A Neuro-Fuzzy System for Automatic Multi-Level Image Segmentation using KFCM and Exponential Entropy

G. Raghotham Reddy[1], E. Suresh[2] , S.Uma Maheshwar[3] and M. Sampath Reddy[4]

[1,2,3] *Lecturer, Kakatiya Institute of Technology and Science, Warangal, A. P., India*
[4] *Lecturer, Ramappa Engineering College, Mahabubabad, A. P., India*
E-mail: grr_ece@yahoo.com

Abstract

An auto adaptive neuro-fuzzy segmentation and edge detection architecture is presented. This system consists of a multilayer perceptron (MLP)-like network that performs image segmentation by adaptive thresholding of the input image using labels automatically pre-selected by kernel based fuzzy clustering technique. The proposed architecture is feed forward, but unlike the conventional MLP the learning is unsupervised. The output status of the network is described as a fuzzy set. Fuzzy entropy is used as a measure of the error of the segmentation system as well as a criterion for determining potential edge pixels. Exponential entropy was employed to overcome the drawbacks of using conventional logarithmic entropy. The proposed system is capable to perform automatic multilevel segmentation of images, based solely on information contained by the image itself. No a priory assumptions whatsoever are made about the image (type, features, contents, stochastic model, etc.). Such an "universal" algorithm is most useful for applications that are supposed to work with different (and possibly initially unknown) types of images. The proposed system can be readily employed, "as is," or as a basic building block by a more sophisticated and/or application-specific image segmentation algorithm. By monitoring the fuzzy entropy relaxation process, the system is able to detect edge pixels

Keywords: Image Segmentation, Adaptive Tresholding, Error backpropagation Neural Network System and Kernal Fuzzy C-means Clustering algorithm.

1.0 ADAPTIVE NEURO-FUZZY SYSTEM WITH KFCM

The Adaptive Neuro-Fuzzy system consists of a multilayer neural network that performs adaptive, multilevel thresholding of the image using labels automatically pre selected by a fuzzy clustering technique. The learning technique employed is self-supervised allowing, therefore, automatic adaptation of the NN. The output status of the network is described as a fuzzy partition. Fuzzy entropy is used as a measure of the error of the system as well as a criterion for determining potential edge pixels. Given an input image, the system is forced to evolve toward a minimum fuzzy entropy state in order to obtain image segmentation. Pixels most affected by the consecutive training iterations (due to the amount of their contribution to the fuzzy entropy of the system) are labeled as edge pixels.

1.1 Description of Adaptive Neuro-Fuzzy System

Block diagram of the system is shown in fig. 1. Labels are found by applying the KFCM algorithm to the image histogram. Then, the information about the labels is employed to build the network activation and error functions. The input to a neuron in the input layer is normalized between [0-1], proportionally to the gray value of the correspondent pixel. The image information is first propagated forward using (1) to get the output status of the network. The output value of each neuron lies in the interval [0-1]. Then, the output error is calculated and back propagated to update the weights [(4)]. Training continues either until a minimum error or until a maximum number of iterations reached. The output of the system at this stage constitutes the segmented image. Integrating (summing) the thresholded (binared) differences between the outputs at consecutive epochs yield the edge image.

Please use the following format when citing this chapter:

Reddy, G.R., Suresh, E., Maheshwar, S.U., Reddy, M.S., 2006, in IFIP International Federation for Information Processing, Volume 228, Intelligent Information Processing III, eds. Z. Shi, Shimohara K., Feng D., (Boston: Springer), pp. 367–372.

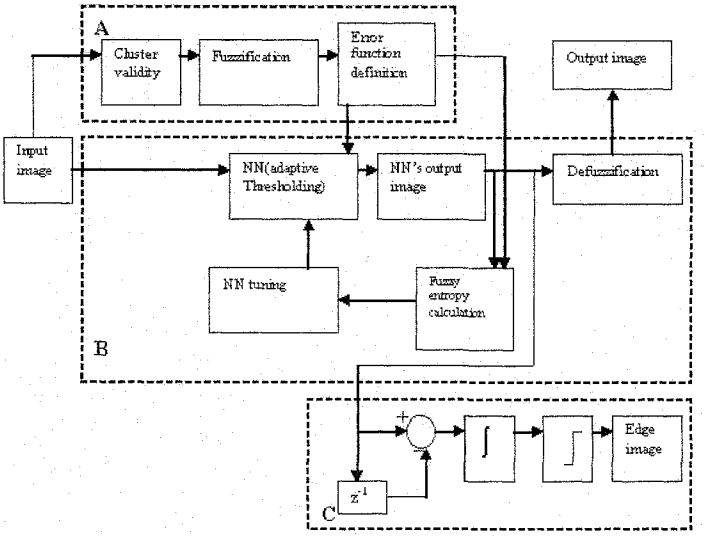

Fig. 1: Block Diagram of Neuro Fuzzy System

1.1.1 Error Function Definition Block

The purpose of this block is to provide the objective error function to be used by the adaptive thresholding block. First, the cluster validity block determines the number of objects in the input image, then the fuzzification block divides the input image into that number of fuzzy sets using KFCM as shown in Fig. 2(b), and then the error function definition block generates error function by determining the contribution of each gray level to the fuzzy entropy of the partition as shown in Fig. 2(c). The cluster validity block automatically determines the number of objects in the input image, for this it iterates the KFCM algorithm for a range of hypothesized number of clusters and chooses best option based on a cluster validity measure (e.g., the partition coefficient and the partition entropy).

1.1.2 Adaptive Thresholding Block

This contains the Neural Network (NN) block, the fuzzy entropy calculation block and NN tuning block. Its inputs are the input image and the error function determined by the block (A), and its output is the segmented image.

Neural Network: The neural network block performs adaptive thresholding of the input image. The network architecture is shown in Fig. 3. It consists of an input layer, an output layer and at least one hidden layer. Each layer consists of M x N neurons, every neuron corresponding to an image pixel. Each neuron in the one layer is only connected to the corresponding neuron in the previous layer and the neurons in its d-th order neighborhood.

A neighborhood system over a M x N lattice L is defined as $n^d = \{n_{ij}^d \subset L : (i,j) \in L\}$

where n_{ij}^d, called the d-th order neighborhood of (i, j), is such that

- $(i,j) \notin n_{ij}^d$;
- $(k,l) \in n_{ij}^d$

There are no connections between neurons in the same layer. The NNs' weights cannot be randomly initialized or they will alter the input image. In this work, all weights were initialized to 1, but it is also possible to initialize the weights using some kind of weighting window within the neighborhood of each pixel.

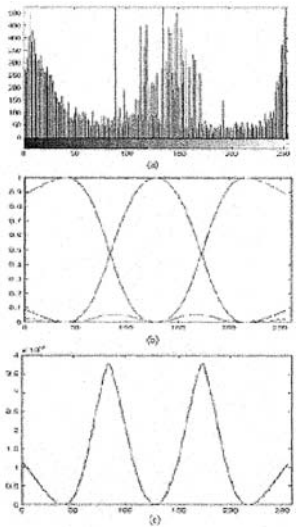

Fig. 2: (a) Histogram of the Panda image (b) Partion found by KFCM (c) Error function

Activation function: A *multi-sigmoid* activation function was used to allow more than two stable states of the neuron output. The *multi-sigmoid* function is defined as (Fig.4).

$$f(x)=\sum_k\left(\frac{y_k-y_{k-1}}{1+e^{-(x-\theta_k)/\theta_0}}+y_{k-1}\right)\times\left\{u(x-y_{k-1}*d^2)-u(x-y_k*d^2)\right\} \quad (1)$$

where

u	step function;
θ_k	thresholds;
y_k	target level of each sigmoid, will constitute the systems' labels;
θ_0	steepness parameter;
d	size of the neighborhood, as defined in the previous section

The thresholds and the target values are obtained from the error function, as the gray levels with the maximal and with the minimal levels of fuzziness respectively. Because the range of the neuron input levels depend on the number of neurons in the previous layer to which it is connected (the size of the neighborhood), the threshold values are adapted to reflect this dependency (by multiplying them by d^2, the number of input links).

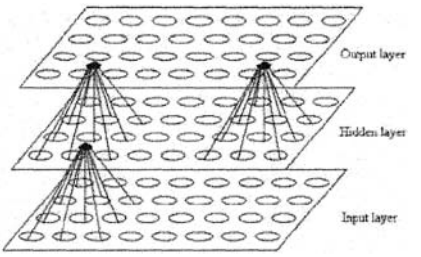

Fig. 3: Neural Network

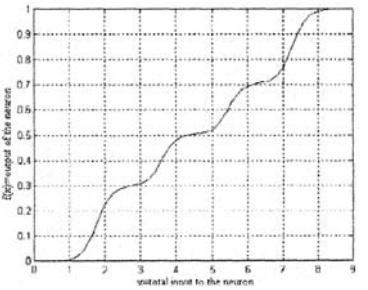

Fig. 4: Multi Sigmoid Activation function

Training: The back-propagation algorithm is employed for training. As we apply input image the neurons in the first layer receives the input, and will apply it to the Linear Combiner and the Activation Function and produce the output this output, will become the input for the neurons in the next layer. So the next layer will feed forward the data, to the next layer. And so on, until the last layer is reached We compare the desired and actual output compute the error as the difference between desired output and actual output. Once we decided what adjustment we need to do to the neurons in the output layer, we back propagate the changes to the previous layers of the network. Indeed, as soon as we have desired outputs for the output layer, we make adjustment to reduce the error (the difference between the output and the desired output). Adjustment will change weights of the input nodes of the neurons in the output layer. The weights are updated as follows:

$$\Delta w_{ji} = \begin{cases} \eta\left(-\frac{\partial E}{\partial o_j}\right)\frac{\partial o_j}{\partial I_j} o_i & output layer \\ \eta\left(\sum_k\left(-\frac{\partial E}{\partial o_k}\frac{\partial o_k}{\partial I_k} w_{kj}\right)\right)\frac{\partial o_j}{\partial I_j} o_i & other layer \end{cases} \quad (2)$$

where

I_i total input to the i th neuron;

w_{ji} weight of link from neuron i in one layer to neuron j in the next layer;

o_i output of the i th neuron in the previous layer;

E error in the network's output (relative to the desired target image);

η learning rate.

Note: *For simplicity 1-D indexes in the above equations are used, the extension to fit the 2-D NN is straightforward.*

For a *multi-sigmoid* as previously defined

$$\frac{\partial o_j}{\partial I_j} = o_j(y_n - y_{n-1} - o_j) \quad (3)$$

and the equations for Δw_{ji} become

$$\Delta w_{ji} = \begin{cases} \eta\left(-\frac{\partial E}{\partial o_j}\right) o_j(y_n - y_{n-1} - o_j) o_i \\ \eta\left(\sum_k\left(-\frac{\partial E}{\partial o_k}\frac{\partial o_k}{\partial I_k} w_{kj}\right)\right) o_j(y_n - y_{n-1} - o_j) o_i \end{cases} \quad (4)$$

for the output and the other layers respectively

Defuzzification: The output of the neural network is initially obtained in terms of the gray levels, which are then "fuzzyfied" in order to determine the error. In the idle case when the network converges with no error at all (E=0), the outputs have only values who's membership values are "1" or "0," defuzzification is not necessary. When the network does not converge completely (whether stopped intentionally or not), the fuzzification of the output image does not result in merely crisp membership values. The information about the membership values of the pixels might be useful for further processing, depending on the application at hand. If crisp labeling is required, a defuzzification stage must be added. For display purposes, the simplest defuzzification method is thresholding the fuzzy partition, so that each pixel is uniquely assigned to the class in which it has the highest membership value.

1.1.3 Edge Detection Block

This subsystem is based on the assumption that the edge pixels have the most ambiguous values in the image, i.e., they give the largest contribution to the fuzzy entropy of the output image at each iteration. Thus, these pixels are those that undergo the changes during the training/tuning of the system. Here, monitoring the changes that take place in the pixels' values between two consecutive iterations and integrating these changes over the whole training period obtain the edge image.

2.0 RESULTS

The Adaptive Neuro-Fuzzy system is implemented in MATLAB environment. The execution time of the Neural Network training epochs depends on the image size and the neighborhood size. Theoretically, the number of required epochs depends on the error and activation functions (which intern depend on the nature of the data), on the learning rate and on the required precision. Practically, the training may usually be stopped after about ten epochs. In terms of runtime memory requirements, these systems require four floating-point matrices of size $d^2 * M * N$ (neighborhood size multiplied by the size of the image) are needed for the two layers of weights and their corresponding updates, and three floating-point matrices of size $M * N$ (the image size) are needed to store the three layers of the network (input, hidden, and output).

2.1 Segmentaion Results

The output of the system is the segmented image enhancing the object over the background. As we can observe in the figures followed. The effect of using KFCM, Selecting thresholds by second derivative of the image histogram and considering exponential form of fuzzy entropy as error function can be seen very clearly. They give smoother image which is more robust to noise. Moreover by employing KFCM instead of FCM makes it applicable to wider range of images, i.e., for those having spherical and non spherical edges. Use of Kfcm made the Adaptive Neuro-Fuzzy system robust to some real life 'complication' like the addition of noise, and changing illumination conditions. Kfcm is also capable of handling uneven sized clusters.

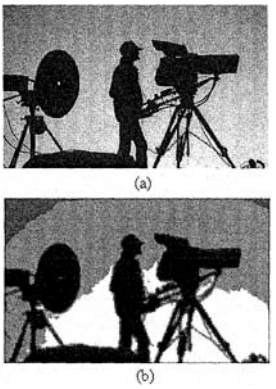

Fig. 5: (a) Original Image (b) Segmented image

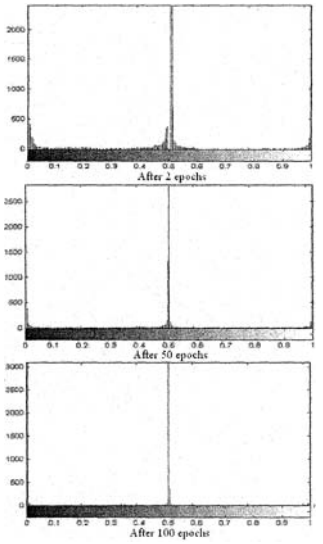

Fig. 6: Convergence Analysis

2.2 Convergence Analysis

The convergence of the adaptive threshold system of Adaptive Neuro-Fuzzy system is visualized in Fig.6. This figure shows the histogram of the output image after 2 and 50 and 100 training epochs (when applied to the Panda image), clearly indicating the convergence of the pixel values to the chosen labels.

2.3 Edge Detection Results

The edge detection subsystem of Neuro-fuzzy system was found to perform poorly compared to some of the better edge detected algorithms existing today, sometimes even worse than the classical, gradient type edge detector (Prewitt, Sobel, etc.). Fig.7 below shows the results of the edge detection subsystem of Neuro-Fuzzy system applied to the cameraman image, compared to the results of the Sebel operator.

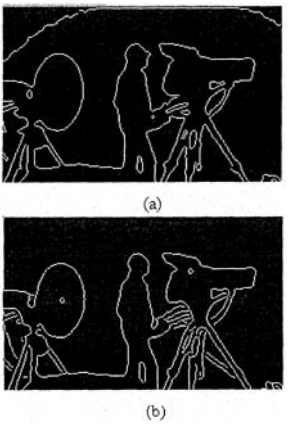

Fig. 7: (a) Edge image of Neuro-fuzzy System (b) Edge image using Sobel operators

3.0 CONCLUSION

The proposed adaptive neuro fuzzy system has been proven to be efficient than many other existing methods of segmentation. It does not require any priori assumptions of the input. Employing KFCM made the system applicable wider range of images. As you can observe in the results it worked for spherical edges in panda and non spherical edges in cameraman image. Use of exponential entropy yielded smoother images at less number of iterations as compared to logarithmic form .It is also robust to noise. Increase of neighbourhood results in more loss of details. This system can also be extended for edge detection but this isn't as efficient as general canny or sobel operators

REFERENCE

[1] Victor Boskovitz and Hugo Guterman (2002), "An Adaptive Neuro-Fuzzy System for automatic Image Segmentation and Edge Detection", *IEEE Trans. On Fuzzy Systems*, Volume_10(No. 2), pp. 247-252.

[2] Rafael C. Gonzalez and Richard E. Woods, (2002), "Digital Image Processing", Pearson Education, New Delhi