Color Image Segmentation with a Hyper-Conic Multilayer Perceptron

Juan Pablo Serrano, Arturo Hernández, and Rafael Herrera

Center for Research in Mathematics
Computer Science Department
Guanajuato, Guanajuato, Mexico
{jpsr,artha,rherrera}@cimat.mx
http://www.cimat.mx

Abstract. We apply the Hyper-Conic Artificial Multilayer Perceptron (HC-MLP) to color image segmentation, where we consider image segmentation as a classification problem distinguishing between foreground and background pixels. The HC-MLP was designed by using the conic space and conformal geometric algebra. The neurons in the hidden layer contain a transfer function that defines a quadratic surface (spheres, ellipsoids, paraboloids and hyperboloids) by means of inner and outer products, and the neurons in the output layer contain a transfer function that decides whether a point is inside or outside a sphere. The Particle Swarm Optimization algorithm (PSO) is used to train the HC-MLP. A benchmark of fifty images is used to evaluate the performance of the algorithm and compare our proposal against statistical methods which use copula gaussian functions.

Keywords: Color Image Segmentation, Artificial Neural Networks, Geometric Algebra.

1 Introduction

Image segmentation continues to be an important research area in computer vision applications. A great variety of segmentation algorithms and its applications have been proposed in the literature [1,2,3,4,5]. In this paper we use the Hyper-Conic Multilayer Perceptron (HC-MLP) [6] for color image segmentation, since the color image segmentation problem can be considered as a supervised pixel classification problem. We compare the performance of the HC-MLP with two classifiers: the copula-based probabilistic algorithm and the independent probabilistic model. The input data space consists of the RGB color space vectors of each pixel in the image without any preprocessing.

There exist several proposals in the literature [7,8,9] for color image segmentation using artificial neural networks. Our proposal is to use the higher order hyper-conic neuron of the HC-MLP instead of the hyperplane neuron of the classical MLP. The neurons in the hidden layer (the hyper-conic neurons) and output layer (spherical neurons) of the HC-MLP have transfer functions that define quadratic surfaces and spherical surfaces [10] respectively. The hyper-conic

R. Klette, M. Rivera, and S. Satoh (Eds.): PSIVT 2013, LNCS 8333, pp. 360–371, 2014.

and spherical neurons allow us to simplify the topology of the multilayer perceptron and, at the same time, produce more complex and flexible non-linear decision regions to classify the points in the input data space. The design of the neurons is based on geometrical ideas of (conformal) geometric algebra by using the inner and outer products.

The rest of this paper is organized as follows: In Section 2, we provide an overview of conformal geometric algebra and the conic space. In Section 3, we review the PSO algorithm. In Section 4, we give a detailed implementation of the HC-MLP. In Section 5 we describe the experiments and compare our results with the supervised probabilistic classifiers. Finally, conclusions and future work are discussed in Section 6.

2 Geometric Preliminaries

2.1 Conformal Geometric Algebra

The conformal model provides a way of dealing with Euclidean Geometry within a higher dimensional space. Let $e_0, e_1, ..., e_n, e_\infty$ be a basis of $\mathbb{R}^{n+2}$. We will consider the Euclidean points $x = x_1e_1 + x_2e_2 +, ..., x_ne_n \in \mathbb{R}^n$ to be mapped to points in $\mathbb{R}^{n+1,1}$ according to the following conformal transformation:

$$\mathbb{P}(x) = (x_1e_1 + x_2e_2 + ... + x_ne_n) + \frac{1}{2}x^2e_\infty + e_0, \tag{1}$$

where $\mathbb{R}^{n+1,1}$ is a $(n+2)$-dimensional real vector space that includes the Euclidean space $\mathbb{R}^n$ and has two more independent directions generated by two basic vectors e_0 and e_∞ [11]. Furthermore, it is endowed with a scalar product of vectors satisfying: $e_i \bullet e_i = 1$, $e_i \bullet e_j = 0$, $e_i \bullet e_\infty = 0$, $e_i \bullet e_0 = 0$, $e_\infty \bullet e_0 = -1$, $e_\infty \bullet e_\infty = 0$, $e_0 \bullet e_0 = 0$, where $1 \leq i, j \leq n$. Note that the Euclidean points $x \in \mathbb{R}^n$ are now mapped into points of the null cone in $\mathbb{R}^{n+1,1}$. The vectors e_0 and e_∞ represent the origin and a point at infinity respectively. The number x^2 is:

$$x^2 = x \bullet x = \sum_{i=1}^{n} x_i^2. \tag{2}$$

A hyper-sphere in $\mathbb{R}^n$ is determined by its center $c \in \mathbb{R}^n$ and its radius $\rho \in \mathbb{R}$. Its conformal model representation is the point:

$$\mathbb{S}(c, \rho) = \mathbb{P}(c) - \frac{1}{2}\rho^2 e_\infty. \tag{3}$$

The representation of the sphere s in *Outer Product Null Space* notation (OPNS) in $\mathbb{R}^n$, can be written with the help of $n+1$ conformal points that lie on it, i.e.

$$s^* = \bigwedge_{i=1}^{n+1} \mathbb{P}(x_i). \tag{4}$$

A hyper-plane Θ in $\mathbb{R}^n$ is represented by:

$$\Theta(n, \delta) = n + \delta e_\infty, \tag{5}$$

where n is the normal vector to the hyper-plane in $\mathbb{R}^n$ and $\delta > 0$ is its oriented distance (with respect to n) to the origin. A hyper-plane in $\mathbb{R}^n$ can be defined by n points belonging to it. Thus, in the conformal model, it is represented by the outer product of the n image conformal points plus the vector e_∞, i.e.

$$\Theta^* = \left(\bigwedge_{i=1}^{n} P_i \right) + e_\infty. \tag{6}$$

Distance between points

The inner product between the points $\mathbb{P}(x)$ and $\mathbb{P}(y)$ is directly proportional to the square of the Euclidean distance of the points $\{x, y\}$ multiplied by -2,

$$\mathbb{P}(x) \bullet \mathbb{P}(y) = x \cdot y - \frac{1}{2}y^2 - \frac{1}{2}x^2 = -\frac{1}{2}(x - y)^2 \tag{7}$$

$$(x - y)^2 = -2(\mathbb{P}(x) \bullet \mathbb{P}(y)) \tag{8}$$

The conformal points $\mathbb{P}(x)$ and $\mathbb{P}(y)$ are null and as a consequence, $\mathbb{P}(x) \bullet \mathbb{P}(y) = 0$ if and only if $x = y$.

Distance between a point and a hyper-plane

Similarly, it is possible to calculate the signed distance between a point and a hyper-plane. For a point $\mathbb{P}(x)$ and hyper-plane $\Theta(n, \delta)$, the signed distance of the point to the hyper-plane is given by:

$$\mathbb{P}(x) \bullet \Theta(n, \delta) = x \cdot n - \delta. \tag{9}$$

The distance is zero if the point x lies on the hyper-plane. The distance is positive if the point x is on the same side of the plane as the normal vector n and negative if it is on the opposite side [11].

Point inside or outside a hyper-sphere

The inner product between a point $\mathbb{P}(x)$ and a hyper-sphere $\mathbb{S}(c, \rho)$

$$\mathbb{P}(x) \bullet \mathbb{S}(c, \rho) = \frac{1}{2}\rho^2 - \frac{1}{2}(c - x)^2 \tag{10}$$

can be used to decide whether a point is inside or outside of the hyper-sphere [11]:

- $\text{Sign}(\mathbb{P}(x) \bullet \mathbb{S}(c, \rho)) > 0 : x$ is inside the hyper-sphere.
- $\text{Sign}(\mathbb{P}(x) \bullet \mathbb{S}(c, \rho)) < 0 : x$ is outside the hyper-sphere.
- $\mathbb{P}(x) \bullet \mathbb{S}(c, \rho) = 0 : x$ is on the hyper-sphere.

2.2 Conic Space Model

By identifying points $\overline{x} = (x_1, x_2) \in \mathbb{R}^2$ with points $\overrightarrow{x} = (x_1, x_2, 1)$, it is well known that for a given symmetric 3×3 matrix A, the set of points $(x_1, x_2) \in \mathbb{R}^2$ such that

$$\begin{bmatrix} x_1 & x_2 & 1 \end{bmatrix} \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{1,2} & a_{2,2} & a_{2,3} \\ a_{1,3} & a_{2,3} & a_{3,3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix} = 0, \tag{11}$$

lie on a conic. Equation (11) can be written as the scalar product between two vectors of $\mathbb{R}^6$:

$$\mathbb{D}(x_1, x_2) \cdot \mathcal{T}(A) = 0 \Longleftrightarrow a_{1,1}x_1^2 + 2a_{1,2}x_1x_2 + 2a_{1,3}x_1 + a_{2,2}x_2^2 + 2a_{2,3}x_2 + a_{3,3} = 0 \tag{12}$$

where

$$\mathbb{D} : (x_1, x_2) \in \mathbb{R}^2 \mapsto \begin{bmatrix} x_1^2 & x_1x_2 & x_1 & x_2^2 & x_2 & 1 \end{bmatrix} \in \mathbb{R}^6 \tag{13}$$

and $\mathbb{D}(\mathbb{R}^2) = \mathbb{D}^2$,

$$\mathcal{T} \ : \ A \ \in \mathbb{R}^{3\times 3} \mapsto \begin{bmatrix} a_{1,1} & 2a_{1,2} & 2a_{1,3} & a_{2,2} & 2a_{2,3} & a_{3,3} \end{bmatrix}^T \tag{14}$$

The Inner Product Null Space (IPNS) of $a = \mathcal{T}(A) \in \mathbb{R}^6$ is the set of all the points $X = \mathbb{D}(x_1, x_2) \in \mathbb{D}^2 \subset \mathbb{R}^6$ satisfying $X \cdot a = 0$, i.e. the points belonging to the conic represented by A. Given $a = \mathcal{T}(A), b = \mathcal{T}(B) \in \mathbb{R}^6$, the bivector $a \wedge b$ represents the intersection of the conics defined by A and B. The vectors a and b are linearly independent if they represent different conics. More precisely, the IPNS of $a \wedge b$ is the set of points X such that $0 = X \rfloor (a \wedge b) = (X \cdot a)b - (X \cdot b)a$, i.e. X belongs to both conics. The symbol $\rfloor$ denotes contraction of multivectors.

The Outer Product Null Space (OPNS) of a multivector $C \in Cl_6$ is the set of points X satisfying $X \wedge C = 0$. A conic through $(\overline{x}_1, \overline{x}_2, \overline{x}_3, \overline{x}_4, \overline{x}_5)$ is represented by $C = X_1 \wedge X_2 \wedge X_3 \wedge X_4 \wedge X_5$, where $X_i = \mathbb{D}(\overline{x}_i)$ for $i = 1, .., 5$. For instance, the distance from a point $X_0 = \mathbb{D}(x_{1,0}, x_{2,0}) \in \mathbb{D}^2$ and a multivector $C_{<5>} \in Cl(\mathbb{R}^6)$ can be obtained as follows:

$$dist(X, C_{<5>}) = (X \wedge C_{<5>})^* = (X \wedge C_{<5>})I_{<6>}^{-1}. \tag{15}$$

If $C = X_1 \wedge X_2 \wedge X_3 \wedge X_4 \wedge X_5$ then

$$\begin{aligned} dist(X_0, C_{<5>}) &= (X_0 \wedge X_1 \wedge X_2 \wedge X_3 \wedge X_4 \wedge X_5)^* \\ &= \det \begin{vmatrix} x_{1,0}^2 & x_{1,0}x_{2,0} & x_{1,0} & x_{2,0}^2 & x_{2,0} & 1 \\ x_{1,1}^2 & x_{1,1}x_{2,1} & x_{1,1} & x_{2,1}^2 & x_{2,1} & 1 \\ x_{1,2}^2 & x_{1,2}x_{2,2} & x_{1,2} & x_{2,2}^2 & x_{2,2} & 1 \\ x_{1,3}^2 & x_{1,3}x_{2,3} & x_{1,3} & x_{2,3}^2 & x_{2,3} & 1 \\ x_{1,4}^2 & x_{1,4}x_{2,4} & x_{1,4} & x_{2,4}^2 & x_{2,4} & 1 \\ x_{1,5}^2 & x_{1,5}x_{2,5} & x_{1,5} & x_{2,5}^2 & x_{2,5} & 1 \end{vmatrix} \\ &= a_{1,1}x_{1,0}^2 + 2a_{1,2}x_{1,0}x_{2,0} + 2a_{1,3}x_{1,0} \\ &\quad + a_{2,2}x_{2,0}^2 + 2a_{2,3}x_{2,0} + a_{3,3} \\ &= \mathcal{T}(\overrightarrow{x}^T\overrightarrow{x}) \cdot \overline{a}, \end{aligned} \tag{16}$$

3 Particle Swarm Optimization (PSO)

A PSO algorithm consists of a swarm of particles, each of which is a potential solution to the optimization problem, which are dispersed in a multidimensional search space as follows: the position of each particle is adjusted according to its own experience and those of its neighbors [12]. The social component of the particle velocity update reflects information obtained from all the particles in the swarm. In this sense, the social information of the particle i is the best position found by the swarm, referred to by $\overline{g}_i$. A cognitive component represents the personal search of each particle, referred to by the vector $\overline{l}_i$. The velocity $v_{ij}(t+1)$ of particle i in the j-th canonical direction, $j = 1, ..., n$, at time step $t+1$, is calculated as follows:

$$v_{ij}(t+1) = wv_{ij}(t) + c_1 r_{1j}(t)[l_{ij}(t) - x_{ij}(t)] + c_2 r_{2j}(t)[g_{ij}(t) - x_{ij}(t)] \quad (17)$$

where w is the inertia weight, $x_{i,j}(t)$ is the position of particle i in the j-th direction at time step t, c_1 and c_2 are positive acceleration constants used to scale the contribution of the cognitive and social components respectively. Finally, $r_{1j}(t), r_{2j}(t) \sim U(0,1)$ are random values in the range $[0,1]$, sampled from a uniform distribution. The pseudo-code of the PSO algorithm is illustrated in Figure 1.

4 Hyper-Conic Neural Network

The hyper-conic neuron contains a transfer function that defines decision boundaries such as spheres, ellipsoids, paraboloids and hyperboloids. Figure 2 shows the architecture of the HC-MLP whose hyper-conic neurons in the hidden layer use the outer product in conic space. The input signals $x = [x_1, x_2, ..., x_n]$ are propagated through the network from left to right. The symbols $o_1, ...o_k, ..o_p$ denote the output vector $\overline{o}$ from the hidden layer to the output layer. The output vector of the output layer are described by the vector $\overline{y} = [y_1, y_2, ..., y_k, ..., y_q]$. Therefore, the output of the neuron o_i in the hidden layer can be written as a function composition of a non-linear associator and the sigmoid function G as follows:

$$o_i = \frac{1}{1 + exp(-(X \wedge C^i)I^{-1})}, \quad (18)$$

where

$$\beta = \frac{n^2 + 3n + 2}{2} \quad (19)$$

is the dimension of the conic space $\mathbb{R}^\beta$, $C^i = \mathbb{D}(\overline{x}_1) \wedge \mathbb{D}(\overline{x}_2) \wedge \cdots \wedge \mathbb{D}(\overline{x}_{\beta-1})$ is a $(\beta-1)$-blade, the points $\overline{x}_j \in \mathbb{R}^2$ are the points to be estimated, and I denotes the pseudoscalar of the conic space $\mathbb{R}^\beta$. The distance computed in the non-linear associator in (18) can, in general, be positive or negative, depending on the position of the point with respect to the conic. As we have seen, the distance is zero if the point is on the conic surface.

Algorithm 1. PSO Algorithm

for all particle i **do**
  //Create and initialize an $n-dimensional$ swarm, S
  //position $\overline{x} = x_1, .., x_n$ and velocity $\overline{v} = v_1, .., v_n$.
end for
while stop criteria not met **do**
  for all particle i **do**
    if f($S_i.\overline{x}$) < f($S_i.\overline{lb}$) **then**
      $S_i.\overline{l} = S_i.\overline{x}$ //set personal best $\overline{l} = \{l_1, ..., l_n\}$ as best position found so far by the particle.
    end if
    if f($S_i.\overline{x}$) < f($S_i.\overline{gb}$) **then**
      $S_i.\overline{g} = S_i.\overline{x}$ //set global best $\overline{g} = \{g_1, ..., g_n\}$ as best position found so far by the whole swarm.
    end if
  end for
  for all particle i **do**
    for each canonical direction indexed by j **do**
      $S_i.v_j(t+1) = w \bullet S_i.v_j(t) + c_1 \bullet r_{1j}(t) \bullet [S_i.l_j(t) - S_i.x_j(t)]$
        $+ c_2 \bullet r_{2j}(t) \bullet [S_i.g_j(t) - S_i.x_j(t)]$ //update the velocity.
      $S_i.x_j(t+1) = S_i.x_j(t) + S_i.v_j(t+1)$ //update position
    end for
  end for
end while

Fig. 1. Particle Swarm Optimization Algorithm

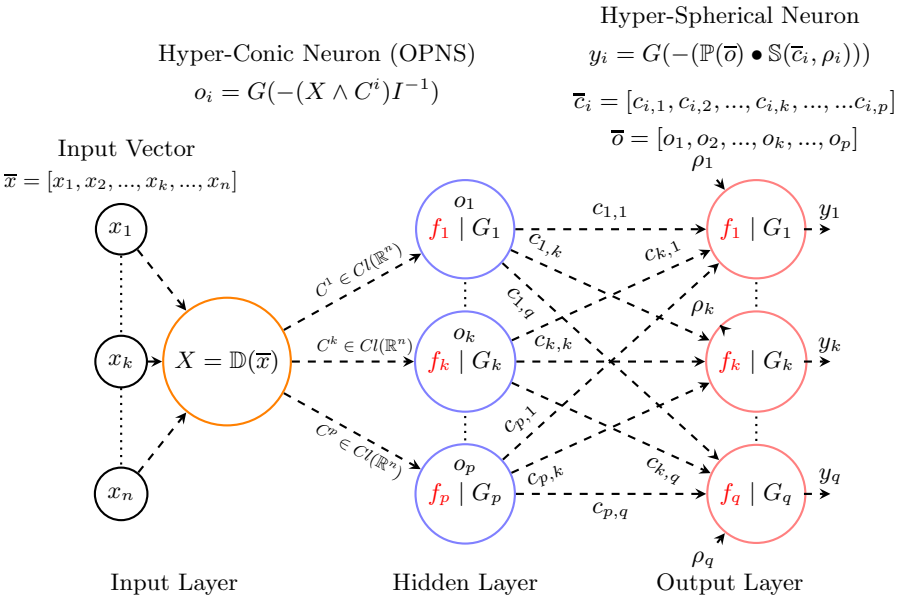


Fig. 2. HC-MLP Model where the HCN is implemented by using the outer product null space

The output y_i for the neuron i in the output layer is written as a function composition as follows:

$$y_i = \frac{1}{1 + exp(-(\mathbb{P}(\overline{o}) \bullet \mathbb{S}(\overline{c}_i, \rho_i)))} \tag{20}$$

where $\mathbb{P}(\overline{o})$ is the conformal point of the output vector of the neurons in the hidden layer, $\mathbb{S}(\overline{c}_i, \rho_i)$ is the hyper-sphere (represented in conformal space) whose center is the point $\overline{c}_i$ given by the weight vector of the neuron i in the output layer and its radius is ρ_i. A particle swarm algorithm is used for training the HC-MLP. The training stage of the HC-MLP finds the vector of parameters that determines the quadratic surfaces of the neurons in the hidden layer, as well as, the centers and radii of the spheres in the output layer. The parameters are enconded in a vector that represents a particle for the PSO algorithm.

5 Experimental Setup and Results

Since color image segmentation can be interpreted as a supervised classification problem, we use the HC-MLP to classify the pixels of 50 test images. The experimental setup is as follows:

- The patterns which are used during the training and testing stages belong to 3-dimensional space.
- The components of the input vector of each pattern are given by the values of the RGB color model.
- We work with the images from the databases used in [14,2] and available online at [15]. Three images of the database are presented in Figure 4.
- The training patterns are taken from labelling-lasso images which are divided into two sets referenced as a training set and a testing set.
- Pixels of each image were normalized by the min-max normalization method in the interval $[-1, +1]$.
- The outputs are encoded as -1 for foreground and +1 for background.
- The objective function is based on the mean square error function (MSE).
- The stop criteria is given by the number of epochs in the interval [1,100] or whether the MSE on the validation data set presents an increment during the training.
- The population size for the PSO algorithm is set to 20 particles initialized in the interval (-100,+100).
- The parameters used in the PSO algorithm are given as follows [13]:
  - inertia weight value is set to 0.7;
  - social component value is set to 1.49618;
  - cognitive component value is set to 1.49618.
- For comparison purposes, the tree evaluation measures which are used in this work are (see Figure 3):
  - accuracy;
  - sensitivity;
  - specificity.

The sensitivity and specificity measures explain the percentage of well classified pixels for each class, foreground and background, In the last column of the figure 4, we present two examples of classifications done with the HC-MLP. For

		Truth	
		Positive	Negative
Model	Positive	tp	fp
	Negative	fn	tn

(a)

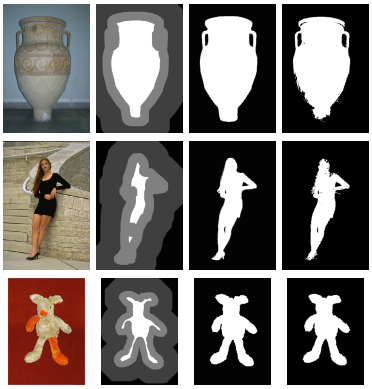
$$accuracy = \frac{tp+tn}{tp+fp+fn+tn}$$

$$sensitivity = \frac{tp}{tp+fn}$$

$$specificity = \frac{tn}{tn+fp}$$

(b)

Fig. 3. (a) A confusion matrix for binary classification. (b) Definitions of accuracy, sensitivity and specificity



(a) (b) (c) (d)

Fig. 4. (column a) The color image. (column b) The labelling-lasso image with training data for background (dark gray), for foreground (white) and the validation data (gray). (column c) The correct classification with foreground (white) and background (black). (column d) Classification with the HC-MLP.

comparison purposes, Table 1 shows the minimum, median, mean, maximum and standard deviation values of the specificity, sensitivity and accuracy values obtained by using the I-M and the GC-M models [2]. We present the results obtained by the HC-MLP when using 1, 2 and 3 neurons in the hidden layer.

According to Table 1, the HC-MLP presents a better performance than the I-M model for all evaluation measures, except for the minimum values. The mean accuracy of the HC-MLP when using 1, 2 or 3 neurons in the hidden layer is better than the mean accuracy of the I-M model (79.5%). For instance, the I-M model presents 8.12% more errors than the HC-MLP with 2 neurons in the hidden layer.

We found that the best architecture for the HC-MLP for this classification problem is the one using 2 neurons in the hidden layer. The mean value for the specificity measure given by the HC-MLP is better than the average of the specificity produced by the GC-M model. The average of the accuracy value using the HC-MLP is slightly better (approximately 0.5%) than the average accuracy produced by the GC-M. The maximum values for all evaluations using HC-MLP are better than those produced by the GC-M. However, the average of the sensitivity by using the GC-M is better than that produced by the HC-MLP, and so are minimum values. On the other hand, there are cases in which

Table 1. Descriptive results for all evaluation measures. BG stands for the background class and FG stands for the foreground class. (The best values are typed in boldface.)

Measure	Minimum	Median	Mean	Maximum	Std. deviation
I-M					
Specificity - BG	0.4040	0.8570	0.8170	0.9930	0.1330
Sensitivity - FG	0.3950	0.7630	0.7680	1.0000	0.1720
Accuracy	0.5710	0.7920	0.7950	0.9760	0.1070
GC-M					
Specificity - BG	**0.5510**	0.9240	0.8850	0.9940	0.1080
Sensitivity - FG	**0.4840**	0.8750	**0.8540**	0.9980	0.1270
Accuracy	**0.5870**	0.8890	0.8710	0.9870	0.0830
HC-MLP - 1 Neuron					
Specificity - BG	0.1487	0.9565	0.9130	**1.000**	0.1376
Sensitivity - FG	0.0344	**0.8813**	0.7995	**1.000**	0.2227
Accuracy	0.5244	0.8814	0.8648	**1.000**	0.1070
HC-MLP - 2 Neurons					
Specificity - BG	0.0993	0.9684	0.9269	**1.000**	0.1363
Sensitivity - FG	0.0000	0.8686	0.8085	**1.000**	0.2157
Accuracy	0.5196	**0.8899**	**0.8762**	**1.000**	0.1054
HC-MLP - 3 Neurons					
Specificity - BG	0.2069	**0.9762**	**0.9358**	**1.000**	0.1196
Sensitivity - FG	0.0409	0.8336	0.7712	**1.000**	0.2464
Accuracy	0.5206	0.8884	0.8648	**1.000**	0.1144
HC-MLP - from 1 to 3 Neurons					
Specificity - BG	0.1487	0.9597	0.9298	1.000	0.1265
Sensitivity - FG	0.1585	0.8962	0.8421	1.000	0.1752
Accuracy	0.5244	0.9183	0.8919	1.000	0.0942

either the classification problem is linearly separable or the decision regions can be efficiently designed using only one conic surface. For instance, Figure 5 shows two images of different complexity according to their data distributions, which are shown in the cubes of the third column. It is possible to see that for the first image, it is enough to use only one conic surface, while for the second one it is necessary to have a higher number of conics to produce the correct decision boundary separating background and foreground. In this sense, we selected the best values obtained by using 1,2 and 3 neurons in the hidden

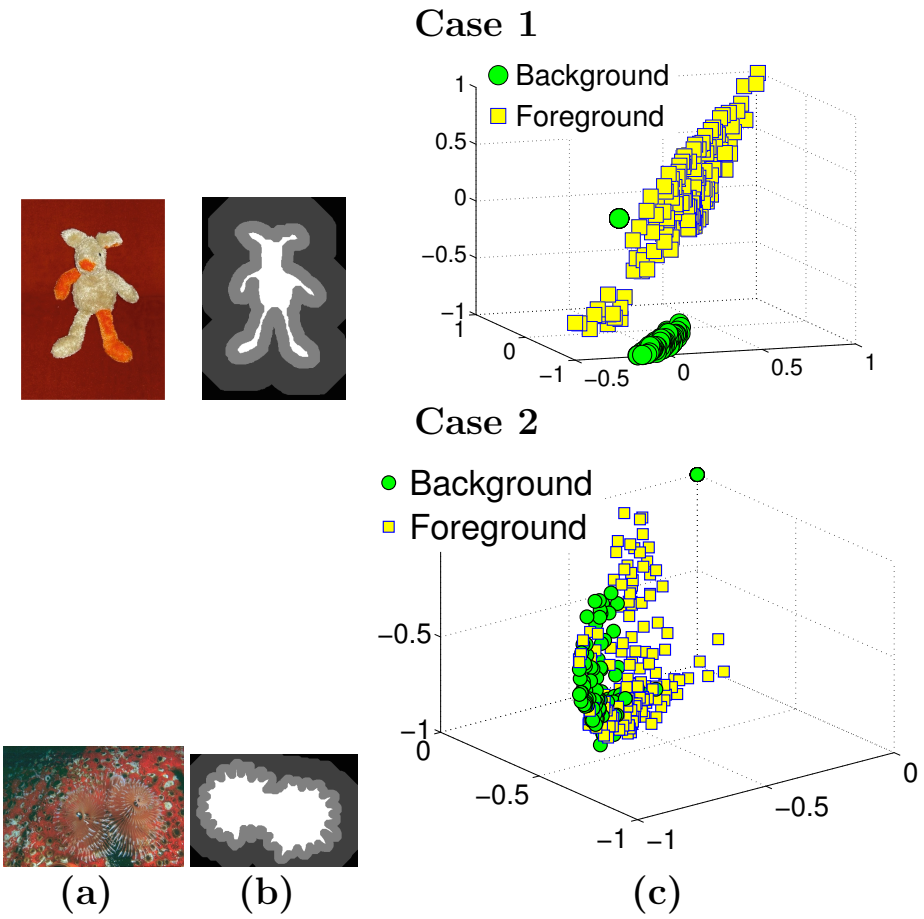


Fig. 5. (column a) Original Image. (column b) The labelling-lasso image with training data for background (dark gray), for foreground (white) and the validation data (gray). (column c) Space structure of the training set and validation set.

layer for each image. The results obtained are presented at the end of Table 1. The HC-MLP presents a better performance than those of the I-M and the GC-M for all evaluation measures, except for the minimum values produced by the I-M and GC-M models.

6 Conclusions

In this paper, we use the Hyper-Conic Multilayer Perceptron for color image segmentation, since the color image segmentation problem can be considered as a supervised pixel classification problem. We found that the best architecture for the HC-MLP for this classification problem is the one using 2 neurons in the

hidden layer. HC-MLP presents a better performance than the I-M model. The average of the accuracy value using HC-MLP is slightly better then the average accuracy produced by the GC-M. According to Table 1, HC-MLP obtains the best value to classify the foreground and background with respect to the I-M and GC-M models on most images. However, there are images for which the number of neurons (1 to 3) used in the hidden layer is not sufficient.

On the other hand, we have also selected the best values produced by the different architectures composed by 1, 2 and 3 neurons in the hidden layer of the HC-MLP. The aim is to show that it is neccesary to apply different architectures depending on the pixel distribution of the images in order to achieve the segmentation. In this paper, we have chosen 1, 2 and 3 neurons to simplify the topology of the HC-MLP for this image database. The immediate future work is to use a HC-MLP endowed with a higher number of neurons in order to deal with those images which require a more complex architecture. Furthermore, we will also investigate the design of an algorithm to adapt the number of neurons in the hidden layer according to the pixel distribution of each image.

Acknowledgments. The authors gratefully acknowledge the financial support from the National Council for Science and Technology of Mexico (CONACyT) and from the Center for Research in Mathematics (CIMAT). The third author would like to thank the International Centre for Theoretical Physics for its hospitality and support.

References

1. Cheng, H.-D., Jiang, X.H., Sun, Y., Wang, J.: Color image segmentation: advances and prospects. Pattern Recognition 34(12), 2259–2281 (2001)
2. Salinas-Gutiérrez, R., Hernández-Aguirre, A., Rivera-Meraz, M.J.J., Villa-Diharce, E.R.: Supervised Probabilistic Classification Based on Gaussian Copulas. In: Sidorov, G., Hernández Aguirre, A., Reyes García, C.A. (eds.) MICAI 2010, Part II. LNCS, vol. 6438, pp. 104–115. Springer, Heidelberg (2010)
3. Szeliski, R.: Computer vision: algorithms and applications, pp. 235–269. Springer (2011)
4. Hernandez-Lopez, F.J., Rivera, M.: Binary segmentation of video sequences in real time. In: MICAI, pp. 163–168. IEEE Proceedings (2010)
5. Hernandez-Lopez, F.J., Rivera, M.: Change detection by probabilistic segmentation from monocular view. In: Machine Vision and Applications (2012) (to appear)
6. Serrano Rubio, J.P., Hernández Aguirre, A., Herrera Guzmán, R.: A Conic Higher Order Neuron Based on Geometric Algebra and Its Implementation. In: Batyrshin, I., Mendoza, M.G. (eds.) MICAI 2012, Part II. LNCS, vol. 7630, pp. 223–235. Springer, Heidelberg (2013)
7. Lescure, P., Meas-Yedid, V., Dupoisot, H., Stamon, G.: Color segmentation on biological microscope images. In: Proceeding of SPIE, Application of Artificial Neural Networks in Image Processing IV, San Jose, California, January 28-29, pp. 182–193 (1999)
8. Rae, R., Ritter, H.J.: Recognition of human head orientation based on artificial neural networks. IEEE Trans. Neural Network 9(2), 257–265 (1998)

9. Fang, Y., Pan, C., Liu, L.: On-line training of neural network for color image segmentation. In: Wang, L., Chen, K., S. Ong, Y. (eds.) ICNC 2005. LNCS, vol. 3611, pp. 135–138. Springer, Heidelberg (2005)
10. Perwass, C., Banarer, V., Sommer, G.: Spherical decision surfaces using conformal modelling. In: Michaelis, B., Krell, G. (eds.) DAGM 2003. LNCS, vol. 2781, pp. 9–16. Springer, Heidelberg (2003)
11. Dorst, L., Fontijne, D., Mann, S.: Geometric Algebra for Computer Science: An Object-Oriented Approach to Geometry. The Morgan Kaufmann Series in Computer Graphics, pp. 23–57, 355–389. Morgan Kaufmann Publishers Inc., San Francisco (2007)
12. Engelbrecht Andries, P.: Fundamental of Computational Swarm Intelligence. Wiley (2005)
13. Van den Bergh, F., Engelbrecht, A.P.: A study of particle swarm optimization particle trajectories. Information Sciences 176(8), 937–971 (2006)
14. Blake, A., Rother, C., Brown, M., Perez, P., Torr, P.: Interactive image segmentation using an adaptive GMMRF model. In: Pajdla, T., Matas, J(G.) (eds.) ECCV 2004. LNCS, vol. 3021, pp. 428–441. Springer, Heidelberg (2004)
15. Rother, C., Kolmogorov, V., Blake, A., Brown, M.: Image and video editing, `http://research.microsoft.com/en-us/um/cambridge/projects/visionimagevideoediting/segmentation/grabcut.htm`