Geometric Neural Networks and Support Multi–Vector Machines

Eduardo Bayro-Corrochano and Refugio Vallejo
Centro de Investigaciones Matemáticas, CIMAT, Computer Science Department
Apartado Postal 402, 36000-Guanajuato, Gto, Mexico
email: edb,vallejo@fractal.cimat.mx

Abstract

The representation of the external world in biological creatures appears to be defined in terms of geometry. In this regard the author uses the Clifford geometric algebra for the development of geometric type neural networks. The contribution of this paper is the extension of our past work including the use of the SV–Machines in the Clifford algebra framework. Thus geometric MLPs and RBF networks can be generated using SV–Machines straightforwardly. In this way we expanded the sphere of applicability of the SV–Machines by the treatment of multivectors which encode the geometry of the data manifold in a rich manner.

1 Introduction

The computational efficiency of geometric algebra has been shown in various challenging areas of mathematical physics [2]. Preliminary attempts for applying the geometric algebra in neural geometry have been already done [3, 1]. Analyzing other mathematical 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. Section two presents the geometric feedforward neural networks. Section three introduces the Support Multi-Vector Machines. The last section discusses the suitability of the geometric neural nets using the SVM approach.

2 Geometric Neural Networks

Real, complex and quaternion 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 and the scalar product will be substituted by the Clifford or geometric product.

The function for a n-dimensional multivector $\boldsymbol{m}$ reads

$$\boldsymbol{f}(\boldsymbol{m}) = f(m_0) + f(m_1)\sigma_i + f(m_2)\sigma_j + ... + f(m_j)\sigma_i\wedge\sigma_j + ... + f(m_n)\sigma_1\wedge\sigma_2\wedge\ldots\wedge\sigma_n, \quad (1)$$

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

$$\mathbf{w}^T\mathbf{x} + \theta \qquad \Rightarrow \qquad \boldsymbol{w}\boldsymbol{x} + \boldsymbol{\theta} = \boldsymbol{w}\cdot\boldsymbol{x} + \boldsymbol{w}\wedge\boldsymbol{x} + \boldsymbol{\theta} \tag{2}$$

The geometric neuron outputs a more rich kind of pattern comprising multivector components like planes, volumes, ..., hyper-volumes. Let us illustrate this with an example in $\mathcal{G}_{3,0}$

$$\boldsymbol{o} = \boldsymbol{f}(\boldsymbol{w}\boldsymbol{x} + \boldsymbol{\theta}) = f(s_0) + f(s_1)\sigma_1 + f(s_3)\sigma_2 + ... + f(s_5)\sigma_1\sigma_3 + f(s_6)\sigma_2\sigma_3 + f(s_7)\sigma_1\sigma_2\sigma_3, \tag{3}$$

where $\boldsymbol{f}$ is the activation function defined in Eq. (1) and $s_i \in \mathcal{R}$.

2.1 Feedforward geometric neural networks

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

$$\begin{aligned} \boldsymbol{o}_j &= \boldsymbol{f}_j(\sum_{i=1}^{N_i} \boldsymbol{w}_{ji}\cdot\boldsymbol{x}_{ji} + \boldsymbol{w}_{ji}\wedge\boldsymbol{x}_{ji} + \boldsymbol{\theta}_j) \\ \boldsymbol{y}_k &= \boldsymbol{f}_k(\sum_{j=1}^{N_j} \boldsymbol{w}_{kj}\cdot\boldsymbol{o}_{kj} + \boldsymbol{w}_{kj}\wedge\boldsymbol{o}_{kj} + \boldsymbol{\theta}_k) \end{aligned} \tag{4}$$

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 $\boldsymbol{D}_i = e^{\alpha\frac{\bar{i}\bar{i}}{2}}$ [2], i.e.

$$\boldsymbol{y}_k(\boldsymbol{x}) = \sum_{j=1}^{N} \boldsymbol{w}_{kj}\boldsymbol{G}_j(\boldsymbol{D}_j(\boldsymbol{x}_{ji} - \boldsymbol{t}_j)\tilde{\boldsymbol{D}}_j) \tag{5}$$

Note that the activation function uses as components Gauss functions.

2.2 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 and the quaternion MLP [3] are simply especial cases of this extended rule. The norm of a multivector $\boldsymbol{x}$ for the learning rule reads

$$|\boldsymbol{x}| = (\boldsymbol{x}|\boldsymbol{x})^{\frac{1}{2}} = (\sum_{A}[\boldsymbol{x}]_A^2)^{\frac{1}{2}}. \tag{6}$$

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

$$\mathcal{Y}_t : (\mathcal{G}_{p,q})^n \to (\mathcal{G}_{p,q})^m, \tag{7}$$

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

$$E = \frac{1}{2}\int_{\boldsymbol{x}\in\boldsymbol{X}} ||\mathcal{Y}_w - \mathcal{Y}_t||^2, \tag{8}$$

where $\boldsymbol{X}$ is some compact subset of the Clifford module $(\mathcal{G}_{p,q})^n$ involving the product topology derived from equation (6) for the norm and $\mathcal{Y}_w$ and $\mathcal{Y}_t$ are the learned and target mapping functions respectively. The back-propagation algorithm is a procedure for updating the weights and biases. This algorithm is a function of the negative derivative of the error function (Eq. (8)) with respect to the weights and biases themselves. The updating equation for the multivector weights of any hidden $j-$layer reads

$$\boldsymbol{w}_{ij}(t+1) = \eta[(\sum_{k}^{N_k} \delta_{kj} \otimes \overline{\boldsymbol{w}_{kj}}) \odot \boldsymbol{F}'(\boldsymbol{net}_{ij})] \otimes \overline{\boldsymbol{o}_i} + \alpha \boldsymbol{w}_{ij}(t), \tag{9}$$

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

$$\boldsymbol{w}_{jk}(t+1) = \eta[(\boldsymbol{y}_{k_t} - \boldsymbol{y}_{k_a}) \odot \boldsymbol{F}'(\boldsymbol{net}_{jk})] \otimes \overline{\boldsymbol{o}_j} + \alpha \boldsymbol{w}_{jk}(t), \tag{10}$$

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

$$\boldsymbol{w}_{jk}(t+1) = \eta(\boldsymbol{y}_{k_t} - \boldsymbol{y}_{k_a}) \otimes \overline{\boldsymbol{o}_j} + \alpha \boldsymbol{w}_{jk}(t), \tag{11}$$

where $\boldsymbol{F}$ is the activation function defined in equation (1), 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 antinvolution (reversion or conjugation).

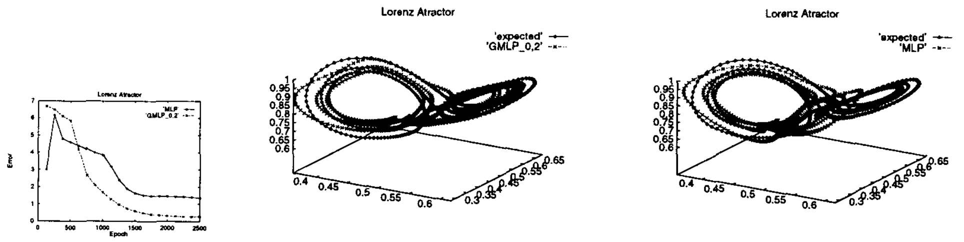

Figure 1: *a) Training error b) Prediction by GMLP$_{0,2}$ and expected trend c) Prediction by MLP and expected trend.*

2.3 Coordinate free learning in a chaotic process

Let us show an application of a geometric multi-layer perceptron to distinguish the geometric information in a chaotic process. 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 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 1.a show the error during training, note that the GMLP$_{0,2}$ converges faster than the MLP. Interesting is how they behave by the prediction. The Figures 1b-c shows that the GMLP$_{0,2}$ predicts better than the MLP. Analyzing the covariance parameters of the MLP [0.96815, 0.67420,0.95675] and of the 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, see Figure 1.b . 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.

3 Support Vector Machines using Geometric Algebra

The SVM approach of Vladimir N. Vapnik [4] uses optimization methods for learning. Using SV-Machines we can generate MLP and RBF networks and networks with other kernels. Our idea is to generate neural networks using the SVM machines in the geometric algebra framework. In this way we are using the potential of SV-Machines for the processing of multivectors. We will call our approach *Support Multi-Vector Machine* (SMVM). Let us review briefly the SVM and then explain the SMVM.

3.1 Support Vector Machines

Basically the SVM maps the input space R^d into a high-dimensional feature space H given by $\Phi : R^d \Rightarrow H$ satisfying a Kernel $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \Phi(x_i) \cdot \Phi(x_j)$ which fulfill the Mercer's condition [4]. The SMV constructs an optimal hyperplane in the feature space to divide two data clusters.
SV-Machines build the mapping

$$f(\boldsymbol{x}) = sign\Big(\sum_{supportvectors} y_i \alpha_i K(\boldsymbol{x}_i, \boldsymbol{x}) - b \Big). \tag{12}$$

We find the coefficients α_i in the the separable case (analogously in the non-separable case) which fulfill the maximum of the functional

$$W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j}^{l} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

subject to the constraints $\sum_{i=1}^{l} \alpha_i y_i = 0$, where $\alpha_i \geq 0$, i=1,2,...,l. This functional coincides with the functional for finding the optimal hyperplane. Examples of SV Machines include: polynomial learning machines $K(\boldsymbol{x}, \boldsymbol{x}_i) = [(\boldsymbol{x} \cdot \boldsymbol{x}_i) + 1]^d$, radial basis functions machines $K_\gamma(|x - x_i|) = exp\{-\gamma |\boldsymbol{x} - \boldsymbol{x}_i|^2\}$ and two layer neural networks $K(\boldsymbol{x}, \boldsymbol{x}_i) = S\big(v(\boldsymbol{x} \cdot \boldsymbol{x}_i) + c\big)$.

3.2 Support Multi-Vector Machines

A Support Multi-Vector Machine maps the multivector input space of $\mathcal{G}_{p,q}$ into a high-dimensional feature space $\Phi : R^d \Rightarrow R^{2^n}$, where R^{2n} $(n = p + q)$ is the space spanned by the multivector basis of $\mathcal{G}_{p,q}$. The SMVM constructs an optimal separating hyperplane in the multivector feature space using in the nonlinear case the kernels

$$\sum_{m=1}^{Ngradedspaces} K_m(\boldsymbol{x}_{mi}, \boldsymbol{x}_{mj}) = \sum_{m=1}^{Ngradedspaces} \Phi(x_{mi}) \cdot \Phi(x_{mj}), \tag{13}$$

which fulfill the Mercer's conditions. SMV–Machines for multivectors $\boldsymbol{x}$ of a geometric algebra $\mathcal{G}_{p,q}$ implement the multivector mapping

$$\begin{aligned}\{y_1, y_2, ..., y_m\} &= \sum_{m=1}^{Ngradedspaces} f_m(\boldsymbol{x}_{mi}) \\ &= \sum_{m=1}^{Ngradedspaces} sign\Big(\sum_{supportvectors} y_{mi}\alpha_{mi}K(\boldsymbol{x}_{mi}, \boldsymbol{x}_m) - b_m\Big), \end{aligned} \quad (14)$$

where m is the amount of grades spaces. The coefficients α_{mi} in the the separable case (analogously in the non–separable case) are found maximizing the functional:

$$W(\alpha_i^m) = W\Big(\sum_{m=1}^{Ngradedspaces} \sum_{i}^{l} \alpha_{mi}\Big) =$$
$$\sum_{m=1}^{Ngradedspaces} \sum_{i=1}^{l} \alpha_{mi} - \frac{1}{2} \sum_{m=1}^{Ngradedspaces} \sum_{i,j}^{l} \alpha_{mi}\alpha_{mj}y_{mi}y_{mj}K(x_{mi}, x_{mj})$$

subject to the following constraint

$$\sum_{m=1}^{Ngradedspaces} \sum_{i=1}^{l} \alpha_{mi}y_{mi} = 0, \quad (15)$$

where $\alpha_{mi} \geq 0$, for m=1,...,N graded spaces and i=1,2,...,l. This functional coincides with the functional for finding the optimal separating hyperplane for the multivector feature space.

3.3 Clustering Multivectors

This section shows the use of SVM to find the support vectors in the 2D and 3D case. These examples were chosen, because we can depict the feature space and show clearly to the reader how the SVM and the SMVM work. First we take two clusters in R^2 and apply a nonlinear SVM and a nonlinear SMVM using in both cases a RBF kernel. Figure 2.a shows the original 2D data, Figure 2.b the support vectors found by SVM and 2.c the support multivectors found by the SMVM in $\mathcal{G}_{2,0,0}$. The multivectors were coded as $\boldsymbol{m}_i = \boldsymbol{x}_i + \frac{1}{3}\sum_{j=i}^{i+2} \boldsymbol{x}_j + (\boldsymbol{x}_i - \boldsymbol{x}_{i+1}) \wedge (\boldsymbol{x}_i - \boldsymbol{x}_{i+2})$. Note that the areas follow the borders of the clusters, whereas in the SVM case some support vectors are in the interior of the clusters. Figure 3.a shows the original 3D data, Figure 3.b individual support vectors found by linear SVM when the curvature of the data manifold changes and 3.c similarly the individual support multivectors found by the linear SMVM in $\mathcal{G}_{3,0,0}$. The multivectors coding the data are of the form: $\boldsymbol{m}_i = \boldsymbol{x}_i + \frac{1}{4}\sum_{j=i}^{i+3} \boldsymbol{x}_j + (\boldsymbol{x}_i - \boldsymbol{x}_{i+1}) \wedge (\boldsymbol{x}_i - \boldsymbol{x}_{i+2}) \wedge (\boldsymbol{x}_i - \boldsymbol{x}_{i+3})$, which represent pyramidal volumes. Note that the wide of the volume basis follow the curvature change of the surface. This experiment shows how the multivector capture the geometric properties of the manifold and it is interesting that only a linear SMVM is enough to find the optimal support multivector. Note that in Figure 3.c there is a multivector inside the lower cluster which was not consider by the SMVM, whereas in the SVM case this was unnecessary used as support vector, see Figure 3.b .

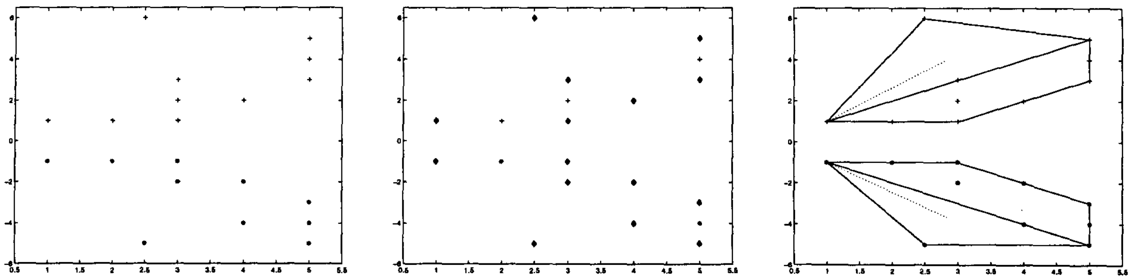

Figure 2: *2D case using RBF Kernel: a) two 2D clusters data b) SVM (support vectors indicated with diamonds) c) SMVM using $\mathcal{G}_{2,0,0}$ (support multivectors by triangles).*

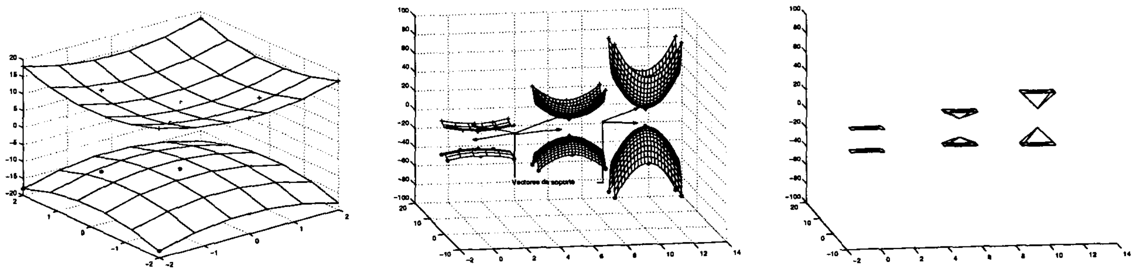

Figure 3: *3D case: a) two 3D clusters b) linear SVM lineal by change of curvature of the data surface (one support vector) c) linear SMVM using $\mathcal{G}_{3,0,0}$ by change of curvature of the data surface (one multivector changing its shape according the surface curvature).*

4 Conclusions

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 feed-forward neural networks. The reader can see that real-, complex- and quaternion-valued neural networks are simple particular cases of the geometric algebra multidimensional neural networks and that they can be generated via Support Multi-Vector Machines. The presented experiments reveal the role of geometric learning. The use of SVM in the geometric algebra framework expands its sphere of applicability and troughs new light in the comprehension of th SV-Machines for multi-dimensional learning.

References

[1] Bayro-Corrochano E., Buchholz S., Sommer G. 1996. Selforganizing Clifford neural network *IEEE ICNN'96 Washington, DC*, June, pp. 120-125.

[2] Hestenes, D. and Sobczyk, G. Clifford Algebra to Geometric Calculus: A Unified Language for Mathematics and Physics, 1984.

[3] 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.

[4] Vapnik V. Statistical Learning Theory. Wiley, New York, 1998.