Color Image Segmentation with SOCNet algorithm

Philippe Biela Enberg [1] [2]; Jack-Gérard Postaire [2]

(1) ERASM - HEI
13 rue de Toul
59046 Lille Cedex - France
Philippe.Biela@hei.fr

(2) LAGIS - UMR CNRS 8146
Université des Sciences et Technologies de Lille
59655 Villeneuve d'Asq – France
Jack-Gerard.Postaire@univ-lille1.fr

Abstract **- In this paper we present a neural network algorithm for color image segmentation. We use for image segmentation some recent results from research in the field of unsupervised classification of multidimensional observations with self-organizing circular networks (SOCNet). The segmentation is performed thanks to a set of networks trained in parallel, using a competitive strategy based on unsupervised learning rules. The learning data set is composed of the color pixels of the image. At the end of the learning phase, each network represents a specific cluster of pixels scattered in the RGB color data space. A quantization task, which consists of a classification procedure, is used to replace the color of each pixel in the image by an equivalent color extracted from the closest network.**

I. Introduction

Segmentation is an important stage in artificial vision system, which includes region detection by texture and color labeling. The segmentation operation will produce a simplified image containing labeled regions with similar attributes; the new image can provide information to higher vision modules tasks to allow an easier description of the scene. Techniques for segmentation are commonly: pixel-based techniques, region-based techniques, edge-based techniques and fuzzy or neural network techniques. In this paper we purpose a new approach for segmentation of color images with neural network technique as Self-Organizing Circular Networks.

II. Self-Organizing Circular Networks

Our method for segmentation of color images is based on the use of competitive neural networks with circular architecture [1].

The neural network and the learning procedure are related to Kohonen's maps, which makes it possible to adapt the structure of the map to the density distribution of the observations presented inside the space of representation thanks to the self-organizing and convergence properties of Kohonen's maps [2] [3]. These properties have been especially studied with 1D dimensional structure and have given important mathematical results for self-organization and convergence properties and in the same way with data distribution representation capacities [4].

Figure (1) shows the architecture of such a Self Organizing Circular Network (SOCNet) composed of K neurons arranged as a closed chain, so that each unit is attached to its two contiguous neighbors. The chain consists then in K neurons units denoted N_k , k = 1, .., K. Each neuron is associated with a weight vector W_k , k = 1, .., K which corresponds to the position of the neuron inside the data space observation. It means that weight vectors and data observations have the same dimension D. We denote $X = [x_1, x_2, \ldots, x_d, \ldots, x_D]^T$ the vector for data observation and consider that the data set sample is constituted by N observations X as $\chi = \{X_1, \ldots X_n, \ldots X_N\}$.

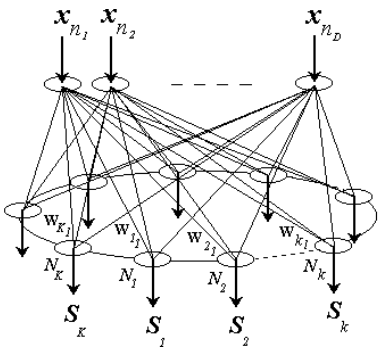

Figure 1: Self Organizing Circular Network architecture (SOCNet)

It means that all attributes x_d are valued by weight w_{kd} before being computed by an activation function F(.) for each neurone N_k. This function is the same for every neurone and consists in computing the Euclidean distance between the position of the data X and the neurone N_k indicated by its weight vector W_k inside the data space observation. The result is present in output O_k of neurone N_k.

We use a classical learning rule as LVQ [5] to adapt the neurones in the observation data space according to the density distribution of the data. This procedure will adapt the weights of the neurones in such a way that positions of the neurones become close to the positions of parts of the space where density distribution of the data is the more important. At the end of the learning procedure the shape of the neural architecture shows a one dimensional structure which curves among the data in such a way that neighboring neurones by their index (k) inside the SOCNet correspond to the data which are relatively close each one the other by their positions inside the data space representation.

The algorithm uses in a parallel way a few identical SOCNets designed as S^r, $r = 1, 2, ..., R$. Each one of these networks will indicate a cluster of homogeneous data inside the data space observation. The number of SOCNets used will correspond to the number of clusters searched. In case of applications in the field of color image segmentation, the number of clusters searched is often chosen in an emperical way in connection with the nature of the images.

The learning procedure for SOCNets is composed of two levels: one concerns adaptation of networks in global position, while the second level concern local adaptation of neurons inside each SOCNet. We present these two levels with spots to underline their complementarities in clustering process for data samples.

The global step level is based on an iterative adaptative process. At each duration step (t), an observation $X(t)$ is selected among the sample data set χ and presented simultaneously to all SOCNets. Then, Euclidean distance $dE_k^r(t)$ (Eq. 1) is computed for every neuron from any SOCNet S^r. The output of neurone N_k^r is updated with the result of the distance computed between the position of neurone N_k^r given by W_k^r and the data $X(t)$.

$$dE_k^r(t) = \sum_{d=1}^{D} \left(x_d(t) - w_{kd}^r(t) \right)^2 \qquad (1)$$

with:

$$X(t) = [x_1(t), ..., x_d(t), ..., x_D(t)]^T$$

$$W_k^r(t) = \left[w_{k1}^r(t),.., w_{kd}^r(t),.., w_{kD}^r(t) \right]^T$$

A Mahalanobis distance $dM^r(t)$ between the data $X(t)$ and the global position of the SOCNet S^r is simultaneously computed for each SOCNet. Equation (2) gives details for distance dM^r where $\overline{W}^r$ represents the average position of the SOCNet and Σ^r represents the covariance matrix for neurones positions inside SOCNet S^r.

$$dM^r(t) = \frac{\exp\left(-\frac{1}{2} \left(X(t) - \overline{W}^r(t) \right)^T \left(\Sigma^r(t) \right)^{-1} \left(X(t) - \overline{W}^r(t) \right) \right)}{(2\pi)^{D/2} \left[\Sigma^r(t) \right]^{1/2}} \qquad (2)$$

with:

$$\overline{W}^r(t) = \frac{\sum_{k=1}^{K} W_k^r(t)}{K}$$

$$\sigma^{2r}_{ij} = \sum_{k=1}^{K} (w_{ki}^r - \overline{w}_i^r)(w_{kj}^r - \overline{w}_j^r)$$

As we dispose of Euclidean dE_k^r and Mahalanobis dM^r distances, we determine which neurone and which SOCNet are respectively closest from the selected data $X(t)$. The best matching neurone and the best matching network are respectively designed as "winner neuron" and "winner SOCNet". At this time we mark the winner neuron: $N_{k_o}^{r_1}$ where indexes *ko* and *r1* design respectively the winner neurone and the SOCNet which contains it.

We mark the winner SOCNet with index r_2 as R^{r2}, which is the closest from the selected data with Mahalanobis distance.

Thanks to distances $dM^r(t)$ and $dE_k^r(t)$, we develop a competitive strategy among neurons and SOCNets when the networks marked by index r_1 and r_2 are the same. It means that the closer network S^{r_2} to the observation $X_n(t)$ contains also the closest neuron $N_{k_o}^{r_1}$ to the observation $X_n(t)$.

We note that otherwise, if the networks S^{r_1} and S^{r_2} are different, the procedure will not continue the current adaptation but will restart with a new randomly chosen observation $X_m(t)$. The step time iteration (t) is not incremented in this case.

The adaptation of the network S^{r_0} (with $r_o = r_1 = r_2$) will be done for the neurons placed in the neighborhood of the winner neuron $N_{k_o}^{r_0}$ with a traditional competitive learning scheme as LVQ, then the closest neuron and its neighbors inside the SOCNet will be adapted. Nevertheless, all the neighbors of the winner are not updated in the same way. This competitive rule among the neurons inside the SOCNet S^{r_0} makes it possible to distribute the neurons in an effective way in the local data space set by each cluster as each spare units will be ready to learn some future new input vectors.

The learning algorithm is based on the weights updating rule as defined in equation (3) :

$$W_k^r(t+1) = W_k^r(t) + \alpha(t)\left(X_n(t) - W_k^r(t) \right) \quad for \; N_k^r \in V_A\left(k, v(t) \right) \tag{3}$$

The adjustable parameters are the learning rate $\alpha(t)$ and the adaptation neighborhood size $V_A(t)$. To increase the learning phase procedure inside each network we also use a Rival Penalty Strategy. L. Xu has used this rule for the purpose of multidimensional data classification with Radial Basis Functions (RBF) neural networks [6]. In its first presentation, the mechanism of this adaptation was to move the weights of the winner in the direction given by the position of the observation X(t) and also to repulse in opposite direction, the position of the rival neurone which corresponds to the second closer neuron from the position of the observation X (t). For that purpose we define the winner SOCNet and the rival SOCNet as S^{r_0} for the winner and $S^{r_0'}$ to the rival. First we look at which SOCNet S^r as $(r \neq r_0)$ contains the "second" closer neuron N_k^r from the selected observation $X_n(t)$. The distance used here for the rival penalty strategy is the Euclidean distance. When the procedure has finished to determine the second closest neuron (marked with index k_0') to the rival network (marked with index r_0'), the neuron $N_{k_0'}^{r_0'}$ and its neighborhood inside a rival topologic neighboring are pushed away from the observation vector $X_n(t)$.

Equation (4) gives the rule of Rival Penalty Strategy. The rule is controlled by parameters as repulse rate $\beta(t)$ and rival neighboring $V_R(t)$.

$$W_k^{r_0'}(t+1) = W_k^{r_0'}(t) - \beta(t)\left(X_n(t) - W_k^{r_0'}(t) \right) \quad for \; N_k^{r_0'} \in V_R\left(k, r_0'(t) \right) \tag{4}$$

When we use a large adaptation neighboring $V_A(t)$ and rival neighboring $V_R(t)$ at beginning of adaptation process, we allow the weights of the neurones to globally take almost the same value, forcing the neurones of the SOCNet to take place at a local average position, which tends to mark the center of one of the clusters inside the data space observation. In the same way, with a large neighboring for rival penalty, we can share and distribute the SOCNets to some parts of the data space occupied by homogeneous and consistent clusters.

We gave some relative important initial values to the parameters α_o et β_o, considering that we need to strongly organize the SOCNets in the beginning of the adaptation and rival penalty procedures to give each one a global position and shape which correctly reflect the distribution of the data inside the observation data space.

III. SOCNet Algorithm: application to segmentation of color images

We present here some results obtained in the field of color images segmentation. We use Lena's image in 24 bit true color BMP format. The size of selected image is 256 rows by 256 lines. The data set sample concerns in each case pixels from images: each step time (t) a pixel is the selected

from the image in a random way. The attributes of the data are the color values of the selected pixel. As we work with BMP image file format it means that we select the (r, g, b) values as data attributes. Each attribute takes its value in the range [0 … 255]. The number of final colors inside the segmented image result is defined by the user and set to R in the general case. We show the results on reference image for different values of R: (4,5,6,7). The pixels are assigned to the different clusters according their distances compared to the networks representative of each cluster.

We seek the nearest neuron for each pixel of the image tested. The network to which this neuron belongs indicates the cluster to which the pixel observation considered is assigned.
One modifies then the initial color of the pixel by giving to it the color representative of the cluster, determined by the average position of the SOCNet in space color as defined by equation (2) to $\overline{W}^{r}$.

We present the results for image "Lena" (figure 2) given by the SOCNet algorithm, which have been tested with 4, 5, 6 and 7 SOCNets: we obtain respectively images with 4, 5, 6 and 7 final colors (figures 3 to 6). The parameters used for the experiments have been tuned as follow:

Figure 2: Image « Lena »: 256x256 pixels RGB

The different parameters for SOCNet algorithm have been set as follows:

- Number of neurones per SOCNet K = 120
- Maximum number of iterations T_{max} = 20.000
- Initial neighboring adaptation size V_{A_o} = 60
- Initial neighboring repulsive adapt. size V_{R_o} = 60
- Initial value of gain for adaptation α_o = 0.5
- Initial value of gain for repulsion β_o = 0.001

Experimentation has shown that the value for parameters is not crucial for the segmented image result.

Figure 3: 4 SOCNets ; 120 neurones / SOCNet

Figure 4: 5 SOCNets ; 120 neurones / SOCNet

Figure 5: 6 SOCNets ; 120 neurones / SOCNet

Figure 6 : 7 SOCNets ; 120 neurones / SOCNet

We do comparison of segmentation image results by SOCNets with another method as K-means. Results are given in table (1), we give for each case evaluation parameters as number of colors (R) inside segmented image which corresponds with the number of SOCNets, number of regions inside segmented image (R(I)) and global mean square error (MSE).

	Kmeans	***SOCNets***	*Kmeans*	***SOCNets***
Image	*Fig. 3*	***Fig. 3***	*Fig. 4*	***Fig. 4***
R	*4*	***4***	*5*	***5***
MSE	*435*	***228***	*387*	***186***
R(I)	*784*	***1002***	*1201*	***907***

	Kmeans	***SOCNets***	*Kmeans*	***SOCNets***
Image	*Fig. 5*	***Fig. 5***	*Fig. 6*	***Fig. 6***
R	*6*	***6***	*7*	***7***
MSE	*331*	***181***	*271*	***155***
R(I)	*1075*	***1303***	*1216*	***1788***

Table 1 : Evaluation for segmentation results of the "Lena" image of figure (2)

Conclusion

In this article we have presented an application of the SOCNets to segmentation task for color images. The SOCNet method which usually is used for clustering problems with multidimensional data show that they are particularly well adapted to the segmentation of color images.

We have seen how efficient it could be to compute few neural networks as the SOCNets in a parallel way to carry out an unsupervised classification task, here dedicated to a segmentation color images.

The joint use of different nature as Euclidean and Mahalanobis distances and calling upon antagonistic adaptive mechanisms, make it possible to make evolve simultaneously and in parallel to the Self-Organizing Circular Networks. In the end of the training period each network will reveal clusters of pixels with homogeneous characters in color data space representation.

References

[1] P. Biela Enberg, D. Hamad, J.-G. Postaire. *"Unsupervised Data Classification with Neural Elastic Networks and Genetic Algorithms"* MENDEL 98 4th Int. Mendel Conference, Brno, pp. 249-254, 24-26 June 1998.

[4] C. Bouton, G. Pagès, *"Convergence in distribution of the one-dimensional Kohonen algorithm when the stimuliare not uniform"* Advanced in Applied Probability, 26,1,March 1994.

[2] T. Kohonen. *"A simple paradigm for the self-organized formation of structured maps"*, Lecture notes in biomathematics, S. Amari and M.A. Arbib, Eds, Berlin: Spinger-Verlag, pp. 248-266, 1982.

[3] T. Kohonen, K. Makisara, O. Simula. *"Self-organizing maps: Optimization approach"*, in Artificial Neural Networks, Elseiver Science, New-York, pp. 981-990, 1991.

[5] T. Kohonen, "*Self-Organizing maps*" , 3rd Edition Spinger-Verlag Berlin Heidelberg New York, 2001.

[6] L. Xu, A. Krzyzak, E. Oja. "*Rival Penalized competitive learning for clustering analysis, RBF net, and curve detection*", IEEE Trans. On Patt. Analysis and Machine Intell., vol. PAMI-4, pp. 636-649, 1993.