

# Quaternion Support Vector Classifier

G. López-González, Nancy Arana-Daniel, and Eduardo Bayro-Corrochano

CINVESTAV - Unidad Guadalajara,  
Av. del Bosque 1145, Colonia el Bajo, Zapopan, Jalisco, México  
{gelopez,edb}@gdl.cinvestav.mx

**Abstract.** This paper presents the Quaternion Support Vector Machines for classification as a generalization of the real- and complex- valued Support Vector Machines. In this framework we handle the design of kernels involving the Clifford or quaternion product. The QSVM allows to change the metric involved in the quaternion product. The application section shows experiments in pattern recognition and colour image processing.

**Keywords:** Support Vector Machines, Clifford geometric algebra, classification, complex SVM, quaternion SVM.

## 1 Introduction

This paper presents the theory and application of the Quaternion Support Vector Machine (QSVM) and its use for applications in pattern classification and image processing. This work is a continuation of a first works on the generalization of SVMs, see [1,2].

In order to take advantage of certain geometric characteristics of the data and to avoid the use of many binary output SVMs, we were motivated to develop in the Quaternion Algebra framework SVM-based algorithms for classification. The quaternion algebra framework allows us to express in a compact way a variety of functions of geometric entities. By using the QSVM with one kernel (involving the quaternion product) we obtain non-linear mappings reducing the complexity of the computation greatly.

The organization of this paper is as follows: Section 2 outlines the quaternion algebra. Section 3 explains the kernel and sign functions. Section 4 introduces the Quaternion Support Vector Machines for Classification. Section 5 presents the use of QSVM for applications. The last section is devoted to the conclusion.

## 2 Quaternion Algebra: An Outline

The quaternion algebra  $\mathbb{H}$  was invented by W. R. Hamilton in 1843. It is an associative, non-commutative, four-dimensional algebra

$$\mathbb{H} = \{q = s + xi + yj + zk = s + \mathbf{q} \mid s, x, y, z \in \mathbb{R}\}, \quad (1)$$

where the orthogonal imaginary numbers  $i$ ,  $j$ , and  $k$  obey the following multiplicative rules:  $i^2 = j^2 = -1$ ,  $k = ij = -ji \rightarrow k^2 = -1$ . The conjugate of a quaternion is given by  $\bar{q} = s - xi - yj - zk$ . A quaternion in a polar representation is given by

$$q = |q|e^{i\phi}e^{k\psi}e^{j\theta}. \tag{2}$$

Given two quaternions  $q_a = s_a + \mathbf{q}_a$ ,  $q_b = s_b + \mathbf{q}_b$ , the quaternion anticommutative product is given by

$$q_c = s_c + \mathbf{q}_c = q_a q_b = (s_a s_b - \mathbf{q}_a \cdot \mathbf{q}_b) + (s_a \mathbf{q}_b + s_b \mathbf{q}_a + \mathbf{q}_b \times \mathbf{q}_a) \tag{3}$$

### 3 Quaternion Kernel and Sign Functions

#### 3.1 Kernel Functions

The kernel identity for linear classification is given by

$$K(x_m, x_n)_I = x_m^H x_n. \tag{4}$$

The next kernels are used for non linear classification. Polynomial quaternion based kernel are formulated as an extension of the polynomial kernels for a real valued SVM [3] as follows

$$K(x_m, x_n)_I = (x_m^H x_n + c)^d, \tag{5}$$

where  $c \in \mathbb{H}$  and  $d \in \mathbb{R}$  is the power of the binomial, multiplied with itself  $d$  times via the quaternion product.

The normalized Gaussian kernel is given by

$$g(x_m, x_n) = \frac{1}{\sqrt{2\pi\rho}} \exp^{-\frac{\|x_m - x_n\|^2}{2\rho^2}} \in \mathbb{R} \tag{6}$$

The Gaussian quaternion Gabor kernel function is

$$K(x_m, x_n)_q = g(x_m, x_n) (\cos(\mathbf{w}_0^T x_m) \cos(\mathbf{w}_0^T x_n) + \cos(\mathbf{w}_0^T x_m) \sin(\mathbf{w}_0^T x_n) i + \sin(\mathbf{w}_0^T x_m) \cos(\mathbf{w}_0^T x_n) j + \sin(\mathbf{w}_0^T x_m) \sin(\mathbf{w}_0^T x_n) k). \tag{7}$$

where the variables  $w_0, x_m - x_n$  stand for the the frequency and space domains respectively.

#### 3.2 Sign Functions

The decision function of the QSVM reads

$$y = qsign_m \left[ \sum_{j=1}^l (\alpha_j \circ y_j) (k(x_j, x) + b) \right]. \tag{8}$$

where  $m$  stands for the state valency, this is the number of classes considered for classification. The output  $y \in \mathbb{H}$  can classify up to  $2^4$  classes. We can decide wich parts of the labels  $y$  are we going to use. We regularly use the scalar part when  $m$  is equal to 2. We use the scalar and first imaginary when  $m$  is equal to 4. And the pure quaternion for state valency of 8, see figure 1 for this case.

The operation "o" is defined as

$$(\alpha_j \circ y_j) = \langle \alpha_j \rangle_s \langle y_j \rangle_s + \langle \alpha_j \rangle_i \langle y_j \rangle_i i + \langle \alpha_j \rangle_j \langle y_j \rangle_j j + \langle \alpha_j \rangle_k \langle y_j \rangle_k k,$$

simply one consider the multiplications of the corresponding coefficients of the quaternion basis.

### 4 Quaternion Support Vector Machines for Classification

For the case of the Quaternion SVM for classification we represent the data set in the quaternion algebra  $\mathbb{H}$ , where a quaternion is given by a four dimensional entity  $q = s + xi + yj + zk$ . Each data  $i$ th-vector has quaternion entries  $x_i = [q_{i1}, q_{i2}, \dots, q_{iD}]^T$ , where  $q_{ij} \in \mathbb{H}$  and  $D$  is the dimension of this input vector. Thus the  $i$ th-vector dimension is  $D \times 4$ . Each data  $i$ th-vector  $x_i \in \mathbb{H}^D$  of the  $N$  data vectors will be associated with a one quaternion at the output as follows:  $[x_{i1}, x_{i2}, \dots, x_{ij}, \dots, x_{iD}]$ , where each output  $y_i = y_{is} + y_{ij} + y_{ik} \in \{\pm 1 \pm i \pm j \pm k\}$  or can be even increase from  $2^4$  to  $m \times 2^4$  according the equation (8):  $y = qsign_m[f(x)]$ . Since the output is a quaternion, the  $m \times 2^4$  classification problem is to separate these quaternion-valued samples into  $m \times 2^4$  classes by selecting a right function from the set of functions  $f(x) = w^{*T}x + b, x, w \in \mathbb{H}^D, b \in \mathbb{H}^D$ . The optimal weight vector will be

$$w = [w_1, w_2, \dots, w_D]^T \in \mathbb{H}^D. \tag{9}$$

Let us see in detail the last equation

$$f(x) = w^H x + b = [w_1^*, w_2^*, \dots, w_D^*][x_1, x_2, \dots, x_D] + b = \sum_{i=1}^D w_i^* x_i + b, \tag{10}$$

where  $w_i^* x_i$  corresponds to the quaternion product of two vectors, where  $w_i^*$  is the conjugated of the vector  $w$ .

We introduce now a structural risk functional similar to the real valued and complex valued [5] SVM but now using quaternions  $\xi_i \in \mathbb{H}^D$ . For the quaternion valued SVM

$$\min \frac{1}{2} w^{*T} w + C \cdot \sum_{j=1}^l ((\xi_{1j}, \xi_{2j}, \dots, \xi_{Dj}))$$

subject to

$$Coef_s(y_{ij})Coef_s(f(x_{ij})) \geq 1 - Coef_s(\xi_{ij}), Coef_i(y_{ij})Coef_i(f(x_{ij})) \geq 1 - Coef_i(\xi_{ij})$$

$$Coef_j(y_{ij})Coef_j(f(x_{ij})) \geq 1 - Coef_j(\xi_{ij}), Coef_k(y_{ij})Coef_k(f(x_{ij})) \geq 1 - Coef_k(\xi_{ij})$$

$$Coef_s(\xi_{ij}) \geq 0, Coef_i(\xi_{ij}) \geq 0, Coef_j(\xi_{ij}) \geq 0, Coef_k(\xi_{ij}) \geq 0 \quad j = 1, \dots, l,$$

where the subindex  $i = 1, \dots, D$ , each  $Coef_i(\xi_{ij})$  extracts the real coefficient of the multivector  $\xi_{ij}$  with respect to a quaternion basis vector, e.g.  $Coef_k(\xi_{ij})$  extracts with respect to the quaternion basis  $k$ .

The dual expression of this problem can be derived straightforwardly. Firstly let us consider the expression of the orientation of optimal hyperplane. Since the  $w_i = [w_{i1}, w_{i2}, \dots, w_{iD}]^T$ , each of the  $w_{ij}$  is given by the quaternion

$$w_{ij} = w_{is} + w_{ii}i + w_{ij}j + w_{ik}k. \tag{11}$$

Each component of these weights are computed as follows:

$$w_{is} = \sum_{j=1}^l ((\alpha_{is})_j (y_{is})_j) (x_{is})_j, w_{ii} = \sum_{j=1}^l ((\alpha_{ii})_j (y_{ii})_j) (x_{ii})_j, \tag{12}$$

$$w_{ij} = \sum_{j=1}^l ((\alpha_{ij})_j (y_{ij})_j) (x_{ij})_j, w_{ki} = \sum_{j=1}^l ((\alpha_{ik})_j (y_{ik})_j) (x_{ik})_j.$$

According the Wolfe dual programming [4] the dual form reads

$$L_D = \min \frac{1}{2}(w^H w) - \sum_{j=1}^l [(\alpha_{j1s} + \alpha_{j1i} + \alpha_{j1j} \alpha_{j1k}) + \dots + (\alpha_{jDs} + \alpha_{jDi} + \alpha_{jDj} + \alpha_{jDk})] \quad (13)$$

subject to  $a^T \cdot \mathbf{1} = 0$ , where the entries of the vector  $a = [a_s, a_i, a_j, a_k]$  are given by

$$\begin{aligned} a_s^T &= [((\alpha_{1s1})(y_{1s1}), (\alpha_{2s1})(y_{2s1}), \dots, (\alpha_{Ds1})(y_{Ds1})), \dots, [(\alpha_{1s1})(y_{1s1}), (\alpha_{2s1})(y_{2s1}), \dots, (\alpha_{Ds1})(y_{Ds1})]], \\ a_i^T &= [((\alpha_{1i1})(y_{1i1}), (\alpha_{2i1})(y_{2i1}), \dots, (\alpha_{Di1})(y_{Di1})), \dots, [(\alpha_{1i1})(y_{1i1}), (\alpha_{2i1})(y_{2i1}), \dots, (\alpha_{Di1})(y_{Di1})]], \\ a_j^T &= [((\alpha_{1j1})(y_{1j1}), (\alpha_{2j1})(y_{2j1}), \dots, (\alpha_{Dj1})(y_{Dj1})), \dots, [(\alpha_{1j1})(y_{1j1}), (\alpha_{2j1})(y_{2j1}), \dots, (\alpha_{Dj1})(y_{Dj1})]], \\ a_k^T &= [((\alpha_{1k1})(y_{1k1}), (\alpha_{2k1})(y_{2k1}), \dots, (\alpha_{Dk1})(y_{Dk1})), \dots, [(\alpha_{1k1})(y_{1k1}), (\alpha_{2k1})(y_{2k1}), \dots, (\alpha_{Dk1})(y_{Dk1})]], \end{aligned} \quad (14)$$

note that each data *ith*-vector,  $i = 1, \dots, N$ , has  $D$  quaternion entries and after the training we take into account not  $N$  but  $l$  *ith*-vectors which is the number of the found support vectors each one belonging to  $\mathbb{H}^D$ . Thus  $a^T$  has the dimension:  $(D \times l) \times 2^4$ , the latter multiplicand corresponds to the length of a quaternion of  $\mathbb{H}$ .

In  $a^T \cdot \mathbf{1} = 0$ ,  $\mathbf{1}$  denotes a vector of all ones, and all the Lagrange multipliers should fulfil  $0 \leq (\alpha_{is})_j \leq C, 0 \leq (\alpha_{ii})_j \leq C, 0 \leq (\alpha_{ij})_j \leq C, 0 \leq (ik)_j \leq C$  for  $i = 1, \dots, D$  and  $j = 1, \dots, l$ .

And we can generalize this for non-linear classification as

$$L_D = \frac{1}{2} \sum_{m=s,i,j,k} \sum_{n=s,i,j,k} [a_m^T K_{m,n} a_n] - \sum_{j=1}^l [(\alpha_{j1s} + \alpha_{j1i} + \alpha_{j1j} \alpha_{j1k}) + \dots + (\alpha_{jDs} + \alpha_{jDi} + \alpha_{jDj} + \alpha_{jDk})] \quad (15)$$

where  $K_{m,n}$  represents the part of the kernel result that correspond to the product of the parts  $m$  and  $n$ , for example, the part of  $i$  and  $j$  is  $k$ . Please note that the order of the products affect the result, just as the complex value SVM case, this kernels are not commutative, with the exception of the Gaussian kernel.

As explained in [2] we can have a compact an easy representation if we are using the Gaussian quaternion Gabor kernel, this will help for the programming of the algorithm. After some algebraic manipulations, we can rewrite equation (13) as a compact equation as follows

$$\begin{aligned} \min \quad & \frac{1}{2} w^{*T} w + C \cdot \sum_{j=1}^l [\xi_{1j}, \xi_{2j}, \dots, \xi_{Dj}] = \frac{1}{2} a^{*T} H a + C \cdot \sum_{j=1}^l [\xi_{1j}, \xi_{2j}, \dots, \xi_{Dj}] \quad (16) \\ \text{subject, to} \quad & \text{Coef}_s(y_{ij}) \text{Coef}_s(f(x_{ij})) \geq 1 - \text{Coef}_s(\xi_{ij}) \\ & \text{Coef}_i(y_{ij}) \text{Coef}_i(f(x_{ij})) \geq 1 - \text{Coef}_i(\xi_{ij}) \\ & \text{Coef}_j(y_{ij}) \text{Coef}_j(f(x_{ij})) \geq 1 - \text{Coef}_j(\xi_{ij}) \\ & \text{Coef}_k(y_{ij}) \text{Coef}_k(f(x_{ij})) \geq 1 - \text{Coef}_k(\xi_{ij}) \\ & \text{Coef}_s(\xi_{ij}) \geq 0, \text{Coef}_i(\xi_{ij}) \geq 0, \text{Coef}_j(\xi_{ij}) \geq 0, \text{Coef}_k(\xi_{ij}) \geq 0, , j = 1, \dots, l, \end{aligned} \quad (17)$$

where  $a$  is given by equation (4).

$H$  is a positive semi-definite matrix which is the expected quaternion *Gram* matrix. This matrix in terms of the matrices of the  $t$ -grade parts of  $\langle x^* x \rangle_t$ , is written as follows:

$$H = \begin{bmatrix} H_s & H_i & H_j & H_k \\ H_i^T & H_s & H_j & H_k \\ H_j^T & H_i^T & H_s & H_k \\ H_k^T & H_j^T & H_i^T & H_s \end{bmatrix}, \tag{18}$$

note that the diagonal entries equal to  $H_s$  and since  $H$  is a symmetric matrix the lower matrices are transposed. The optimal weight vector  $w$  is as given by equation 9.

The threshold  $b \in \mathbb{H}_n^D$  can be computed by using KKT conditions with the Quaternion support vectors as follows

$$\begin{aligned} b &= [b_1 b_2 b_3 \dots b_D] \\ &= [(b_{1s} + b_{1i}i + b_{1j}j + b_{1k}k)(b_{2s} + b_{2i}i + b_{2j}j + b_{2k}k), \dots, \\ &\quad (b_{Ds} + b_{Di}i + b_{Dj}j + b_{Dk}k)] \end{aligned} \tag{19}$$

$$= \sum_{j=1}^l (y_j - w^{*T} x_j) / l. \tag{20}$$

### 5 Experimental Analysis and Applications

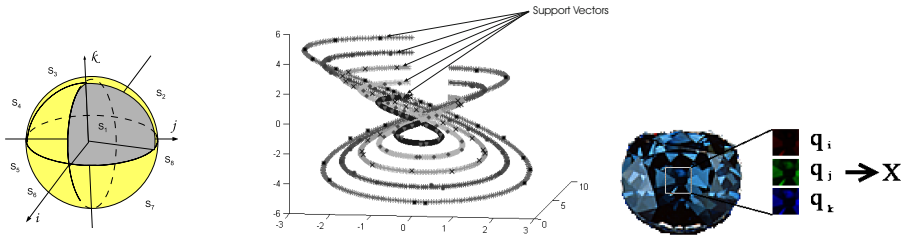
In this section we present three interesting experiments: solving the 3D XOR problem, the 3D spiral problem and classification of image colours. Each data *ith*-vector has quaternion entries  $x_i = [q_{i1}, q_{i2}, \dots, q_{iD}]^T$ , where  $q_{ij} \in \mathbb{H}$  and  $D$  is the dimension of this input vector.

**3D XOR: Nonlinear Classification.** We test with an extension of the XOR problem in 3D. The data *ith*-vector has one quaternion  $x_i = [q_{i1}] = [0, x_{i1}, y_{i1}, z_{i1}]$  where  $D=1$ . After training with 80 data vectors (10 for each quadrant), we get a 100 % efficiency using 8000 (1000 per quadrant) input quaternions non used during the training with 20% of additive noise, i.e. in the range of  $\pm 1$  a deviation of 0.4

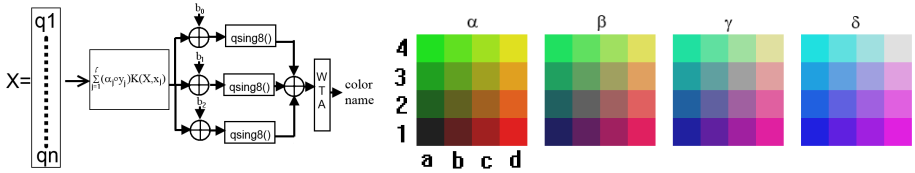
**3D Spiral: Nonlinear Classification Problem.** We extended the well known 2-D spiral problem to the 3-D space. This experiment should test whether the QSVM would be able to separate five 1-D manifolds embedded in  $\mathbb{R}^3$ . For this application, we used QSVM, this allows us to have quaternion inputs and outputs. The five functions were generated as follows:

$$f_i(t) = [x_i(t), y_i(t), z_i(t)] = [z_i * \cos(\theta) * \sin(\theta), z_i * \sin(\theta) * \sin(\theta), z_i * \cos(\theta)] \tag{21}$$

for  $i = 1, \dots, 5$ . In Figure 1.b one can see that the problem is high non-linear separable. The data *ith*-vector has nine quaternion entries  $x_i = [q_{i1}]$ ,  $D=1$ . The QSVM uses for training 50 input quaternions of each of the five functions, since these have three coordinates we use simply the imaginary quaternion basis of the quaternion, namely  $x_i = x_i(t)i + y_i(t)j + z_i(t)k \equiv [0, x_i(t), y_i(t), z_i(t)]$ . The QSVM used the kernel given by (7). Note that the QSVM indeed manage to separate the five classes.



**Fig. 1.** a) 8-state using pure quaternions  $\mathbb{H}$ . b) 3D spiral with five classes. The marks represent the support multivectors found by the CSVM. c) Quaternion sampling of a RGB image.



**Fig. 2.** a) The hyperplane with a bias varying  $[b_0, b_1, b_2]$  QSVM colour classifier. b) The RGB space  $[x, y, z]$ , divided in 64 sub cubes  $x=1, 2, 3, 4$ ;  $y=a, b, c, d$ ;  $z=\alpha, \beta, \gamma, \delta$ .

















**Colour Classification**

The colour classification is a difficult task principally due two aspects: colour on a surface is distributed not homogeneous and the recognition on of certain colour is intrinsically a human capacity dependent of the mud and culture of the observer. One can go even more deep in the analysis of the human capacity for colour recognition similar as the by other humans senses like touch, taste and smell. The problem becomes more difficult, if we want to build a sensor which mimics the human colour recognition capacity. The task is to design a classifier for 64 colours which can be used for system for visual impaired people, thus it makes no sense a device which classifies a bigger number of distinct colours. Important was to assign to each class the name of certain colour which is commonly accepted. We chose a QSVM with a input vector with  $D=64$  quaternions (sampling the colour image with a  $8 \times 8$  window, see Figure 1.c), with a identity kernel which first compute in equation (8), the orientation of the optimal hyperplane data  $\sum_{j=1}^l (\alpha_j \circ y_j)(x_j, x)$  and the varies the bias  $b$  for four levels:  $b_0, b_1, b_3$ , see Figure 2.a, the idea here is to shift this hyperplane with the offsets dividing the 3D quadrants of the RGB space in sub-cubes, thus we divide the RGB space in  $8x2^3$  cubes which yields 64 cubes for the clustering. The 4th valency state output will be  $y = qsing_8(f(x))$ . Apparently a simple approach, our classifier proof to be very efficient. The classifier depicted in Figure 2.a has at the output a Winner Take All (WTA) to assign a colour according the task in question. This 64 outputs of the classifier were index for 64 different colours variations shown in Figure 2.b .

For the test of our classifier, we use it to recognize colors in diamonds and wood. We select this kind of objects, because their surfaces have many variations in color. For the diamond experiment we select 8 types and proceed to classify them. As we can see in the Figure 3.a, the obtained color is similar to the one given by a human. For example,

the last two are categorized as blue, and we can see that one is a combination of cyan and blue, and the other a dark blue, so the WTA will be adjusted to give us a blue. The only error is the fourth diamond, which is classified as an orange one, instead of the expected yellow, however increasing the division of the colour scale, the classifier will be able to distinguish these different colours as well.

We repeat the same for wood type classification as shown in Figure 3.b. We can observe that similar kind of wood gave us the same color, like the second (special walnut) and third (red mahogany), and the fourth (early american) and sixth (cherry). One interesting detail is that only the clear colours have a component in blue, like the last two (oak and maple).

							
d,1, $\alpha$	d,3, $\delta$	d,3, $\alpha$	d,3, $\alpha$	a,3, $\beta$	c,2, $\delta$	a,3, $\delta$	a,1, $\beta$
							
d,2, $\alpha$	b,1, $\alpha$	b,1, $\alpha$	c,2, $\alpha$	d,3, $\alpha$	c,2, $\alpha$	d,3, $\beta$	d,4, $\gamma$

**Fig. 3.** a) (upper row) diamond colour types and the classification results. b) (lower row) wood colour types and the classification results.

## 6 Conclusions

This paper generalizes the real valued SVM to the quaternion valued SVM and it is used for classification. The QSVM accepts multiple multivector inputs like a MIMO architecture, that allows us to have multi-class applications. A key feature is that the QSVM formulation allows to change the metric involved in the quaternion product. The application section shows experiments in pattern recognition and colour image processing. The extension of the real valued SVM to the Quaternion SVM appears promising particularly by using high dimensional geometric primitives for geometric computing and their applications like graphics, augmented reality, robot vision and humanoids.

**Acknowledgement.** We are very thankful to the project CONACYT-SEP 2012 No. 17622 and the CONACYT PhD scholarship program for supporting this work.

## References

1. Bayro-Corrochano, E., Arana-Daniel, N., Vallejo-Gutierrez, R.: Design of Kernels for Support Multivector Machines Involving the Clifford Geometric Product and the Conformal Geometric Neuron. In: Int. Joint Conference on Neural Networks, Portland-Oregon, USA, pp. 2893–2898 (2003)

2. Bayro-Corrochano, E., Arana-Daniel, N.: Clifford support vector machines for classification, regression and recurrence. *IEEE Transactions on Neural Networks* 7(17), 1016–1035 (2007)
3. Burges, C.J.C.: A tutorial on Support Vector Machines for Pattern Recognition. *Knowledge Discovery and Data Mining* 2(2), 1–43 (1998)
4. Vapnik, V.: *Statistical Learning Theory*. Wiley, New York (1998)
5. Zhang, L., Zhou, W., Jiao, L.: Complex-valued support vector classifiers. *Digital Signal Processing* 20(3), 944–955