop on Theoretical Foundation of Computer Vision, Dagstuhl, March 13-19, 1996, Springer Wien.
22. Rumelhart D.E. and McClelland J. L.. 1986. Parallel Distributed Processing: Explorations in the Microestructure of Cognition. 2 Vols. Cambridge: MIT Press.