3. Explain Competitive Learning with some suitable example.

Or

Differentiate between Supervised and Unsupervised learning.

Unit-III

4. Explain Backpropagation algorithm?

Or

Explain the XOR problem.

Unit-IV

5. Explain Phonetic typewriter.

Or

Explain handwritten digital recognition?

Unit-V

6. Explain Fuzzy Associative Memories.

Or

Explain Fuzzy Truck Backer Upper Control System? Give example of Fuzzy Logic.

Printed Pages – 4 Roll No. : .....

AS-4230

B. Tech. (Seventh Semester) Examination, 2013

(Computer Science and Engg. Branch)

SOFT COMPUTING

Time Allowed : Three hours

Maximum Marks : 60

Note : *Attempt questions of all sections as directed.*

Section-'A'

(Objective Type Questions) **10×2=20**

Note : *Attempt all questions. Each question carries 2 marks.*

1. (i) ..... stable states may also arise when there is some delay in the feedback of the outputs from other

processing units to the current unit, even, though the weights are exactly symmetric.

(ii) In a few cases it is possible to predict the global pattern behaviour, if it is possible to show the existence of an energy function called ..... .

(iii) The type of learning is determined by the manner in which the ..... changes takes place.

(iv) A prescribed set of well defined rules for the solution of a learning problem is called a ..... .

(v) ..... learning is used for pattern association required in pattern recognition.

(vi) In case of ..... learning every input vector has a corresponding desired output.

(vii) One of the most successful applications of neural network principles is in solving optimization problems. There are many situations where a problem can be formulated as ..... or maximization of some cost function or objective function subject to certain constraints.

(viii) It is possible to map optimization problem onto a feedback network, where the units and connections

strengths are identified by comparing the cost function of the problem with the ..... function of the network expressed in terms of the state values of the units and the connection strengths.

(ix) Neural Networks became very popular because of the ability of ..... neural network to form complex decision regions in the pattern space for classification.

(x) Many pattern recognition problems, especially character or other symbol recognition and vowel recognition, have been implemented using a ..... neural network.

Section-'B' **5×8=40**

*Note : Solve any **one** questions from each unit. Each question carries 8 marks.*

Unit-I

2. Design AND Gate using ANN.

Or

Explain feedback from neuron to ANS.

Unit-II

Friday 2009

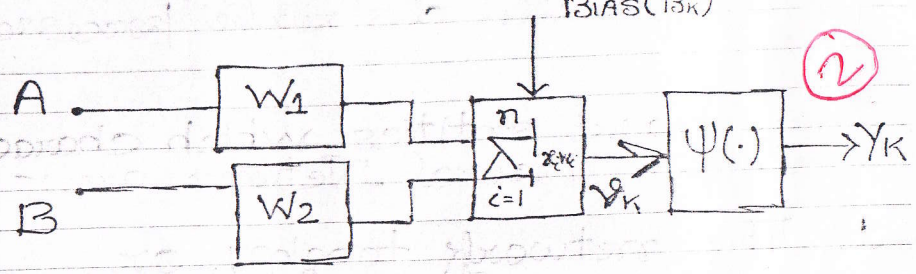
Question:- Design AND Gate using ANN

Construction of Logic gates using the Neural Network framework:-

(a) AND Gate

A.B ← LOGIC for AND Gate

where A and B are Input values



(2)

$$x_K = \left(\sum_{i=1}^{n} x_i w_i + B_K\right) = u_K + B_K \quad (1)$$

$$\left[\because u_K = \sum_{i=1}^{n} x_i w_i\right]$$

So, from the Neural Network framework we are able to analyse that there are four parameters which are required for generating output Y_K i.e. A, B, W_1 and W_2.

Therefore, the truth table for the AND Gate will be like the following table

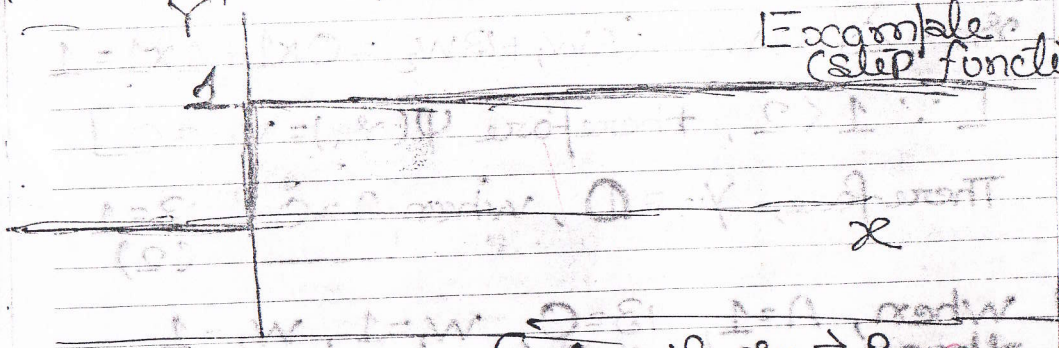
Saturday February 2009 28

TRUTH TABLE (AND GATE)

A	B	$Y_K = A.B$
0	0	0
0	1	0
1	0	0
1	1	1

Taking $W_1 = W_2 = 1$ (1)

In this case we will be using threshold function (step function) as the activation function. In threshold function if the value of local induced field is above a certain value than only an output of "1" is produced otherwise output of "0" is produced.



$$\psi(x_K) = \begin{cases} 1 & \text{if } x_K \geq 2 \\ 0 & \text{if } x_K < 2 \end{cases} \quad (2)$$

[Take threshold value be > 2]

2 Monday March 2009

Now, we will consider the four input patterns

a) When, $A=0$, $B=0$, $W_1=1$, $W_2=1$ (2)

$y_K = \sum_i x_i w_i = 0\times1 + 0\times1 = (AW_1 + BW_2) = 0$

[$\because 0<2$ therefore $\psi(y_K) = y_K = 0$]

∴ Therefore, $y_K = 0$ when, $A=0$ & $B=0$ ——— (1)

b) When, $A=0$, $B=1$, $W_1=1$, $W_2=1$ then,

$y_K = \sum_i x_i w_i = (AW_1 + BW_2) = 0\times1 + 1\times1 = 1$

[$\because 1<2$, therefore $\psi(y_K) = y_K = 0$]

Therefore, $y_K = 0$, when $A=0$ & $B=1$ ——— (2)

c) When, $A=1$, $B=0$, $W_1=1$, $W_2=1$ then,

$y_K = \sum_i x_i w_i = (AW_1 + BW_2) = 1\times1 + 0\times1 = 1$

[$\because 1<2$, therefore, $\psi(y_K) = y_K = 0$]

Therefore, $y_K = 0$, when $A=1$ & B ——— (3)

d) When, $A=1$, $B=1$, $W_1=1$, W_2 then,

$y_K = \sum_i x_i w_i = (AW_1 + BW_2)$

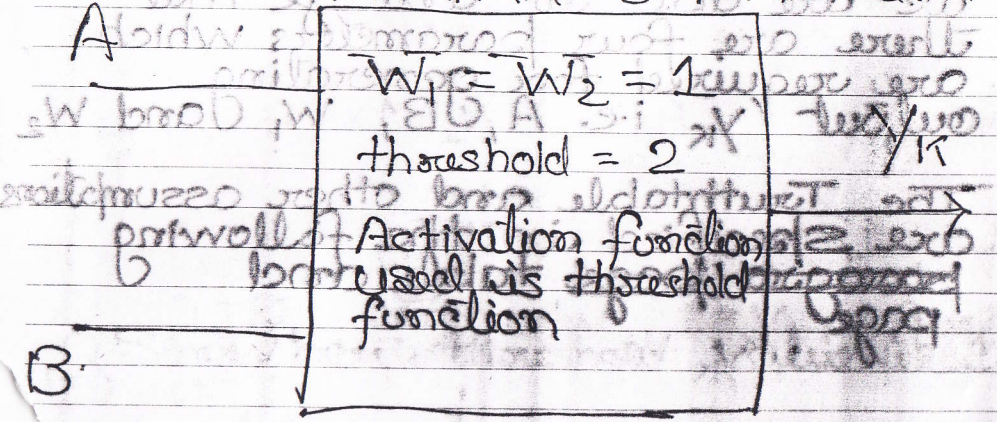
$= (1\times1 + 1\times1) = (1+1) = 2$

[$\because 2=2$, therefore, $\psi(y_K) = y_K = 1$]

Therefore, when $A=1$ & $B=1$ in that case $y_K = 1$ ——— (4)

From (1), (2), (3) & (4) we are able to conclude that when $W_1 = W_2 = 1$ and threshold value set to 2 [utilizing threshold function as the step function] we are able to construct an AND Gate utilizing the Neural Network framework.

ANN (as AND GATE)



Explain feedback from neuron to ANS

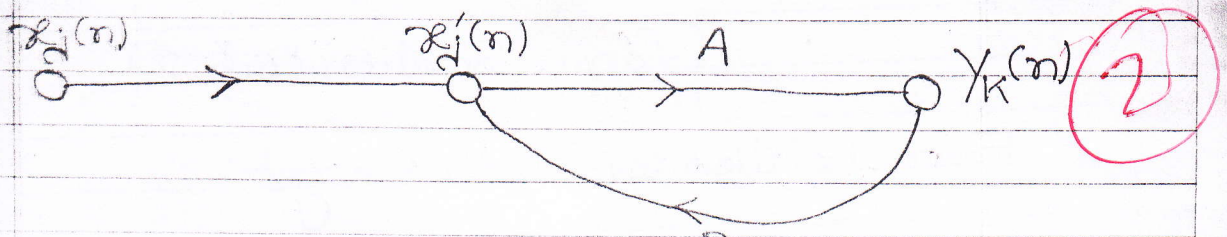
~~LECTURE 4~~

BRICE
Page No.
Date: / /

Feedback

Def:-

Feedback is said to exist in a dynamic system whenever the output of an element in the system influences in part the input applied to that particular element, thereby giving rise to one or more closed paths for the transmission of signals around the system (1)



(2)

Figure :- Signal flow Graph of a single loop feedback system

NOTE :-

It plays a major role in the study of a special class of neural networks known as Recurrent Networks

Here,

$x_j(n)$ = Input Signal

$x'_j(n)$ = Internal Signal

$Y_K(n)$ = Output Signal

From the figure we get relation

$$Y_K(n) = A[x_j'(n)] \quad \ldots\ldots\ldots (i)$$

(fanin Relation Rule 2)

$$x_j'(n) = B[Y_K(n)] + x_j(n) \quad \ldots\ldots (ii)$$

(fanin Relation Rule 2)

From Relation (i) & (ii) we get

$$x_j'(n) = \frac{Y_K(n)}{A} \quad \ldots\ldots\ldots\ldots (I)$$

$$\frac{Y_K(n)}{A} = B[Y_K(n)] + x_j(n)$$

$$Y_K(n)\left(\frac{1}{A} - B\right) = x_j(n)$$

$$\frac{Y_K(n)}{A}(1 - AB) = x_j(n)$$

$$\boxed{Y_K(n) = \frac{A}{1-AB}\, x_j(n)}$$

We refer to $\frac{A}{1-AB}$ as the closed-loop operator of the system & AB as the open-loop operator. In general, the open-loop operator is non-commutative in that $BA \neq AB$

here $y_k \neq x_j'(n)$

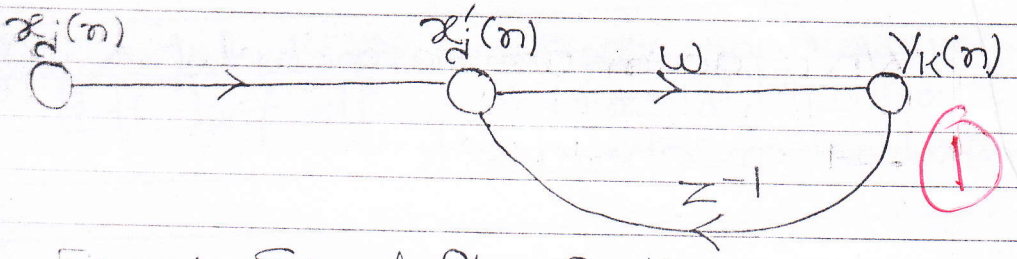


Figure:- Signal flow Graph

Here, A is represented by fixed weight W
B is unit delay operator, z^{-1}

Whose output is delayed with respect to the input by one time unit.

We may express the closed loop operator of the system as

$$\frac{A}{1-AB} = \frac{W}{1-Wz^{-1}}$$

$$= W(1-Wz^{-1})^{-1}$$

Using Binomial expansion

$$(1-x)^{-1} = (1+x+x^2 \ldots\ldots\ldots)$$

We may write the closed loop operator of the system as

$$\frac{A}{1-AB} = \omega \sum_{l=0}^{\infty} w^l z^{-l}$$

$$y_k(n) = w \sum^{l} w^l z^{-l} [x_j(n)]$$

Where again we have included square brackets to emphasize the fact that z^{-l} is an operator

$$z^{-l}[x_j(n)] = x_j(n-l)$$

$$\therefore y_k(n) = \sum_{l=0}^{\infty} w^{l+1} x_j(n-l) \quad (2)$$

The dynamic behavior of the system is controlled by the weight w. In particular, we may distinguish two specific cases :-

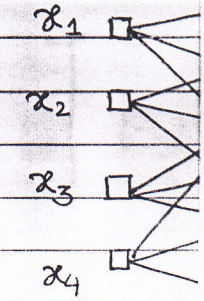
(1) $|w| < 1$, for which the output signal $y_k(n)$ is exponentially convergent, that is the system is stable

(2) $|w| \geq 1$, for which the o/p signal $y_k(n)$ divergent, that is system is unstable. If $|w| = 1$ the divergence is linear and if $|w| > 1$ the divergence is exponential.

Explain Competitive Learning with some suitable example

~~COMPETITIVE LEARNING~~

COMPETITIVE LEARNING



Layer of Source nodes

In competitive learning, as the name implies, the output neurons of a neural network compete among themselves to become active (fired). In competitive learning only a single output neuron is active at any one time.

NOTE :-
It is because of the feature of competitive learning that only one single neuron is active at any one time that makes competitive learning highly suited to discover statistically salient features that may be used to classify a set of input patterns.

Rumelhart and Zipser, 1985 proposed ~~the~~ Three basic elements of a competitive learning rule :-

1. A set of neurons that are all the same except for some randomly distributed synaptic weights, and which therefore respond differently to a given set of input patterns
2. A limit imposed on strength of each neuron
3. A mechanism that permits the neurons to compete for the right to respond to a given subset of inputs, such that only one output neuron, or only one neuron per group, is active at a time. The neuron that wins the competition is called "winner-takes-all neuron"

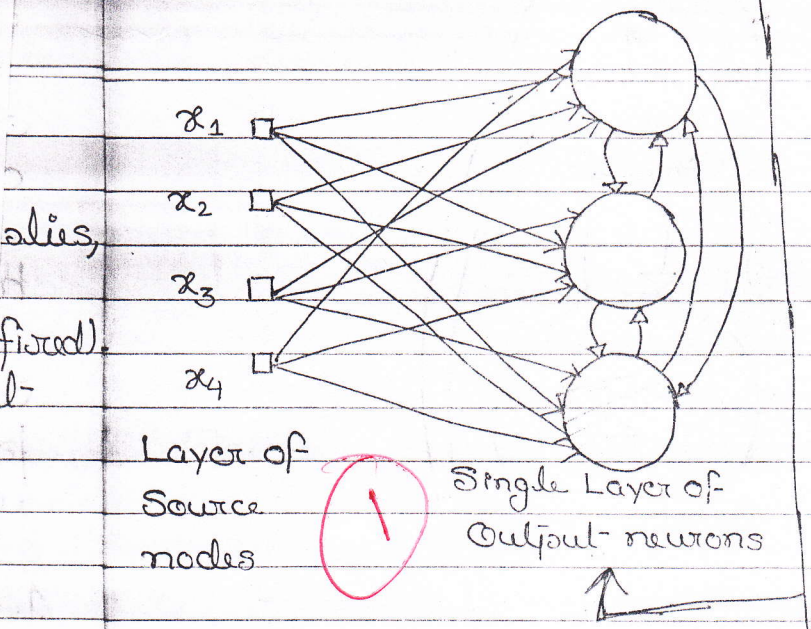


FIGURE :-
Architectural graph of a simple competitive learning network with feedforward (excitatory) connection from the source nodes to the neurons, and lateral (inhibitory) connections among the neurons; the lateral connections are signified by open arrows.

For a neuron k to be the winning neuron, its induced local field v_k for a specified input pattern x must be the largest among all the neurons in the network. The output signal y_k of winning neuron k is set equal to one; the output signals of all the neurons that lose the competition are set equal to zero. We thus write

$$y_k = \begin{cases} 1 & \text{if } v_k > v_j \text{ for all } j \neq k \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

Note: different inputs

$$\sum_j w_{kj} = 1 \quad (1)$$

$$\Delta w_{kj} = \begin{cases} \eta (x_i - w_{kj}) & \text{if neuron } k \text{ wins the competition} \\ 0 & \text{if neuron } k \text{ loses the competition} \end{cases} \quad (1)$$

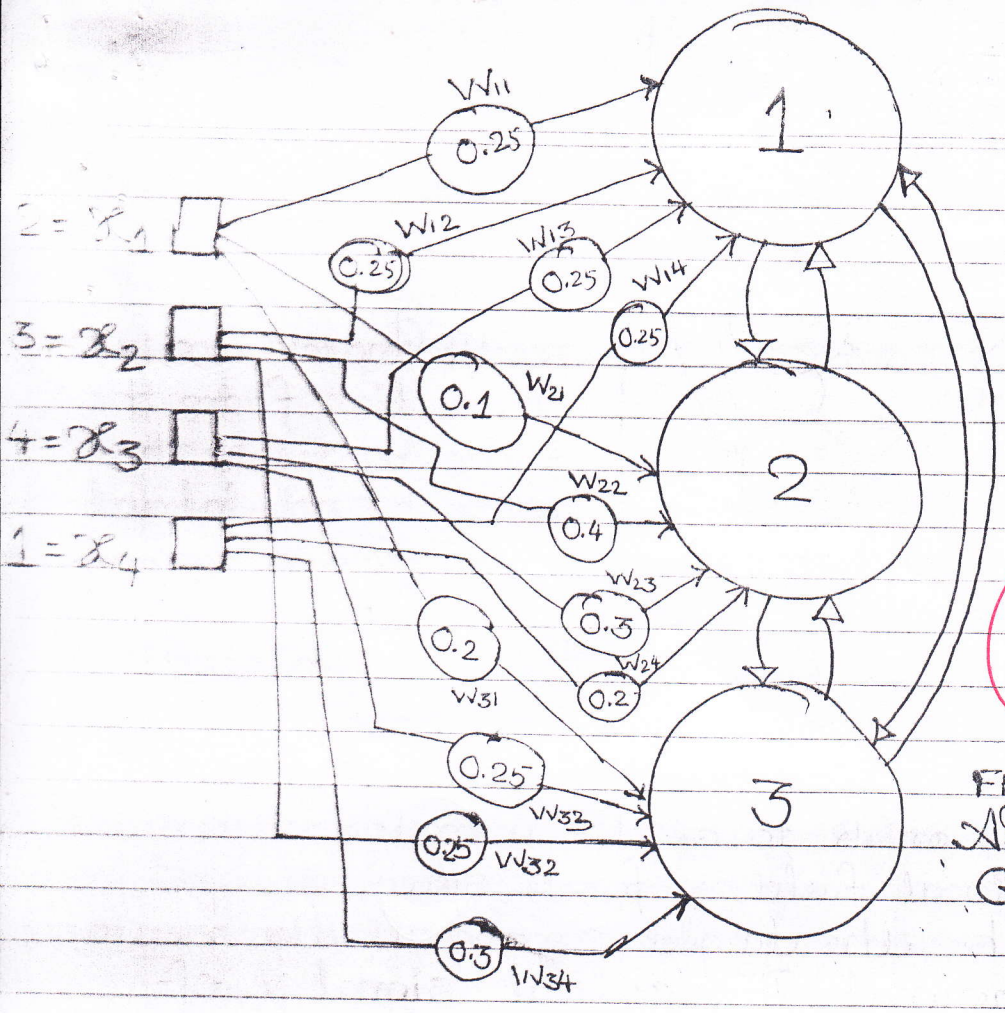


Figure :- Architectural Graph

$(2 \times 0.25 + 3 \times 0.25 + 4 \times 0.25 + 1 \times 0.25) = y_1 = 2.50$

$(2 \times 0.1 + 3 \times 0.4 + 4 \times 0.3 + 1 \times 0.2) = y_2 = 2.80$

$(2 \times 0.2 + 3 \times 0.25 + 4 \times 0.25 + 1 \times 0.3) = y_3 = 2.45$

Here,

$y_2 > y_1 > y_3$ (Therefore output signal of only neuron 2 will be active and rest will be deactivated)

A3) Comparison of Supervised and Unsupervised Learning

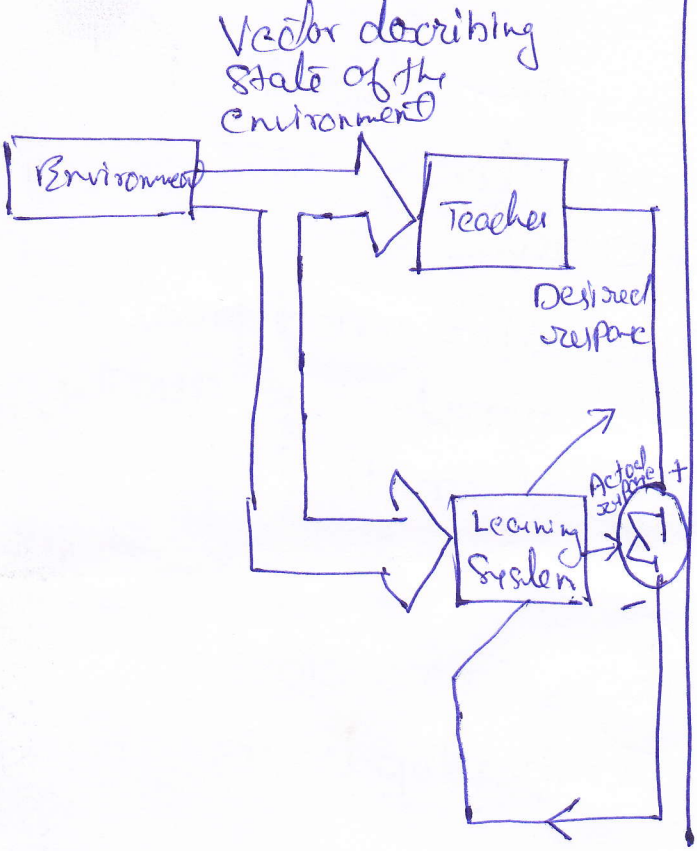
Supervised Learning	Unsupervised Learning
1 Learning associated with a teacher	1. There is no teacher.
2 Resulting error can be used to correct parameters for improving network	2. Rules of adaptation are used for improvements
3 Learning algorithm is accurate	3. Learning algorithm is comparatively less accurate
4. This refer to "search" in weight space along error gradients	4. This refer modifications of parameters is quasi biological method
5. There is user class membership concept	5. There is no class membership basis.

Quasi :- अर्थात्, मानो, प्रायः
as if, almost, that is to say

Supervised Learning

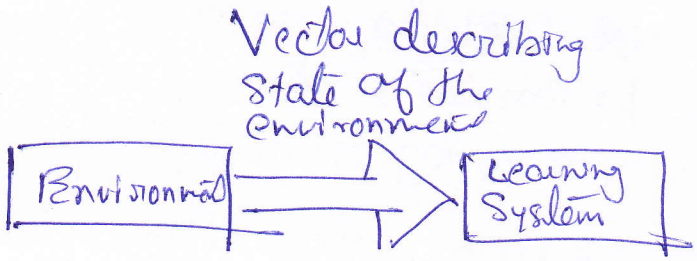
Application

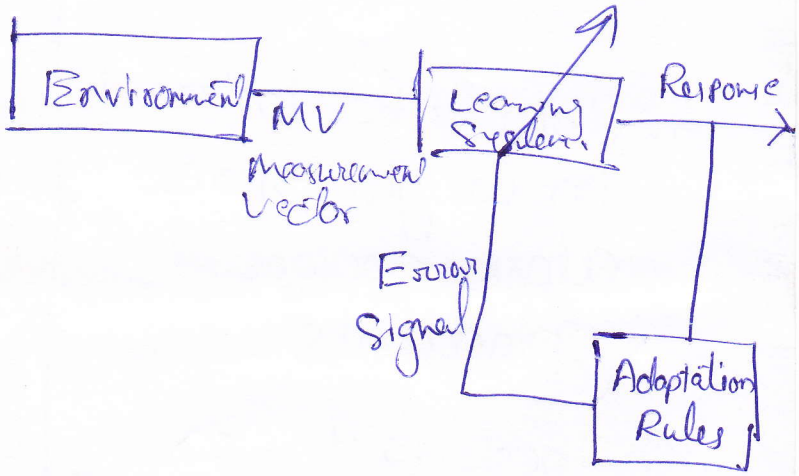
6) Supervised learning is used for pattern association required in pattern recognition



Unsupervised Learning

6) SOM





Answer 4

BACKPROPAGATION

Training the backpropagation network requires the steps which follows

Step 1

Select the next training pair from the training set ; apply the input vector to the network input.

Step 2

Calculate the output of the network

Step 3

Calculate the error between the network output and the desired output (the target vector from the training pair)

Step 4

Adjust the weights of the network in a way that minimizes the error

Step 5

Repeat Steps 1 through 4 for each vector in the training set until the error for the entire set is acceptably low.

The instantaneous error energy $\mathcal{E}(n)$ and therefore the average error energy $\mathcal{E}_{av}$ is a function of all the free parameters (i.e. synaptic weights and bias levels) of the network.

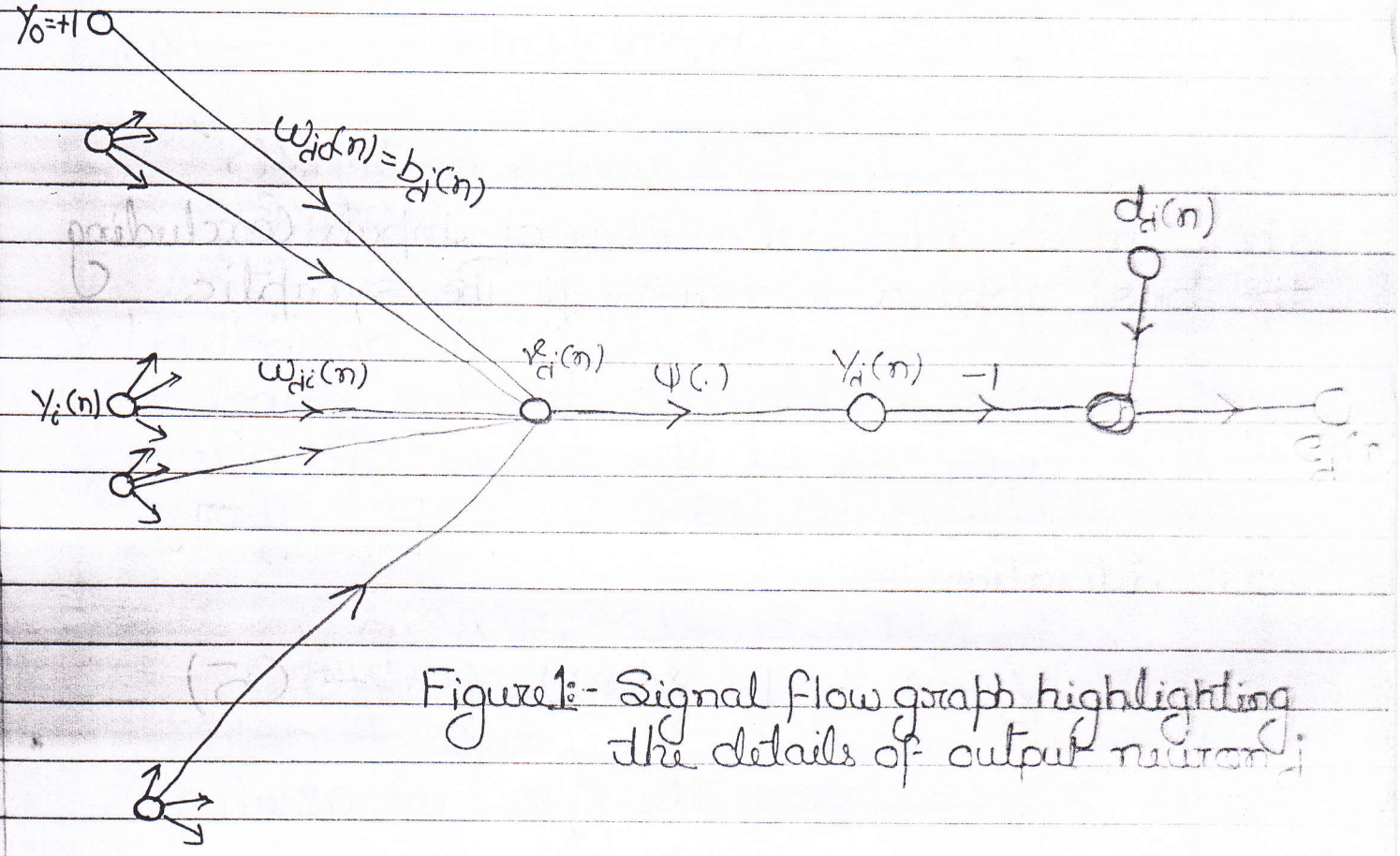


Figure 1:- Signal flow graph highlighting the details of output neuron j

The arithmetic average of these individual weight changes over the training set is therefore an estimate of the true change that would result from modifying the weights based on minimizing the cost function $\mathcal{E}_{av}$ over the entire training set. We will address the quality of the estimate later in this section.

Consider then Figure 1 which depicts neuron j being fed by a set of function signals produced by a layer of neurons to its left. The induced local field $v_j(n)$ produced at the input of the activation function associated with neuron j is therefore

$$v_j(n) = \sum_{i=0}^{m} w_{ji}(n)\, y_i(n) \quad \text{---------- (4)}$$

where m is the total number of inputs (excluding the bias) applied to neuron j. The synaptic weights w_{j0} (corresponding to the fixed input $y_0 = +1$) equals the bias b_j applied to neuron j. Hence the function signal $y_j(n)$ appearing at the output of neuron j at iteration n is

$$y_j(n) = \varphi_j(v_j(n)) \quad \text{---------- (5)}$$

According to the chain rule of calculus we may express the gradient as

$$\frac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)} = \frac{\partial \mathcal{E}(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_{ji}(n)} \quad \text{---------- (6)}$$

Differentiating both sides of equation 2 with respect to $e_j(n)$ we get

$$\frac{\partial \mathcal{E}(n)}{\partial e_j(n)} = e_j(n) \quad \text{---------- (7)}$$

Differentiating both sides of Eq(1) w.r.t $y_j(n)$ we get

$$\frac{\partial e_j(n)}{\partial y_j(n)} = -1 \quad \text{---------- (8)}$$

Differentiating Eq(5) w.r.t. $v_j(n)$ we get

$$\frac{\partial y_j(n)}{\partial v_j(n)} = \varphi_j'(v_j(n)) \quad \text{---------- (9)}$$

Where the use of prime signifies the differentiation with respect to the argument

Finally differentiating Eq(4) w.r.t $w_{ji}(n)$ yield

$$\frac{\partial v_j(n)}{\partial w_{ji}(n)} = y_i(n) \quad \text{---------- (10)}$$

The use of equations (6) (7) (8) (9) & (10) yields

$$\frac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)} = -e_j(n)\varphi_j'(v_j(n))y_i(n) \quad \text{---------- (11)}$$

The correction $\Delta w_{ji}(n)$ applied to $w_{ji}(n)$ is defined by the delta rule

$$\Delta w_{ji}(n) = -\eta \frac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)} \quad \text{---------- (12)}$$

where η is the learning rate parameter of the back propagation algorithm. The use of the minus sign in eqn 12 accounts for gradient descent in weight space (i.e. seeking a direction for weight change that reduces the value of $\mathcal{E}(n)$. Accordingly, the use of Eq (11 and (12) yields

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n) \quad \cdots (13)$$

where the local gradient $\delta_j(n)$ is defined by

$$\delta_j(n) = -\frac{\partial \mathcal{E}(n)}{\partial v_j(n)}$$

$$= -\frac{\partial \mathcal{E}(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \quad \cdots (14)$$

$$= e_j(n) \varphi'(v_j(n))$$

The local gradient points to required changes in synaptic weights. According to Eq (14) the local gradient $\delta_j(n)$ for output neuron j is equal to the product of the corresponding error signal $e_j(n)$ for that neuron and the derivative $\varphi'(v_j(n))$ of the associated activation function.

Explain the XOR problem

XOR PROBLEM

In the elementary (single-layer) perceptron there are no hidden neurons. Consequently, it cannot classify input patterns that are not linearly separable. However nonlinearly separable patterns are of common occurrence. For example, this situation arises in the Exclusive OR (XOR) problem, which may be viewed as a special case of a more general problem, namely that of classifying points in the unit hypercube. Each point in the hypercube is either in class 0 or class 1. However, in the special case of the XOR problem, we need consider only the four corners of the unit square that correspond to the input patterns (0,0), (0,1), (1,1) and (1,0). The first and third input patterns are in class 0 as shown by

$$0 \oplus 0 = 0$$

and

$$1 \oplus 1 = 0$$

where $\oplus$ denotes the Exclusive OR Boolean function operator. The input patterns (0,0) and (1,1) are at opposite corners of the unit square, yet they produce the identical output 0. On the other hand, the input patterns (0,1) and (1,0) are also at opposite corners of the square, but they are in class 1 as shown by

$$0 \oplus 1 = 1$$

and

$$1 \oplus 0 = 1$$

We first recognize that the use of a single neuron with two inputs results in a straight line for a decision boundary in the input space. For all points on one side of this line, the neuron outputs 1; for all points on the other side of the line, it outputs 0. The position and orientation of the line in the input space are determined by the synaptic weights of the neuron connected to the input nodes, and the bias applied to the neuron. With the input patterns (0,0) and (1,1) located on opposite corners of the unit square and likewise for the other two input patterns (0,1) and (1,0), it is clear that we cannot construct a straight line for a decision boundary so that (0,0) and (1,1) lie in one decision region and (0,1) and (1,0) lie in the other decision region. In other words an elementary perceptron cannot solve the XOR problem.

15. Explain Phonetic typewriter

LVQ = Learning Vector Quantization

[study 222, 30 ...

PHONETIC TYPEWRITER

The objective in speech recognition is to transform a given utterance into a sequence of phoneme-like units and convert this sequence into the text corresponding to the spoken utterance

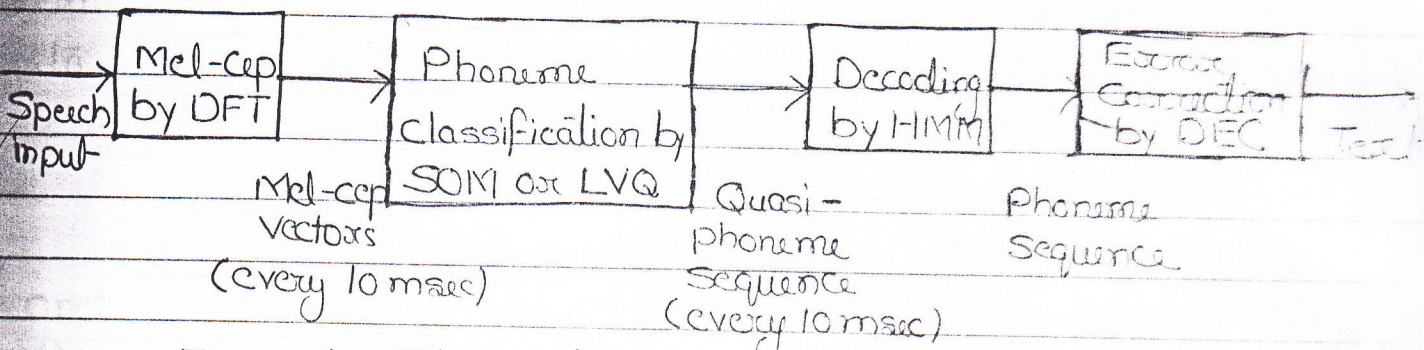


Figure 1 Block diagram of a phonetic typewriter

A block diagram of the phonetic typewriter developed by Kohonen is given in Figure 1. The input speech signal to the phonetic typewriter is processed to obtain a spectral representation using 20 mel-scale cepstral coefficients for every 10 ms segments of data. The sequence of coefficient vectors is given as input to a phoneme classifier, one vector at a time, to obtain the quasi-phoneme sequence as output. The phoneme classification is achieved by using either LVQ or SOM learning. The sequence of phoneme-like units is converted to the phonetic transcription using a multiple codebook Hidden Markov Model (HMM) technique. The errors in the phonetic decoding by the HMM are corrected using the Dynamically Expanding Context (DEC) algorithm and then converted into the text corresponding to the input utterance. The phonetic typewriter was able to produce letter accuracies of 95% for the Finnish language. The alphanumeric

was reported to have worked well for languages whose orthography and phonemic transcriptions have simple correspondence.

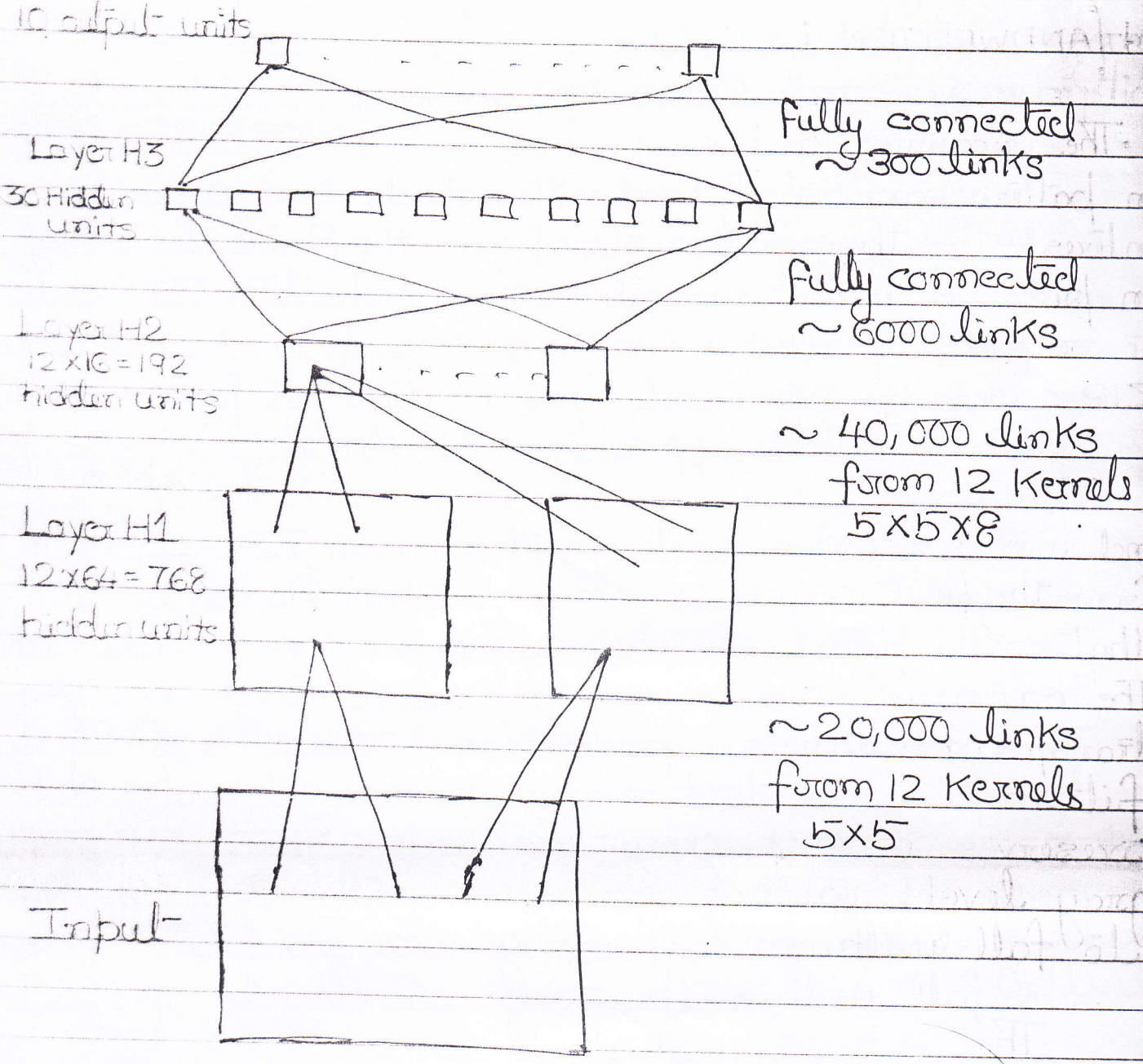
Que Explain handwritten digital recognition

HANDWRITTEN DIGITS RECOGNITION

The recognition of handwritten digits is a classic problem in pattern recognition. Specifically the Postal Service is interested in the recognition of handwritten ZIP codes on pieces of mail. A backprop network has been designed to recognize segmented numerals digitized from handwritten ZIP codes that appeared on U.S. mail.

The neural network was trained on 7291 examples and was tested on 2007 new examples. As can be seen from the figure 1, the data set contained numerous examples that are ambiguous, unclassifiable, or even misclassified. The examples are preprocessed through a simple linear transformation that makes the raw segmented digits fit in a 16X16 gray-level image. The transformation preserves the aspect-ratios of the digits. The gray levels in each image were scaled and translated to fall within the range -1 to +1.

The network consisted of three hidden layers H1, H2 and H3 and an output layer. Layer H1 is connected to the input image, and layer H3 feeds its outputs into the output layer. The output layer has 10 units and uses 1 out-of-10 coding. Layer H3 has 30 units and is fully connected to H2. The output layer is fully connected to H3. On the other hand, "weight-sharing" interconnections are used between the inputs and layer H1 and between layers H1 and H2. The network is represented in figure 1.



Weight sharing refers to having several connections controlled by a single weight. It imposes equality constraints among connection strengths, thus reducing the number of free parameters in the network, which leads to improved generalization. Another motivation for employing weight sharing is to encourage the hidden layers H1 and H2 to develop feature-selection properties that simplify the classification

task ultimately implemented by layer H3 and the output layer. This strategy is crucial for high-accuracy classification performance, since one is dealing here with low-level input images, as opposed to using a relatively small-sized training set consisting of invariant features

The first hidden layer (H1) is composed of "feature maps". All units in a feature map share the same set of weights (except for the threshold, which may differ from unit to unit). There are 12 groups of 8x8 feature maps (64 units per feature map). Each unit in a feature map receives inputs from a 5x5 window of the input image. Two neighboring units in a feature map in H1 have their receptive 5x5 field (in the input layer) two pixels apart. Here, the motivation is that the exact position of a feature need not be determined with high precision. Each of the 64 units in a given feature map performs the same operation on corresponding parts of the image. The function performed by a feature map can thus be interpreted as detecting the presence of 1 of 12 possible microfeatures at arbitrary positions in the input image.

layer H2 is also composed of 12 feature maps, each containing 4x4 units. The connection scheme between layers H1 and H2 is quite similar to the one described above between H1 and the input layer, but with slightly more complicated

due to the multiple 8x8 feature maps in H1. Each unit in H2 receives input from a subset (8 in this case) of the 12 maps in H1. Its receptive field is composed of a 5x5 window centered at identical positions within each of the selected subset maps in H1. Again, all units in a given map in H2 share their weights.

As a result of this structure, the network has 1000 units, 64660 connections and 9760 independent weights. All units use a hyperbolic tangent activation function. Before training, the weights were initialized with a uniform random distribution between −2.4 and +2.4 and further normalized by dividing each weight by the fan-in of the corresponding unit.

Backprop based on the approximate Newton method was employed in an incremental mode. The network was trained for 23 cycles through the training set. The percentage of misclassified patterns was 0.14 percent on the training set and 5.0 percent on the test set. Another performance test was performed employing a rejection criterion, where an input pattern was rejected if the levels of the two most active units in the output layer exceeded a given threshold. For this given rejection threshold, the network classification error on the test-patterns was reduced to 1 percent but resulted in a 12 percent

rejection rate. Additional weight-pruning based on information theoretical ideas on a four-hidden-layer architecture similar to the one described above resulted in a network with only about one-quarter as many free parameters as that described above and improved performance to 99 percent generalization error with a rejection rate of only 9 percent. For comparison, a fully interconnected feedforward neural network with 40 hidden units (10690 free weights) employing no weight sharing, trained on the same task produced a 1.6 percent misclassification on the training set and 19.4 percent rejections for a 1 percent error rate on the test set.

Thus, when dealing with large amounts of low-level information (as opposed to carefully preprocessed feature data), proper constraints should be placed on the network so that the number of free parameters in the network is reduced as much as possible without overly reducing its computational power. Also, incorporating a priori knowledge about the task (such as the architecture for developing translation-invariant features implemented by layers H1 and H2 in the preceding network) can be very helpful in arriving at a practical solution to an otherwise difficult problem.

6. Explain Fuzzy Associative Memories

Fuzzy Associative Memory

We shall focus on fuzzy systems $S: I^n \to I^p$ that map balls of fuzzy sets in I^n to balls of fuzzy sets in I^p. These continuous fuzzy systems behave as associative memories. They map close inputs to close outputs. We shall refer to them as Fuzzy associative memories, or FAMs.

The simplest FAM encodes the FAM rule or association (A_i, B_i) which associates the p-dimensional fuzzy set B_i with the n-dimensional fuzzy set A_i. These minimal FAMs essentially map one ball in I^n to one ball in I^p. They are comparable to simple neural networks. But we need not adaptively train the minimal FAMs. As discussed below, we can directly encode structured knowledge of the form "If traffic is heavy in this direction, then keep the stop light green longer" in a Hebbian-style FAM correlation matrix. In practice we sidestep this large numerical matrix with a virtual representation scheme. In place of the matrix the user encodes the fuzzy-set association as a single linguistic entry in a FAM-bank linguistic matrix.

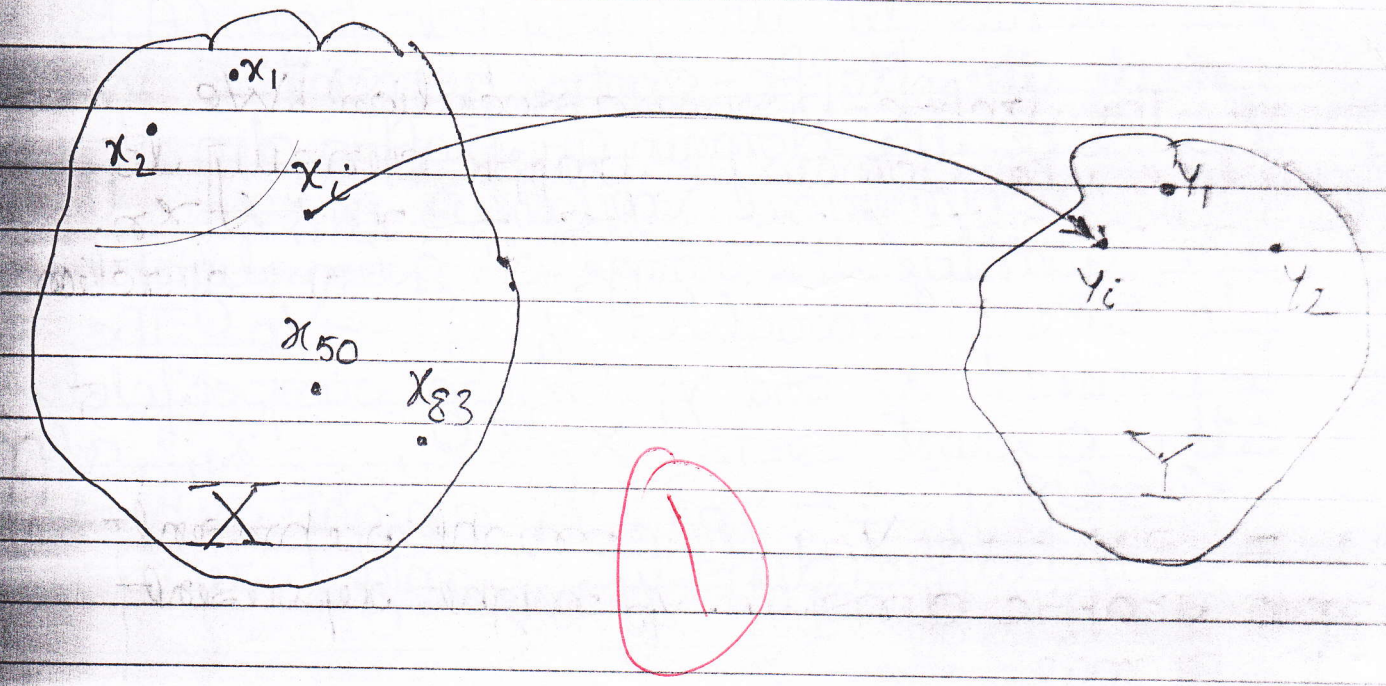
In general a FAM system $F: I^n \to I^p$ encodes and processes in parallel a FAM bank of m FAM rules $(A_1, B_1) \cdots\cdots (A_m, B_m)$. Each input A to the FAM system activates each stored FAM rule to different degree. The minimal FAM that stores (A_i, B_i) maps input A to B_i', a partially activated version of B_i. The more A resembles

output fuzzy set B combines these partially activated fuzzy sets $B'_1 \cdots B'_m$. B equals a weighted average of the partially activated sets:

$$B = w_1 B'_1 + \cdots + w_m B'_m$$

where w_i reflects the credibility, frequency, or strength of the fuzzy association (A_i, B_i). In practice we usually "defuzzify" the output waveform B to a single numerical value y_j in Y by computing the fuzzy centroid of B with respect to the output universe of discourse Y. An adaptive FAM (AFAM) is a time-varying FAM system.

Fuzzy associative memories (FAMs) are transformations. FAMs map fuzzy sets to fuzzy sets. They map unit cubes to unit cubes as in Figure 1. In the simplest case the FAM system consists of a single association. In general the FAM system consists of a bank of different FAM associations. Each association corresponds to a different numerical FAM matrix, or a different entry in a linguistic FAM-bank matrix. We do not combine these matrices as we combine or superimpose neural-network associative-memory (outer product) matrices). We store the matrices separately and access them in parallel.



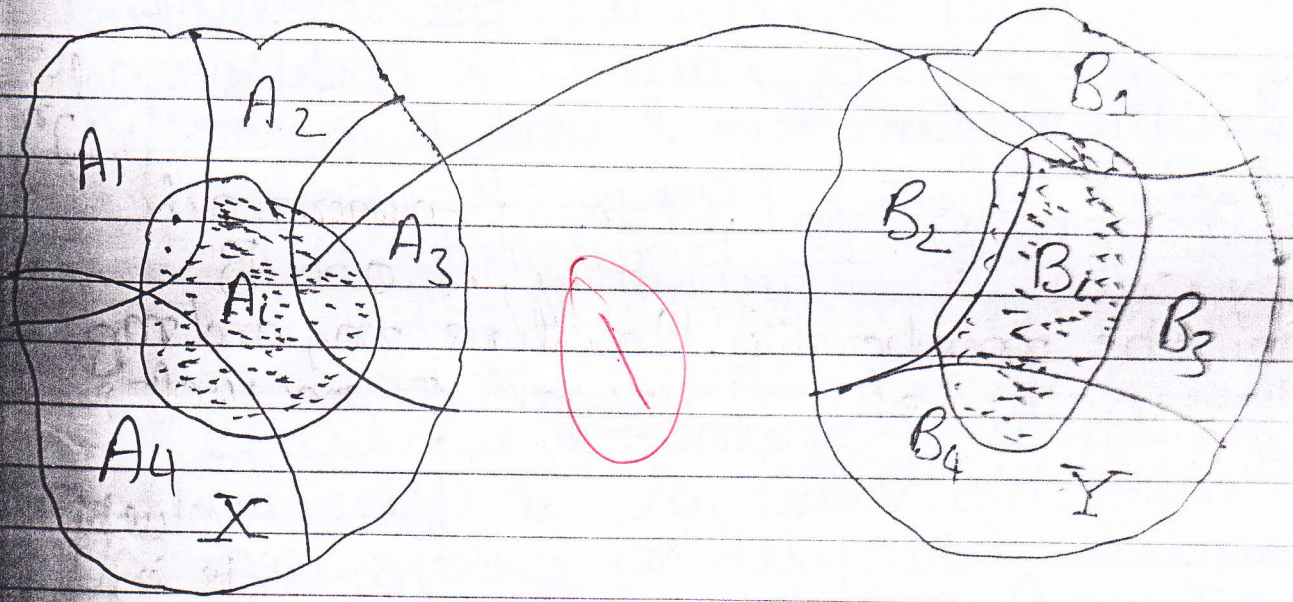
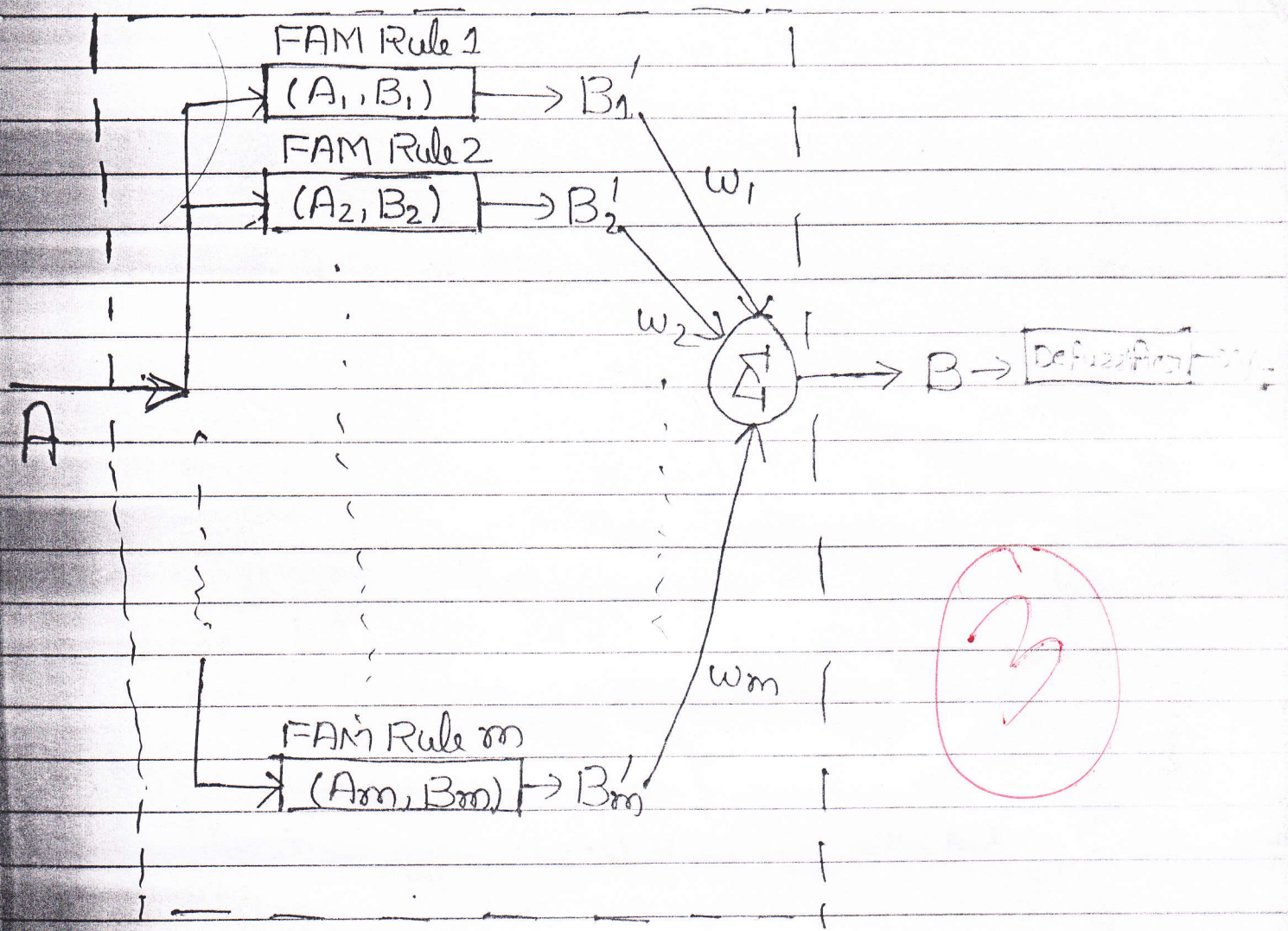


Figure 1 Function f maps domain X to range Y. In the final illustration we use several numerical-point samples (x_i, y_i) to estimate $f: X \to Y$. In the second case we use only a few fuzzy subsets A_i of X and B_j of Y. The fuzzy association (A_i, B_j) represents system structure, as an adaptive clustering algorithm infer or as an expert might articulate

We begin with single-association FAMs. For concreteness let the fuzzy-set pair (A, B) encode the traffic-control association. We quantize the domain of traffic density to the n-numerical variables $x_1, x_2, \ldots, x_n$. We quantize the range of green-light duration to the p variables $y_1, y_2, \ldots, y_p$. The elements x_i and y_i belong respectively to the ground sets $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_p\}$. x_1 might represent zero traffic density. y_p might represent 10 seconds.

The fuzzy sets A and B are multivalued or fuzzy subsets of X and Y. So A defines a point in the n-dimensional unit hypercube $I^n = [0,1]^n$ and B defines a point in the p-dimensional fuzzy cube I^p. Equivalently, A and B define the membership functions m_A and m_B that map the elements x_i of X and y_i of Y to degrees of membership in $[0,1]$. The membership values, or fit (fuzzy unit) values indicate how much x_i belongs to or fits in subset A, and how much y_i belongs to B. We describe this with the abstract functions $m_A : X \to [0,1]$ and $m_B : Y \to [0,1]$. We shall freely view sets both as functions and as points in fuzzy power sets.



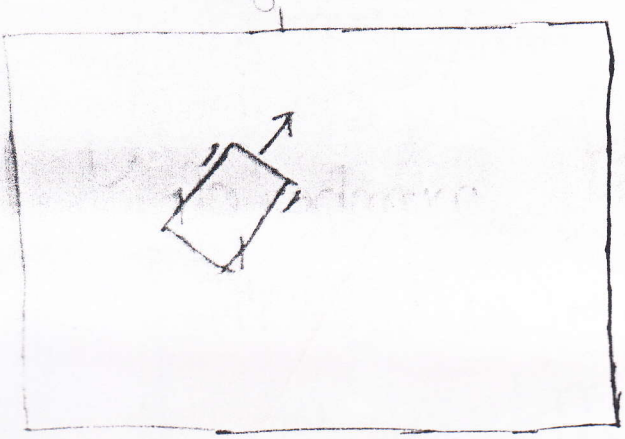
FAM system

Figure FAM system architecture. The FAM system maps fuzzy sets in the unit cube I^n to fuzzy sets in the unit cube I^p. Binary input sets model exact input data. In general only an uncertainty estimate of the system state confronts the FAM system. So A is a proper fuzzy set. The user can defuzzify output fuzzy set B to yield exact output data, reducing the FAM system to a mapping between Boolean cubes

Question6 Describe Fuzzy Truck Backer Upper Control System? Give example of Fuzzy Logic

Answer:

loading dock (x_f, y_f)



≤1 Mark

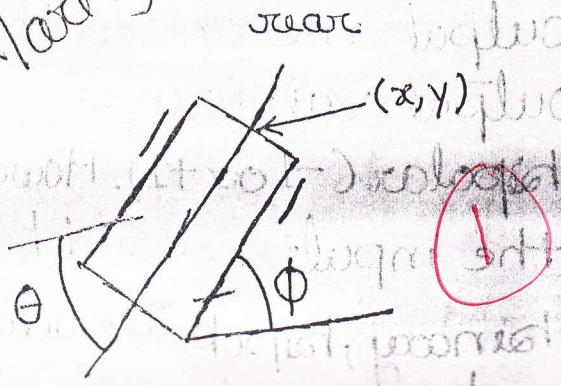


Figure1:- Diagram of simulated truck and loading zone

Figure1 shows the simulated truck and loading zone. The truck corresponds to the cab part of the neural truck i.e. the Nguyen-Widraw neural truck backer-upper system. The three state variables ϕ, x and y exactly determine the truck position. ϕ specifies the position of the rear center of the truck in the plane.

The goal was to make the truck arrive at the loading dock at a right angle ($\phi_f = 90°$) and to align the position (x, y) of the truck with the desired loading dock (x_f, y_f). We considered only backing up. The truck moved backward by some fixed distance at every stage. The loading zone corresponded to the plane $[0, 100] \times [0, 100]$, and (x_f, y_f) equaled (50, 100)

At every stage fuzzy controllers should produce the steering angle θ that backs up the truck to the loading dock from any initial position and from any [illegible] in the loading zone.

zy membership functions can have different shapes depending on the designer's preference or experience. In practice fuzzy engineers have found triangular and trapezoidal shapes help capture the modeler's sense of fuzzy numbers and simplify computation.

Next we specified the fuzzy "rule base" or bank of fuzzy associative memory (FAM) rules. Fuzzy associations or "rules" (A, B) associate output fuzzy sets B of control values with input fuzzy sets A of input-variable values. We can write fuzzy associations as antecedent-consequent pairs or IF-THEN statements.

In the truck backer-upper case, the FAM bank contained the 35 FAM rules in Figure 2

	LE	LC	CE	RC	RI
RB	1 PS	2 PM	3 PM	4 PB	5 PB
RU	6 NS	7 PS	8 PM	9 PB	10 PB
RV	11 NM	12 NS	13 PS	14 PM	15 PB
ZE	16 NM	17 NM	18 ZE	19 PM	20 PM
LV	21 NB	22 NM	23 NS	24 PS	25 PM
LU	26 NB	27 NB	28 NM	29 NS	30 PS
LB	31 NB	32 NB	33 NM	34 NM	35

X

Next we specified the fuzzy-set values of the input and output fuzzy variables. The fuzzy sets numerically scaled linguistic terms, the sort of linguistic words we expect might use to describe the control system's behavior. We chose the fuzzy-set values of the fuzzy variables as follows

Angle ϕ

RB: Right Below
RU: Right Upper
RV: Right Vertical
VE: Vertical
LV: Left Vertical
LU: Left Upper
LB: Left Below

x-position x

LE: Left
LC: Left Center
CE: Center
RC: Right Center
RI: Right

{2 Mark} (1 Mark)

Steering-angle signal θ —

NB: Negative Big
NM: Negative Medium
NS: Negative Small
ZE: Zero
PS: Positive Small
PM: Positive Medium
PB: Positive Big

Fuzzy subsets contain elements with degrees of membership. A fuzzy membership function $m_A: Z \to [0,1]$ assigns a real number between 0 and 1 to every element z in the universe of discourse Z. This number (z) indicates the degree to

Fuzzy Truck Backer-Upper System

We first specified each controller's input and output variables. The input variables were the truck angle ϕ and the x-position coordinate x. The output variable was the steering-angle signal θ. We assumed enough clearance between the truck and the loading dock so we could ignore the y-position coordinate. The variable ranges were as follows

$$0 \leq x \leq 100$$
$$-90 \leq \phi \leq 270$$
$$-30 \leq \theta \leq 30$$

{1 Mark}

Positive values of θ represented clockwise rotations of the steering wheel. Negative values represented counter-clockwise rotations. We discretized all values to reduce computation. The resolution of ϕ and θ was one degree each. The resolution of x was 0.1.

Next we specified the fuzzy-set values of the input and output fuzzy variables. The fuzzy sets numerically represented linguistic terms, the sort of linguistic terms an expert might use to describe the control system's behavior. We chose the fuzzy-set values of the fuzzy variables as follows

Angle ϕ

RB: Right Below
RU: Right Upper
RV: Right Vertical
VE: Vertical
LV: Left Vertical
LU: Left Upper
LB: Left Below

x-position x

LE: Left
LC: Left Center
CE: Center
RC: Right Center
RI: Right

{2 Mark} (1 Mark)

Steering-angle signal θ

NB: Negative Big
NM: Negative Medium
NS: Negative Small
ZE: Zero
PS: Positive Small
PM: Positive Medium
PB: Positive Big

Fuzzy subsets contain elements with degrees of membership. A fuzzy membership function $m_A: Z \to [0,1]$ assigns a real number between 0 and 1 to every element z in the universe of discourse Z. This number (z) indicates the degree to

$$A_1(x) = \begin{cases} 1 & \text{when } x \leq 20 \\ (35-x)/15 & \text{when } 20 < x < 35 \\ 0 & \text{when } x \geq 35 \end{cases}$$

$$A_2(x) = \begin{cases} 0 & \text{when either } x \leq 20 \text{ or } \geq 60 \\ (x-20)/15 & \text{when } 20 < x < 35 \\ (60-x)/15 & \text{when } 45 < x < 60 \\ 1 & \text{when } 35 \leq x \leq 45 \end{cases}$$

$$A_3(x) = \begin{cases} 0 & \text{when } x \leq 45 \\ (x-45)/15 & \text{when } 45 < x < 60 \\ 1 & \text{when } x \geq 60 \end{cases}$$

To illustrate the concepts, we consider three fuzzy sets that represent the concepts of a young, middle-aged and old person.