# A Connectionist Model for Category Perception: Theory and Implementation

Jayanta Basak, C. A. Murthy, Santanu Chaudhury, *Member, IEEE,* and Dwijesh Dutta Majumder

***Abstract*—A connectionist model for learning and recognizing objects (or object classes) has been presented here. The learning and recognition system uses confidence values for the presence of a feature. The network can recognize multiple objects simultaneously when the corresponding overlapped feature train is presented at the input. An error function has been defined and it is minimized for obtaining the optimal set of object classes. The model is capable of learning each individual object in the supervised mode. The theory of learning is developed based on some probabilistic measures. Experimental results have been presented. The model can be applied for the detection of multiple objects occluding each other.**

***Index Terms*—neural network, confidence value, multiple object recognition, conditional probability, supervised learning, state updation, bottom-up link, top-down link, hidden nodes.**

## I. Introduction

ASSOCIATIONS between a set of features characterizing objects or concepts or abstracts or otherwise, is an important characteristic of human intelligence along with our ability to recall from partial information. The problems inheriting this property in fields of pattern recognition and artificial intelligence are implementable with connectionist models. But in the present state-of-the-art connectionist models the simultaneous recall of multiple concepts is nonexistent. In this paper we propose a model which realizes this objective. A connectionist network, satisfying all the above requirements can play a key role in designing problem solving systems which can have general purpose reasoning capabilities for various applications.

Various neural network models have been proposed so far for producing the desired output pattern from the given input pattern. The basic concepts of neural networks are presented in various surveys [17], [3], [4], [7]. Carpenter and Grossberg [2] have developed the adaptive resonance theory for classifying the input patterns into different categories. Both supervised and unsupervised modes of learning are possible in the adaptive resonance theory of network models. The perceptron model [9] was developed to determine classes using simple interclass boundary. To have nonlinear boundary among the classes, multilayered perceptron and the corresponding back-propagation rule was developed [8]. Amari [11] developed a self-organizing model for concept formation and the corresponding orthogonal and covariance learning techniques. Kohonen [19] developed a self-organizing model for automatically producing the topologically correct maps of the features of observable events. Anderson *et al.* [1] have developed the brain-state-in-a-box model to produce the association between the input and the output patterns and applied for categorical perception. They also discussed the probability learning technique in this model. Hopfield [12], [13] proposed a simple but strong model for retrieving output patterns from noisy input patterns. Various application specific models, utilizing the self-organization behavior of the models and the capability of categorization, were also developed. Fukushima [15] developed a multilayered network model for visual pattern recognition which is capable of recognizing characters in a position and size invariant way. Zemel *et al.* [18] developed a multilayered model for object recognition which learns the transformations among the input features and the output objects using back-propagation techniques. Various other application specific models also have been developed so far. The research on neural networks as a optimization problem solver was initiated by Hopfield and Tank [14]. After that, various constraint satisfaction problems were solved by the network models. Stochastic network models were developed in this trend of research. Hinton *et al.* [10] developed a simple but strong learning algorithm for Boltzmann machines.

But most of the methods developed so far are inherently suitable for determining a particular category from the given input pattern. But, as discussed earlier, a feature vector of the observable categories may be generated due to presence of more than one category. This really leads to the superposition of feature vectors. In literature, there are very few works which can really tackle such problems. Peng and Reggia [10] proposed a model for medical diagnosis, by posing the problem as a constraint satisfaction problem. In their work the connection strengths are preassigned depending on the conditional probability values of the diseases given the present symptoms. But in their work there is no explicit algorithm for learning these probability values. Also, in the work of Zemel *et al.* [18], the multiple objects are not recognized simultaneously. But a model which is capable of recognizing the minimum possible set of categories (or object classes) from the input pattern and able to learn the associations of the patterns with the categories is still required to be developed. In this paper a network model which meets both the requirements of forming association and composition is presented. The

Manuscript received October 22, 1991; revised June 4, 1992. This work was supported in part by DoE/UNDP under project IND/85/072.

J. Basak, C. A. Murthy, and D. Dutta Majumder are with the National Center for Knowledge Based Computing Electronics and Communication Sciences Unit, Indian Statistical Institute, Calcutta 700 035, India.

S. Chaudhury is with the Department of Electrical Engineering, Indian Institute of Technology, Hauz Khas, New Delhi, India.

model is capable of learning in the supervised mode. It can be applied for different applications like object recognition, medical diagnosis, etc.

In Section II the general description and the formulation of the problem are presented. The structure of the network model and its dynamical behavior are presented in Section III. Section IV deals with the learning issues and the complexity of the network. Section V discusses the experimental results. Section VI draws the conclusion and gives a line of future research.

## II. Description of the Problem and Formulation

The objective of this section is to provide a mathematical formulation for identifying the categories (or object classes) from a given set of input features. The problem can be looked upon as a similarity-based induction hypothesis as posed by Stanfill and Waltz [6]. The goal of this hypothesis is to make decisions by looking for object classes in the given feature set. Informally it can be said that if an object class be present then the features of the object class should be present.[1] Each feature can be associated to more than one object classes. Initially, each feature should activate its corresponding object classes (formation of initial hypotheses). On the otherhand, each object class should support its constituent features.[2] If a valid feature (present in the scene) does not get proper support from its corresponding object classes, it would strengthen the decision about its corresponding object classes more to receive proper support. On the otherhand, if a feature, which is not really present, gets a substantial support, it would try to enervate the decisions about its corresponding object classes. By this method of iterative reasoning the proper decision about the presence or absence of the object classes can be performed.

The feature set can be looked upon as an input confidence vector, where each confidence value represents the presence or absence of the corresponding feature. In the same way the set of objects can be looked upon as an output confidence vector. The confidence values can be discrete, i.e., 0 or 1, or continuous in the interval $[0, 1]$. The formulations of the problem for two different situations have been discussed in the following subsections.

### *A. Formulation for Discrete Confidence Values*

Let us assume that at most $n$ features can appear in the input, and there are at most $k$ possible output object classes. Let the entire set of features be $\{f_1, f_2, \cdots, f_n\}$, and the set of output objects be $\{o_1, o_2, \cdots, o_k\}$. The feature set for each object can be represented as an input confidence vector

$$\boldsymbol{c} = [c_1, c_2, \cdots, c_n]$$

where $c_i \in \{0, 1\}$ represents the confidence of feature $f_i$, i.e.,

$$c_i = \begin{cases} 0 & \text{if the feature } f_i \text{ is absent} \\ 1 & \text{if the feature } f_i \text{ is present.} \end{cases}$$

Similarly, the set of objects can be described in terms of an output confidence vector given as

$$\boldsymbol{c}^o = [c_1^o, c_2^o, \cdots, c_k^o]$$

where $c_j^o \in \{0, 1\}$ represents the confidence of object $o_j$, i.e.,

$$c_j^o = \begin{cases} 0 & \text{if object } o_j \text{ is absent} \\ 1 & \text{if object } o_j \text{ is present.} \end{cases}$$

For each object class, there is a set of associated features. Let $F_j$ be the feature set associated with the object class $o_j$ or in other words it can be stated that if any object in the object class $o_j$ be present in the scene then all the features in the set $F_j$ are expected to be present. Therefore the set $F = \bigcup_{j=1}^{k} F_j$ represents the complete set of features under consideration. Let the association of a feature with an object class be expressed by a matrix $\boldsymbol{a}$ such that

$$a_{li} = \begin{cases} 1 & \text{if } f_i \in F_l \\ 0 & \text{if } f_i \notin F_l. \end{cases}$$

Mathematically,

$$\max_{i=1,\cdots,k} c_l^o a_{li} = c_i', \qquad \text{for all } i, \quad 1 \le i \le n$$

where $c_i'$ is the confidence of feature $i$ due to different object hypotheses. Observe that $c_i = c_i'$ for all $i$ when there is no noise.

The formulation just described finds out the set of all possible object hypotheses for a given input feature set by simple logical reasoning. But the formulation does not deal with the following prominent points. Firstly, it is very sensitive to noise because if the confidence of any input feature becomes zero due to presence of noise, it eliminates all the possible object hypotheses corresponding to it. Secondly, the method gives the complete set of possible objects. Possibly a proper subset of objects could have explained the input feature train. But the above method does not essentially find out the minimal set. Thirdly, here it is assumed that the features are present in the scene with either zero or full confidence which is very ideal and may not be always possible in practical cases. To avoid these problems, we have to consider continuous confidence values in the interval of $[0, 1]$, and find out the possible object set with minimum cardinality satisfying the given input feature vector.

### *B. Formulation for Continuous Confidence Values*

In this section we are considering that the features can suffer from noise, and therefore the confidence values will not be perfectly 0 or 1, rather they will be in the interval $[0, 1]$. Thus the confidence values here, to some extent, represent the vagueness of the knowledge about the presence of the features. If the confidence $c_i = 0.5$, it indicates a don't know condition about the presence or absence of the feature. And if $c_i > 0.5$, then the possibility of the feature $f_i$ to be present is more than the possibility that it is absent. Since the input confidences take continuous values, the output confidences, as a consequence, are continuous and in the interval $[0, 1]$. Hence the confidences of the input features will not be perfectly matched to the support from the output objects. The recognition problem

[1] Inpractical situation, occlusion may result in loss of feature. That is taken care of in actual formulation.

[2] The word support is used lucidly. The concrete formulation is given later.

in this case can be solved by finding the output confidence values such that the error between the input confidences and the support from the output objects is minimized. Thus an optimization function needs to be defined such that it holds both for discrete and continuous values. Such a function has been defined below as

$$E_1 = 1/2 \sum_{i=1}^{n} \left( \max_{l=1,\cdots,k} c_l^o a_{li} - c_i \right)^2 .$$

Observe that $E_1$ measures the square of the difference of the input confidence and the feedback support over all the features. The constant 1/2 is considered in the expression of $E_1$ for the sake of normalization.

Note that in the output confidence vector obtained as the solution of the above optimization problem will have elements with high confidence values for all the object classes which are supported by the input feature vector. But subsets of the set of output objects thus inferred, in many situations, will account for all the input features because a feature can support more than one object. Among these subsets the one with minimum cardinality can be taken as the simplest explanation of the input vector. To ensure the minimum number of objects, the total summation of the output confidence values is to be minimized. So in addition to $E_1$, another factor is to be minimized, which can be given as

$$E_2 = 1/2 \sum_{l=1}^{k} (c_l^o)^2 .$$

The total optimization function that is to be minimized can be taken as

$$E = \kappa_1 E_1 + \kappa_2 E_2$$

where $\kappa_1$ and $\kappa_2$ are two constants which determine the relative importance of $E_1$ and $E_2$. In situations like English character recognition no more than one character can occur together, and therefore $\kappa_2$ should be large. On the other hand, in industrial parts recognition more than one part is presented frequently, and thereby $\kappa_2$ should be kept small.[3] The actual form of optimization function becomes

$$E = \kappa_1/2 \sum_{i=1}^{n} \left( \max_{l=1,\cdots,k} c_l^o a_{li} - c_i \right)^2 + \kappa_2/2 \sum_{l=1}^{k} (c_l^o)^2 . \quad (1)$$

The function $E$ can be minimized by the gradient descent technique given below.

$$\Delta c_l^o = -\frac{\partial E}{\partial c_l^o} .$$

The solution obtained by this gradient descent technique will at least be a suboptimal one. Thus the solution does not really give the minimal set of objects explaining the input features, rather it gives a more nonredundant set of objects covering all the features. The above equation can be written as

$$\Delta c_l^o = \kappa_1 \sum_{i=1}^{n} e_{il} - \kappa_2 c_l^o \quad (2)$$

where

$$e_{il} = \begin{cases} (c_i - c_l^o a_{li}) a_{li} & \text{if } c_l^o a_{li} \geq c_m^o a_{mi} \quad \forall m \neq l \\ 0 & \text{otherwise.} \end{cases}$$

Equation (2) can be physically interpreted. The value of $a_{li}$ will be 1 if the feature $f_i$ belongs to the object $o_l$, and will be 0 if it does not. So $e_{il}$ basically measures the difference of the input confidence and the maximum feedback support.

## III. Neural Network Model

In the methodology described in Section II-B, the confidence values of the objects are iteratively increased or decreased depending on the support from the features. Note that at each step the change in the confidence value of one object does not depend on that of the other objects. As the process of updation of confidence values of the objects are inherently parallel, this iterative process can be successfully implemented on a neural network (or connectionist model), where the node activities totally depend on the local computations. A network model has been designed here for this particular problem solving paradigm. The model structure and the state updations of the nodes are described in the following subsections.

### A. Structure of the Connectionist Model

The model proposed for solving the problem is shown in Fig. 1. The network is three-layered model, where the output layer consists of nodes whose activations represent the confidences of the output objects. The total number of output nodes denote the maximum number of objects the network is able to recognize. In the input layer the node activations represent the input confidence values of the features, and the number of input nodes indicates the maximum dimension of the input feature vector. The feature-object association matrix [**a**] is embedded in the connection strengths in the network. From (2) it is clear that the supports of a feature to different objects are different, and this depends on the feedback support from the objects. To incorporate this preferential support ($e_{il}$) into the network, a set of hidden nodes is associated with each input node. All these nodes constitute the hidden layer lying between the input and output layer. Each hidden node is connected to exactly one input node and exactly one output node, and plays the key role of associating the input node with the output node. The output nodes are actually instantiated from the input layer only through the hidden nodes, and there is no direct connection from the input layer to the output layer. Each hidden node is connected to a single output node through two kinds of links. The activations from the hidden nodes proceed to the output nodes through the *bottom-up* links, and the activations of the output nodes are fed back to the hidden nodes through the *top-down* links. There is exactly one unidirectional link from each input node to each hidden

[3] Here the large and small are used only to bring the sense, and not how to select them. It will be shown in the next section that $\kappa_1$ and $\kappa_2$ are absorbed in the network parameters, and the selection of parameters are described later.

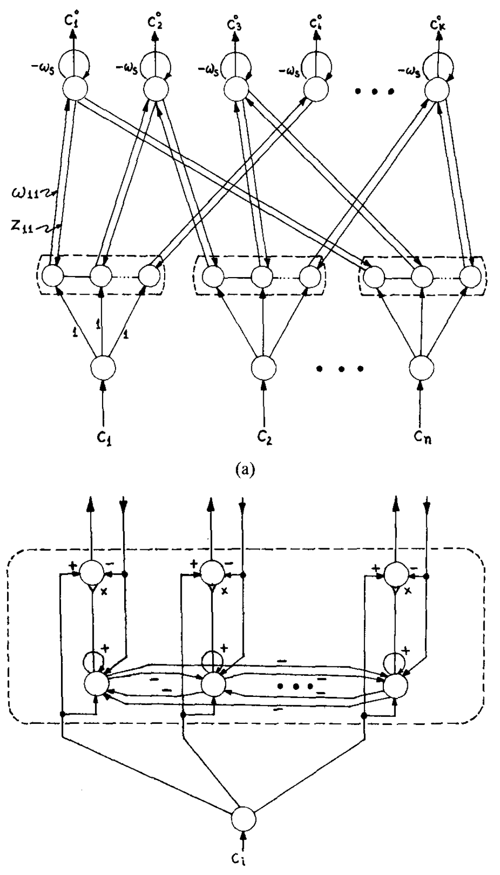


Fig. 1. (a) A schematic diagram of the structure of the network model. (b) Structure and interconnection of the hidden nodes associated with an input node.

nodes associated with it, and the signal can flow from the input nodes to the hidden nodes but not in the opposite direction. The unidirectional links from the input layer to the hidden layer are of fixed weights, and in the present work the weights are set as unity. Each output node is associated with a negative self-feedback. All the hidden nodes connected to a common input node have lateral inhibitory connections among themselves and each one has a self-excitatory feedback.

With the present model, whenever a feature train is presented at the input, the input layer activates the hidden nodes, and activation propagates from the hidden nodes to the output nodes through the bottom-up links. Due to the presence of negative self-feedback in the output, the activations reduce slightly in the output layer, and again propagate to the hidden nodes through the top-down links. Since there is no path from the hidden nodes to the input nodes, activations cannot flow back to the input nodes. As soon as the hidden nodes receive activations through the top-down links, they start competing among themselves through the lateral inhibitory connections. Due to presence of self-excitatory feedback, only that particular hidden node with maximum activation through top-down link will be the winner-take-all[4] [17] and remains enabled, the other hidden nodes will be disabled. In this way corresponding to each input node there will be at most one enabled hidden node.[5] The difference of the activations coming from the input node and through the top-down link to the enabled node is propagated through the bottom-up link to the corresponding output node. Again the same process repeats. Each output node retains its activation for a period over which the competition in the hidden layer takes place. Each hidden node has functionally two parts, one of them retains the activation coming from the output layer, the other part competes and makes the node enabled or disabled. The negative self-feedback in the output layer ensures that the activations of output nodes getting no support from the hidden layer will reduce to zero.

In this process, if the activation from input layer is less than the feedback from output layer to any enabled hidden node, the activation of the corresponding output node will be negated. The effect is just opposite when the input activation is higher than the feedback from output layer. Since the input activations represent the input feature confidences, the effect is just the same as that of $e_{il}$ described in (2). The weights of the top-down links emulate the factor $a_{li}$, and the weights of the bottom-up links emulate the constant $\kappa_1$, while the weights of the self-feedback in the output layer emulate the constant $\kappa_2$. With this network updation the output nodes corresponding to the "true" objects explaining the feature train, will remain highly activated, while activations of other output nodes will disappear. The dynamic behavior of the output nodes are explained in the next subsection.

### *B. Dynamic Behavior of Output Nodes*

The dynamic behavior of the output nodes are described mathematically in this section. Without loss of generality it can be considered that the $l$th object is represented by the $l$th output node, and the $i$th feature is represented by the $i$th input node. In input and output layers the nodes are numbered independently. The hidden nodes are numbered according to the associated input and output nodes, viz. the hidden node connected to $i$th input node and $l$th output node will be numbered as an ordered pair $(i, l)$. The notations used here are given below.

| | |
|---|---|
| $v_l$ | output (or state) of $l$th output node; |
| $u_l$ | instantaneous input to $l$th output node; |
| $w_{il}$ | weight of the bottom-up link from $(i, l)$th hidden node to $l$th output node; |
| $z_{li}$ | weight of the top-down link from $l$th output node to $(i, l)$th hidden node; |
| $w_s$ | weight of negative self-feedback of each output node; |
| $c_i$ | input to $i$th input node. |

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

$$\frac{du_l}{dt} = \sum_{i=1}^{n} w_{il}\delta_{il} - w_s v_l \tag{3}$$

[4]This is due to MAXNET principle.

[5]If the feedback through top-down link is zero to all hidden nodes, then there will be no winner-take-all-node corresponding to that input node.

where $\delta_{il}$ measures the difference of the input activation from $i$th input node and the feedback from $l$th output node, provided the $(i,l)$th hidden node is enabled (winner-take-all node). Mathematically,

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

The output $v_l$ is related to the instantaneous input $u_l$ by a semilinear nondecreasing gain function $g(\cdot)$, i.e., $v_l = g(u_l)$. In the present work $g(\cdot)$ is chosen as an S-function given as

$$\begin{aligned} g(u) &= 0, && \text{for } u \leq 0 \\ &= \tfrac{1}{2}(2u)^{1/\epsilon}, && \text{for } 0 < u \leq 0.5 \\ &= 1 - \tfrac{1}{2}(2-2u)^{1/\epsilon}, && \text{for } 0.5 < u \leq 1 \\ &= 1, && \text{for } u > 1. \end{aligned}$$

In the transfer function $g(\cdot)$, the value of $\epsilon$ controls the gain of the transfer function. The value of $\epsilon$ varies in the range $[0,1]$. If $\epsilon$ is 0, then output takes values only 0 and 1. On the other hand if $\epsilon$ is 1, then the gain is simply a linear one.

The dynamic system described by (2) can be shown to converge to stable states by having an energy function $\mathcal{E}(t)$ defined as

$$\mathcal{E}(t) = 1/2\sum_{i=1}^{n}\lambda_i'\left(\max_{l=1,\cdots,k} z_{li}v_l - c_i\right)^2 + 1/2w_s\sum_{l=1}^{k}v_l^2 \quad (4)$$

where $n$ is the number of input nodes and $k$ is the number of output nodes. $\lambda_i'$ is a multiplication factor such that

$$\lambda_i' = w_{is}/z_{si}, \qquad \text{if } z_{si}v_s = \max_{l=1,\cdots,k} z_{li}v_l.$$

The change of energy with the parameter $t$ (time) can be written as

$$\frac{d\mathcal{E}}{dt} = \sum_{l=1}^{k}\frac{\partial\mathcal{E}}{\partial v_l}\frac{dv_l}{dt}. \quad (5)$$

But,

$$\frac{\partial\mathcal{E}}{\partial v_l} = -\left(\sum_{i=1}^{n} z_{li}\lambda_{il}\delta_{il} - w_s v_l\right)$$

i.e.,

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

So, (5) can be written as

$$\frac{d\mathcal{E}}{dt} = -\sum_{l=1}^{k} g_{-1'}(v_l)\left(\frac{dv_l}{dt}\right)^2. \quad (6)$$

From (6) it is evident that $(d\varepsilon/dt) \leq 0$ for all $t \geq 0$. As $t \to \infty$, $(d\mathcal{E}/dt) \to 0$, and thereby the dynamic system converges to the local energy minima.

Comparing the energy function $\mathcal{E}(t)$ in (4) and the optimization function $E$ in (1), it is clear that the weights of the top-down links play the same role as the quantity $a_{li}$ in (1). The weights of the top-down links should measure the relative importance of the features with respect to the objects. The factor $\lambda_i'$ to some extent imitates the constant $\kappa_1$. Actually in the optimization function $E$, the effects of the errors due to the mismatch of the input confidence and the feedback support were taken to be the same for all features. But in practice the situation should be different; if a feature is very important with respect to some object, the effect of the mismatch corresponding to that feature-object association should be more. Actually, these preferential effects of mismatch are taken care of in the energy function (by the use of different weights in the bottom-up links), and thereby in the dynamic behavior of the output nodes. How to select and automatically learn the weights of the links will be discussed in the next section.

## IV. Learning the Object Categories

So far, the network structure and the dynamic behavior of the network have been discussed. The network will be able to correctly recognize the objects from the input feature train, and thereby will be able to explain the input features with a nonredundant set of objects in the output if and only if the weights of the top-down and bottom-up links are set properly. In the current work, supervised mode of learning has been considered for obtaining the weights of these links. In this learning mode, a single object (training sample) is presented to the network at a time, and the input confidence vector of the corresponding feature set is presented at the input. At the output layer the node corresponding to the training sample is enabled, and all other nodes are disabled. The weights of the links are then modified according to the learning rules, as discussed in the following subsections. The process of learning is continued until the change in the weights becomes insignificant and then the process is said to have converged.

In the process of learning, the links of the network are also built up. Initially, there exists a pool of input nodes, hidden nodes, and output nodes in the network. Whenever a training sample is presented, one output node and some of the input nodes are enabled. If there is no association between the enabled output node and any enabled input node (i.e., both of them are not connected to a common hidden node), then an arbitrary hidden node (if it is not already associated with an input–output pair) will be connected to that input and output node (bottom-up and top-down links are created).

If corresponding to an input–output association, there already exists a hidden node, then the weights of the links are modified. In the modification process, the rate of change of weights should not be the same for all the links. This is due to the fact that as the learning process goes on, the rate of learning decreases which ensures the convergence (discussed in Section IV-C). Therefore, if a link is created in the earlier part of learning process, its rate will be decreased compared to a newly created link. Each node (input and output) has an attribute which determines for how much time the node has been enabled. Henceforth, this attribute will be termed as *agility factor.* Actually, agility factor decreases with the amount of time for which node has been enabled. Although each hidden node is enabled for a less amount of time compared to its associated input node, it has no separate agility factor, and this is the same as that of the associated input node.

The rate of learning of any link at any instant will be governed by the agility factors of the nodes at its two ends. For learning of the weights, a suitable measure of the weights of the links are considered and discussed in the following subsections.

### *A. Measure of Weights*

The importance of any feature with respect to an object model primarily depends on two factors. Firstly, to what extent the feature is consistent with respect to that particular object, i.e., given the object is present, what is the likelihood that the feature will appear in the input. Actually, this factor ensures that if a feature is erratic and very prone to noise then it is better not to associate much importance with that particular feature-object association. This is supported by the fact that the change in output confidence of any output object due to mismatch in any input feature is proportional to $a_{li}$ (2), where $a_{li}$ is 1 or 0 depending on whether the $i$th feature belongs to the $l$th object or not. If the value of $a_{li}$ is extended to the continuous domain in $[0, 1]$ then $a_{li}$ basically gives a measure of the likelihood of the appearance of the $i$th feature with respect to the $l$th object.

Secondly, the importance of a feature with respect to a particular object means how likely the object is, given that the feature is present. Therefore if a feature is consistent and unique with respect to a particular object, then the feature should have high importance with respect to that object. On the otherhand, despite consistency, if the feature is shared evenly by large number of objects, the importance of the feature will be low with respect to any one of the objects.

Considering both the factors just described, the weights of the bottom-up links are taken to be proportional to the likelihood of the appearances of the features with respect to the objects, and as well as the likelihood of the appearances of the objects with respect to the features. A measure of importance of any feature $f_i$ with respect to an object $o_l$ can be given as

$$m(o_l, f_i) = m_1(o_l, f_i) m_2(o_l, f_i)$$

where $m_1$ gives a measure of the chance of the appearance of $o_l$ if $f_i$ is present and $m_2$ gives a measure of the chance of the occurrence of $f_i$ given that $o_l$ is present. Without loss of generality, $m_1$ and $m_2$ can be considered to be semilinear nondecreasing functions of the corresponding conditional probabilities. Here we have taken $m_1(o_l, f_i)$ to be $p(o_l|f_i)$ and $m_2(o_l, f_i)$ to be $p(f_i|o_l)$, where $p(o_l|f_i)$ is the conditional probability of occurrence of $o_l$ given that $f_i$ is present (similar explanation for $p(f_i|o_l)$).[6]

As described in the dynamic behavior of the network in (3), the effect of output activation is modulated by the weight of the top-down link. The effect is the same as multiplying $c_l^o$ by $a_{li}$ in (2). Actually if a feature is erratic with respect to an object, it should not get proper support from the object, and the weight of the top-down link measures the consistency of the feature with respect to the object. Therefore the learning process should be designed so that after convergence, the weight of the top-down link, should capture the measure $m_2$, i.e., $z_{li} \propto m_2(o_l, f_i)$. Again $m_2(o_l, f_i)$ is 1 if feature $f_i$ is fully consistent with object $o_l$, i.e., $a_{li} = 1$ in case of perfect features, as described in Section II-B. The weight of the top-down link, therefore, can be taken exactly equal to $m_2(\cdot)$, i.e., $z_{li} = m_2(o_l, f_i)$. The measure $m_2(\cdot)$ actually gives a value proportional to $w_{il}/z_{li}$. In (3) describing the dynamic behavior of the nodes, $\lambda_{il}$ is therefore proportional to $m_1(o_l, f_i)$, which supports the discussion presented in Section II-B. We will denote the measure $m_2(o_l, f_i)$ as $\lambda_{il}$.

In selection of the actual weights (i.e., after learning what should be the weights), the weights of the bottom-up links are selected in such a way that the total input going to an output node, with perfect input feature train presented, be unity. This is done because the gain function of each node is chosen as an S-function, which saturates if the input is greater than or equal to unity. In order that an output node does not saturate for a noisy input confidence vector and reaches the edge of saturation for a perfect input confidence vector, the sum of all the bottom-up weights going to that output node must be unity. For this purpose, the weights of the bottom-up links are selected as

$$w_{il} = \frac{m(o_l, f_i)}{\sum_{j=1}^{n} m(o_l, f_j)}.$$

If the feature set of an object is a subset of the feature set of another object, and the probabilities of appearance, of both the objects are same, then whenever the larger feature set is presented during recognition mode, both the objects will be fully activated. To prevent such a situation, the measure of the weights of the bottom-up links are modified according to Weber's law as described in ART [2]. The modified measure is given as

$$w_{il} = \frac{m(o_l, f_i)}{\gamma + \sum_{j=1}^{n} m(o_l, f_j)}. \tag{7}$$

where $\gamma$ is a constant. It will be shown that $\gamma$ actually determines the rate of learning also.

### *B. Learning of the Weights*

The weights of the links in the network are changed in the supervised learning mode in such a way that they become the same as the measures, as described above, after a sufficient number of learning trials. The conditional probability $p(o_l|f_i)$ can be considered to be equal to $N_l'/N_i$ (almost everywhere) for large values of $N_l'$ and $N_i$. $N_i$ represents the total number of occurrences of feature $f_i$, and $N_l'$ is the number of occurrences of object $o_l$ given that feature $f_i$ has occurred. Initially the ratio does not give actual probability values, but if the measure is made equal to the ratio, then as the number of presentations becomes very high, the measure converges to the actual probability value. Therefore at any instant of time $t$,

$$\lambda_{il}(t) = \frac{N_l'}{N_i}$$

where $N_l'$ and $N_i$ are now interpreted as the respective number of occurrences up to the instant $t$. Similarly, the weights of

[6]Here for simplicity of representation, $o_l$ has been used to denote the occurrence of $o_l$.

bottom-up links take the values given as

$$z_{li}(t) = \frac{N_i'}{N_l}$$

where $N_l$ is the total number of occurrences of object $o_l$, and $N_i'$ is the number of occurrences of feature $f_i$ given the object $o_l$ is present up to instant $t$. If the feature $f_i$ is present at time instant $t+1$ then the value of $\lambda_{il}(t)$ changes as

$$\lambda_{il}(t+1) = \begin{cases} \frac{N_l'+1}{N_i+1} & \text{if } o_l \text{ is present at instant } t+1 \\ \frac{N_l'}{N_i+1} & \text{if } o_l \text{ is absent at instant } t+1 \end{cases}$$

If the intervals of occurrences are considered instead of the number of occurrences, then $\lambda_{il}(t)$ can be written as

$$\lambda_{il}(t) = \frac{T_l'(t)}{T_i}$$

where $T_i$ is the total time for which the feature $f_i$ is presented, and $T_l'$ is the total interval for which $o_l$ was present given that $f_i$ was present.

In the supervised learning process, every time a teaching vector is presented at the output layer, a single output node is enabled and the other nodes are disabled. Let the teaching vector be denoted by

$$\boldsymbol{y}(t) = [y_1(t), y_2(t), \cdots, y_k(t)]$$

i.e., for each output node there is a teaching input, which changes over time $t$. If the feature $f_i$ is present for the interval $\Delta t$ then the value of $\lambda_{il}(t)$ changes to

$$\lambda_{il}(t+\Delta t) = \frac{T_l' + y_l(t)\cdot\Delta t}{T_i + \Delta t}.$$

The equation is valid if $\Delta t$ is small enough, and if the teaching input $y_l(t)$ is considered to remain unchanged over the interval $\Delta t$. The change of $\lambda_{il}(t)$ can be written as

$$\Delta\lambda_{il}(t) = \lambda_{il}(t+\Delta t) - \lambda_{il}(t)$$

i.e.,

$$\Delta\lambda_{il}(t) = \begin{cases} 0 & \text{if } c_i = 0 \\ \frac{T_l'+y_l(t)\cdot\Delta t}{T_i+\Delta t} - \frac{T_l'}{T_i} & \text{if } c_i \neq 0 \end{cases}$$

i.e.,

$$\Delta\lambda_{il}(t) = \frac{\Delta t(T_i y_l(t) - T_l')c_i}{T_i(T_i+\Delta t)}.$$

In the above expression it is considered that $c_i$ can take values of 0 and 1 only. But the same expression can be taken to be valid for fractional values of $c_i$ in the range of $[0, 1]$. The convergence of the above expression for continuous values of $c_i$ will be given in the next section. The rate of change of $\lambda_{il}(t)$ can be written as

$$\frac{\Delta\lambda_{il}(t)}{\Delta t} = \frac{(y_l - \lambda_{il}(t))c_i}{T_i + \Delta t}.$$

Letting $\Delta t \to 0$, the differential change of $\lambda_{il}(t)$ with time $t$ becomes

$$\frac{d\lambda_{il}}{dt} = \alpha_i c_i(y_l - \lambda_{il}) \tag{8}$$

where

$$\alpha_i = 1/T_i.$$

$\alpha_i$ decreases with the time of activation of the $i$th input node. $\alpha_i$ is actually the agility factor associated with the $i$th input node. Initially in the network, agility factors for all nodes are unity. As the nodes are enabled in the learning mode, the agility factor decreases. The change of agility factor for the $i$th input node in time $\Delta t$ can be given as

$$\Delta\alpha_i(t) = \begin{cases} 0, & \text{if } c_i = 0 \\ 1/T_i - 1/(T_i+\Delta t), & \text{if } c_i \neq 0. \end{cases}$$

Letting $\Delta t \to 0$, the rate of change of $\alpha_i(t)$ becomes

$$\frac{d\alpha_i}{dt} = -\alpha_i^2 c_i. \tag{9}$$

Considering the measure $m_1(o_l, f_i)$, the rate of change of weight of the top-down links can be given as

$$\frac{dz_{li}}{dt} = \alpha_l^o y_l(c_i - z_{li}) \tag{10}$$

where $\alpha_l^o$ is the agility factor of the $l$th output node. The rate of change of $\alpha_l^o$ can be given as

$$\frac{d\alpha_l^o}{dt} = -{\alpha_l^o}^2 y_l. \tag{11}$$

The total weights of the bottom-up links should be such that for a set of inputs the output equals the teaching input vector. Therefore the expression for $w_{il}$ after convergence can be written as

$$w_{il} = \frac{\lambda_{il} z_{li} g^{-1}(y_l)}{\gamma + \sum_{j=1}^{n} \lambda_{jl} z_{lj} c_j}. \tag{12}$$

The explanation for $\gamma$ is given in the previous section. The gain $g(\cdot)$ is considered to be the same for all nodes. With this expression for $w_{il}$, the total input $(u_l)$ to the $l$th output node becomes

$$u_l = \frac{g^{-1}(y_l)\sum_{i=1}^{n}\lambda_{il} z_{li} c_i}{\gamma + \sum_{j=1}^{n}\lambda_{jl} z_{lj} c_j}.$$

By mathematical restructuring, the expression can be written as

$$u_l = g^{-1}(y_l) - \frac{\gamma w_{il}}{\lambda_{il} z_{li}}$$

i.e.,

$$y_l = g\left(u_l + \frac{\gamma w_{il}}{\lambda_{il} z_{li}}\right).$$

The constant $\gamma$ is considered to be very small. Using Taylor's expansion, and restoring up to second term, the expression can be written as

$$y_l - o_l = \frac{\gamma w_{il} g'(u_l)}{\lambda_{il} z_{li}}$$

i.e.,

$$w_{il} = \frac{\varepsilon_l \lambda_{il} z_{li}}{\gamma g'(u_l)}$$

where

$$\varepsilon_l = y_l - o_l.$$

The factor $\varepsilon_l$ measures the error between the teaching input and the actual output at the $l$th output node. The required rate of change of $w_{il}$ should be

$$\frac{dw_{il}}{dt} = \frac{\varepsilon_l}{\gamma g'(u_l)}\left(z_{li}\frac{d\lambda_{il}}{dt} + \lambda_{il}\frac{dz_{li}}{dt}\right).$$

Let $\delta_l = (\varepsilon_l / \gamma g'(u_l))$. Using (8) and (10), the above expression can be written as

$$\frac{dw_{il}}{dt} = \overbrace{\left(\alpha_i \delta_l z_{li} + \alpha_l^o\left(\frac{w_{il}}{z_{li}}\right)\right)c_i y_l}^{\text{I}} - \overbrace{(\alpha_i c_i + \alpha_l^o y_l) w_{il}}^{\text{II}}. \tag{13}$$

In (13), the factor $1/\gamma$ determines the rate of learning. If $\gamma$ increases, the rate decreases, and vice-versa. Part I of (13) indicates that the weight of a link changes with the product of the activations of the nodes at the two ends, modulated by the error at the output node. This, in fact, supports the Hebbian rule of learning [16]. Part II shows a decay in the weight of the link, which actually supports the associative decay of the weights of the links [2]. In the next section we will be considering the convergence of the weights of the links to the proposed measures.

### C. Convergence of the Weights

In this section, convergence of the weights of the top-down and bottom-up links have been considered. If the presentation of the patterns are considered to be random then $\boldsymbol{y(t)}$ is a stochastic process, and the appearances of the input activations are also random, i.e., $\boldsymbol{c(t)}$is a stochastic process. Initially the agility factors for all the nodes are set to unity. The agility factor of a node (input or output) decreases whenever it is activated. Whenever an input–output association is formed through a hidden node, the weight of the corresponding top-down link is set to unity. In the learning process, if the input–output association already exists in the network, the weight of the corresponding top-down link will change according to (10). The weight of the bottom-up links are initially set to zero, and they increase according to (13). The initial values of the link weights and the agility factors can be summarized as follows:

$$z_{li}(0) = 1, \qquad w_{il}(0) = 0, \qquad \alpha_i(0) = 1, \qquad \alpha_l^o(0) = 1$$

where $t = 0$ indicates the association of $i$th input node and $l$th output node is just formed. The convergence of the bottom-up weights need not be proved separately, rather if the convergence of $\lambda_{il}$ and $z_{li}$ can be shown, then it can be said that $w_{il}$ converges for a given $\gamma$. The convergence of the weights, with the given initial conditions, can be proved both for perfect binary inputs and noisy inputs.

For perfect binary inputs, $\alpha_i$ decreases and $\lambda_{il}$ changes only when $c_i = 1$. Let $c_i = 1$ at the time intervals $(\tau_0, \tau_1), (\tau_2, \tau_3), (\tau_4, \tau_5), \cdots$ and so on. Let us denote the total time for which $c_i = 1$ as $\tau_i'$, i.e.,

$$\tau_i' = \sum_{r=0} (\tau_{2r+1} - \tau_{2r}).$$

For nonzero probability of the $i$th feature, $\tau_i' \to \infty$ as $t \to \infty$. Since there is no change in the values of either $\alpha$ or $\lambda$ when $c_i \neq 1$ (i.e., $c_i = 0$), $\lambda_{il}$ will converge as $\tau_i' \to \infty$ if $\lambda_{il}$ converges for $t \to \infty$, and vice versa. Therefore the convergence of $\lambda_{il}$ can be proved considering $c_i = 1$ for all $t \geq 0$. Equation (9) can be rewritten as

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

Therefore $\alpha_i = 1/(t+1)$ with the initial value 1. From (9) and (8), the equation for $\lambda_{il}$ can be rewritten as

$$\frac{d\lambda_{il}}{d\alpha_i} = \frac{\lambda_{il} - y_l}{\alpha_i}.$$

From the above equation it is evident that $\lambda_{il}$ will solely depend on the change of $y_l$. Let $y_l = 1$ in the intervals $(t_0, t_1)$, $(t_2, t_3)$, $(t_4, t_5)$, $\cdots$ and so on. Separately integrating for $y_l = 1$ and $y_l = 0$, we have

$$\left.\begin{aligned} \log\left(\tfrac{1-\lambda_{il}(t_{2r+1})}{1-\lambda_{il}(t_{2r})}\right) &= \log\left(\tfrac{\alpha_i(t_{2r+1})}{\alpha_i(t_{2r})}\right) \\ \text{and } \log\left(\tfrac{\lambda_{il}(t_{2r+2})}{\lambda_{il}(t_{2r+1})}\right) &= \log\left(\tfrac{\alpha_i(t_{2r+2})}{\alpha_i(t_{2r+1})}\right) \end{aligned}\right\} \quad \forall r \geq 0.$$

From the above equation the sequence of values of $\lambda_{il}$ can be written as

$$\left.\begin{aligned} \lambda_{il}(t_{2r+1}) &= 1 - \tfrac{\alpha_i(t_{2r+1})}{\alpha_i(t_{2r})}(1 - \lambda_{il}(t_{2r})) \\ \text{and } \lambda_{il}(t_{2r+2}) &= \tfrac{\alpha_i(t_{2r+2})}{\alpha_i(t_{2r+1})}\lambda_{il}(t_{2r+1}) \end{aligned}\right\} \quad \forall r \geq 0.$$

By simple algebraic calculation, the value of $\lambda_{il}(t_{2k+1})$ can be written as

$$\lambda_{il}(t_{2k+1}) = \sum_{r=0}^{2k+1} (-1)^{2k-r+1}\frac{\alpha_i(t_{2k+1})}{\alpha_i(t_r)}.$$

Since $\alpha_i(t) = 1/(t+1)$, the above expression can be written as

$$\lambda_{il}(t_{2k+1}) = \frac{\sum_{r=0}^{k}(t_{2k+1} - t_{2r})}{1 + t_{2k+1}}.$$

In the above expression the numerator represents the time for which the $l$th output node is active during supervised learning, i.e., $y_l = 1$ upto the time instant $t_{2k+1}$. Therefore

$$\lim_{k\to\infty} \lambda_{il}(t_{2k+1}) = \Pr(y_l = 1 | c_i = 1)$$

almost everywhere by strong law of large numbers [5]. This ensures that $\lambda_{il}(t)$ converges to the conditional measure proposed in the previous section. By similar reasoning it can be proved that

$$\lim_{k\to\infty} z_{li}(t_{2k+1}) = \Pr(c_i = 1 | y_l = 1).$$

Therefore the product $\lambda_{il} z_{li}$ converges, and $w_{il}$ converges.

TABLE I
FEATURE VECTORS OF DIFFERENT OBJECTS

| objects | features | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| object 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| object 2 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| object 3 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| object 4 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| object 5 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| object 6 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| object 7 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |

TABLE II
PROBABILITIES OF APPEARANCES OF THE OBJECTS

| objects | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| probabilities | 0.1 | 0.14 | 0.16 | 0.2 | 0.1 | 0.15 | 0.15 |

If the input feature train is contaminated by noise (and also the teaching input) then the reasoning about the presence or absence of feature no more holds good. Then the convergence of the weights can be proved by considering the inputs and outputs to be continuous signals, i.e., in this case a signal is considered to be present at each input node and at each output node. If a perfect feature is present at an input node, the activation at that node reaches its full strength (i.e., unity), and if the feature is noisy, the activation is less than unity. Similarly, if the feature is absent without any additive noise, the activation at the corresponding input node is zero, and for nonzero additive noise a nonzero activation is present at that node. Thus after simple algebraic manipulations, it can be shown that

$$\lambda_{il}(t) = \frac{\int_0^t c_i(\tau) y_l(\tau)\, d\tau}{1 + \int_0^t c_i(\tau)\, d\tau} \tag{14}$$

where the value of $\lambda_{il}$ at time instant $t$ is denoted as $\lambda_{il}(t)$. In (14) if $c_i$ and $y_l$ take values in $\{0, 1\}$, then the value of $\lambda_{il}(t)$ converges to the conditional probability measure as $t \longrightarrow \infty$. With similar considerations it can be shown that $z_{li}(t)$ also converges as $t \longrightarrow \infty$, and thereby $w_{il}$ converges.

## V. EXPERIMENTAL RESULTS AND ANALYSIS

The network has been simulated on SUN 3/60 workstations and tested for different synthetic patterns. The experiment has been performed in two stages. In the first stage the network is allowed to learn the input patterns with suitable teaching vector at the output nodes. After a suitable number of learning trials the network is tested for recognition task in the second stage. This section describes the results of these experiments and analyses the results.

In learning stage, the pattern vectors to be learned are considered to have preassigned probabilities of appearances in the network. Input vectors are applied to the input in a frequency determined by their preassigned probability values. With each input vector the information about the corresponding object is stored, and from this stored information the output vector is generated and then acts as the teaching vector in the supervised mode of learning. In actual network formation, a hidden node is connected in the network whenever a new input–output pair appears. In the simulated network, for sake of simplicity, the total number of hidden nodes is considered to be equal to the product of the number of input and output nodes. Although there may be a lot of redundant nodes in the hidden layer, the performance of the network in terms of learning the input vectors and recognizing the objects will not be affected at all. The weights of the bottom-up links corresponding to the uninstantiated hidden nodes happen to be zero and do not in any way affect the recognition task. Only the time performance of the network may degrade to some extent during recognition on a sequential machine.

In the first set of experiments, the network is simulated with fourteen input nodes and seven output nodes. Seven different 14-bit vectors are presented to the network during the learning trials. The input vectors are shown in Table I. The preassigned probability values of the seven objects are shown in Table II. The objects are randomly presented to the network according to their probability values. This is done by generating a random number in the range $[0, 1]$ by the system-built pseudo-random number generator. If the number is such that $P_{k-1} \leq N < P_k$, where $P_k = \sum_{i=1}^{k} p_i$, ($p_i$ is the preassigned probability value of the $i$th pattern) then the generated pattern is considered to be the $k$th pattern. The network is trained with 15 000 learning trials and the corresponding results are shown in Table III and Table IV. The value of $\gamma$, determining the rate of learning, is taken as 0.1, and the time step is chosen as 0.005. From Table III it is obvious that the weights of the bottom-up links corresponding to features shared by a large number of objects are comparatively small. Also, the weights of the bottom-up links corresponding to the features shared by a fewer number of objects are comparatively high. During the learning trials the input vectors may be contaminated by noise. In the present case, supervised learning is performed with perfect input vectors. Therefore, if the weight of a top-down link becomes unity, it remains unchanged.

In the second stage of the experiment, the noisy patterns are presented at the input and the network is executed in the recognition mode. Different synthetic patterns (may be caused by one or more objects) with different noise levels are applied

TABLE III
WEIGHTS OF THE BOTTOM-UP LINKS ($w_{ij}$)

| | Outputs ($j$) | | | | | | |
|---|---|---|---|---|---|---|---|
| Inputs ($i$) | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | 0.0 | 0.097 | 0.0 | 0.150 | 0.0 | 0.196 | 0.117 |
| 2 | 0.0 | 0.090 | 0.085 | 0.0 | 0.073 | 0.180 | 0.108 |
| 3 | 0.0 | 0.0 | 0.221 | 0.0 | 0.189 | 0.0 | 0.0 |
| 4 | 0.113 | 0.0 | 0.0 | 0.0 | 0.0 | 0.309 | 0.184 |
| 5 | 0.0 | 0.115 | 0.0 | 0.0 | 0.093 | 0.233 | 0.139 |
| 6 | 0.0 | 0.124 | 0.117 | 0.191 | 0.0 | 0.0 | 0.0 |
| 7 | 0.0 | 0.253 | 0.0 | 0.0 | 0.205 | 0.0 | 0.0 |
| 8 | 0.176 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.287 |
| 9 | 0.0 | 0.096 | 0.091 | 0.147 | 0.0 | 0.0 | 0.115 |
| 10 | 0.0 | 0.0 | 0.127 | 0.206 | 0.109 | 0.0 | 0.0 |
| 11 | 0.0 | 0.0 | 0.161 | 0.261 | 0.0 | 0.0 | 0.0 |
| 12 | 0.409 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 13 | 0.133 | 0.180 | 0.0 | 0.0 | 0.146 | 0.0 | 0.0 |
| 14 | 0.126 | 0.0 | 0.161 | 0.0 | 0.138 | 0.0 | 0.0 |

TABLE IV
WEIGHTS OF THE TOP-DOWN LINKS ($z_{ji}$)

| | Outputs ($i$) | | | | | | |
|---|---|---|---|---|---|---|---|
| Inputs ($j$) | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | 0.0 | 1.0 | 0.0 | 1.0 | 0.0 | 1.0 | 1.0 |
| 2 | 0.0 | 1.0 | 1.0 | 0.0 | 1.0 | 1.0 | 1.0 |
| 3 | 0.0 | 0.0 | 1.0 | 0.0 | 1.0 | 0.0 | 0.0 |
| 4 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 1.0 |
| 5 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 1.0 | 1.0 |
| 6 | 0.0 | 1.0 | 1.0 | 1.0 | 0.0 | 0.0 | 0.0 |
| 7 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 |
| 8 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |
| 9 | 0.0 | 1.0 | 1.0 | 1.0 | 0.0 | 0.0 | 1.0 |
| 10 | 0.0 | 0.0 | 1.0 | 1.0 | 1.0 | 0.0 | 0.0 |
| 11 | 0.0 | 0.0 | 1.0 | 1.0 | 0.0 | 0.0 | 0.0 |
| 12 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 13 | 1.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 |
| 14 | 1.0 | 0.0 | 1.0 | 0.0 | 1.0 | 0.0 | 0.0 |

at the input. The results of the recognition task are shown in Table V. The network is simulated for 500 iterations for each instance of an input pattern with the self-feedback equal to 0.1 and timestep = 0.05. Uniform noise is added to the input by the system-built pseudo-random number generator. The power of the network is tested for overlapped patterns of more than one object. The table shows that whenever the overlapped pattern corresponding to objects 5 and 6 is presented to the network, the network recognizes the objects correctly without any error even with 30% of noise. When the overlapped pattern corresponding to objects 3 and 4 is presented to the network, it recognizes correctly both the objects even with 10% noise in the input. But it fails to recognize the object 4 in 6% of the cases when input is contaminated with 20% of noise. This is due to the fact that when the patterns of object 3 and 4 are superposed, the object 4 retains only one distinctive feature which does not belong to object 3 (feature 1). On the otherhand object 3 has three distinctive features not belonging to object 4 (features 2, 3, and 14). Therefore the initial activation of object 3 is higher than that of object 4, and in the iterative process of recognition, object 3 gets support from the common features (features 6, 9, 10, and 11) and its own distinctive features as well. On the otherhand object 4 gets support from only one distinctive feature. If the input patterns are heavily contaminated by noise, the initial activation of the objects will be very low. In the process of iterative activation, object 4 sometimes does not get much of the support from its single distinctive feature and that is flagged in the output. The results also show that the network fails to recognize object 4 in 16% of the cases when the input is contaminated by 30% of noise. When the overlapped pattern of objects 1 and 2 is presented, the network recognizes the object 7 along with the objects 1 and 2 almost 47% of time with 10% noise. The chance of recognizing the extra object increases with the percentage of noise. This is due to the fact that the difference of the pattern of object 7 with the overlapped pattern of objects 1 and 2 is only in the last three bits. The noise injected in the patterns acts both in additive and subtractive fashion. In the initial settling process of the network, the total percentage of error in the pattern of object 7 and that in the pattern of object 1 and object 2 become comparable. Therefore, the output of the node corresponding to object 7 becomes high initially and therefore it is supported by the features in the overlapped pattern, and the recognition of

TABLE V
OUTPUTS WITH SUPERPOSED FEATURE VECTORS

| | recognition scor | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | noise level = 0.1 | | | noise level = 0.2 | | | noise level = 0.3 | | |
| pattern | true | false | extra | true | false | extra | true | false | extra |
| objects 3 and 4 | 100 | 0 | 0 | 94 | 0 | 0 | 84 | 0 | 0 |
| objects 5 and 6 | 100 | 0 | 0 | 100 | 0 | 0 | 100 | 0 | 0 |
| objects 1 and 2 | 47 | 0 | 53 | 43 | 0 | 57 | 38 | 0 | 62 |
| object 7 | 100 | 0 | 0 | 92 | 0 | 8 | 85 | 0 | 15 |

an extra object occurs. The fourth row of the table shows that when the pattern of object 7 is presented to the network with 20% of noise, the object 6 is recognized along with object 7 in 8% cases. This is due to the fact that object 7 has two extra features than object 6. Therefore whenever object 7 is presented, all the features of object 6 is present in the scene. But the formulation in (12) prevents the object 6 to become fully active when the pattern of object 7 is presented. Again the effectiveness of this discrimination will depend on the constant $\gamma$. With the chosen value of $\gamma$, the effectiveness of this restriction fails in 8% cases with 20% noise, and recognition of an extra object occurs. As the noise increases, the chance of recognizing object 6 along with object 7 increases. However, recognition is not affected up to 10% noise in the input. The results show that whenever an overlapped pattern is presented, the genuine objects are always recognized and sometimes extra objects are recognized depending on the noise level and the nature of the patterns. But the results presented here deal with patterns having a very high overlap. In almost all practical situations, the patterns are expected to have much less overlap, and such problems will seldom arise.

In the second set of experiments, the network is presented with four different synthetic patterns representing four different objects (as shown in Fig. 2). The objects are learned by the network in supervised mode treating each pixel as a separate feature. The objects could have been learned by extracting out the different local features in the objects, but the objective of this present work is to show the capability of the network to learn and recognize multiple objects simultaneously under a noisy environment. The normalized gray level of each pixel is taken as the confidence of the corresponding feature. The weights of the bottom-up and top-down links are set up according to the pixel values in the images. After learning trials are over, the images are superimposed and injected with noise to produce the test patterns for recognition. The different test patterns are shown in Figs. 3–6. Each of the images of different objects consists of 2500 pixels. In the learning, the network is iterated for 100 trials with the time step equal to 0.005 and $\gamma = 5$. Here it is noteworthy that the value of $\gamma$ is much higher than that chosen in the previous experiment. This is due to the fact that the number of features is much higher in this case. Again since the normalized weights of the bottom-up links are much less than that in the previous case (because number of features are much higher) the learning process converges quickly and only 100 learning trials are necessary. The network is executed in recognition mode with the self-feedback equal to 0.1, and the time step = 0.005. The results after 25 iterations with different test patterns and

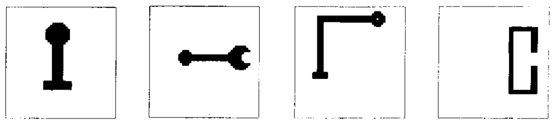


Fig. 2. Four different objects presented at the input.

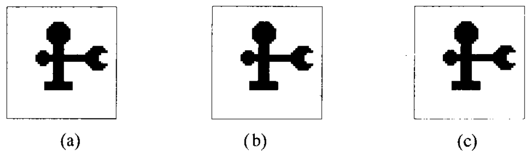


Fig. 3. Superposed pattern of objects 1 and 2 (a) with 10% noise, (b) with 15% noise, (c) with 20% noise.

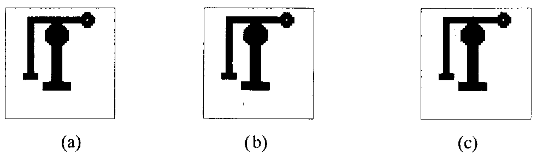


Fig. 4. Superposed pattern of objects 1 and 3 (a) with 10% noise, (b) with 15% noise, (c) with 20% noise.

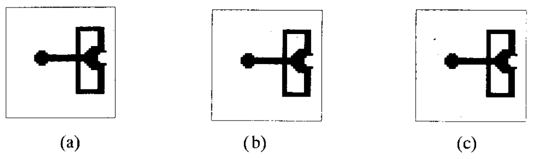


Fig. 5. Superposed pattern of objects 2 and 4 (a) with 10% noise, (b) with 15% noise, (c) with 20% noise.

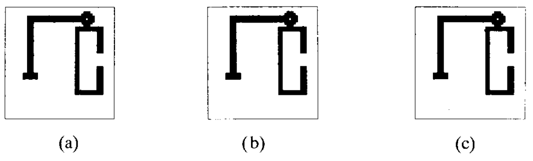


Fig. 6. Superposed pattern of objects 3 and 4 (a) with 10% noise, (b) with 15% noise, (c) with 20% noise.

TABLE VI
RESPONSE AT THE OUTPUT LAYER WITH THE INPUT PATTERN PRESENTED SHOWN IN FIG. 3

| | response at respective output nodes | | |
|---|---|---|---|
| objects | n.l. = 0.1 | n.l. = 0.15 | n.l. = 0.2 |
| object 1 | 0.89 | 0.88 | 0.87 |
| object 2 | 0.91 | 0.90 | 0.88 |
| object 3 | 0.02 | 0.04 | 0.05 |
| object 4 | 0.10 | 0.12 | 0.13 |

TABLE VII
RESPONSES AT THE OUTPUT NODES WITH THE INPUT PATTERN SHOWN IN FIG. 4

| | response at respective output nodes | | |
|---|---|---|---|
| objects | n.l. = 0.1 | n.l. = 0.15 | n.l. = 0.2 |
| object 1 | 0.89 | 0.88 | 0.87 |
| object 2 | 0.09 | 0.10 | 0.10 |
| object 3 | 0.95 | 0.94 | 0.93 |
| object 4 | 0.02 | 0.04 | 0.05 |

TABLE VIII
RESPONSES AT THE OUTPUT NODES WITH THE INPUT PATTERN SHOWN IN FIG. 5

| | response at respective output nodes | | |
|---|---|---|---|
| objects | n.l. = 0.1 | n.l. = 0.15 | n.l. = 0.2 |
| object 1 | 0.07 | 0.08 | 0.10 |
| object 2 | 0.91 | 0.90 | 0.88 |
| object 3 | 0.02 | 0.04 | 0.05 |
| object 4 | 0.96 | 0.94 | 0.93 |

TABLE IX
RESPONSES AT THE OUTPUT NODES WITH THE INPUT PATTERN SHOWN IN FIG. 6

| | response at respective output nodes | | |
|---|---|---|---|
| objects | n.l. = 0.1 | n.l. = 0.15 | n.l. = 0.2 |
| object 1 | 0.02 | 0.04 | 0.05 |
| object 2 | 0.12 | 0.13 | 0.14 |
| object 3 | 0.95 | 0.94 | 0.93 |
| object 4 | 0.96 | 0.94 | 0.93 |

different noise levels are presented in Tables VI–IX. From the results it is clear that as the noise increases the output of the "true" objects decreases, and that of the "false" objects increases.

The network can be applied to actual object recognition, where the local features or primitives would form the input feature vector to the network.

## VI. DISCUSSION

A model for learning and recognizing the objects (viewed as feature vectors) and the corresponding connectionist implementation has been presented so far. In the derivation of the learning rules some measures for the weights of the links are derived from the optimization function, and correspondingly the learning rules are derived. Although the derivation of the learning rules does not assume any explicit biological functionality, ultimately it shows that the learning rules support the Hebbian rule [16] and the associative decay phenomenon [2]. The strong points of the present work lie in two parts. Firstly, it gives an explicit formulation and the corresponding network structure for recognizing multiple objects simultaneously which is mostly guided by the memory based reasoning [6]. Secondly, it derives the learning rules by defining explicit measures for the weights, which really draws a connecting link between the probabilistic measures and the connectionist learning paradigms.

The present network model is comparable to other existing network models and superior to some of them. If the agility factors of the nodes always happen to be unity, then the learning rules basically reduce to the learning rules in the Carpenter/Grossberg's model operated in supervised mode. In the present model, since the probabilities of appearances of the objects are considered, this is supposed to be stronger in learning the object classes adaptively. The comparisons with other network models are already presented in the introduction.

In Section III it has been noted that the network is capable of learning the object classes in supervised mode. The unsupervised mode of learning can also be implemented in this network model. During unsupervised mode also, as discussed in supervised mode, it is considered that a single object is present in the scene. Then in a noiseless condition, the output node corresponding to the presented object will have maximum output (equal to unity) in the stable condition, and all other outputs will be zero. On the other hand, if the presented input does not match with any exemplar pattern of the objects stored in the network then outputs of all the output nodes will be much less than unity. Even under a noisy condition, if a single object is present then the output of the node representing that object will be much higher than the other output nodes, although it completely depends on the amount of noise present. If the maximum output in the output layer is found to be greater than a certain threshold (can be referred as *vigilant threshold*) then the input pattern can be considered to represent the object corresponding to maximum output. On the other hand, if the maximum output is less than the vigilant threshold, the input pattern can be considered to represent a new object and a completely new output node can be allocated for the pattern.

The model is also extendible to multiple layers. In the development of the model the features are considered to be independent. But in practical problems, like object recognition, a group of features occur together in different objects. The group of features can be found out and considered as a macrofeature or subpart of the different objects. Again for objects having movable parts (i.e., nonrigid objects) these subparts can be found by extracting out the common subset of features in different instances of the same object. The macrofeatures or subparts can appear in a larger group of features and thus form a hierarchy. In the formation of the network if an intermediate node be allocated to each macrofeature, a multi-layered hierarchical network model will be formed for learning and recognizing the objects. Presently we are involved in the extension of the model to the hierarchical model with unsupervised category formation capability.

## References

[1] J. A. Anderson, J. W. Silverstein, S. R. Ritz, and R. S. Jones "Distinctive features, categorical perception, and probability learning," *Psych. Rev.*, vol. 84, pp. 413–451, 1977.

[2] G. A. Carpenter and S. Grossberg, "A massively parallel architecture for a self-organizing neural pattern recognition machine," *Computer Vision, Graphics and Image Processing*, vol. 37, pp. 34–115, 1987.

[3] J. A. Feldman, "Dynamic connections in neural networks," *Biol. Cybern.*, vol. 46, pp. 27–39, 1982.

[4] J. A. Feldman and D. H. Ballard, "Connectionist models and their properties," *Cognitive Science*, vol. 6, pp. 205–254, 1982.

[5] R. B. Ash, *Real Analysis and Probability.* London, U.K.: Academic Press, 1972.

[6] C. Stanfill and D. Waltz, "Toward memory-based reasoning," *Commun. ACM*, vol. 29, pp. 1213–1228, 1986.

[7] S. E. Fahlmann and G. E. Hinton, "Connectionist architecture for artificial intelligence," *IEEE Computer*, vol. 20, pp. 100–109, 1987.

[8] D. E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing: Explorations in Microstructures of Cognition (Ed.), Vol. I.* Cambridge, MA: Bradford Books/MIT Press, 1986.

[9] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychol. Rev.*, vol. 65, pp. 386–408, 1958.

[10] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A learning algorithm for Bolzmann machines," *Cognitive Science*, vol. 9, pp. 147–169, 1985.

[11] S. I. Amari, "Neural theory of association and concept formation," *Biol. Cybern.*, vol. 26, pp. 175–185, 1977.

[12] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," in *Proc. Nat. Academy Sciences*, 1982, pp. 2554–2558.

[13] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two state neurons," in *Proc. Nat. Academy Sciences*, 1984, pp. 3088–3092.

[14] J. J. Hopfield and D. W. Tank, "Neural computation of decision in optimization problems," *Biol. Cybern.*, vol. 52, pp. 141–152, 1985.

[15] K. Fukushima, "Neural network model for selective attention in visual pattern recognition and associative recall," *Appl. Opt.*, vol. 26, pp. 4985–4992, 1987.

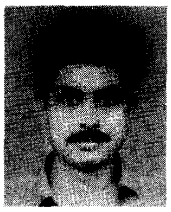
[16] D. O. Hebb, *The Organization of Behavior.* New York: Wiley, 1949.

[17] R. P. Lippmann, "An introduction to computing with neural nets," *IEEE Acout., Speech, Signal Processing*, vol. 4, pp. 4–22, 1987.

[18] R. S. Zemel, M. C. Mozer, and G. E. Hinton, "TRAFFIC: A model of object recognition based on transformation of feature instances," Tech. Rep. CRG-TR-7, University of Toronto, Toronto, 1988.

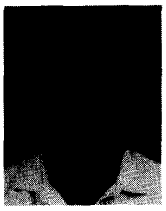
[19] T. Kohonen, *Self-Organization and Associative Memory.* Berlin, Germany: Springer-Verlag, 1988.

[20] Y. Peng and J. A. Reggia, "A connectionist model for diagnostic problem solving," *IEEE Syst., Man, Cybern.*, vol. 19, pp. 285–298, 1989.



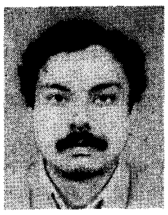
**Jayanta Basak** was born on September, 1965. He completed his undergraduate course in electronics and telecommunication engineering from Jadavpur University in 1987. He received the M.E. degree in computer science and engineering from Indian Institute of Science (IISc), Bangalore in 1989.

He worked as a Computer Engineer in the FGCS/KBCS project at the Indian Statistical Institute, Calcutta from 1989 to April 1992. He joined as a programmer in the same Institute in April 1992. His current research interests are neural networks and Computer Vision.



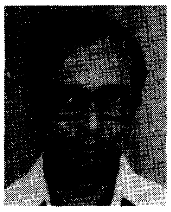
**C.A. Murthy** was born in Ongole, India in 1958. He received the B. Stat (Hons), the M.Stat, and the Ph.D. degree from the Indian Statistical Institute (ISI) Calcutta.

He worked in the National Centre for Knowledge Based Computing. Currently he is employed as a lecturer in the Electronics and Communication Science Unit (ECSU), ISI. His fields of interest include pattern recognition, cluster analysis, computer vision, fuzzy sets, and neural networks.



**Santanu Chaudhury** (M'91) received the B.Tech degree in electronics and electrical communication engineering in 1984 and the Ph.D. degree in computer science and engineering in 1989 from the Indian Institute of Technology, Kharagpur, India. From 1989–1992 he was with the Nodal Centre for Knowledge Based Computing, ECSU, ISI, Calcutta. From 1990 to 1991 a faculty member in the Department of Electronics and Electrical Communication Engineering I.I.T, Kharagpur.

Currently, he is as assistant professor in the Department of Electrical Engineering, I.I.T, Delhi, India. His current research interests are connectionist networks for computer vision problems, document image understanding, image interpretation, and object recognition.



**Dwijesh Dutta Majumder** received the M.Sc(Tech.) from Calcutta University (C.U.). In 1962 he received the Ph.D. degree in memory technology from the same university. In 1955 he joined Electronic Computer Division of ISI. From 1972 until 1992 he was head ECBU in ISI. He has published more than 300 research papers and six books in the fields of memory technology, speech and other pattern recognition, image analysis, computer vision, artificial intelligence, numeral modeling, cybernetics, and knowledge based computing.

Dr. Majumder is chairman of National Centre for Knowledge Based Computing, Professor Emeritus of ISI, Distinguished HCL Professor of IIT, and Emeritus Scientist of CSIR. He is a fellow of INSA, INAE, I.A.SC, IETE, and CSI and is a member of the Governing Boards of International Association of Pattern Recognition, the World Organization of Cybernetics Systems and the International Fuzzy Systems Association.