FUZZY
sets and systems

Fuzzy Sets and Systems 72 (1995) 331–348

Use of fuzziness measures in layered networks for object extraction: a generalization

Ashish Ghosh*

Machine Intelligence Unit, Indian Statistical Institute, 203 B.T. Road, Calcutta 700035, India

Received April 1994; revised August 1994

Abstract

A generalized self-organizing multilayer neural network incorporating fuzziness measures is designed for object extraction. Every neuron in each layer corresponds to an image pixel. A neuron in one layer is connected to the corresponding neuron in the previous layer and the neighbors of that neuron. Neurons in the output layer are also connected to the corresponding neurons in the input layer. The output of the neurons in the output layer is viewed as a *fuzzy set* and measures of fuzziness is used to model the error (instability of the network) of the system. Results of a simulation study using synthetic and real images are seen to be quite satisfactory.

Keywords: Image processing; Neural networks; Fuzzy sets; Object extraction; Neuro-fuzzy computing

1. Introduction

Fuzzy set theoretic [24, 7–9, 17, 20] models try to mimic human reasoning and the capability of handling uncertainty arising from ill-defined, incomplete, defective and imprecise input. Neural network models [21, 19, 22], on the other hand, attempts to emulate the architecture and information representation schemes of human brain by providing a mathematical model of combination of numerous neurons connected in a network. Integration of the merits of these two technologies therefore promises to provide, to a great extent, more intelligent systems (in terms of parallelism, fault tolerance, adaptivity and uncertainty management) to handle real life recognition problems.

These promises have motivated (during the last five years or so) a large number of researchers to exploit these modern concepts in the field of pattern recognition and machine vision under a new branch called *neuro-fuzzy* computing [3, 2, 15, 16]. The fusion or integration is mainly tried out in the following ways or in any combination of them: (i) incorporating fuzziness into the neural network frameworks, (ii) designing neural networks guided by fuzzy logic formalism, (iii) changing the basic characteristics of the neurons so that they perform the operations used in fuzzy set theory, (iv) making the individual neurons fuzzy, and (v)

* E-mail: ash@isical.ernet.in.

modeling the error or instability or energy function of a neural network based system using measures of fuzziness/uncertainty of a set.

The present work is an attempt to design a generalized multilayer neural network capable of incorporating various fuzziness measures for performing (unsupervised) the self-organizing task of object extraction. Though a preliminary attempt was made in [6] to design a network architecture for such a task, the present article provides a more general version capable of dealing with different types of fuzziness measures. The architecture used previously was capable of working with those types of fuzziness measures where the derivative of the fuzziness measure with respect to the status of any output node is dependent only on that node. The present architecture overcomes this restriction and can work with different types of measures.

Various fuzziness measures have been used to extract objects from noisy environments using this network architecture. A simulation study was done using a synthetic image corrupted by $N(0, \sigma^2)$ additive noise and a real image. The results obtained were found to be quite satisfactory. A comparative study among the different fuzziness measures, to find out their suitability for object extraction, is also made in this context.

2. Measures of fuzziness of a fuzzy set

A fuzziness measure [9] of a fuzzy set [24] expresses the average amount of ambiguity in making a decision whether an element belongs to the set or not.

Several authors [5, 23, 10, 18] have made attempts to define such measures. A few measures relevant to the present work are described here.

(i) *Index of fuzziness*: The index of fuzziness of a fuzzy set A having n supporting elements is defined as [8]

$$\gamma_p(A) = \frac{2}{n^{1/p}} l^p(A, \bar{A})$$

$$= \frac{2}{n^{1/p}} \left[\sum_{i=1}^{n} \{\min(\mu_A(x_i), 1 - \mu_A(x_i))\}^p \right]^{1/p}, \tag{1}$$

when $l^P(A, \bar{A})$ denotes the distance between fuzzy set A and its nearest ordinary set $\bar{A}$. An ordinary set $\bar{A}$ nearest to the fuzzy set A is defined as

$$\mu_{\bar{A}}(x) = \begin{cases} 0 & \text{if } \mu_A(x) \leqslant 0.5, \\ 1 & \text{if } \mu_A(x) > 0.5. \end{cases} \tag{2}$$

The value of p in the above equation depends on the type of distance measure used.

(ii) *Entropy* of a fuzzy set: As defined by De Luca and Termini [5] is given by

$$H(A) = \frac{1}{n \ln 2} \sum_{i=1}^{n} \{S_n(\mu_A(x_i))\} \tag{3}$$

with

$$S_n(\mu_A(x_i)) = -\mu_A(x_i) \ln\{\mu_A(x_i)\} - \{1 - \mu_A(x_i)\} \ln\{1 - \mu_A(x_i)\}, \tag{4}$$

and that of Pal and Pal [18] is given by

$$H(A) = \frac{1}{n(\sqrt{e} - 1)} \sum_{i=1}^{n} \{S_n(\mu_A(x_i)) - 1\} \tag{5}$$

with

$$S_n(\mu_A(x_i)) = \mu_A(x_i)\, e^{1-\mu_A(x_i)} + \{1 - \mu_A(x_i)\}\, e^{\mu_A(x_i)}. \tag{6}$$

Another definition of entropy which involves the distance of a fuzzy set from its furthest ordinary set is given by Bart Kosko [10]. It says

$$R_p(A) = \frac{l^p(A, \bar{A})}{l^p(A, \underline{A})} = \frac{[\sum_{i=1}^{n} \{\min(\mu_A(x_i), 1 - \mu_A(x_i))\}^p]^{1/p}}{[\sum_{i=1}^{n} \{\max(\mu_A(x_i), 1 - \mu_A(x_i))\}^p]^{1/p}}, \tag{7}$$

where $\underline{A}$ is an ordinary set furthest to the fuzzy set A, defined by

$$\mu_{\underline{A}}(x) = \begin{cases} 0 & \text{if } \mu_A(x) \geqslant 0.5, \\ 1 & \text{if } \mu_A(x) < 0.5. \end{cases} \tag{8}$$

(iii) *Fuzzy correlation*: A concept of correlation, giving a measure of relationship between two membership functions representing fuzzy sets, was introduced by Murthy et al. [13]. The concept was later on extended and applied to image segmentation by Pal and Ghosh [14]. Correlation between a fuzzy property and its nearest two tone property represents the degree of closeness of the fuzzy set to the nearest ordinary set. In other words, correlation also provides a measure of information about the distance of a fuzzy set (represented by μ_1) from its nearest ordinary set (represented by μ_2) and is expressed as

$$C(\mu_1, \mu_2) = 1 - \frac{4}{X_1 + X_2} \sum_{i=1}^{n} \{\mu_1(i) - \mu_2(i)\}^2, \tag{9}$$

with

$$X_1 = \sum_{i=1}^{n} \{2\mu_1(i) - 1\}^2, \tag{10}$$

$$X_2 = \sum_{i=1}^{n} \{2\mu_2(i) - 1\}^2; \tag{11}$$

$0 \leqslant C(\mu_1, \mu_2) \leqslant 1$.

In the following sections we will use these measures to compute the error or measure of instability of a multilayer self-organizing neural network.

3. Multilayer networks

In this section let us brief the structure and working principle of the existing multilayer neural networks.

3.1. The multilayer perceptron

The multilayer perceptron (MLP) is made up of sets of nodes arranged in layers. Nodes of two consecutive layers are connected by links or weights; but there is no connection among the nodes of the same layer. The layer where the inputs are presented is known as the *input layer*. On the other hand, the output producing layer is called the *output layer*. The layers in between the input and the output layers are known as *hidden layers*.

The output of nodes in one layer is transmitted to nodes in another layer via links that amplify or attenuate or inhibit such outputs through weighting factors. Except for the input layer nodes, the total input to each node is the sum of weighted outputs of the nodes in the previous layer. Each node is activated in accordance with the input to the node and the activation function of the node. The total input (I_j) to the jth unit of any layer is,

$$I_j = \sum_i w_{ji} o_i \tag{12}$$

with o_i as the output of the ith neuron in the previous layer and w_{ji} is the connection weight between the jth node of one layer and the ith node of the previous layer. The output of a node j is obtained as

$$o_j = f(I_j), \tag{13}$$

where f is the activation function [21]. Mostly the activation function is sigmoidal, with the form (Fig. 1)

$$f(x) = \frac{1}{1 + e^{-(x-\theta)/\theta_0}}. \tag{14}$$

The function is symmetrical around θ and θ_0 controls the steepness of the function. θ is known as the threshold/bias value.

Initially very small random values are assigned to the links/weights. In the learning phase (*training*) of such a network we present the pattern $X = \{x_i\}$, where x_i is the ith component of the vector X, as input and ask the net to adjust its set of weights in the connecting links and also the thresholds in the nodes such that the desired output $\{t_i\}$ is obtained at the output nodes. After this, we present another pair of X and $\{t_i\}$, and ask the net to learn that association also. In fact, we desire the net to find a set of weights and biases that will be able to discriminate among all the input/output pairs presented to it. This process can pose a very strenuous learning task and is not always readily accomplished. Here the desired output $\{t_i\}$ basically acts as a teacher which tries to minimize the error.

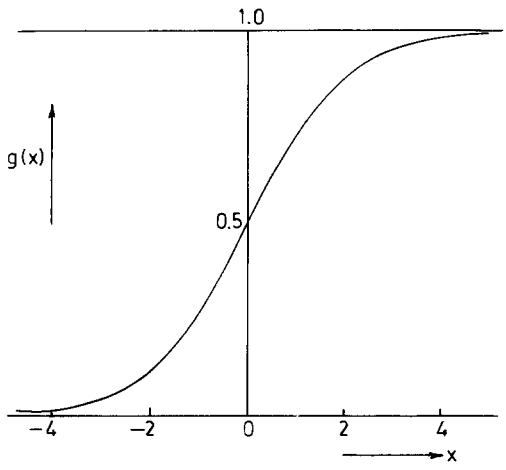

Fig. 1. Sigmoidal activation function.

In general, the output $\{o_i\}$ will not be the same as the target or desired value $\{t_i\}$. For a pattern p, the error is,

$$E = \frac{1}{2}\sum_i (t_i - o_i)^2, \tag{15}$$

where the factor of one half is inserted for mathematical convenience. The procedure for learning the correct set of weights is to vary the weights in a manner such that the error E is reduced as rapidly as possible. This can be achieved by moving in the direction of negative gradient of E. In other words, the incremental change for a particular pattern p is

$$\Delta w_{ji} \propto -\frac{\partial E}{\partial w_{ji}} = -\eta\frac{\partial E}{\partial w_{ji}} = -\eta\frac{\partial E}{\partial I_j}\frac{\partial I_j}{\partial w_{ji}}$$

$$= \eta\delta_j o_i \quad \text{(from (12))} \tag{16}$$

with

$$\delta_j = -\frac{\partial E}{\partial I_j} = -\frac{\partial E}{\partial o_j}\frac{\partial o_j}{\partial I_j}$$

$$= -\frac{\partial E}{\partial o_j} f'(I_j) \quad \text{(from (13))}. \tag{17}$$

As E can be directly calculated in the output layer, for the links connected to the output layer the change in weight is given by

$$\Delta w_{ji} = \eta\left(-\frac{\partial E}{\partial o_j}\right) f'(I_j) o_i. \tag{18}$$

If the links do not affect the output nodes directly (for links between the input and the hidden layer, and also between two consecutive hidden layers), the factor $\partial E/\partial o_j$ cannot be computed directly. In this case we use [21] the chain rule to write

$$\frac{\partial E}{\partial o_j} = \sum_k \frac{\partial E}{\partial I_k}\frac{\partial I_k}{\partial o_j} = \sum_k \frac{\partial E}{\partial I_k}\frac{\partial}{\partial o_j}\sum_i w_{ki} o_i = \sum_k \frac{\partial E}{\partial I_k} w_{kj}$$

$$= \sum_k (-\delta_k) w_{kj}. \tag{19}$$

Hence

$$\Delta w_{ji} = \begin{cases} \eta(-\partial E/\partial o_j) f'(I_j) o_i, \\ \eta(\sum_k \delta_k w_{kj}) f'(I_j) o_i \end{cases} \tag{20}$$

for the output layer and other layers, respectively.

In particular, if

$$o_j = \frac{1}{1 + \exp(-(\sum_i w_{ji} o_i - \theta_j))} \tag{21}$$

then

$$f'(I_j) = \frac{\partial o_j}{\partial I_j} = o_j(1 - o_j) \tag{22}$$

and thus we get

$$\Delta w_{ji} = \begin{cases} \eta(-\partial E/\partial o_j)\, o_j(1 - o_j)\, o_i, \\ \eta(\sum_k \delta_k w_{kj})\, o_j(1 - o_j)\, o_i \end{cases} \tag{23}$$

for the output layer and other layers, respectively.

It may be mentioned here that a large value of η corresponds to rapid learning but might result in oscillations. A *momentum* term of $\alpha \Delta w_{ji}(t)$ $(0 < \alpha < 1)$ can be added to increase the learning rate (without leading to oscillation) and thus expression (16) can be modified as

$$\Delta w_{ji}(t + 1) = \eta \delta_j o_i + \alpha \Delta w_{ji}(t), \tag{24}$$

where the quantity $(t + 1)$ is used to indicate the $(t + 1)$th time instant, and α is a constant which determines the effect of previous weight changes on the current direction of movement in weight space. The second term is used to specify that the change in w_{ji} at $(t + 1)$th instant should be somewhat similar to the change undertaken at instant t.

3.2. A self-organizing multilayer network

Several authors [4, 1, 12, 11] have used the multilayer perceptron for image segmentation/texture discrimination. These perceptron based techniques require a set of images of known classes for learning (supervised) which may not always be available in real life situations.

If the images to be processed come from a set of classes (i.e., images of the same class have common characteristics) then one can train the network with a set of images (with known output classes), and use the trained network on future images. However, if the images do not share some common features and a set of images with known targets (may be synthetic images) is not available, the multilayer perceptron as such, may not be useful for image processing. In the following sections we describe a multilayer neural network which overcomes these constraints.

3.2.1. Description and operation of the network

Architecture. In Fig. 2 we depict the 3-layered version of a *self-organized multilayer neural network* (SOMLNN). In each layer there are $M \times N$ neurons (for an $M \times N$ image). Each neuron corresponds to a single pixel. Besides the *input and output* layers, there can be a number of *hidden* layers (more than zero). Neurons in the same layer do not have any connection among themselves. Each neuron in a layer is connected to the corresponding neuron in the previous layer and to its neighbors (over dth ordered neighborhood N^d); thus each neuron in layer i $(i > 1)$ will have $|N^d| + 1$ (where $|N^d|$ is the number of pixels in N^d) links to the $(i - 1)$th layer. For N^1, a neuron has 5-links whereas for N^2, 9-links will be associated with every neuron. However, for boundary nodes (pixels) number of links may be less than $|N^d| + 1$. Every neuron in the output layer is also connected to the corresponding neuron in the input layer. It may be noted that this architecture differs from the standard MLP in the following major points:

- the distribution of links, and
- the feed back connection from the output layer to the input layer.

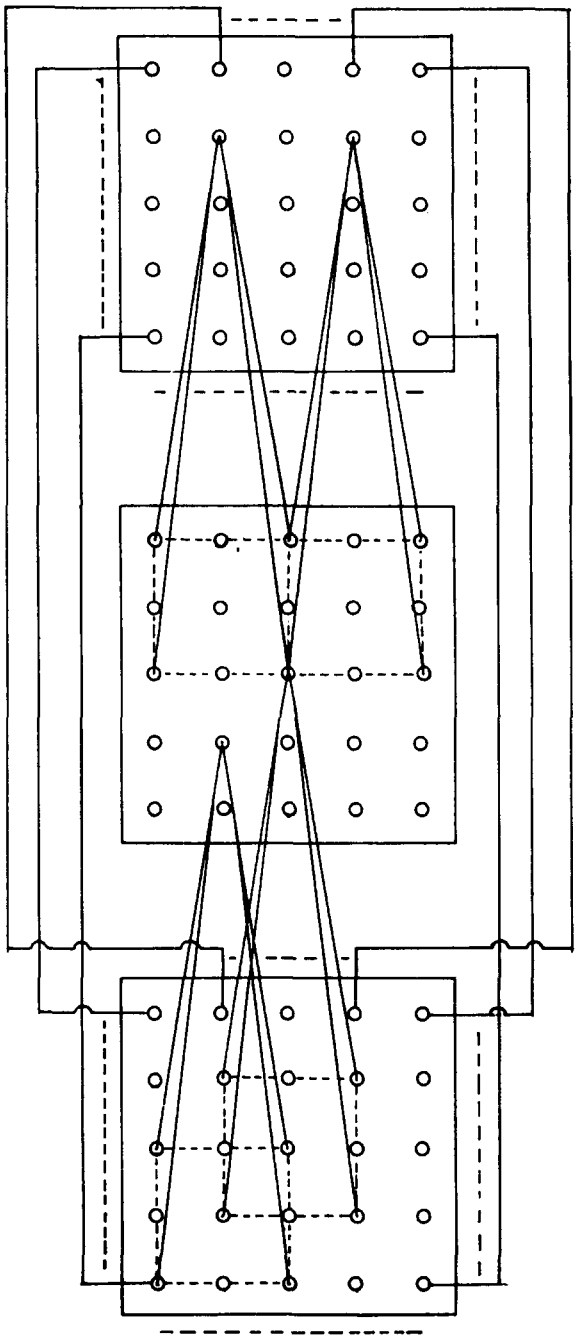

Fig. 2. Schematic representation of Self-organizing multilayer neural network.

Initialization. The input to a neuron in the input layer is given as a real number in $[0, 1]$ which is proportional to the gray value of the corresponding pixel. Since we are trying to eliminate noise and extract spatially compact regions, all initial weights are set to one (1). No external bias is imposed on the weights.

Random initialization (of weights) may act as a pseudo noise and the compactness of the extracted regions may be lost. As all the weights are set to unity, the total input (initially) to any node lies in $[0, n_f]$ (where n_f $(= |N^d| + 1)$ is the number of links a neuron has); hence the most unbiased choice for the threshold value θ (for the input/output transfer function, Eq. (13)) would be $\frac{1}{2} n_f$ (the middle most value of the total input range).

Operation. The input (I_j) to any neuron in the jth layer (except the input layer) is computed using (12). The transfer function f (Eq. (13)) is then applied to get the output status of the neurons in this layer. These outputs are then fed as input to the next layer. Starting from the input layer, this way the input image is passed on to the output layer and the corresponding output states are calculated. The output value of each neuron lies in $[0, 1]$.

Here our intention is to extract spatially homogeneous regions through the process of self-organization using only one noise corrupted realization of a scene (it does not require a set of images with known output classes for learning). The way the network gets organized, under ideal condition when the image is not noisy, the output status of most of the neurons in the output layer will be either 0 or 1. But due to the effect of noise the output status of the neurons in the output layer usually will be in $[0, 1]$ and thus the status value will represent the degree of brightness (darkness) of the corresponding pixel in the image. Therefore, the output status in the output layer may be viewed to represent a *fuzzy set "bright (dark) pixels"*. The measure of *fuzziness* of this set, on the global level, may be considered as the *error or instability of the whole system* as this will reflect the deviation from the desired state of the network. Thus, when we do not have any a priori target output value, we can take the fuzziness value as a measure of system error and back propagate it to adjust the weights (mathematical expressions for this are given later) so that the system error reduces with passage of time and in the limiting case it becomes zero. The error measure E can also be taken as a suitable function of a fuzziness measure, i.e.,

$$E = g(I), \tag{25}$$

where I is a *measure of fuzziness* (Eqs. (1), (3), (5), (7) and (9)) of a fuzzy set.

After the weights have been adjusted properly, the output of the neurons in the output layer is fed back to the corresponding neurons in the input layer. The second pass is then continued with this as input. The iteration (updating of weights) is continued as in the previous case until the network stabilizes, i.e., the error value (measure of fuzziness) becomes negligible. When the network stabilizes the output status of the neurons in the output layer becomes either 0 or 1. Neurons with output value 0 constitute one group and those having output value 1 constitute the other group. It may be mentioned here that the scene can have any number of compact regions (objects).

4. Weight correction and network architecture for various fuzziness measures

The mathematical derivation for weight updating rules with different fuzziness measures (Eqs. (1), (3), (5), (7) and (9)) are as follows. The derivations are given only for correcting the weights of the links connected to the output layer. For other layers similar expressions, as in the second part of (23) are applicable.

4.1. Weight correction for index of fuzziness

Let us consider

$$E = \{\nu_p(A)\}^p = \frac{2^p}{n} \sum_j \{\min(o_j, 1 - o_j)\}^p, \tag{26}$$

n being the number of neurons in the output layer. Here,

$$\frac{\partial E}{\partial o_j} = \begin{cases} (2^p/n)\, p(o_j)^{p-1} & \text{if } 0 \leqslant o_j \leqslant 0.5, \\ -(2^p/n)\, p(1-o_j)^{p-1} & \text{if } 0.5 \leqslant o_j \leqslant 1.0. \end{cases} \tag{27}$$

Thus from (20) we get

$$\Delta w_{ji} = \begin{cases} \eta(-2^p/n)\, p\, o_j^{p-1} f'(I_j)\, o_i & \text{if } 0 \leqslant o_j \leqslant 0.5, \\ \eta(2^p/n)\, p(1-o_j)^{p-1} f'(I_j)\, o_i & \text{if } 0.5 \leqslant o_j \leqslant 1.0. \end{cases} \tag{28}$$

For our simulation purpose we will be using $p = 2$ (i.e., the quadratic index of fuzziness [8]).

4.2. *Weight correction for entropy*

We consider, $E = H$, where H is the entropy of a fuzzy set (Eqs. (3) and (5)). For entropy measures the weight correction is made as

$$\Delta w_{ji} \propto -\frac{\partial E/\partial o_j}{|\partial E/\partial o_j|^q}, \quad q > 1, \tag{29}$$

where $|\partial E/\partial o_j|$ represents the magnitude of the gradient. The above formula is used to correct the weights instead of that of (16) to ensure that the weight correction is maximum when the network is most unstable (i.e., when all the output values are 0.5) and vice versa. In other words, for a neuron the weight correction for its links should be maximum when its output status is close to 0.5 and is minimum when its ouput status is close to 0 or 1. For our implementation we will choose $q = 2$.

Now from (3)

$$\frac{\partial H}{\partial o_j} = -\frac{1}{n \ln 2} \ln \frac{o_j}{1-o_j}. \tag{30}$$

Thus from (20) and (29)

$$\Delta w_{ji} = \eta(n \ln 2)^{q-1} \frac{\ln(o_j/(1-o_j))}{|\ln(o_j/(1-o_j))|^q} f'(I_j)\, o_i. \tag{31}$$

Similarly, with exponential entropy (Eq. (5)) if we consider

$$E = H \tag{32}$$

then,

$$\frac{\partial H}{\partial o_j} = \frac{1}{n(\sqrt{e}-1)} \{(1-o_j)e^{1-o_j} - o_j e^{o_j}\}. \tag{33}$$

Thus,

$$\Delta w_{ji} = -\eta\{n(\sqrt{e}-1)\}^{q-1} \frac{(1-o_j)e^{1-o_j} - o_j e^{o_j}}{|(1-o_j)e^{1-o_j} - o_j e^{o_j}|^q} f'(I_j)\, o_i. \tag{34}$$

As the expressions for weight correction (Δw_{ji}) with the previously described error measures involve only the output status of the jth neuron of one layer and the ith neuron of the previous layer (i.e., involves o_j and o_i) the self-organizing multilayer neural network architecture described in Section 3.2.1 as such can implement the object extraction algorithm with these error measures.

4.3. Weight correction for Kosko's entropy measure

Let us consider

$$E = (R_p)^p$$

$$= \frac{\sum_j \{\min(o_j, 1 - o_j)\}^p}{\sum_j \{\max(o_j, 1 - o_j)\}^p} = \frac{a}{b} \quad \text{(say)}. \tag{35}$$

Then,

$$\frac{\partial E}{\partial o_j} = \begin{cases} ((a + b)/b^2)\, p o_j^{p-1} & \text{if } 0 \leqslant o_j \leqslant 0.5, \\ -((a + b)/b^2)\, p(1 - o_j)^{p-1} & \text{if } 0.5 \leqslant o_j \leqslant 1.0. \end{cases} \tag{36}$$

Therefore,

$$\Delta w_{ji} = \begin{cases} -\eta((a + b)/b^2)\, p o_j^{p-1} f'(I_j)\, o_i & \text{if } 0 \leqslant o_j \leqslant 0.5, \\ \eta((a + b)/b^2)\, p(1 - o_j)^{p-1} f'(I_j)\, o_i & \text{if } 0.5 \leqslant o_j \leqslant 1.0. \end{cases} \tag{37}$$

For the present simulation p is taken as 2. Note that with passage of time (i.e., when o_j approaches 0 or 1) the value of a decreases and that of b increases. Thus $(a + b)/b^2$ decreases. Also note that the value of $(a + b)/b^2$ is dependent on the present status of the whole network. Hence for the updating of a single weight the overall status of the network is required. Thus the architecture described in Section 3.2.1 is not sufficient for implementing an object extraction algorithm involving Kosko's entropy as an error measure.

In order to implement this algorithm we need to add a subnetwork (with the architecture in Fig. 2) which calculates $(a + b)/b^2$. The block takes the form as shown in Fig. 3. It is worth noting that the product of the output of the block in Fig. 3 and η acts as the learning rate (for the present architecture) which decreases as the network approaches stability.

4.4. Weight correction for correlation measure

Let us choose,

$$E = 1 - C, \tag{38}$$

where C is the correlation between μ_1 and μ_2 as defined in (9). Thus

$$\frac{\partial E}{\partial o_j} = \frac{8(X_1 + X_2)\{\mu_1(j) - \mu_2(j)\} - 16X_3\{2\mu_1(j) - 1\}}{(X_1 + X_2)^2}, \tag{39}$$

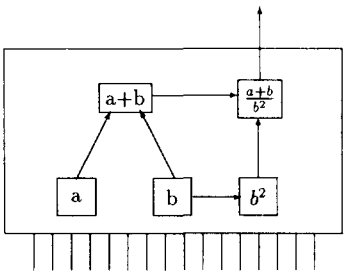

Fig. 3. Block diagram for computing information about the overall status of the system with Kosko's entropy as error measure.

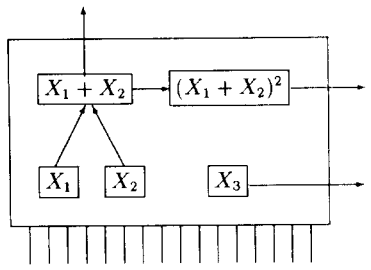

Fig. 4. Block diagram for computing information about the overall status of the system with fuzzy correlation as error measure.

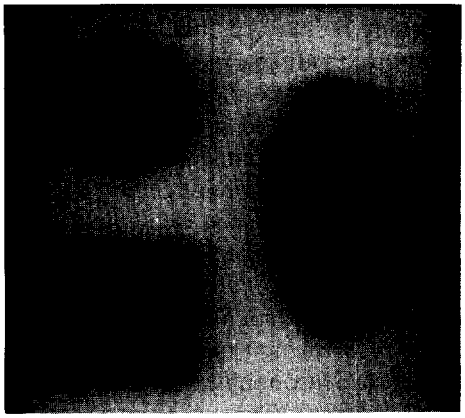

Fig. 5. Original synthetic image.

where

$$X_3 = \sum_j \{\mu_1(j) - \mu_2(j)\}^2. \tag{40}$$

Hence,

$$\Delta w_{ji} = -\eta \frac{\partial E}{\partial o_j} f'(I_j) o_i. \tag{41}$$

In this case the network architecture will need a subnetwork as shown in Fig. 4 in addition to the block in Fig. 2. Please note that for updating the weights we will need the output values of the block in Fig. 4.

5. Computer simulation and results

In order to check the effectiveness of the proposed technique, computer simulation has been done on a synthetic bitonic image (Fig. 5) which contains object regions of three different shapes (in general, it can contain any number of objects with any shape) corrupted by noise. (We did the simulation with a number of

different types of images amongst which one is chosen here for presentation.) The corrupted versions were obtained by adding noise from $N(0, \sigma^2)$ distribution with different values of σ (10, 20, 32). Three noisy inputs are shown in Figs. 6(a), 7(a) and 8(a). The images are of dimension 128×128 and have 32 levels. Simulation study has also been done on a real image of a *noisy tank* (Fig. 9(a)). The noisy tank image is of size 64×64 with 64 gray levels. For the simulation study η value has been taken as 0.2 and the neurons are assumed to be connected to its second ordered neighbors (N^2 neighborhood is used). A neuron thus gets input from nine neurons in the previous layer. The threshold value θ in this case is $\frac{9}{2} = 4.5$. The input gray values are mapped in $[0, 1]$ by a linear transformation and is given as input to the network. The network is then allowed to be settled. When the network is stabilized, the neurons having status value 0 constitute one region type, say, object (background), and the remaining neurons with output status 1 constitute the other region type, say, background (object).

The objects extracted by the proposed technique with different expressions of error for different noisy versions of the synthetic image are included in Figs. 6–8. Fig. 9 depicts the objects extracted from the *noisy tank* image with different error models.

Examining the results it can very easily be inferred that, as the noise level increases, the quality of the output, as expected, deteriorates; but approximate shapes and outlines are maintained. Note that the outputs are independent of the number of object regions (and their shapes) present in the scene. Comparing results of different error models, it is noticed that outputs with index of fuzziness measure are better than those obtained by entropy measures. Among the two different entropy measures, the exponential function is found to be more immune to noise. Results corresponding to Kosko's entropy measure and correlation measure are comparable to those of index of fuzziness measure and better than those of entropy measures. Between Kosko's entropy measure and correlation measure the latter one is seen to provide better results compared to the former.

This is possibly due to different learning rates. For a fixed value of η, the learning rate is low for the index of fuzziness, Kosko's entropy measure and correlation measure, whereas it is higher for entropy measures. When the learning rate is high, a particular neuron influences its neighbors to a great extent; thus the noisy elements affect the results strongly. The system thus fails to remove all the noisy elements. Of the two entropy measures the exponential one is more noise immune due to its lower learning rate at the initial stage of learning.

A critical examination of the results reveal that the index of fuzziness, Kosko's entropy measure and the correlation measure are consistently better than entropy measures for maintaining the compactness of the extracted objects (as determined by their boundaries). But shapes of objects are better preserved by entropy measures. This observation can be explained as follows: since for the fuzziness, Kosko's entropy and correlation measures, the rate of learning is slow, it smoothes out noises and creates compact regions, while for entropy measures because of rapid learning all noisy pixels may not be removed, particularly when σ value is very high. On the other hand, entropy measures enable the network to preserve object boundaries as learning rate is very high near the most ambiguous region ($o_j \simeq 0.5$).

6. Discussions and conclusion

The limitations of the feed forward multilayer perceptron with back propagation of error for image processing (segmentation/object extraction) have been addressed.

A generalized self-organizing multilayer neural network suitable for image processing applications is proposed in this regard. Though a preliminary attempt was made to design a neural network architecture for such a task, the present work provides a more general architecture capable of dealing with different types of fuzziness measures. The neural network architecture used previously was capable of working with those types of fuzziness measures whose derivatives with respect to the status of any output node are dependent

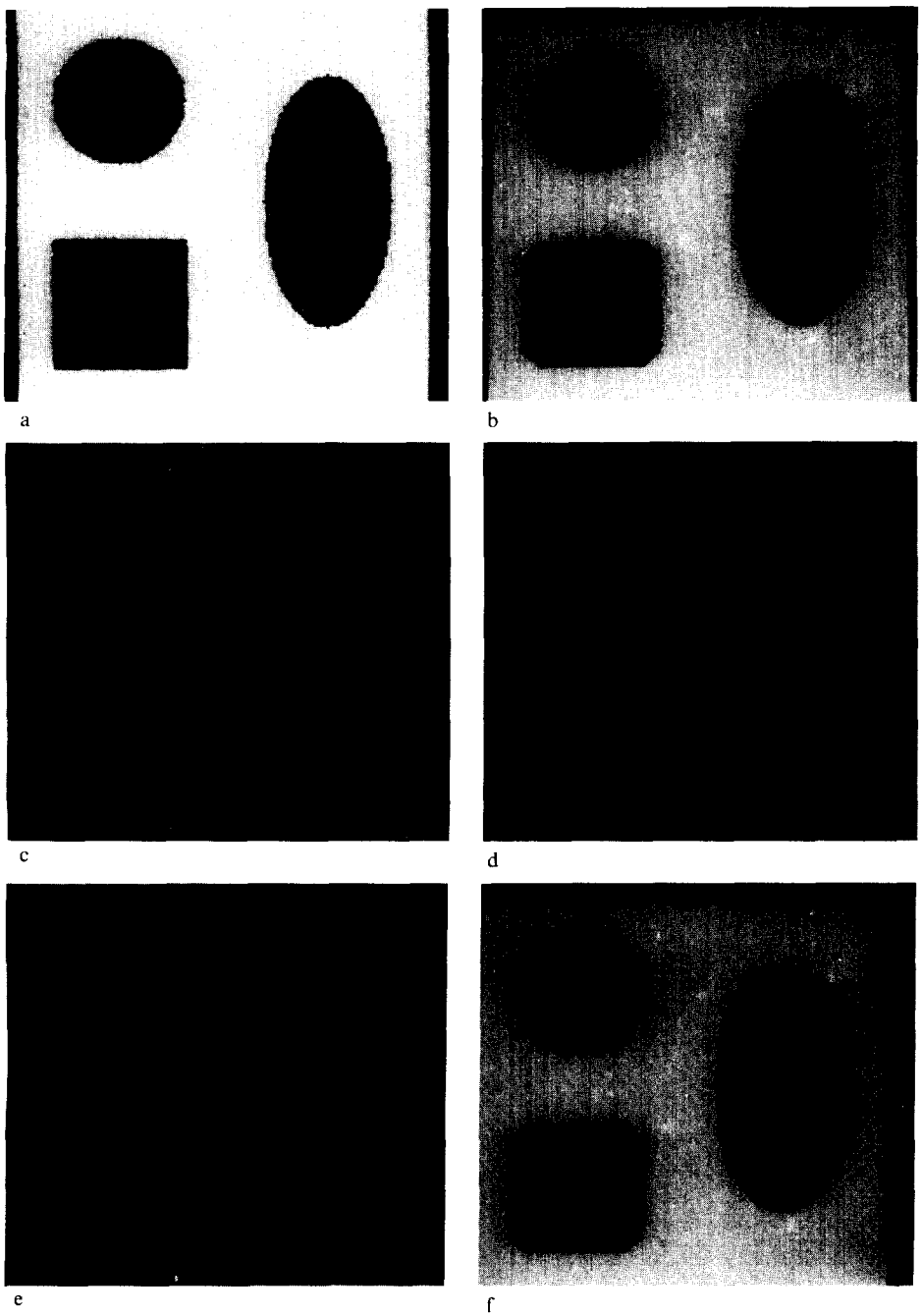

Fig. 6. Results for a noisy version ($\sigma = 10$) of the synthetic image: (a) Input; (b) Extracted object with index of fuzziness; (c) Extracted object with logarithmic entropy; (d) Extracted object with exponential entropy; (e) Extracted object with Kosko's entropy; (f) Extracted object with fuzzy correlation.

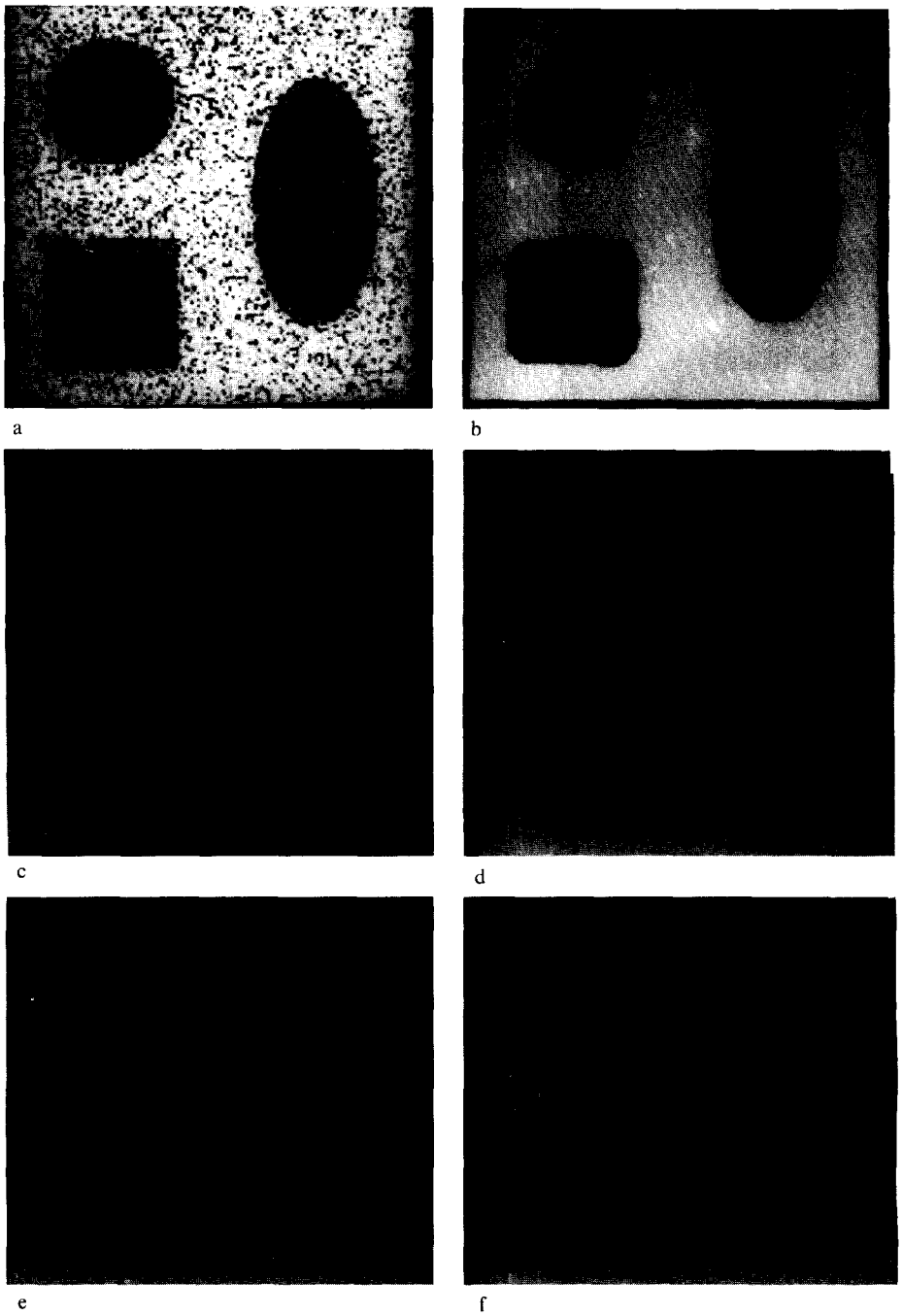

Fig. 7. Results for a noisy version ($\sigma = 20$) of the synthetic image: (a) Input; (b) Extracted object with index of fuzziness; (c) Extracted object with logarithmic entropy; (d) Extracted object with exponential entropy; (e) Extracted object with Kosko's entropy; (f) Extracted object with fuzzy correlation.

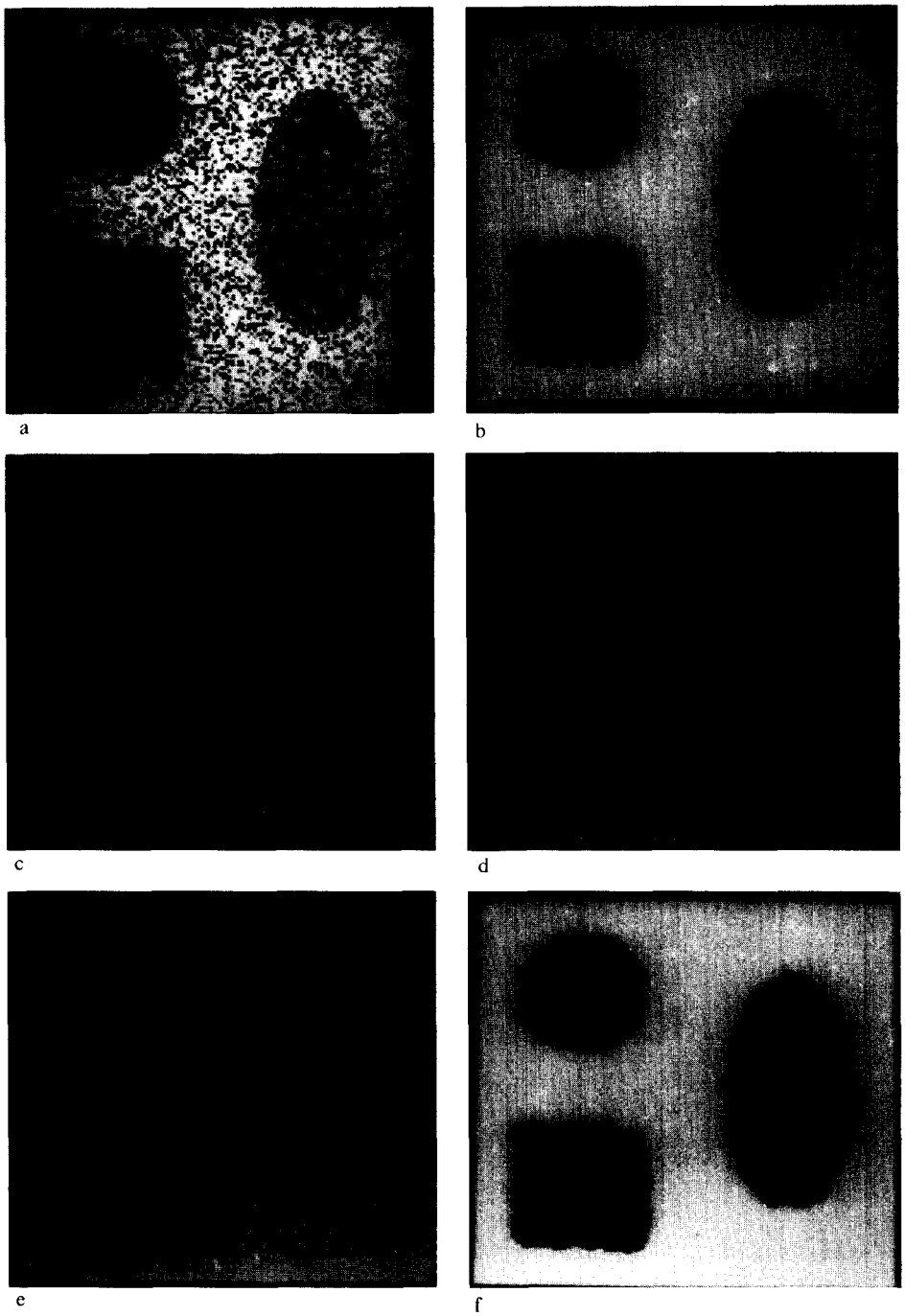

Fig. 8. Results for a noisy version ($\sigma = 32$) of the synthetic image: (a) Input; (b) Extracted object with index of fuzziness; (c) Extracted object with logarithmic entropy; (d) Extracted object with exponential entropy; (e) Extracted object with Kosko's entropy; (f) Extracted object with fuzzy correlation.

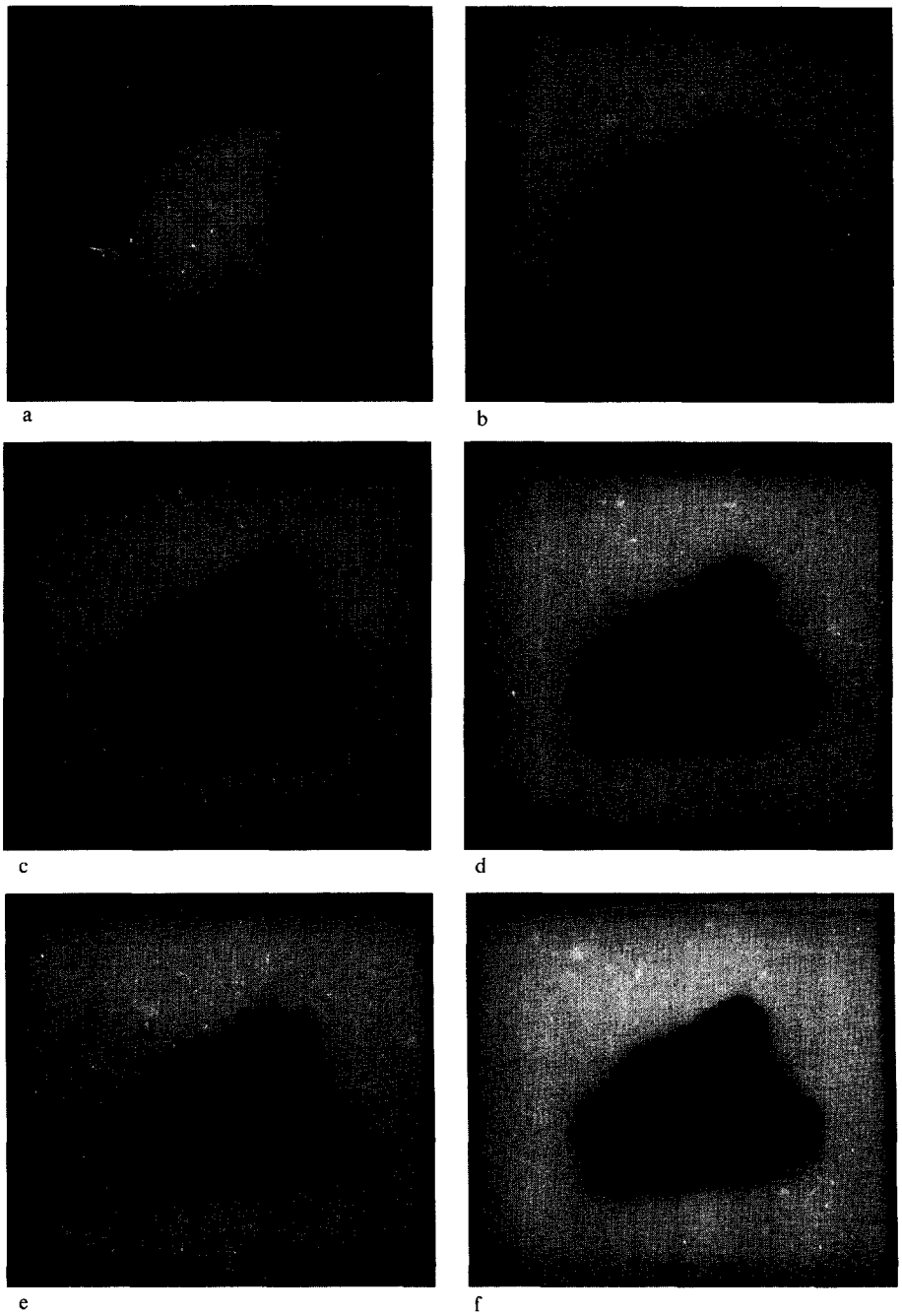

Fig. 9. Results for the noisy tank image: (a) Input; (b) Extracted object with index of fuzziness; (c) Extracted object with logarithmic entropy; (d) Extracted object with exponential entropy; (e) Extracted object with Kosko's entropy; (f) Extracted object with fuzzy correlation.

only on that node. The present architecture overcomes this restriction and can work with different types of fuzziness measures.

Each neuron in the network corresponds to an image pixel. A neuron in one layer is connected to the corresponding neuron in the previous layer and the neighbors of that neuron. Neurons in the output layer are also connected to the corresponding neurons in the input layer. The output of the neurons in the output layer has been viewed as a *fuzzy set* and measures of fuzziness have been used to model the error (instability of the network) of the system.

An application of the proposed architecture has been shown in object extraction problem from noisy environments. The algorithm has been implemented on a set of noisy images and the approximate shapes and boundaries of the extracted objects are found to be satisfactory even for very high noise level establishing the noise immunity of the proposed technique. Results also show that the rate of learning affects the output, specially when the noise level is very high. The outputs are better for lower learning rates and deteriorates with increase of rate of learning. Thus when the noise level is low, methods with higher learning rates are preferred; but when the noise level is high, methods with lower learning rates will be suitable.

Acknowledgements

The author gratefully acknowledges Prof. Sankar K. Pal for his constructive criticism and interest for this work. A part of this work was done when the author held a *Research Associateship* of the Council of Scientific and Industrial Research, India.

References

[1] N. Babaguchi, K. Yamada, K. Kise and Y. Tezuku, Connectionist model binarization, *Internat. J. Pattern Recognition Artif. Intell.* **5**(4) (1991) 629–644.

[2] J.C. Bezdek, On the relationship between neural networks, pattern recognition, and intelligence, *Internat. J. Approx. Reason.* **6**(2) (1992) 85–107.

[3] J.C. Bezdek and S.K. Pal (Eds.), *Fuzzy Models for Pattern Recognition: Methods that Search for Structures in Data* (IEEE Press, New York, 1992).

[4] W.E. Blanz and S.L. Gish, A real time image segmentation system using a connectionist classifier architecture, *Internat. J. Pattern Recognition Artif. Intell.* **5**(4) (1991) 603–617.

[5] A. Deluca and S. Termini, A definition of a non probabilistic entropy in the setting of fuzzy set theory, *Inform. and Control* **20** (1972) 301–312.

[6] A. Ghosh, N.R. Pal and S.K. Pal. Self-organization for object extraction using multilayer neural network and fuzziness measures, *IEEE Trans. Fuzzy Systems* **1**(1) (1993) 54–68.

[7] A. Kandel, *Fuzzy Mathematical Techniques with Applications* (Addison-Wesley, Reading, MA, 1986).

[8] A. Kaufmann, *Fuzzy Subsets – Fundamental Theoretical Elements* (Academic Press, New York, 1980).

[9] G.J. Klir and T. Folger, *Fuzzy Sets, Uncertainty and Information* (Prentice-Hall, Englewood Cliffs, NJ, 1988).

[10] B. Kosko, Fuzzy entropy and conditioning, *Inform. Sci.* **40** (1986) 165–174.

[11] J. Lee, R.C. Weger, S.K. Sengupta and R.M. Welch, A neural network approach to cloud classification, *IEEE Trans. Geoscience Remote Sensing* **28**(5) (1990) 846–855.

[12] B.S. Manjunath, T. Simchony and R. Chellappa, Stochastic and deterministic networks for texture segmentation, *IEEE Trans. Acoustic. Speech and Signal Process.* **38**(6) (1990) 1039–1049.

[13] C.A. Murthy, S.K. Pal and D.D. Majumder, Correlation between two fuzzy membership functions, *Fuzzy Sets and Systems* **7** (1985) 23–38.

[14] S.K. Pal and A. Ghosh, Image segmentation using fuzzy correlation, *Inform. Sci.* **62**(3) (1992) 223–250.

[15] S.K. Pal and A. Ghosh, Neuro-fuzzy image processing: relevance and feasibility, in S. Mitra, W. Kraske and M.M. Gupta, Eds., *Neural and Fuzzy Systems: The Emerging Science of Intelligence and Computing* (SPIE Press, New York, 1994) 160–184.

[16] S.K. Pal and A. Ghosh, Neuro-fuzzy computing for pattern recognition, *IEEE Trans. Neural Networks*, communicated.

[17] S.K. Pal and D.D. Majumder, *Fuzzy Mathematical Approach to Pattern Recognition* (Wiley, New York, 1986).

[18] N.R. Pal and S.K. Pal, Object background segmentation using a new definition of entropy, *IEE Proc.* Part E (1989) 284–295.
[19] Y.H. Pao, *Adaptive Pattern Recognition and Neural Networks* (Addison-Wesley, New York, 1989).
[20] W. Pedrycz, Fuzzy sets in pattern recognition: methodology and methods, *Pattern Recognition* **23**(1/2) (1990) 121–146.
[21] D.E. Rumelhart, J. McClelland et al., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vol. 1 (MIT Press, Cambridge, MA, 1986).
[22] P.D. Wassermann, *Neural Computing: Theory and Practice* (Van Nostrand Reinhold, New York, 1990).
[23] W.X. Xie and S.D. Bedrosian, An information measure for fuzzy sets, *IEEE Trans. Systems Man Cybernet.* **14** (1984) 151–156.
[24] L.A. Zadeh, Fuzzy sets, *Inform. and Control* **8** (1965) 338–353.