ESANN'2000 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 26-28 April 2000, D-Facto public., ISBN 2-930307-00-5, pp. 377-382

Quaternionic Spinor MLP

Sven Buchholz, Gerald Sommer

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

Abstract. This paper introduces a novel quaternion–valued MLP–type network called the Quaternionic Spinor MLP (QSMLP). In contrast to another recently proposed Quaternionic MLP it uses spinors in its propagation function. This allows very efficiently the processing of 3D vector data, which is demonstrated by experiments. The QSMLP is proven to be a universal approximator and a learning algorithm for it is derived.

1. Introduction

There are a couple of neural architectures proposed in the literature to process multidimensional data. In particular a lot of research was done in the complex domain [1]. However, any further extension requires non–basic algebraic structures as e.g. Clifford Algebras [2]. A domain of special interest was also always that of 3D vector space, as it arises naturally in important applications as robotics and control theory. Yet, direct architectures as proposed in [6] fail for algebraical reasons [1], [2]. More efficient and robust is the use of quaternions, which have become very common in these applications [3]. Recently, in [1] a Quaternionic MLP (QMLP) was developed. However, this QMLP performs not in a natural way on 3D vector data. In this paper we propose instead the Quaternionic Spinor MLP (QSMLP), which has such properties in addition. The paper starts with an introduction to quaternions. The third section is describing the QSMLP in terms of architecture, approximation properties and learning algorithm. In section 4 experimental results are reported that support our claims on the performance of QSMLP vs. QMLP.

2. Quaternions

Quaternions are generalized complex numbers of the form

$$\boldsymbol{q} = q_0 + q_1\,\mathrm{i} + q_2\,\mathrm{j} + q_3\,\mathrm{k} \tag{1}$$

with $q_0, q_1, q_2, q_3 \in \mathbb{R}$ and imaginary units $\{\mathrm{i}, \mathrm{j}, \mathrm{k}\}$. The imaginary units of a quaternion behave similar to the imaginary unit of complex numbers

$$\mathrm{i}^2 = \mathrm{j}^2 = \mathrm{k}^2 = -1\,. \tag{2}$$

ESANN'2000 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 26-28 April 2000, D-Facto public., ISBN 2-930307-00-5, pp. 377-382

They can be seen as spatial orthogonal vectors, since the following relations hold

$$\mathrm{j}\,\mathrm{k} = -\mathrm{k}\,\mathrm{j} = \mathrm{i} \quad \mathrm{k}\,\mathrm{i} = -\mathrm{i}\,\mathrm{k} = \mathrm{j} \quad \mathrm{i}\,\mathrm{j} = -\mathrm{j}\,\mathrm{i} = \mathrm{k}\,. \tag{3}$$

Thereof, a quaternion consists of a scalar part q_0 and a vector part denoted by $\vec{q} := q_1\,\mathrm{i} + q_2\,\mathrm{j} + q_3\,\mathrm{k}$. The 4–tuple notation of a quaternion is given by

$$\boldsymbol{q} = (q_0, q_1, q_2, q_3) = (q_0, \vec{q})\,. \tag{4}$$

Furthermore, let $[\boldsymbol{q}]_i$ $(i \in \{0, \ldots, 3\})$ denote the i-th projection. Together with the postulation that $(1, \vec{0})$ should be its identity, multiplication of quaternions $\otimes$ is already fully determined by (2) and (3)

$$\boldsymbol{q} \otimes \boldsymbol{p} = (q_0\, p_0 - \vec{q} \cdot \vec{p},\; q_0\, \vec{p} + p_0\, \vec{q} + \vec{q} \times \vec{p})\,. \tag{5}$$

Through this, quaternions become a real associative (but not commutative) division algebra $\mathbb{H} := (\mathbb{R}^4, +, \otimes)$. As in the case of complex numbers, the norm of a quaternion is defined via conjugation. The conjugate of a quaternion is $\boldsymbol{q}^* := (q_0, -\vec{q})$. The norm of a quaternion is then given by $|\boldsymbol{q}| := \sqrt{\boldsymbol{q}^* \boldsymbol{q}}$.

3. The Quaternionic Spinor MLP

After these preliminaries we follow the way outlined in the introduction.

3.1. QSMLP Architecture

Let us start with reviewing briefly the architecture of the MLP and the QMLP to be able to show how the QSMLP differs from them. Of course all named neural networks have in common, that they consist of layers of neurons with weighted feed–forward only connections between all the neurons of consecutive layers. More precisely, the output of the j-th (non input) neuron in layer l of an MLP is given by

$$f(\sum_i w_{ij}^{(l)} \cdot x_i^{(l-1)} + \theta_j^{(l)})\,, \tag{6}$$

where $w_{ij}^{(l)}$ is the weight connecting the i–th node in layer $(l-1)$ with the j–th node in layer l, θ_j is the appropriate bias and $x_i^{(l-1)}$ is the i-th input. All these entities are real numbers. The choice of the activation function f will be discussed later on. The QMLP introduced in [1] uses quaternionic entities and replaces the scalar product by the quaternionic product

$$\boldsymbol{f}(\sum_i \boldsymbol{w_{ij}}^{(l)} \otimes \boldsymbol{x_i}^{(l-1)} + \boldsymbol{\theta_j}^{(l)})\,. \tag{7}$$

However, this definition makes the main benefits of quaternions, namely the operating on 3D vectors and its applications, not easy to use. The multiplication of a 3D vector $\vec{v} := (0, \vec{v})$ by a single quaternionic weight results not in a vector again. To overcome this drawback we suggest the QSMLP with neurons of the form

$$\boldsymbol{f}(\sum_i \boldsymbol{r_{ij}}^{(l)} \otimes \boldsymbol{x_i}^{(l-1)} \otimes \boldsymbol{r}^{*}_{\boldsymbol{ij}}{}^{(l)} + \boldsymbol{\theta_j}^{(l)})\,. \tag{8}$$

For all $\boldsymbol{r}, |\boldsymbol{r}| = 1$ the mapping $\vec{v} \mapsto \boldsymbol{r} \otimes \vec{v} \otimes \boldsymbol{r}^*$ is a Euclidean 3D rotation [5]. Thus, in the case of processing 3D vector data the QSMLP performs a dilatation–rotation as weight association. This is also true for the vector part of arbitrary quaternionic data. Hence, also then the QSMLP has the advantage that its parameters are easier to interpret than those of the QMLP. All quaternions with unit norm form the so called Spin-group, which is a double cover of SO(3). The elements of the Spin–group are named spinors, which is where the name QSMLP comes from. We still have to define the activation function that should be used. The activation functions of MLPs are often sigmoidal ones, among the most popular is $\sigma : x \mapsto 1/(1+\exp(-x))$. Based on it the function

$$\boldsymbol{\sigma}(\boldsymbol{x}) = \sigma([\boldsymbol{x}]_i) \quad (i \in \{0, \ldots, 3\}) \tag{9}$$

was proposed for the QMLP [1], which we will adopt.

3.2. QSMLPs are Universal Approximators

In order to show the validity of the section headline we have to prove the following theorem.

Theorem 1 *Let X be a compact subset of $\mathbb{H}^n$. Then there exists a natural number N such that the space*

$$\left\{ \sum_{j=1}^{N} \lambda_j \, \boldsymbol{\sigma}((\sum_{i=1}^{n} \boldsymbol{r_i} \otimes \boldsymbol{x_i} \otimes \boldsymbol{r_i}^*) + \boldsymbol{\theta_j}) \right\} \tag{10}$$

is dense in the space of all continuous functions from X to $\mathbb{H}$.

Proof In [4] the fundamental density theorem for MLPs with sigmoidal activation functions was proven. A set of quaternionic–valued functions has the universal approximation property, iff it is a universal approximator for any of the (real–valued)component functions. The density theorem for the QMLP could be proven in [1] by this argument, since it allows reduction to the real–valued case in [4]. Thus, we only have to show that at least any projection $[\boldsymbol{w} \otimes \boldsymbol{x}]_i$ can be written as the finite sum of spinor multiplications $\boldsymbol{r_i} \otimes \boldsymbol{x_i} \otimes \boldsymbol{r_i}^*$. But, there always exists $u \in \mathbb{H}$ such that for all $i \in \{0, \ldots, 3\}$

$$[\boldsymbol{w} \otimes \boldsymbol{x}]_i = [\boldsymbol{w} \otimes \boldsymbol{x} \otimes \boldsymbol{w}^* + \boldsymbol{u} \otimes \boldsymbol{x} \otimes \boldsymbol{u}^*]_i\,. \tag{11}$$

Due to the limited available space we have to omit detailed calculations. □

ESANN'2000 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 26-28 April 2000, D-Facto public., ISBN 2-930307-00-5, pp. 377-382

3.3. QSMLP Learning Algorithm

F or the sake of simplicity let us consider a QSMLP with only one hidden layer. Using similar notations as in section 3 we define at first

- hidden node activation and output value

$$\boldsymbol{S_m}^{(1)} := \sum_n \boldsymbol{r_{nm}}^{(1)} \otimes \boldsymbol{x_n} \otimes \boldsymbol{r_{nm}}^{*(1)} + \boldsymbol{\theta_m}^{(1)} \qquad \boldsymbol{h_m} := \boldsymbol{\sigma}(\boldsymbol{S_m}^{(1)}) \quad (12)$$

- output node activation and output value

$$\boldsymbol{S_p}^{(2)} := \sum_m \boldsymbol{r_{mp}}^{(2)} \otimes \boldsymbol{h_m} \otimes \boldsymbol{r_{mp}}^{*(2)} + \boldsymbol{\theta_p}^{(2)} \qquad \boldsymbol{o_p} := \boldsymbol{\sigma}(\boldsymbol{S_p}^{(2)}) \,. \quad (13)$$

We want to minimize by gradient descent the error function

$$E = \frac{1}{2} \sum_p (\boldsymbol{y_p} - \boldsymbol{o_p})^2 \,, \quad (14)$$

whereby $\boldsymbol{y_p}$ stands for the p-th expected output value. First, we have to compute the weigh ts of the output layer according to

$$\nabla E_{\boldsymbol{r_{mp}}^{(2)}} = \sum_i \frac{\partial E}{\partial [\boldsymbol{r_{mp}}^{(2)}]_i} \,. \quad (15)$$

The chain rule applied to each term of (15) gives

$$\frac{\partial E}{\partial [\boldsymbol{r_{mp}}^{(2)}]_i} = \sum_j \frac{\partial E}{\partial [\boldsymbol{S_p}^{(2)}]_j} \frac{\partial [\boldsymbol{S_p}^{(2)}]_j}{\partial [\boldsymbol{r_{mp}}^{(2)}]_i} \,. \quad (16)$$

The partial derivativ es of the error function E wrt. $\boldsymbol{S_p}^{(2)}$ are given b y

$$\frac{\partial E}{\partial [\boldsymbol{S_p}^{(2)}]_j} = \frac{\partial E}{\partial [\boldsymbol{y_p}]_j} \frac{\partial [\boldsymbol{y_p}]_j}{\partial [\boldsymbol{S_p}^{(2)}]_j} = ([\boldsymbol{y_p}]_j - [\boldsymbol{o_p}]_j)\, \dot{\boldsymbol{\sigma}}([\boldsymbol{S_p}^{(2)}]_j) \,. \quad (17)$$

F urthermore, with $(r_0, \vec{r}) := \boldsymbol{r_{mp}}^{(2)}$, $(q_0, \vec{q}) := \boldsymbol{h_m}$ w e obtain

$$\frac{\partial [\boldsymbol{S_p}^{(2)}]}{\partial \boldsymbol{r_{mp}}^{(2)}} = \nabla (r_0^2 q_0 + (\vec{r} \cdot \vec{r}) q_0,\, r_0^2 \vec{q} + (\vec{r} \cdot \vec{q}) \vec{r} + 2 r_0 (\vec{r} \times \vec{q}) - \vec{r} \times \vec{q} \times \vec{r}) \,. \quad (18)$$

Thus, we get the following update rule for the weights of the output lay er

$$\Delta \boldsymbol{r_{mp}}^{(2)} = [\, \underbrace{(\boldsymbol{y_p} - \boldsymbol{o_p}) \odot \dot{\boldsymbol{\sigma}}(\boldsymbol{S_p}^{(2)})}_{\boldsymbol{\delta_p}^{(2)}} \,] \otimes \frac{\partial [\boldsymbol{S_p}^{(2)}]}{\partial \boldsymbol{r_{mp}}^{(2)}} \,, \quad (19)$$

ESANN'2000 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 26-28 April 2000, D-Facto public., ISBN 2-930307-00-5, pp. 377-382

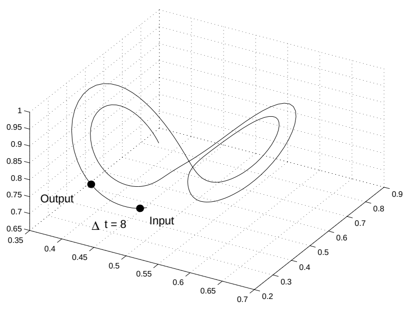
whereas $\odot$ denotes scalar multiplication component by component. The update rule for the weigh ts of the hidden layer is

$$\Delta \boldsymbol{r_{nm}}^{(1)} = \underbrace{[\,(\sum_p \boldsymbol{r_{mp}}^{*(2)} \otimes \boldsymbol{\delta_p}^{(2)}) \odot \dot{\boldsymbol{\sigma}}(\boldsymbol{S_m}^{(1)})\,]}_{\boldsymbol{\delta_m}^{(1)}} \otimes \frac{\partial[\boldsymbol{S_m^{(1)}}]}{\partial \boldsymbol{r_{nm}}^{(1)}} \, . \qquad (20)$$

The derivativ e$\frac{\partial[\boldsymbol{S_m^{(1)}}]}{\partial \boldsymbol{r_{nm}}^{(1)}}$ is the same as in (18) with $(r_0, \vec{r}) := \boldsymbol{r_{nm}}^{(1)}$, $(q_0, \vec{q}) := \boldsymbol{x_n}$. Finally, we have $\Delta\, \boldsymbol{\theta_p}^{(2)} = \boldsymbol{\delta_p}^{(2)}$ and $\Delta\, \boldsymbol{\theta_m}^{(1)} = \boldsymbol{\delta_m}^{(1)}$.

4. Experiments

We considered the task of short term prediction of the chaotic Lorenz attractor. The Lorenz attractor is generated by the system $\{\dot{x} = \sigma(x-y), \dot{y} = xz + rx - y, \dot{z} = xy - bz\}$ with parameter values $\sigma = 10, r = \frac{8}{3}, b = 28$ and starting state $(x_0, y_0, z_0) = (0, 1, 0)$. From the time interv al (12s,17s) 1000 points (sampling rate $\Delta t = 0.005$) were tak en, from which the first 250 formed the training set and the last 750 the test set. The prediction step rate was set to 8. The sets are shown in Figure 1 below.



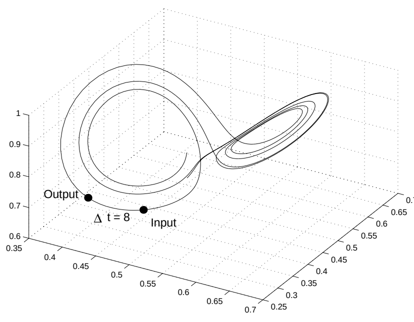


Figure 1: T raining set (left) and test set (right)

Both a QMLP and QSMLP wit one hidden layer of 6 nodes were trained ov er 10000 epoc hs. The 3D data w ascoded in the vector part of one input and output node, respectively. Averaged ov er 10 trials the QMLP achiev ed a MSE (training/test) of 0.005/0.023. Instead, the QSMLP achieved 0.001/0.0013. Actually, the generalization of the QSMLP w as ev en better than indicated n umerically. This can be seen from Figure 2 by comparing size and the right loop. No better performance could be achiev ed with either one of the netw orks b y using more hidden nodes. Thus, the QSMLP outperformed the QMLP on this task due to its ability to model the intrinsic properties of the data.

ESANN'2000 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 26-28 April 2000, D-Facto public., ISBN 2-930307-00-5, pp. 377-382

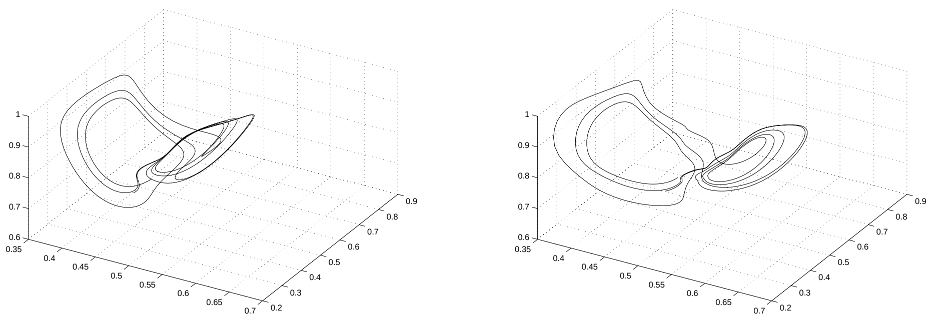


Figure 2: Generalization of the QMLP (left) and the QSMLP (right)

5. Conclusion

We proposed a nov el MLP–type neural netw orkoperating in the quaternion algebra via spinors as propagation function, which we hav e called the Quaternionic Spinor MLP (QSMLP). We believe that this proposed arc hitecture is more suitable for the processing of 3D vector data than both the MLP and the QMLP. In a first simulation a better performance as the Quaternionic MLP in [1] was achiev ed already Future work will go on in evaluating the QSMLP on tasks in the field of robotic vision. The property of the QSMLP to have weigh ts that code directly rigid 3D motions might there show useful, for example to get control parameters immediately.

Acknowledgment This work was supported by the Deutsche F orsc h ungsgemeinschaft under grant So-320-2-1.

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. *Intr oduction to Neural Computation in Clifford A lgebra*, chapter 13. G. Sommer (Ed.), Geometric Computing with Clifford Algebra. Springer–Verlag, 2000 (to appear).

[3] J. C. K. Chou. Quaternion Kinematic and Dynamic Differential Equations. *IEEE Trans. on Rob otics and Automation*, 8(1):53–64, 1992.

[4] G. Cybenko. Approximation b y superposition of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2:303–314, 1989.

[5] H.-D. Ebbinghaus et al. *Numbers.* Springer–Verlag, 3rd edition, 1995.

[6] T. Nitta. A Back–Propagation Algorithm for Neural Networks Based on 3D Vector Product. In *Pr oceedings IJCNN'93*, pages 589–592, 1993.