ELSEVIER

Neurocomputing 10 (1996) 341–358

NEUROCOMPUTING

A self-organizing network for mixed category perception

Jayanta Basak, C.A. Murthy, Sankar K. Pal *

Machine Intelligence Unit, Indian Statistical Institute, Calcutta 700 035, India

Received 3 January 1994, accepted 4 August 1994

Abstract

A neural network model capable of self-organizing in presence of multiple or mixed categories is presented. A certainty factor is derived about the decision on how well the features (due to single or mixed categories) have been interpreted by the network. One part of the model, the, *monitor*, controls the performance of the other part, the, *categorizer* in the self-organization process. The network automatically adjusts the number of nodes in the hidden and output layers, depending on the nature of overlap between the patterns from different categories. Mathematical derivations of the bounds on the number of nodes have been presented. The capability of the model is demonstrated experimentally both on one-dimensional binary strings and visual patterns.

Keywords: Self-organization; Monitor; Categorizer; Certainty factor; Mixed category perception

1. Introduction

Various neural network models have been built so far for producing the desired output for a given input pattern [10,13,6]. But most of the models used for pattern recognition problems determine a single category for a given input. That is, an input feature vector is assumed to be generated from a single category. However, in real life situation, a feature vector of observable categories may be generated from a combination (or presence) of more than one category. In other words, the input vector may result from the superimposition of more than one individual class feature vector. For example, in the field of industrial inspection, several objects may appear simultaneously in the scene and can occlude each other. In that case, the feature vector does not correspond to any single object. Similarly, in medical

* Corresponding author.

0925-2312/96/$15.00 © 1996 Elsevier Science B.V. All rights reserved
SSDI 0925-2312(95)00055-7

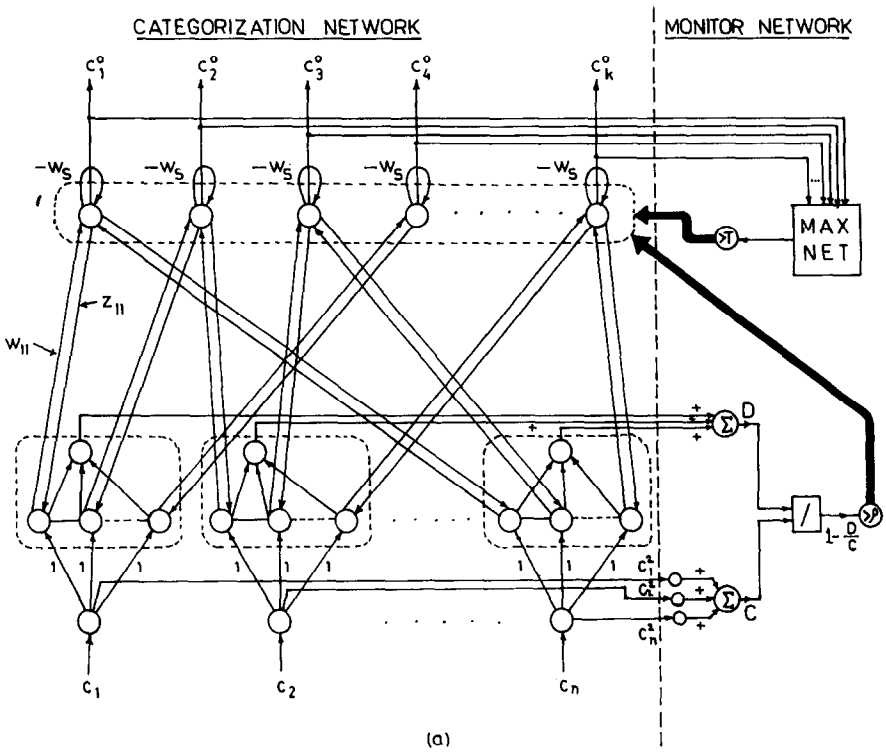
diagnosis problem, the symptoms of a patient may not correspond to any single disease. In literature, there exist very few investigations which can really tackle such a problem [12].

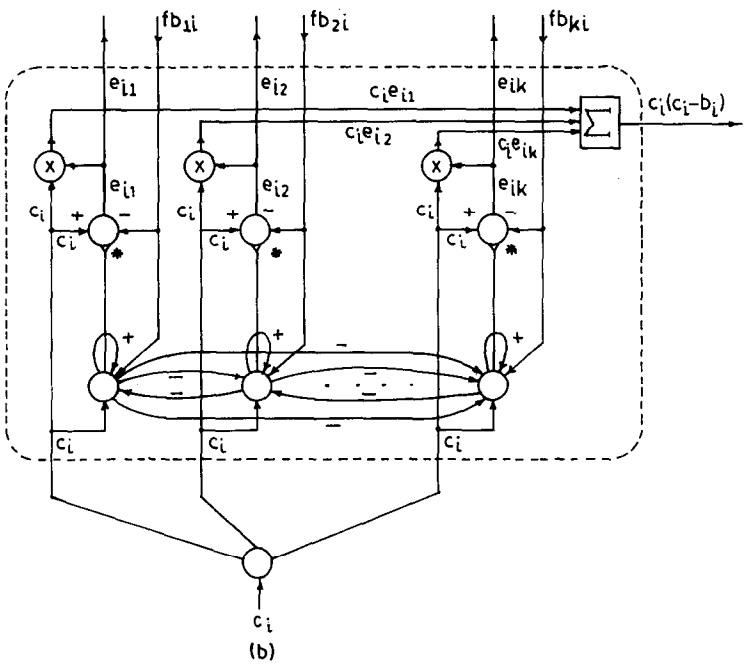
This problem becomes even harder if the connectionist model needs to be operated under unsupervised mode. The task is difficult because the model should be able to decide if the input pattern has been generated by the presence of more than one candidate feature vector, and at the same time be able to automatically associate the features with their corresponding categories without the help of any external teacher. It may be noted here that the biological systems are able to self-organize in such cases.

In literature, there exist several attempts on self-organization using connectionist models [1,2,4,7,8,9,11]. In most of them, the concept of self-organization is analogous to the idea of clustering in the literature of pattern recognition [5], and it works only when the input pattern represents a single category. In the present problem, it is not sensible to compute any kind of distance between the input pattern and the templates corresponding to the categories (as performed in most of the self-organization models) because due to superimposition, the resulting feature vector may be widely different from its constituent feature vectors.

In the present article, we attempt to develop a connectionist model for performing the task of self-organization in the presence of mixed categories by interpreting a feature vector generated due to the presence of more than one category. Instead of using the conventional concepts of self-organization, the concept of similarity based induction hypothesis [14] has been used. It may be mentioned that, operationally the model has a similarity to ART [4]. But, unlike ART, the present model is able to categorize in the presence of more than one category. The categories are initially hypothesized depending on the feature vector presented to the network. After the formation of initial hypothesis, the presence or absence of each category is iteratively verified depending on the support it gets from its constituent features. Note that it does not use any order search mechanism as employed in adaptive resonance theory. Rather, with each feature a measure of ambiguity is associated which indicates how well that feature has been interpreted. The certainty factor about a decision (whether it is correct or not) is measured based on the ambiguity value present in the feature vector. The confidence value indicates whether the feature vector should be considered as a valid feature vector (i.e. single or mixed instances of the learned categories) or should be treated as a new category. The network model incrementally adjusts the number of nodes with the incoming train of patterns. The total number of nodes in the network for a given set of patterns has been theoretically estimated. The effectiveness of the model has been demonstrated on both binary and visual patterns.

Fig. 1 (next page). (a) Structure of the connectionist model for self-organization. Bold lines represent the control paths from the monitor network to the categorization network. (b) Structure and connections of the hidden nodes in the network.


CATEGORIZATION NETWORK
MONITOR NETWORK
MAX NET
D
C
(a)


(b)

2. Neural network model

The proposed model (Fig. 1) consists of two subnetworks. One of them categorizes the input patterns (*categorizer*) and another part monitors the performance of the categorizer (*monitor*). In the present section the structure of the categorizer is described. The monitor network will be described in Section 4. The categorizer consists of three layers: input, hidden and output layer. The number of nodes in the input and output layers are equal to the number of features and number of possible object classes respectively. Each input node accepts an activation value equal to the confidence level about the presence of the corresponding feature. Similarly, an output node activation represents the confidence level about the presence of the corresponding object. (If an entity is absent then the confidence level is zero; if it is present then it is unity; and some intermediate confidence level represents more or less present.) The hidden layer associates the input node activations to the output node activations. To each input node a group of hidden nodes is connected which represents the group of the objects to which the input feature belongs.

Each hidden node is connected to exactly one input node through unidirectional links of unit weights, and is connected to a single output node through bottom-up and top-down links. Each output node has a self-negative feedback. Each hidden node has functionally two parts. One of them stores the activation value received from the output layer through top-down links. The other part takes the key role in competition with the other hidden nodes. (The hidden nodes connected to the same input node compete between themselves.) The network works as follows.

Initially, when a pattern or a set of features is presented to the network, the input nodes get activated according to the confidence levels of the corresponding features. The input nodes send their activation values to the hidden nodes. The hidden nodes do not compete in the initialization process; rather the activation values received by the hidden nodes from the input layer are propagated to the output layer through the bottom-up links. In this process, each output node receives an activation value equal to the sum of all activation values appearing through the bottom-up links. (This is equivalent to initial hypothesis formation.) The output of each output node is related to its input activation by an S-function [3]. After the initialization process, the output nodes send their activations back to the hidden layer through the top-down links. Once the hidden nodes receive the activation values from output layer, the competition process starts (this is performed by the second part of each hidden node). The hidden nodes connected to the same input node, compete between themselves. On the other hand, hidden nodes corresponding to different input nodes do not compete with each other. In the competition process for each input node only one hidden node remains active (the winner-take-all node receiving the maximum amount of feedback). The active hidden node represents the most possible object class to which the feature belongs. Once the competition is over, each WTA hidden node sends the differential support (which is the difference between corresponding input activation and feedback support) to the corresponding output node through bottom-up link. In

this process (referred as *settling process*), each output node gets some differential support from the hidden layer and some inhibition due to negative self-feedback. The settling process continues till the network reaches a stable state. The objects which have their candidate features in the set of features presented to the network, get sufficient differential support from the hidden layer. As a result, the activation values of these nodes reach some stable nonzero values when the support and the self-negation become equal. On the other hand, the objects which do not have their features in the feature train presented to the network, receive only self-negation; consequently, the activation values get down to zero. The settling process approximately corresponds to the iterative verification process, where the objects compete for the features. Next we present the dynamic behavior of the network.

The states of the output nodes are updated according to the differential equation given as

$$\frac{du_l}{dt} = \sum_{i=1}^{n} w_{il} e_{il} - w_s v_l. \tag{1}$$

where u_l and v_l are the total input to and output of the lth output node. w_{il} is the weight of the bottom-up link from the (i, l)th hidden node (connecting ith input node and lth output node) to the lth output node. w_s is the weight of the self-feedback in the output layer. e_{il} (*differential support*) measures the difference of the input activation from ith input node and the feedback from lth output node, provided the (i, l)th hidden node is enabled (winner-take-all node). Mathematically,

$$e_{il} = \begin{cases} c_i - z_{li} v_l & \text{if } z_{li} v_l \geq z_{mi} v_m \quad \forall m \neq l \\ 0 & \text{otherwise} \end{cases}$$

where c_i is the activation at the ith input node, z_{li} is the weight of the top-down link from the lth output node to the (i, l)th hidden node. The output v_l is related to the instantaneous input u_l by a semilinear nondecreasing gain function $g(.)$ (chosen as an S-function [3]).

The dynamic system described by (1) can be shown to converge. Let an energy function $\mathscr{E}(t)$ be defined as

$$\mathscr{E}(t) = \tfrac{1}{2} \sum_{i=1}^{n} \lambda_i' \Big(\max_{l=1,\ldots,m} z_{li} v_l - c_i \Big)^2 + \tfrac{1}{2} w_s \sum_{l=1}^{m} v_l^2 \tag{2}$$

where n is the number of input nodes and m is the number of output nodes. λ_i' is a multiplication factor such that

$$\lambda_i' = w_{is}/z_{si} \quad \text{if } z_{si} v_s = \max_{l=1,\ldots,m} z_{li} v_l$$

The rate of change of energy can be written as

$$\frac{d\mathscr{E}}{dt} = \sum_{l=1}^{m} \frac{\partial \mathscr{E}}{\partial v_l} \frac{dv_l}{dt} \tag{3}$$

But,

$$\frac{\partial \mathscr{E}}{\partial v_l} = -\sum_{i=1}^{n} z_{il}\lambda_{il}e_{il} - w_s v_l$$

i.e.

$$\frac{\partial \mathscr{E}}{\partial v_l} = -\frac{du_l}{dt}.$$

Therefore Eq. (3) can be written as

$$\frac{d\mathscr{E}}{dt} = -\sum_{l=1}^{k} g^{-1'}(v_l)\left(\frac{dv_l}{dt}\right)^2 \tag{4}$$

From (4) it is evident that $(\mathrm{d}\mathscr{E}/\mathrm{d}t) \leq 0$ for all $t \geq 0$. As $t \to \infty$, $(\mathrm{d}\mathscr{E}/\mathrm{d}t) \to 0$, and thereby the system converges to an energy minima (local).

The energy function (2) reveals the fact that the system always tries to minimize the error of mismatch between the input confidence values and the interpreted confidence values of the output layer. The weights of the top-down links are set in such a way that z_{li} is zero if ith feature do not belong to object, and close to unity if it belongs to. As a result, the first part shows the similarity of the problem to the 'set covering' problem. The second part of the energy function ensures the fact that the feature train presented to the network should be interpreted by the minimum possible number of objects. This enables the network to reduce the number of redundant objects or the chance of false alarming.

3. Learning strategy

The recognition would be correct if and only if the weights of the bottom-up and top-down links are set properly. Since the network does not have any apriori knowledge about the nature of associations between the features and the objects, the knowledge has to be acquired adaptively. Presently the strategy for learning or the adaptive acquisition of the knowledge about the relative frequency of the features and objects is presented.

If the network detects some new feature-object pair (by the monitor network in categorization process), a new hidden node is allocated for the corresponding input and output nodes. The hidden node is then connected to the input node with a link of unit weight. The bottom-up and top-down links are created from the hidden node to the corresponding output node. The weight of the bottom-up link is initialized to zero, and weight of the top-down link is initialized to unity. The hidden node is also connected to all other resident hidden nodes connected to the same input node. This enables the hidden node to compete with other nodes in the same group connected to the input node. Then the weights of the bottom-up and top-down links are updated so long as the pattern is present at the input.

The learning rules or the rules for iterative adjustment of the weights are set in such a way that the weight of each link asymptotically reaches a predefined measure. The measure for each weight is defined in such a way that it achieves the ability to capture the relative frequency of appearances of the corresponding feature object pairs. From the expression of the energy function (2), it is intuitively seen that the weight of a top-down link (z_{li}) should be proportional to the probability of appearance of the corresponding feature (i) with respect to the corresponding object (l). On the other hand, the value of λ'_i should be proportional to the probability of appearance of the object (corresponding to the winner) with respect to feature (i). Therefore, the asymptotic values can be given as

$$z_{li} = p(f_i \mid o_l) \tag{5}$$

and

$$\lambda_{il} = p(o_l \mid f_i). \tag{6}$$

The weight of bottom-up links should take a form

$$w_{il} \propto \lambda_{il} z_{li} \tag{7}$$

In the present work, since the transfer function of the output nodes is chosen as an S-function, the output values always saturate if the total input activations exceed unity. The weights of bottom-up links are iterated in such a way that the total activation reaching an output node is always less than unity. Therefore, an additional constraint is imposed on the weights of the bottom-up links which is

$$\sum_i w_{il} \leq 1. \tag{8}$$

Moreover, if two objects (say A and B) are such that the feature set of one object (say A) is a subset of another one (say B) and the probabilities of appearances of both the objects are the same then both A and B would be fully active if the larger feature set (corresponding to B) is presented to the network. In that case, it would not be possible to decide that a single object has been presented to the network. This problem can be taken into account by using Weber's law (as presented in adaptive resonance theory [4]). Considering (7), (8) and Weber's law, the asymptotic measure for the weights of bottom-up links becomes

$$w_{il} = \frac{p(o_l \mid f_i) p(f_i \mid o_l)}{\gamma + \sum_{i=1}^{n} p(o_l \mid f_i) p(f_i \mid o_l)}. \tag{9}$$

The constant γ is used to get the effect of Weber's law.

The weights of the links in the network are changed in such a way that they become equal to the measures after sufficient number of learning trials. In other words, the learning rules should be such that the weights of the bottom-up and top-down links asymptotically reach the measures (Eqs. 5 and 9). The conditional probability values are approximated by the ratio of the number of appearance of

the features and the objects. The detailed derivation of the learning rules is presented in [3]. The learning rules are

$$\frac{dw_{il}}{dt} = \left(\alpha_i \delta_l z_{li} + \alpha_l^o \left(\frac{w_{il}}{z_{li}}\right)\right) c_i y_l - (\alpha_i c_i + \alpha_l^o y_l) w_{il} \tag{10}$$

$$\frac{dz_{li}}{dt} = \alpha_l^o y_l (c_i - z_{li}) \tag{11}$$

where α_i and α_l^0 are the agility factors of the ith input node and the lth output node. The agility factor determines the capability of learning of the links connected to that node. The higher the agility factor, the higher will be the rate of learning and vice versa. Initially, the agility factor of all nodes are set to unity and they are decreased with the learning trials. The agility factor of a hidden node is the same as that of the input node connected to it. The agility factors are changed according to the following rules.

$$\frac{d\alpha_i}{dt} = -\alpha_i^2 c_i. \tag{12}$$

$$\frac{d\alpha_l^o}{dt} = -\alpha_l^{o2} y_l \tag{13}$$

The value of y_l is the desired output value at the lth output node. The desired output is determined by the monitor network. If the monitor network finds a pattern already known to the network, it sets the desired output of the corresponding node(s) to be unity and all other nodes to zero. If it finds a new pattern then it sets the desired output for the newly created node to unity and all other nodes to zero. The value of δ_l is given as

$$\delta_l = \frac{\epsilon_l}{\gamma g'(u_l)} \tag{14}$$

where $\epsilon_l = y_l - o_l$ measures the difference of the actual output at the lth output node from its desired value (as determined by monitor network). γ is the constant used in the asymptotic measure (Eq. 9) which also controls the rate of learning (as expressed in the learning rules).

4. Categorization (unsupervised classification)

Whenever a pattern (corresponding to either single or mixed category) is presented to the network, it produces some output depending on the weights of the bottom-up and top-down links. The output values indicate the confidence levels of the corresponding objects. The recognition mechanism should be such that the error between the feature confidence values and the interpretation of object confidence values ($z_{li} v_l$ in (2)) should be minimum. In other words, the network should be able to interpret all features presented to the input with the

current output confidence values. This is performed with a measure of certainty in the monitor network. The certainty value is measured depending on the ambiguity in the feature set presented to the network. If the activation of an input node does not match with the feedback support received from the output layer then there will be an ambiguity at that node (i.e. the feature is not properly interpreted by the network). Therefore, if a new pattern (representative of a single or mixed caltegory) is presented at the input then the feature set would not be properly interpreted by the output categories. As a result, there would be a high ambiguity in the feature set. The certainty is measured depending on the ambiguity in the feature set. If the certainty is less than some threshold (or ambiguity is greater than some threshold) then the input feature vector is considered to be representative of a new category.

Note that the categorization process is not similar to the clustering problem in pattern recognition. In the clustering problem whenever a new pattern appears, a distance is measured from the seed points of different clusters, and depending on the distances the pattern is considered to be coming from the present clusters or some new cluster. But if a pattern is a representative of some mixture of more than one category then the clustering process would not be able to determine that. On the other hand, the present categorization process is able to self-organize even in the presence of mixed categories. Next we present the measure of certainty factor.

Suppose, the network is presented with a set of n features with input confidence values given by $[c_1, c_2, \cdots, c_n]$. Let, after the network has stabilized, the top-down feedback corresponding to these features be $[b_1, b_2, \cdots, b_n]$. Then the total ambiguity D corresponding to the entire feature set can be defined as

$$D = \sum_{i=1}^{n} c_i(c_i - b_i) \tag{15}$$

The mismatch between the input confidence c_i and the top-down feedback b_i in each feature is modulated by the confidence value of the feature itself, thereby setting the relative importance of the mismatch. If $(c_i - b_i)$ increases, the value of D also increases, and vice-versa. Here, $D \geq 0$, and posseses a maximum value of $\sum_{i=1}^{n} c_i^2 = C$ (say).

The normalized ambiguity measure is given by

$$\mu = \frac{D}{C} \tag{16}$$

The certainty factor is given by

$$CF = 1 - \mu. \tag{17}$$

From (15), it is clear that for some given set of c_i values, if feedback support decreases then the ambiguity will increase and consequently, CF will decrease. But if c_i for each feature i (for a known category) is very small then the ambiguity (or CF) will reflect the fact that there is very little confusion in the network. This is due to the fact that the output confidence value will also be small (since input confidence values are small) and as a result, the feedback will also be small. As a

result, the network will accept the input as a representative of the category which is not desirable. Therefore, the monitor network also measures the maximum activation value present in the output layer. The output activation of any node in the network can be formulated as follows.

Without loss of generality, we can consider all the features to be active, i.e. sending differential activation (e) to a particular output node l. (If some features are shared by other objects and do not send any activation to that output node then they can be omitted from the feature set.) Under stable condition, the positive and negative signals at the output node will cancel each other. Mathematically,

$$w_s o_l = \sum_{i=0}^{n} (c_i - z_{li} o_l) w_{il}. \tag{18}$$

Considering linear gain function of the nodes, the output activation can be written as

$$o_l = \frac{\sum_{i=0}^{n} w_{il} c_i}{w_s + \sum_{i=0}^{n} w_{il} z_{li}} \tag{19}$$

When the network has learnt a particular category, the value of $\sum_{i=0}^{n} w_i z_i$ (we can drop l to represent the validity of the expression for all nodes) is nearly unity (the learning rules are designed in such a way). If w_s is small enough and all c_i values are nearly unity then the output x (maximum output is denoted by x) will reach unity.

The certainty factor is compared with a threshold (*vigilance threshold* (ρ)) and the maximum output activation with another threshold (*output threshold* (T)). If the pattern satisfies the conditions $CF > \rho$ and $x > T$ then it is treated to be a known category or mixture of more than one known category. If either of these conditions fails then it is treated as a new category. The connectionist model works as follows:

Step 1. Present a new pattern. The pattern may be representative of a single category or may be caused by the presence of more than one category.

Step 2. Measure the *Certainty Factor* (CF). Measure the maximum output (x) in the output layer. Note that there does not exist any output node initially. In that case, CF and x are considered as zero. The monitor network measures these two factors.

Step 3. If $CF > \rho$ and $x < T$ then goto Step 7;
if $CF > \rho$ and $x > T$ then make all the output nodes whose activations are greater than T fully active, and goto Step 6.

Step 4. Allocate a new output node in the output layer. If the total number of output nodes is greater than the capacity of the network then exit.

Step 5. Allocate hidden nodes for having input-output associations corresponding to the features which are present at the input. Connect the hidden nodes to the corresponding input nodes and the newly created output node. Initialize the weights of the newly created links, i.e. weights of the bottom-up links are set either to zero or to a small value. The weights of the top-down links are set to unity. Make the newly created output node fully active.

Step 6. Learn the weights of the links, i.e. iterate the weights till they converge.

Step 7. Present another new pattern. Goto Step 2.

The categorization process is controlled by the monitor circuit of the network (Fig. 1). In Section 2, it was mentioned that each hidden node has two operational parts. With each hidden node, another node is added to compute the ambiguity value at that particular node. This node computes the ambiguity value only when the network gets stabilized. The ambiguity values and the input activations are propagated to the monitor circuit where the normalized ambiguity value is computed and the certainty factor is determined. The monitor circuit also has a connection to the output layer. The part connected to the output layer consist of a comparator circuit which determines the maximum activation present in the output layer. The certainty factor is compared with vigilance threshold (ρ) and if it is found to be less then a new output node is allocated at the output layer. The hidden nodes corresponding to new feature-object pairs are also allocated, and the necessary bottom-up and top-down links are created. If CF is found to be greater than ρ then the maximum output activation is checked if it is greater than T. If so then only the network is allowed to execute the learning rules. On the other hand if the maximum output value is found to be less than T then learning rules are not executed.

5. Estimation of the number of nodes

In Section 2 it has been mentioned that the network is formed adaptively along with learning of the input features. The total number of nodes in the network is equal to $n + h + m$, where h is the total number of hidden nodes, m is the total number of output nodes (i.e. the maximum number of categories that can be identified), and n is the number of input nodes (maximum number of input features). Apparently, it seems that the total number of hidden nodes will be approximately equal to the product of the number of input nodes and the number of output nodes, i.e. of $O(mn)$. But in practice h depends on the number of categories that actually share each input feature, because the number of hidden nodes associated with an input node is equal to the maximum number of objects sharing the feature corresponding to that input node. In one extreme case h is of $O(mn)$; in the other extreme, it is equal to the number of input nodes, when each feature belongs to exactly one object. Here, it is assumed that there is no redundant feature, i.e. each feature belongs to at least one output category.

The expected number of hidden nodes can be calculated by considering a particular model of input-output association. It is very unlikely that a feature will belong to all the output categories. Again, the probability that a feature will not belong to any object is zero. Therefore, for any particular feature i, if it is shared by approximately μ_i number of categories, then the probability that the ith feature is shared by μ_i categories will be maximum, and it will decrease as the number of objects increases or decreases. Following the nature of the input-output association, it can be validly assumed that the association follows a truncated Poisson distribution. The probability that the ith feature is shared by j number of categories is given by

$$\Pr(i\text{th feature shared by } j \text{ objects}) = \frac{\dfrac{\exp(-\mu_i)\mu_i^{j-1}}{(j-1)!}}{\displaystyle\sum_{r=1}^{m-1} \frac{\exp(-\mu_i)_i^{r-1}}{(r-1)!}} \tag{20}$$

The expression is normalised because a feature can be shared by at most m categories, i.e. the summation is in the range $j \in [l, m]$. The total number of hidden nodes can be written as $h = \sum_{i=1}^{n} h_i$ where h_i is the number of hidden nodes associated with the input node corresponding to ith feature. The expected value of h_i is given as

$$\begin{aligned} h_i &= \sum_{j=1}^{m} j.\ \Pr(i\text{th feature is shared by } j \text{ objects}) \\ &= 1 + \frac{\displaystyle\sum_{j=0}^{m-1} \frac{j \exp(-\mu_i)\mu_i^j}{j!}}{\displaystyle\sum_{r=0}^{m-1} \frac{\exp(-\mu_i)\mu_i^r}{r!}} \\ &= 1 + \mu_i \left[\frac{\displaystyle\sum_{j=0}^{m-2} \frac{\mu_i^j}{j!}}{\displaystyle\sum_{r=0}^{m-1} \frac{\mu_i^r}{r!}} \right] \\ &= 1 + \mu_i H_i \end{aligned} \tag{21}$$

where

$$H_i = \left[\frac{\displaystyle\sum_{j=0}^{m-2} \frac{\mu_i^j}{j!}}{\displaystyle\sum_{r=0}^{m-1} \frac{\mu_i^r}{r!}} \right] \tag{22}$$

From Eq. (22), it is evident that $H_i < 1$. Again by simple algebraic calculation, H_i can be written as

$$H_i = 1 - \frac{\dfrac{\mu_i^{m-1}}{(m-1)!}}{\sum_{r=0}^{m-1} \dfrac{\mu_i^r}{r!}}$$

But

$$\sum_{r=0}^{m-1} \frac{\mu_i^r}{r!} = \exp(\mu_i) - \sum_{r=m}^{\infty} \frac{\mu_i^r}{r!} < \exp(\mu_i) - \frac{\mu_i^{m-1}}{(m-1)!}$$

Therefore,

$$H_i > 1 - \frac{\dfrac{\mu_i^{m-1}}{(m-1)!}}{\exp(\mu_i) - \dfrac{\mu_i^{m-1}}{(m-1)!}} \tag{23}$$

From Eq. (23), it can be inferred that the total number of hidden nodes in the network is bounded, and the upper and lower bounds in the number of hidden nodes are governed by the bounds of H_i for all i. The total number of hidden nodes h can be written as

$$n + \sum_{i=1}^{n} \mu_i > h > n + \sum_{i-1}^{n} \mu_i - \sum_{i-1}^{n} \frac{\dfrac{\mu_i^m}{(m-1)!}}{\exp(\mu_i) - \dfrac{\mu_i^{m-1}}{(m-1)!}} \tag{24}$$

In Eq. (24), if we assume $\mu_1 = \mu_2 = \cdots = \mu_n = \mu(say)$ then the total number of hidden nodes cannot be greater than $n + n\mu$. For a fixed value of μ, the total number of hidden nodes will be $O(n)$ and thereby the total number of nodes will be of $O(m+n)$.

6. Simulation and experimental results

The network has been simulated on SUN 3/60 workstation. The capability of the network is tested both for binary and visual patterns. Note that the application of the network in practical domains like 2D object recognition, needs domain specific knowledge and is not incorporated here.

The set of binary patterns presented to the network is shown in Table 1. The power of the network is tested with different vigilance and output threshold values.

Table 1
Patterns are presented against features. Full confidence is represented by 1 and no confidence as 0

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}
pat_1	0	0	0	1	0	0	0	1	0	0	0	1	1	1
pat_2	1	1	0	0	1	1	1	0	1	0	0	0	1	0
pat_3	1	0	0	0	0	1	0	0	1	1	1	0	0	0
pat_4	0	1	1	0	1	0	1	0	0	1	0	0	1	1
pat_5	1	1	0	1	1	0	0	0	0	0	0	0	0	1
pat_6	1	1	0	1	1	0	0	1	1	0	0	0	0	0

Table 2
Results of categorization when patterns are presented individually with vigilance factor = 0.9 and noise level = 0.3

Actual class	Category																	
1	1	0	0	0	0	0	42	0	0	0	0	0	0	0	0	0	1	0
2	0	1	0	0	0	0	0	6	0	1	1	26	0	0	8	0	0	1
3	0	0	1	0	0	0	0	0	41	0	0	0	0	0	0	0	1	0
4	0	1	0	39	0	0	0	0	0	0	0	0	1	0	0	1	0	2
5	0	0	2	0	32	0	0	0	0	0	0	0	0	10	0	0	0	0
6	0	0	0	0	0	44	0	0	0	0	0	0	0	0	0	0	0	0

The value of γ is chosen as 0.15. The value of negative self-feedback is set to be 0.05. The patterns are contaminated with additive and subtractive noise. It was found that the network is able to categorize single categories for noise level as high as 30% with an output threshold 0.8. Tables 2 and 3 show the results of categorization for vigilance threshold values 0.9 and 0.65. With a high vigilance factor (ρ), the network is very sensitive to noise and creates a large number of categories. Table 2 illustrates the effect of a large value of ρ on the categorization process. It has been found that 18 categories are formed for six classes, although 12 categories become redundant after repeated presentations of the patterns. Each class gets associated with a particular category, e.g. class 1 goes to 7th category,

Table 3
Results of categorization when patterns are presented individually with vigilance threshold = 0.65 and noise level = 0.3

Actual class	Category						
1	1	0	0	0	0	49	0
2	0	50	0	0	0	0	0
3	0	0	50	0	0	0	0
4	0	0	0	50	0	0	0
5	0	0	0	0	1	0	49
6	50	0	0	0	0	0	0

Table 4
Results of categorization when individual and mixed patterns appear randomly to the network. 'cls' stands for class and 'ctg' stands for category

	ctg1	ctg2	ctg3	ctg4	ctg5	ctg6	ctg7
cls1	0	0	0	0	58	0	0
cls2	0	0	0	0	0	63	0
cls3	0	0	60	0	0	0	0
cls4	74	0	0	0	0	0	0
cls5	0	0	0	56	0	0	0
cls6	0	1	0	0	0	0	58
cls1&2	0	0	0	1	6	5	3
cls1&3	0	0	6	0	6	0	0
cls1&4	6	0	0	0	6	0	0
cls1&5	0	0	0	3	3	0	0
cls1&6	0	0	0	1	4	0	3
cls2&3	0	0	4	0	0	4	0
cls2&4	7	0	0	0	0	7	0
cls2&5	0	0	0	6	0	6	0
cls2&6	0	0	0	0	0	7	7
cls3&4	5	0	5	0	0	3	0
cls3&5	0	0	8	8	0	0	0
cls3&6	0	0	3	0	0	0	3
cls4&5	6	0	0	6	0	0	0
cls4&6	2	0	0	0	0	0	2
cls5&6	0	0	0	5	0	0	7

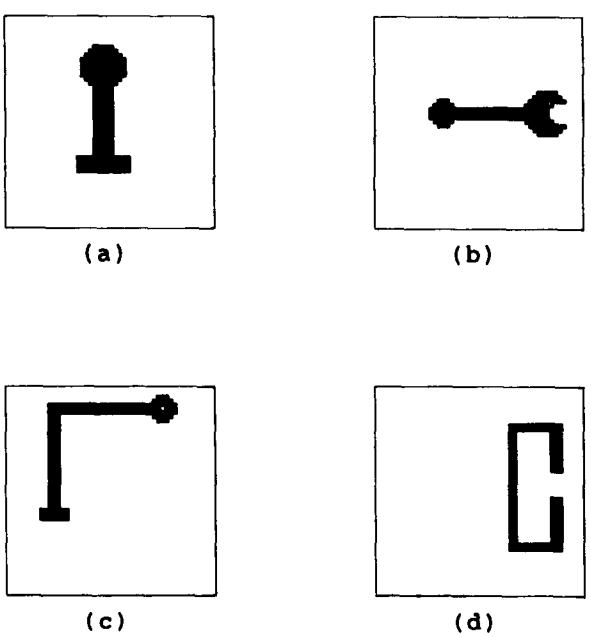


Fig. 2. (a), (b), (c) and (d) represent the objects 1, 2, 3 and 4 respectively used as visual input pattern.

Table 5
Confusion matrix when visual patterns are presented with vigilance threshold = 0.8 and noise level = 0.2. 'cls' stands for class and 'ctg' stands for category

	ctg1	ctg2	ctg3	ctg4
cls1	0	12	0	0
cls2	23	0	0	0
cls3	0	0	19	0
cls4	0	0	0	18
cls1&2	7	7	0	0
cls1&3	0	3	3	0
cls1&4	0	5	0	5
cls2&3	4	0	4	0
cls2&4	2	0	0	2
cls3&4	0	0	7	7

class 2 is mapped to 12th category and so on. Effect of a relatively small value of ρ has been illustrated in Table 3. With a small ρ, the network is less sensitive to noise and as a result, only 7 categories are formed of which only one is redundant.

Next to it the power of the network is tested for categorization of mixed categories. It was found that the network is able to categorize mixed categories correctly. Table 4 presents the results of categorization with 10% contamination (output threshold = 0.8, vigilance threshold = 0.8). It shows that even when the patterns are presented in a mixed form, the network is able to predict that more than one category is present at the input. For example, whenever a mixture of class 2 and class 3 is presented, the model is able to predict that the corresponding categories 3 and 6 are present at the input simultaneously. Similarly, the network response for other mixtures of patterns is illustrated in Table 4. The capability of the network in the categorization of visual patterns is also tested. The patterns (Fig. 2) are contaminated with a noise level of 20% and presented to the network. The network was found to categorize correctly these patterns. Table 5 presents the results of categorization for visual patterns.

7. Discussion

In the present article we have presented a connectionist model which can categorize even in presence of mixed categories. The network architecture is able to incrementally adjust its number of nodes. A theoretical estimate on the number of hidden nodes is also given. The power of the network lies in the categorization of mixed patterns. Although the present work has a similarity with adaptive resonance theory, ART is not able to categorize in presence of mixed categories. Moreover, the network does not consider any ordered search technique to categorize which is essential in adaptive resonance theory. Another advantage of this network is that it is not necessary to give input as 0 or 1; rather it can accept

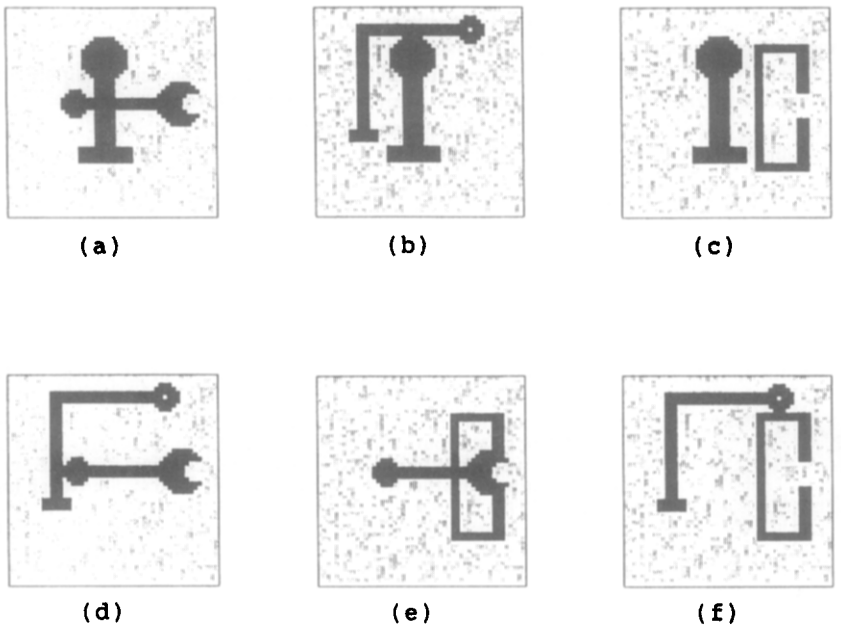


Fig. 3. Mixed patterns with 20% noise level. (a), (b), (c), (d), (e) and (f) represent the mixture of objects 1&2, 1&3, 1&4, 2&3, 2&4, and 3&4 respectively.

intermediate values also. The same model can also be used under supervised mode when an external teacher will operate in place of the monitor network.

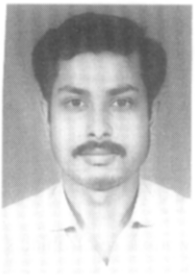
Acknowledgements

The work was done while Prof. S.K. Pal held a Jawaharlal Nehru Fellowship. The computing facility provided by the KBCS center is acknowledged. The authors are thankful to Dr. S. Chaudhury for his helpful discussion.

References

[1] S.I. Amari, Neural theory of association and concept formation, *Biol. Cybernet.* 26 (1977) 175–185.
[2] J.A. Anderson, J.W. Silverstein, S.R. Ritz and R.S. Jones, Distinctive features, categorical perception, and probability learning, *Psychol. Rev.* 84 (1977) 413–451.
[3] J. Basak, C.A. Murthy, S. Chaudhury and D. Dutta Majumder, A connetionist model category perceptron: theory and implementation, *IEEE Trans. Neural Networks* 4 (1993) 257–269.
[4] G.A. Carpenter and S. Grossberg, A massively parallel architecture for a self-organizing neural pattern recognition machine, *Comput. Vision, Graphics and Image Processing* 37 (1987) 34–115.
[5] R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis* (Wiley, New York, 1978).
[6] S.E. Fahlmann and G.E. Hinton, Connectionist architecture for artificial intelligence, *IEEE Comput.* 20 (1987) 100–109.

[7] K. Fukushima, Neural network model for selective attention in visual pattern recognition and associative recall, *Applied Optics* 26 (1987) 4985–4992.
[8] A. Ghosh, N.R. Pal and S.K. Pal, Self-organization for object extraction using multilayer neural network and fuzziness measure, *IEEE Trans. Fuzzy Syst.* 1 (1993) 54–68.
[9] T. Kohonen, *Self-Organization and Associative Memory* (Springer, Berlin, 1988).
[10] R.P. Lippmann, An introduction to computing with neural nets, *IEEE Trans. Acoustics Speech Signal Processing* 4 (1987) 4–22.
[11] S. Mitra and S.K. Pal, Self-organizing neural network as a fuzzy classifier, *IEEE Trans. Syst. Man Cybernet.* 24 (1994) 385–399.
[12] Y. Peng and J. Reggia, A connectionist model for diagnostic problem solving, *IEEE Systems, Man, Cybernet.* 19 (1989) 285–298.
[13] D.E. Rumelhart and J.L. McClelland, eds. *Parallel Distributed Processing, Explorations in Microstructures of Cognition, Vol. I* (Bradford Books/MIT Press, Cambridge, MA, 1986).
[14] C. Stanfill and D. Waltz, Toward memory-based reasoning, *Commun. ACM* 29 (1986) 1213–1228.



Jayanta Basak obtained B.E. in Electronics and Telecommunication Engineering in 1987 from Jadavpur University and M.E. in Computer Science and Engineering in 1989 from Indian Institute of Science, Bangalore. He served as a computer engineer in the KBCS project of the Indian Statistical Institute from 1989 to 1992. He was a programmer in ECSU of the same Institute. Presently he is a programmer in the Machine Intelligence Unit of the Indian Statistical Institute. His research interests are Neural Networks and Computer Vision.

C.A. Murthy was born in Ongole, India in 1958. He obtained his B. Stat (Hons), M. Stat and Ph.D. degrees from the Indian Statistical Institute, Calcutta in 1979, 1980 and 1989 respectively. He visited Michigan State University, USA as UNDP fellow in 1991-92. His fields of interest include Pattern Recognition, Image processing, Computer vision, Fuzzy sets, Neural networks, Fractals and Genetic algorithms.

Sankar K. Pal is a Professor and Founding Head of the Machine Intelligence Unit at the Indian Statistical Institute, Calcutta. He obtained B.Sc. (Hons.) in Physics and B. Tech. and Ph.D. in Radiophysics and Electronics in 1969, 1972, 1974 and 1979 respectively, from the University of Calcutta, India. In 1982 he received another Ph.D. in Electrical Engineering along with DIC from Imperial College, University of London. In 1986 he was awarded a Fulbright Post-doctoral Visiting Fellowship to work at the University of California, Berkeley and the University of Maryland, College Park, USA. In 1989 he received an NRC-NASA Senior Research Award to work at the NASA Johnson Space Center, Houston, Texas, USA.