Learning Geometric Transformations with Clifford Neurons

Sven Buchholz and Gerald Sommer

Department of Computer Science, University of Kiel
Preusserstr. 1-9, 24105 Kiel, Germany
{sbh,gs}@ks.informatik.uni-kiel.de

Abstract. In this paper we propose a new type of neuron developed in the framework of Clifford algebra. It is shown how this novel Clifford neuron covers complex and quaternionic neurons and that it can compute orthogonal transformations very efficiently. The introduced framework can also be used for neural computation of non–linear geometric transformations which makes it very promising for applications. As an example we develop a Clifford neuron that computes the cross-ratio via the corresponding Möbius transformation. Experimental results for confirmation of the proposed novel neural models are presented.

1 Introduction

The aim of neural computation is to understand neural and cognitive processes in biological systems, in particular in the human brain, in computational terms and to design and analyze models and algorithms towards these goals. The connectionist approaches among these are known as *Neural Networks* (NNs).
Neural networks are interconnections of artificial "neurons" that are greatly simplified versions of biological neurons [14].
The first model of an artificial neuron was proposed already more than 50 years ago by McCulloch and Pitts in 1943 [10]. However, the concept reaches wide public in the Artifical Intelligence community not before the end of the 80s, when suitable learning algorithms for training of multi–layer perceptrons (MLPs) were (re–)discovered in [13]. Nowadays artificial neural networks are very common and used in many application fields ranging from classical ones like pattern recognition or computer linguistics to modern ones like financial forecasting, medical diagnosis systems and energy management systems. Therefore, some kind of paradigm shift from symbolic approaches (programming) to neural networks (learning) could be noticed. A change that has also been pushed by many strong mathematical properties of neural networks that could be proven, e.g. the universal approximation property of sigmoidal MLPs [4].
On the other hand, NNs (like any other existing approach) are still far away from brains in any aspect. Besides that distance to the ultimate goal, there are computational problems too. One is the famous bias/variance trade-off formulated exactly in terms of statistical learning theory in [6], which arises from the

fact that NNs can only learn (in practice) from a finite set of examples. Roughly spoken, the only solution is to integrate a priori knowledge. Actually, many other problems that one faces in neural computation reduce in a broader sense to this type of problem.

For example, it is impossible to predict whether a real valued MLP or a complex valued MLP will better perform a task of learning a complex valued function [9]. Both networks are universal approximators. The therefore required high non–linearities can create networks with high "variance", which also makes rule extraction from such networks difficult. In that kind of monolithic structured networks the integration of domain or task specific knowledge ("bias") is not easy to do.

One solution to overcome that "black–box" behavior of NNs is the design of small specific subnetworks that perform simpler tasks and build up more complicated networks from such a functional base. In terms of supervised learning this so called "model–based" approach [3] also has the advantage that optimization is simplified due to constrained lower dimensional weight spaces. In a recent paper [5] the approach is extended to the neural computation of algebraic and geometric structures like eigenvalues of matrices or surface representations.

In this paper we will follow similar intentions, but will focus exclusively on neural computation of geometric transformations. We will do this because of the fundamental importance of geometric transformations in all aspects of the perception–action cycle related to both pattern generation and recognition. Thus, by following Felix Klein's Erlangen program, we will demonstrate how to design NNs as building blocks of geometric expert knowledge by algebraic constraining neural computation. Linear transformations of real vector spaces can be computed easily by a Single Layer Neural Network (SL–NN). This will be our starting point in the next section. There we will also see, that SL–NNs have some drawbacks. The first and fundamental one is obviously the limitation to real linear transformations, which excludes many interesting transformation like projective or Möbius transformations. The second one results from the fact that only special linear transformations are needed in most applications, like affine transformations in object recognition or rigid motions in robotics.

To overcome both drawbacks, we propose a new type of neuron — the Clifford Neuron, which is formulated in terms of Clifford algebra. Since a Clifford algebra is constructed from a real vector space with a quadratic form on it, concepts of metric geometry can be used easily. It is shown that Clifford neurons compute orthogonal transformations very efficiently. Moreover, they can be used to compute non–linear geometric transformations which allows many interesting applications. As an example, we develop a Clifford neuron that computes the cross-ratio via the corresponding Möbius transformation. Experimental results to confirm the novel proposed neural models are presented.

2 Single Layer Neural Networks

A formal neuron is a computational unit of the form shown in Figure 1. A

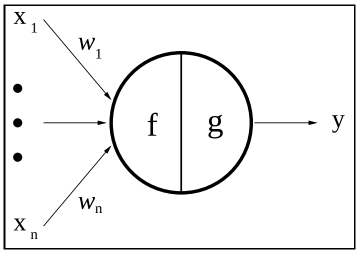

Fig. 1. Formal neuron

propagation function f is used to associate the input vector x with the weight vector w, which comprises the learnable parameters of the neuron. After this an activation function g can be applied. In the linaer case that we want to study g is always set to be the identiy. Association is done via the scalar product, which gives as the output of the neuron

$$y = \sum_{j=1}^{n} w_j x_j + \theta \,, \tag{1}$$

where θ is an additive bias. However, by setting $x^+ := (x, 1)$ and $w^+ := (w, \theta)$ we can rewrite (1) in the form

$$y = \sum_{j=1}^{n+1} w_j^+ x_j^+ =< w^+, x^+ > \ . \tag{2}$$

To formulate a learning rule, we need some more notations. So let us consider a training set $T := \{(x^1, t^1), \ldots, (x^m, t^m)\}$ consisting of input–output pairs (x^i, t^i) with $x^i \in \mathbb{R}^n$, $t^i \in \mathbb{R}$. The goal of learning is to minimize a given error function. The most common error function is the sum-of-squared error (SSE)

$$E = \frac{1}{2} \sum_{i=1}^{m} (t^i - \sum_{j=1}^{n} w_j x_j^i)^2 \,. \tag{3}$$

Minimization is done by using gradient descent. Thus, we have to update the weights of the neuron in each step of the learning algorithm according to

$$\Delta w_j = -\eta \frac{\partial E}{\partial w_j} = \eta \sum_{i=1}^{m} (t^i - \sum_{j=1}^{n} w_j x_j^i) x_j^i \,, \tag{4}$$

where $\eta > 0$ is a suitable learning rate.
Combining real neurons in a single layer that is fully connected to the input gives a SL–NN as shown in Figure 2. The SL-NN computes an ordinary matrix multiplication

$$y = Wx \,, \tag{5}$$

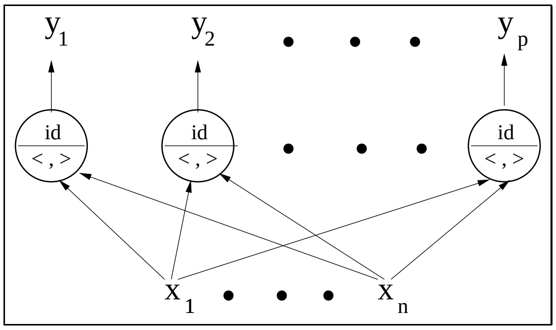

Fig. 2. Single Layer Neural Network (SL–NN)

where W is the weight matrix containing the single neuron weight vectors. If W is square ($x, y \in \mathbb{R}^n$) then a real linear transformation is computed. To overcome the drawback of being limited to the real domain, we could replace every real entity by a complex entity and the scalar product by complex multiplication in (1) to be able to extend the SL–NN to the complex domain. Obviously, every complex neuron would correspond to a single real SL–NN with special weights. Reversing this thought gives the solution of another drawback of computing geometric transformations by SL–NNs in a special case. Suppose we want to compute an Euclidean 2D rotation. Using a SL-NN in this case means to search in the weight space of all 2×2 matrices for the rotation matrix that is the solution. However, using a complex neuron means to search only in the space of all orthogonal 2×2 matrices.
Even more important is the fact that a complex SL–NN would process points in the plane. More general, the problem is the following. Suppose now we want to learn an unknown transformation of an object like that in Figure 3 by giving input/output training views to a SL–NN. More precisely, we have 6 points

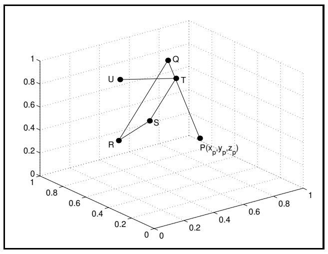

Fig. 3. Wire frame

in space given as input/output. This structural information is lost completely by using a real SL-NN that only sees 1 point in $\mathbb{R}^{18}$. Thus, we have a priori knowledge that can not be integrated.

In the next section we will introduce the framework of Clifford algebra, which contains complex numbers as a special case. That allows to design neurons which integrate geometrical a priori knowledge powerfully.

3 Clifford Algebra and Clifford Neurons

The drawbacks of SL–NN in computing geometric transformations seen before come from the underlying concept. The structural concept of a vector space is too weak for such purposes. An additional structure of a vector space is that implied by a quadratic form Q on it. From this an algebra with the desired properties can be constructed in the following way.
For all $v = (v_1, \ldots, v_n) \in \mathbb{R}^n$ there exists a basis of $\mathbb{R}^n$ such that

$$Q(v) = -v_1^2 - v_2^2 - \ldots - v_p^2 + v_{p+1}^2 \cdots + v_{p+q}^2\,, \tag{6}$$

with $p, q \in \mathbb{N}$ and $p + q = n$. Thus, Q is already determined by (p, q) and so is a scalar product on $\mathbb{R}^n$. Together we call this the quadratic space $\mathbb{R}^{p,q}$. In particular, we obtain from (6) for an orthonormal basis $\{e_1, \ldots, e_n\}$

$$-Q(e_i) = +1 \quad \text{if} \quad i \leq p \tag{7}$$

$$-Q(e_i) = -1 \quad \text{if} \quad i > p\,. \tag{8}$$

This allows the following definition of a Clifford algebra [8].

Definition 1 *An associative algebra with unity 1 over $\mathbb{R}^{p,q}$ containing $\mathbb{R}^{p,q}$ and $\mathbb{R}$ as distinct subspaces is called the Clifford algebra $\mathcal{C}_{p,q}$ of $\mathbb{R}^{p,q}$, iff*
(a) $v \otimes_{p,q} v = -Q(v)\,, \quad v \in \mathbb{R}^{p,q}$
(b) $\mathcal{C}_{p,q}$ *is generated as an algebra by* $\mathbb{R}^{p,q}$
(c) $\mathcal{C}_{p,q}$ *is not generated by any proper subspace of* $\mathbb{R}^{p,q}$.

Condition (a) implies for all $i, j \in \{1, \ldots, p+q\}$

$$e_i \otimes_{p,q} e_j = -e_j \otimes_{p,q} e_i\,. \tag{9}$$

Together with (b) and (c) follows that the dimension of $\mathcal{C}_{p,q}$ is 2^{p+q} [12] and $\mathcal{C}_{p,q}$ is commutative only if $p + q <= 1$. With the notations

$$\mathcal{A} := \{\{a_1, \ldots, a_r\} \in \mathcal{P}(\{1, \ldots, n\}) \mid 1 \leq a_1 \leq \ldots \leq a_r \leq n\}\,, \tag{10}$$

where $\mathcal{P}(\{1, \ldots, n\})$ denotes the power set and defining for all $A \in \mathcal{A}$

$$e_A := e_{a_1} \ldots e_{a_r}\,, \tag{11}$$

we obtain a basis $\{e_A \mid A \in \mathcal{A}\}$ of $\mathcal{C}_{p,q}$. Then the following involutions of $\mathcal{C}_{p,q}$ can be defined

$$\hat{x} = \sum_{A \in \mathcal{A}} (-1)^{|A|}\, x_A e_A \tag{12}$$

$$\tilde{x} = \sum_{A \in \mathcal{A}} (-1)^{\frac{|A|(|A|-1)}{2}} x_A e_A \tag{13}$$

$$\bar{x} = \sum_{A \in \mathcal{A}} (-1)^{\frac{|A|(|A|+1)}{2}} x_A e_A\,. \tag{14}$$

After all these preliminaries we shall look at some Clifford algebra concretely. Any Clifford algebra is isomorphic to a matrix algebra. Table 1 gives an overview in low dimensions, containing the real numbers ($\mathcal{C}_{0,0}$), the complex numbers ($\mathcal{C}_{0,1}$) and the quaternions ($\mathcal{C}_{0,2}$). In this cases we can directly define meaningful Clifford neurons via the algebra multiplication. Complex NNs are well known (see e.g. [7]), whereas the study of quaternionic NNs just started recently [1].

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

Table 1. Low dimensional Clifford algebras
(^{2} denotes direct product; R(x) denotes x×x matrices)

However, our focus here is on the neural computation of geometric transformations. So far, we have only embedded quadratic spaces in Clifford algebras. Our next goal is to determine elements of $\mathcal{C}_{p,q}$ that act as geometric transformations. Therefore, we have to study transformation groups that act on a special subspace $\lambda\mathbb{R}^{p,q} := \mathbb{R} \oplus \mathbb{R}^{p,q}$ of $\mathcal{C}_{p,q}$.

Definition 2 *The Clifford group* $\Gamma_{\mathrm{p,q}}$ *of a Clifford algebra* $\mathcal{C}_{p,q}$ *is defined by its action on* $\lambda\mathbb{R}^{p,q}$ *as*

$$\Gamma_{\mathrm{p,q}} := \{ s \in \mathcal{C}_{p,q}, s \otimes_{p,q} \tilde{s} = \pm 1 \mid \forall x \in \lambda\mathbb{R}^{p,q} : s \otimes_{p,q} x \otimes_{p,q} \hat{s}^{-1} \in \lambda\mathbb{R}^{p,q} \}. \quad (15)$$

This action is in fact an orthogonal transformation of $\mathbb{R}^{p,q+1}$.

Theorem 1 *The map* $\Psi_s : \Gamma_{\mathrm{p,q}} \to O(p, q+1)$; $s \mapsto \psi_s$ *is a group epimorphism, whereby* ψ_s *denotes the function* $x \mapsto s \otimes_{p,q} x \otimes_{p,q} \hat{s}^{-1}$.

Thus, ψ_s implemented as shown in Figure 4 gives a neuron with only one weight s that computes an orthogonal transformation according to Theorem 1.

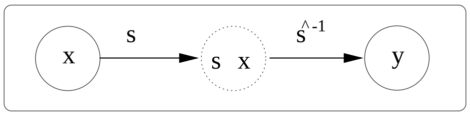

Fig. 4. Clifford neuron (CN)

Let us briefly review the first non trivial case of quaternions $\mathcal{C}_{0,2}$. For a more detailed study the interested reader is refered to [2], which also discusses the complex case that reduces to one multiplication.

Let q_0 denote the scalar part of a quaternion q and $\boldsymbol{q}$ its vector part, respectively. The update rule for the weight s of a quaternionic Clifford neuron is given by

$$\Delta s = -\eta \frac{\partial E}{\partial s} = \eta \sum_{i=1}^{m} (t^i - y^i) \otimes_{0,2} \delta \,, \tag{16}$$

$\delta = \nabla(s_0^2 q_0 + (\boldsymbol{s} \cdot \boldsymbol{s}) q_0,\ s_0^2 \boldsymbol{q} + (\boldsymbol{s} \cdot \boldsymbol{q})\boldsymbol{s} + 2 s_0 (\boldsymbol{s} \times \boldsymbol{q}) - \boldsymbol{s} \times \boldsymbol{q} \times \boldsymbol{s})$. For illustration purposes we made the following simulation with a quaternionic CN and a 3×3 SL-NN. The given task was to learn the transformation consisting of rotation of -60° about the axis [0.5, $\sqrt{0.5}$, 0.5] and translation about [0.2,-0.2,0.3] from 5 randomly chosen points with added normal noise of 20%. Figure 5 gives the obtained results from the trained networks of a test object.

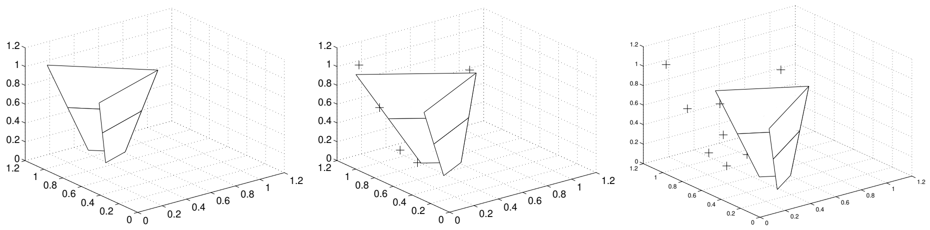

Fig. 5. Expected output test data (left), output quaternionic CN (middle) and output SL-NN (right)

Obviously, the SL–NN just learned the noise in the data which then resulted in a complete wrong pose of the test object. This means that no generalization was possible. On the other hand, the generalization performance of the quaternionic CN was quiet satisfactorily.
In the next section we will see exemplary how the introduced Clifford neurons can be used to compute taks related to non–linear geometric transformations.

4 Computing the Cross–Ratio with Clifford Neurons

The computation of invariants of geometric transformations is fundamental in many areas of Computer Vision and Robot Vision. A special important class is of course that of projective invariants. Among them the cross ratio of planar points (thought as lying in the complex plane) is the basis invariant from which many others can be derived [15].
The cross–ratio of 4 points $z, q, r, s \in \mathbb{C}$ is given by

$$[z, q, r, s] = \frac{(z-q)(r-s)}{(z-s)(r-q)} \,. \tag{17}$$

The cross–ratio depends on the order of the points. The given definition is very common, but not the only in use as remarked in [11]. As far as we know there are now algorithms for the neural computation of the cross–ratio. However, it should be very useful to have a neural method at our disposal, having standard neural network applications like pattern and object recognition in mind. In this section we will show how the cross–ratio can be computed by Clifford neurons. In order to be able to do so, we have to establish the link between the cross-ratio and Möbius transformations first. A Möbius (linear–fractional, bilinear) transformation is a conformal mapping of the extended complex plane of the form

$$w = M(z) = \frac{a + b\,\mathrm{i}}{c + d\,\mathrm{i}}\,, \tag{18}$$

where a, b, c, d are complex constants such that $ad - bc \neq 0$. They form a group $\mathcal{M}$ which can be checked easily. A Möbius transformation is already uniquely determined by 3 points. Moreover, the cross-ratio is invariant under Möbius transformations and the latter is uniquely defined by the correspondence of 2 cross–ratios. Thus, one obtains the following known theorem (see e.g. [11])

Theorem 2 *The cross–ratio* [z,q,r,s] *is the image of z under that Möbius transformation that maps q to 0, r to 1 and s to ∞, respectively.*

With the isomorphisms [12] $M \cong O(1,2) \cong \Gamma_{1,2}$ we can then link the cross-ratio to the 8–dimensional Clifford algebra $\mathcal{C}_{1,2}$, which is according to Table 1 isomorphic to $\mathbb{C}(2)$. However, we can not proceed immediately to the corresponding Clifford neuron, since we do not know yet, how to code the input/output (complex numbers) in it. This coding is given by [2]

$$co(z) = (Re(z), \frac{1}{2}(1 + z\bar{z}), Im(z), \frac{1}{2}(1 - z\bar{z}), 0, 0, 0, 0) \tag{19}$$

or in matrix notation by

$$Co(z) = \begin{pmatrix} z & z\bar{z} \\ 1 & \bar{z} \end{pmatrix}\,. \tag{20}$$

Finally, the Clifford neuron that computes the cross–ratio via the corresponding Möbius transformation has the form

$$y = s \otimes_{1,2} c(z) \otimes_{1,2} \tilde{s}\,, \tag{21}$$

from which we get

$$Co(y) = \lambda \begin{pmatrix} z' & z'\bar{z'} \\ 1 & \bar{z'} \end{pmatrix}\,, \tag{22}$$

with $\lambda = |cz + d|^2$ and $z' = (az + b)/(cz + d)$ [2].

Due to the limited space we have to omit the derivation of the learning algorithm here. In order to test our proposed Clifford neuron to compute cross–ratios we made the following experiment.

The task was to compute the cross–ratio [z,q,r,s] with $q = 0.2 + 0.3\,\mathrm{i}, r = 0.4 - 0.7\,\mathrm{i}$, and $s = 0.6 - 0.2\,\mathrm{i}$. The numerical problem we had to face was of course the coding of ∞ to which the point s should be mapped. After some experiments we have found out, that 10^{15} was a reasonable value with respect to both stability of learning algorithm and accuracy of the results. However, we still had to use a very low learning rate $\eta = 0.00001$ in order to guarantee stability. With this rate the convergence of learning in terms of epochs was only slow, as reported in Figure 6. But, computational time was less than 10 seconds.

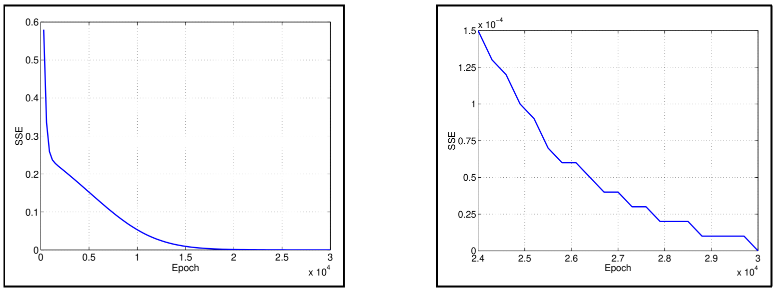

Fig. 6. Convergence of learning: all epochs (left) and last epochs only (right)

The SSE dropped under 0.00001 after 30000 epochs, which gave a weight vector corresponding to the transformation parameters in Table 2. The difference of the learned transformation parameters to the real values was quiet acceptable. Tabel 3 gives as confirmation the results obtained for some test points. The

Parameter	Value	Learned parameters
a	0.2 + 0.5 i	0.20019 + 0.50010 i
b	0.11-0.16 i	0.11001 - 0.15992 i
c	-0.2+ i	-0.20079 + 0.99939 i
d	-0.08-0.64 i	-0.07930 - 0.64007 i

Table 2. Transformation parameters

z	Value	Clifford neuron output
2+3 i	0.3578-0.3357 i	0.3577-0.3364 i
4-7 i	0.4838-0.3044 i	0.4838-0.3051 i
0.3+0.1 i	0.0077-0.6884 i	-0.0082+0.6884 i

Table 3. Cross–Ratios of test points (rounded)

achieved accuracy could still be improved by longer learning times. The Clifford neuron behaved very well in the experiment, so limitation is machine accuracy only.

5 Conclusion

In this paper we presented novel artificial neurons in the framework of Clifford algebra. We showed how they cover complex and quaternionic neurons and how they are related to orthogonal transformations. The potential power of the proposed Clifford neurons was illustrated by the example of computing the cross-ratio with a single Clifford neuron, which is a non–linear task. Future work will concentrate on enlarging the functional basis of Clifford neurons (projective transformations) and then build up networks of it for applications with the emphasis on object recognition.

References

1. P. Arena, L. Fortuna, G. Muscato, and M. G. Xibilia. *Neural Networks in Multidimensional Domains.* Number 234 in LNCIS. Springer–Verlag, 1998.
2. S. Buchholz and G. Sommer. *Introduction to Neural Computation in Clifford Algebra*, chapter 13. G. Sommer (Ed.), Geometric Computing with Clifford Algebra. Springer–Verlag, 2000 (to appear).
3. T. Caelli, D. Squire, and T. Wild. Model–Based Neural Networks. *Neural Networks*, 6:613–625, 1993.
4. G. Cybenko. Approximation by superposition of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2:303–314, 1989.
5. M. Ferraro and T. Caelli. Neural computation of algebraic and geometrical structures. *Neural Networks*, 11:669–707, 1998.
6. S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4:1–58, 1992.
7. G. Georgiou and C. Koutsougeras. Complex domain backpropagation. *IEEE Trans. Circ. and Syst. II*, 39:330–334, 1992.
8. P. Lounesto. *Clifford Algebras and Spinors.* Cambridge University Press, 1997.
9. T. Masters. *Signal and Image Processing with Neural Networks.* John Willey and Sons, 1994.
10. W. S. McCulloch and W. Pitts. A Logical Calculus of Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.
11. T. Needham. *Visual Complex Analysis.* Clarendon Press, Oxford, 1997.
12. I. R. Porteous. *Clifford Algebras and the Classical Groups.* Cambridge University Press, Cambridge, 1995.
13. D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representations by error propagation. In *Parallel Distributed Processing*, volume 1. MIT Press, Cambridge MA, 1986.
14. M. Vidyasagar. *A Theory of Learning and Generalization.* Springer–Verlag, 1997.
15. K. Voss and H. Süsse. *Adaptive Modelle und Invarianten für zweidimensionale Bilder.* Verlag Shaker, Aachen, 1995.