UNIT 2

HOPFIELD NETS, BACK PROPAGATION AND COUNTER PROPAGATION NETWORK

2.1 HOPFIELD NETS, STRUCTURE, TRAINING, APPLICATIONS AND STABILITY–

Q.1 Explain structure and application of Hopfield nets. (KUK, May 2009)
Q.2 Explain training & stability of Hopfield nets. (KUK, May 2010)
Q.3 Draw the architectural graph of the Hopfield network and explain the operational procedure in summarized form. (KUK, May 2011)

Ans. Discrete Hopfield Net – This type of network was described by J.J. Hopfield in 1982. Hopfield while working on the magnetic behaviour of the solids [spin glasses] discribed property of magentic atoms using two states [1 and -1]. The magnetic mutual exchange between the atoms is represent by a mathematical formulation which led to the development of the Hopfield net. The topology of a Hopfield Network is very simple, it has 'n' neurons, which are all networked with each other. A Hopfield network is able to recognize unclear unctear pictures correctly. However, only one picture can be stored at a time. In practial applications one must assume that many pictures will be given, which have to be stored and then classified.

The discrete Hopfield net is a fully interconnected neural net with each unit connected to every other unit. The net has symmetric weights with no self connections i.e. all the diagonal element of the weight matrix of a Hopfield are zero.

$$W_{ij} = W_{ij} \text{ and}$$
$$W_{ii} = 0$$

The two main differences between Hopfield and interative auto associative net [recurrent associative net] are that, in Hopfield net,

- only one unit updates its activation at a time, and
- each unit continues to receive an external signal in addition to the signal form the other units in the net.

The asynchronous discrete time updating of the units allows a function known as an energy function or lyapunov function to be found for the net. This function proves that the net will converge to a stable set of activations. The formulation of the discrete Hopfield net shows the usefulness of the net as a content addressable memory.

1. **Architecture**– The architecture of the discrete Hopfield net is shown in fig. 2.!

The achitecture shown in fig consists of (n) number of x input neurons and y output neurons. It should be noted that a part from receiving a signal from input, they y_1 neuron receives signal from its other neurons also.

This is the same for all other output neuron. Thus, there exists a feedback output returned to each output neuron. That is why Hopfield network is called a feedback network.

2. **Training Algorithm**– Discrete Hopfield net is described for both binary as well as bipolar vector patterns. The weight matrix to store the set of binary input patterns s(P), P=1,.....P, where

$$S(P) = (S_1, (P)..... S_i(P).....S_n(P))$$

canbe determined with the help of Hebb rule. The weight matrix can be determined by the formula

$$W_{ij} = \sum_P [0 - s_i(P) - 1)(2S_j(P) - 1) \text{ for } i \neq j \text{ and } W_{ii} = 0$$

For bipolar input patterns, the weight matrix is given by,

$$W_{ij} = \sum_P s_i(P)\, S_j(P) \text{ for } i \neq j \text{ and } W_{ii} = 0$$

3. **Application Algorithm**– The weights to be used for the application algorithm are obtained from the training algorithm. Then the activations are set for the input vectors. The net input is calculated and applying the activation, the output is calculated. This output is broadcasted to all other units. The process is repeated until the covergence of the net is obtained. The application algorithm of a

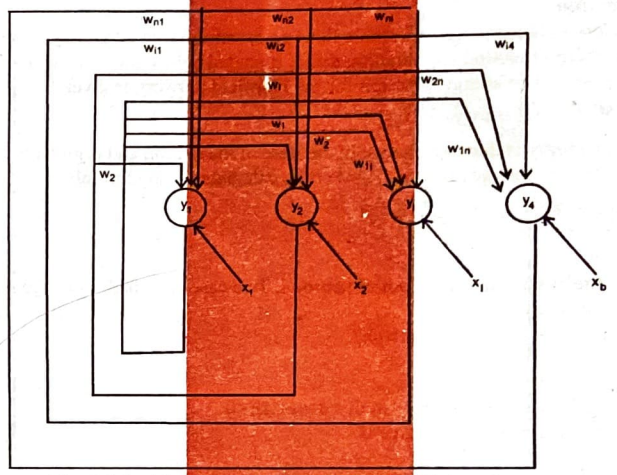

Fig. 2.1 Architecture of Discrete Hopfield Net

discrete Hopfield net is given as follows:

Step 1: Initialize weights to store pattern. While activations of the net are not converged perform step 2 to 8.
Step 2: For each input vector x_i repeat steps 3 to 7.
Step 3: Set initial activations of the net equal to the external input vector x, $y_i = x_i$ [i = 1, -n]
Step 4: Perform Step 5 to 7 for each unit y_i
Step 5: Computer the net input

$$y_{-ini} = x_i + \sum_j y.W_{ji}$$

Step 6: Determine activation [output signal]

$$y_i = \begin{cases} 1, \text{ if } y_{-ini} > \theta_i \\ Yi, \text{ if } y_{-ini} = \theta_i \\ 0, \text{ if } Y_{-ini} < \theta_i \end{cases}$$

Step 7: Broadcast the value of y_i to all other units
Step 8: Test for convergence.

The value of threshold θ is usually taken to be zero. The order of update of the unit is random but each unit must be updated at the same average rate.

4. **Analysis Storage capacity** – In the Hopfield net the number of binary patterns that can be stored and recalled in a net with reasonable accuracy is given by,
P = 0.15n where n is the number of neurons in the net.
If bipolar patterns are used then,

$$P \approx \frac{n}{2\log_2^n}$$

Energy Function

The discrete net will converge to a stable limit point considering an energy function of the system. It is said that the energy function is a function that is bounded below and is a non-increasing function of the state of the system. The energy function for the Hopfield network is given by

$$E = -0.5\Sigma\Sigma y_i y_j w_{ij} - \Sigma x_i y_i + \Sigma \theta_i y_i$$

The change in energy is due to a change in the state of the neuron and is given by ΔE. It is found that the activation of the net changes by Δy_i. This can be calculated using the following equation.

$$\Delta E = -\left[\Sigma y_j\ W_{ij} + x_i - \theta_i\right]\Delta y_i$$

$$\Rightarrow [net_i - \theta_i]\Delta y_i$$

Where Δy_i is the change in the output of neuron i. Two cases in which a change Δy will occur in the activation of neuron y_i are

It y_i is positive, it will change to zero if

$$x_i + \Sigma_j y_j w_{ij} < \theta_i$$

This gives a negative change for y_i; in this cases ΔE<0.

If y_i is zero, it will change to positive if

$$x_i + \Sigma y_j w_{ij} > \theta_i$$

This gives a positive change for y_i in this case ΔE<0.

From the above two case we can show that the energy cannot increase i.e. For both positive and negative change in g_i, the value of ΔE is less than zero [negative]. As a result, the energy is bounded. So the net must reach a stable equilibrium such that the energy does not change with further iteration. The important aspect of the algorithm is that the energy change depends only on the change in activation of one unit and that the weight matrix can be symmetric with zeros present on the diagonal.

5.7 Stability – As with other networks, the weights between layers in this network may be considerd to form a matrix W. Cohen and Grossberg (1983) have shown that recurrent network are stable if the matrix is symmetrical with zero on its main diagonal, that is, if $W_{ij} = w_{ji}$ for i not equal to j, and $W_{ij} = 0$ for all i.

The stability of such a network may be proved through can elegant mathematical technique. Suppose a function can be found that always decreases each time the network changes state. Eventually this function must reach G minimum and stop, thereby ensuring that the networks changes state. Eventually this function must reach a minimum and stop, thenby ensuring that the network is stable. The function that follows is called a Liapunov function and works in just such a manner on the recurrent network presented above

E = (-1/2) 2 ; W/1 Out, OUT 1-2:1/Out 1+2 : Tj out/

where E = an artificial network energy.

W_{ij} = Weight from the output of neuron i to the input of neuron j
Out_j = Output of Neuron j
I_j = external input to neuron j
T_j = threshold of neuron j

The change in energy E_i due to a Change in the state of neuron j_i is

OE = -[2 -(W_{ij} OUT,) IC T_j] on UTI

Where δOUT; is the change in the output of neuron j. The network symmetry criterion is sufficient, but not necessary, to define a stable system. There are may stable is system (eg. all feed forward networks) that not satisfy it. Also, examples can be shown in which minute deviation from symmetry can produce continuous oscillations, however approximate symmetry is useful adequate to produce stable system.

Q.4 What is Hopfield model of neural Networks and differentiate between discrete and continuous time Hopfield Network. **(KUK, May 2010)**

Ans. Hopfield Model of neural Network– See Question (1)

Diserete Hopfield Net – See Question (1)

Continuous Hopfield Net– the continuous Hopfield net is a modified form of discrete Hopfield net. It uses continuous values output functions, which can be used either for associative memory problems or constrained optimization problem. In a discrete Hopfield net, the connections between the units are bidirectional so the weight matrix is symmetric. In continuous Hopfield net, its output signal as, V_i = f(ui).

let us define a energy function as,

$$E = 0.5 \sum_{i=1}^{m}\sum_{j=1}^{m} w_{ij}\ V_i V_j + \sum_{i=1}^{m} \theta_i V_i$$

The minima of the energy function is obtained when,

$$\frac{dE}{dt} \le 0$$

At this stage the net will converge to a stable configuration. The differential equation formed for this assumed energy function. So that the net converges when the activity of the neuron is changed with respect to charge in time is given by,

$$\frac{du_i}{dt} = \frac{-\partial E}{\partial V_i} = -\sum_{j=1}^{h} W_{ij} V_j \ldots\ldots \theta_j$$

In general, the energy function on of a continuous Hopfield net, when ↑ is a time constant, is given as,

$$E = -5\sum_{i=1}^{m}\sum_{j=1}^{m} wij\ V_i V_j - \Sigma\theta_i V_i + \frac{1}{T}\sum_{i=1}^{m}\int_{\vartheta}^{V_i} f_i^{-i}(v)dv$$

when the activity of the each neuron to converge accordingly to the differential equation,

$$\frac{du_i}{dt} = \frac{-u_i}{\tau} + \sum_{j=1}^{h} W_{ij} V_j + \theta_i$$

The time constant in this case generalty taken as 1. The continuous Hopfield net can be used in contorainal op problems like traveling Saleman problem etc.

Algorithm– The algorithm given below discusses the basic procedure for solving constrained optimization problems like traveling salesman problem using a continuous Hopfield net.

Step 1: Initialize activations of all units

Step 2: Initialize a small value of Δt.

Step 3: When stopping condition is false, perform steps 4-8

Step 4: Do steps 5-7 for n^2 times.

Step 5: Choose a unit at random

Step 6: Change activity on selected unit

$$u_{a,i(new)} = u_{a,i(old)} + \Delta t \begin{bmatrix} -x_{a,i(old)} - A \sum_{j \neq i} u_{a,j} \\ -B \sum_{b \neq a} Vb, i + c(n - \sum_{a} \sum_{j} v_{a,j}) \\ -D \sum_{b \neq i} da, b (V_{a,i+1} + V_{a,i-1}) \end{bmatrix}$$

Step 7: Apply the output function

$$V_{a,i} = 0.5 [1 + \tanh(\alpha Ua, j)]$$

Step 8: Test for stopping condition.

The value of A, B, C, D, N and α may as any constant. A steep sigmoid function is formed for large values of α , which the approximates a step function. In the case of travelling salesman problem, the values of N is taken much higher than the number of cities, n.

2.2 Application of Hopfield Networks– Hopfield nets have a wide variety of application in different fields. some of them one given below –

(a) Analog to Digital converter – In recent works, an electrical circuit has been presented that uses a recurrent networks to produce a 4-bit A/D converter. In this networks,

(i) Amplifiers serves as artificial neurons.
(ii) Registors represents weights which connect neurons output to input of all others.
(iii) Weights are symnetrical.
(iv) No Registors converts a neuron's output to its own input.
(v) Amplifies both normal & inventing outputs i.e. the case in which a weights must be negative while permitting the use of ordinary postitive values registers for all weights.
(vi) Each amplifier will have a finite input resistance and input capacitance to characterize the dynamic response.
(vii) this applications assumes that a threshold function is used response.
(viii) This application assumes that a threshold function is used like signoid function.
(ix) All the outputs are charged at the beginning of discrete time intervals called epochs. At the start of each epoch, if the summation of inputs to neurons is greater than the threshold, the output becomes one else it becomes zero.
(x) The objective is to select the sensitive so that a continously increasing voltage X applied to a single-input terminal procedure a set of 4-bit output representation a binary number, the value of which is an approximation to I/P voltage.The energy function is defined as-

$$E = -\frac{1}{2}\left[x - 2; 2i\Delta u\, T_i F + 2:22j - \left[\frac{007}{1} - 007i\right]\right]$$

Where X-input voltage
Where E is minimized, desired output has been reached.

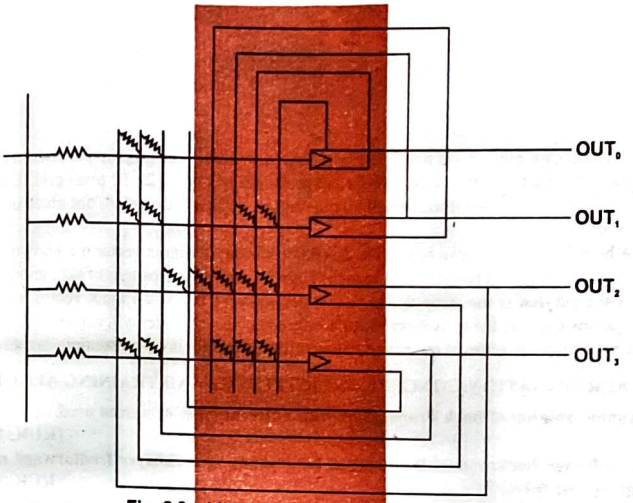

Fig. 2.2 : 4 bit Analog to Digital Converter

(b) Travelling Salesperson problem– It is are optimization task that arises in many practical situations. It is states given a group of cities & distance between each city, find the shortest tour that visits each city only once & return the starting point.

- Solution using recurrent networks is typical in this regard, the result are not guaranteed to be optimal. But the answer is reached so rapidly that the technique may prove useful in certain cases. Suppose the cities to be visited are lettered A, B, C, D and distance between pairs is d_{AB}, d_{bc} & So on.
- The Solution is an ordered set of n cities. The problem is them to map on the computational made using neurons. Each city is respresented as row and n neurons.
- A possible solution is C is visited just, A is long D is third & B is fourth. This requires n_2 neurons & the of such tour is to w_2 is dea + dad + ddb + dbc'.

City	order			Visited
	1	2	3	4
A	0	1	0	0
B	0	0	0	1
C	1	0	0	0
D	0	1	0	0

- Energy function must satisfy two requirements. First it must be low for those solutions that produce a single in each second which is satisfied by three summation energy function. The second requirement favouring short focus is satisfied by adding a term energy function as follows–

$$E = \frac{1}{2D} \frac{\sum_x \sum_y \Sigma dx_y \, Out_x}{(Out_y, i\mu + out_y, i-1)}$$

This term represents the length of any valid tour.

(c) Pattern Completion– A hopfield net is used to encode three graphic patterns: a plan, a tank and a helicopter as memories. These image are defined on a 12× 12 pixel grid. Each figure is respresented by a 144 dimensional bipolar vector: -1 for white pixel, +1 for black pixel and vice versa.

(d) A hopefield network can be used to determine whether are input vector is a known vector or an unknown vector. It recognizes a known vector by producing a pattern of activation on the units of the net that is the same as the vector stored on the net. If the input vector produced is as unknown vector, the activation vector produced as the net iterates will converge to an activation vector that is not one of the stored patterns: such a pattern is called spurious stable state.

2.3 BACKPROPAGATION CONCEPT, ARCHITECTURE AND TRAINING ALGORITHMS–

Q.5 Exaplin training of back Propagation with different types of passed used.
(KUK, May 2009)

Q.6 Describe the backpropagation method for training of multilayer feedforward network in algorithmic form. **(KUK, May 2011)**

Q.7 What is feedforward network? Explain the architecture of back propagation network.

Ans. Feedforward Network– One can differentiate between two basic types of networks, networks with feed back and those with out it. In networks with feed back, the output values can be traced back to the input values. However there are networks wherein for every input vector laid on the network, an output vector is calculated and they can be read from the output neurons. There is no feedback. Hence, only a forward flow of information is present. Networks having this structure are called as feed forward networks. There are various nets that come under the feed forward type of nets. Some of them have already been explained in previous chapter. One of the most important types of feed forward network is the Back Propagation network, which is discussed in the next section. radial basis function is also included in this category. A radial basis function is closely related to the back propagation network except that there is a variation in the function used between both the networks.

Back Propagation Network [BPN]– Back prapagation is a systematic method for training multi-layer artificial neural networks. It has a mathematical foundation that is strong if not highly practical. It is a multi-layer forward network using extend gradient descent based delta-learning rule, commonly known as back propagation of errors rule. Back propagation provides a computationally efficient method for changing the weight in a feed forward network, with differentiable activation function units, to learn a training set of input-output examples. GE Hinton, Rumelhart and R.o. Williams first introduced BPN in 1986. Being a gradient descent method it minimizes the total squared error of the output computed by the net. The network is trained by supervised learning method. The aim of this network is to train the net to achieve a balance between the ability to respond correctly to the input patterns that are used for training and the ability to provide good responses to the input that are similar.

Geeralized Delta learning Rule (or) Back Propagation Rule– The total squared error of output computed by net is minimized by a gradient descent method known as back propagation or generalized delta rule.

Derivation– Consider an arbitary activation function f(x). The derivation of activation function is denoted by f(x). Let

$$y - ink = \Sigma_i z_i wjk$$

$$Z - inj = \Sigma_i V_{ij} x_i$$

$$Y_k = f(y - ink)$$

The error to be minimized is $E = 0.5 \Sigma_k [t_k - y_k]^2$

By use of chain rule we have

$$\frac{\partial E}{\partial wjk} \frac{-\partial}{\partial wjk} \left[0.5 \sum_k (t_k - y_k)^2 \right]$$

$$= \frac{\partial}{\partial KN_{jk}} (0.5[t_k - t(y - ink)]^2)$$

$$= -[t_k - y_k] \frac{\partial}{\partial W_{jk}} f(y - ink)$$

$$= -[t_k - y_k] f(y - ink) \frac{\partial}{\partial W_{jk}} (y - ink)$$

$$= -[t_k - y_k]^{f'} (y - ink) Z_j$$

Let us define $\delta_k = -[t_k - y_k]^{f'} (y - ink)$

weights on connections to the hidden unit Z_j

$$\frac{\partial E}{\partial_{vij}} = -\Sigma_k [t_k - y_k] \frac{\partial}{\partial_{vij}} y_k$$

$$= -\Sigma_k [t_k - y_k] f(y_{ink}) \frac{\partial}{\partial_{vij}} y - ink$$

$$= -\Sigma \delta_k \frac{\partial}{\partial_{vij}} y^{-ink}$$

Rewriting the equation and substituting the value of y-ink

$$= -\Sigma_k \delta_k \frac{\partial}{\partial_{vij}} (\Sigma z_j - w_{jk})$$

$$= -\Sigma_k \delta_k \, w_{jk} \frac{\partial}{\partial V_{ij}} 2j$$

$$= -\Sigma_k \delta_k \, w_{jk} \frac{\partial}{\partial V_{ij}} \partial (Z_{in} \, j)$$

$$= -\sum_K \delta_k \, w_{jk} \, f^1 (Z_{inj}) (x_i)$$

$$\therefore \quad \delta_i = -\sum_K \delta_k w_{jk} f^1(Z_{inj})$$

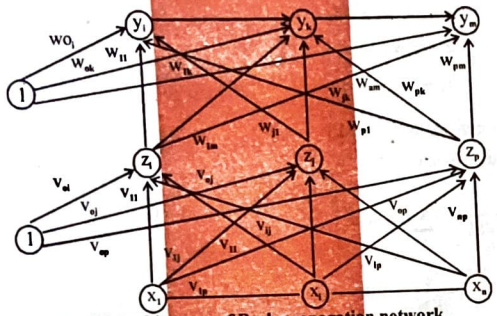

Fig. 2.3 Architecture of Back propagation network

The weight updation for output unit is given by

$$\Delta W_{jk} = -\alpha \frac{\partial E}{\partial_{wjk}}$$

$$= \alpha [t_k - y_k] f^1(y_{ink}) Z_j$$

$$= \alpha f_k z_j$$

The weight updation for the hidden unit is given by :

$$\Delta V_{ij} = -\alpha \frac{\partial E}{\partial_{vij}}$$

$$= \alpha f^1(z - inj) x_i \sum_k \delta_k w_{jk}$$

$$= \alpha \delta_j x_i$$

This is the generalized delta rule used in the back peopagation network during training.

Architecture– A multilayer feed forward back propagation network with one layer of 2 hidden units is shown in the fig. The y output unit has W_{ok} bias and Z hidden unit has V_{ok} as bias.

It is found that both the output units and the hidden units have bias acts like weights on connection from units whose output is always 1. from fig. it is clear that the network has one input layer, one hidden layer and one output layer. There can be any number of hidden layers. The input layer is connected to the hidden layer and the hidden layer is connected to the output layer by means of inter connection weights. Only the feed forward phase of operation is shown. But during the back propagation phase of learning, the signals are sent in the reverse direction. the architecture of back forward network resembles a multi layered feed forward network discussed. The increase in the number of hidden layers results in the computational complexity of the network. As a result, the time taken for convergence and for minimize the error may be very high. The bias is provided for both hidden and the output layer, to act upon the net input to be calculated.

Training Algorithm– The training algorithm of back propagation involves four stages,viz....

1. Initialization of weights
2. Feed forward
3. Back propagation of errors
4. Updation of the weight and biases.

During first stage which is the initialization of weights, some small random values are assigned. During feed forward stage each input unit (x_i) receives an input signal and transmits this signals to each of the hidden units Z1...Zp. Each hidden unit then calculates the activation function and sends its signal Z_j to each output unit. The output unit calculates the activation function to form the response of the net. For the given input pattern.

During back propagation of errors each output unit compares its computed activation y_k with its target value t_k to determine the associated error for that pattern with that unit based on the error, the factor δ_k [k= 1, ...,m} is computed and is used to distribute the error at output unit y_k back to all units in the previous layer. Similarly, the factor δ_j {j = 1..... P} is computed for each hidden unit z_j.

During final stage, the weight and biase are updated using the δ factor and the activation.

Parameters – The various parameters used in the training algorithm are as follows:

x : Input training vector

$x = [x_1, ..., x_i, ..., x_n]$

t = Output target vector

$t = [t_1, ..., t_k, ..., t_m]$

δ_k = error at output unit y_k

δ_j = error at hidden unit z_j

α = learning rate

Voj = bias on hidden unit j

Z_j = hidden unit j

W_{ok} = bias on output unit k

y_k = output unit k.

The training algorithm used in the back propagation network is as follow. The algorithm is given with the various phases.

Initialization of Weights

Steps 1: Initialize weights to small random values.

Steps 2: While stopping condition is false, do steps 3-10.

Steps 3: For each training pair do steps 4-9

Feed Forward

Steps 4: Each input unit receives the input signals x_i and transmits this signals to all units in the layer above i.e. hidden units.

Steps 5: Each hidden unit $[z_j, j = 1,.....P]$ sums its weighted input signals

$$Z - inj = Voj + \sum_{i=1}^{n} x_i v_{ij}$$

applying activation function

$z_j = f(z_{inj})$ and send this signal to all in the above i.e. output units.

Steps 6: Each output unit $\{y_k K = 1,.....m\}$ sums its weight input signals

$$y - ink = W_{ok} + \sum_{j=1}^{p} z_j w_{jk}$$

and applies its activation function calculate the output signal.

$y_k = f(Y\text{-ink})$

Back propagation of Error

Steps 7: Each output $[y_k\ k=1,.....,m]$ receives a target pattern corresponding to an input pattern error information term is calculated as

$$\delta_k = [t_k - y_k]\ f\ [y\text{-}ink]$$

Step 8: Each hidden unit $[z_j, j=1, ...n,]$ sum its delta inputs from units in the layer above

$$\delta - inj = \sum_{k=1}^{m} \delta_j w_{jk}$$

The error information term is calculated as

$$\delta_j = \delta_{-inj}\ f[Z_{inj}]$$

Updation of Weight and Biases

Step 9: Each output unit $[y_{k1}k=1,.....m]$ updates its bias and weights $\{j = O_j.....P\}$

The weight correction term is given by

$$\Delta W_{jk} = \alpha\ \delta k z_j$$

and the bias correction term is given by

$$\Delta W_{ok} = \alpha\ \delta k$$

Therefore, $W_{jk\ (new)} = W_{jk(old)} + \Delta W_{jki} W_{ok\ (new)} = W_{ok(old)} + \Delta W_{ok}$

Each hidden unit $[zj,j=1,.....p]$ updates its bias and weights $[i = 0, ...n]$

The weight correction term

$$\Delta V_{ij} = \alpha \delta_j x_i$$

The bias correction term :

$$\Delta V_{oj} = \alpha\ \delta_j$$

Therefore, V_{ij} (new) V_{ij}(old) + $\Delta\ V_{ijj}$ Voj(New) = Voj (old) + ΔVoj

Step 10: Test the stopping condition.

The stopping condition may be the minimization of the errors, number of epochs etc.

Q.8 Implement a backpropagation network to simulate the following EX-OR function:

I/P1	I/P2	O/p
1	1	0
1	0	1
0	1	1
0	0	0

Try the architecture in which there is only a single hidden unit and all units connected with each other (input units are also connected with the output units). (KUK, June 2008)

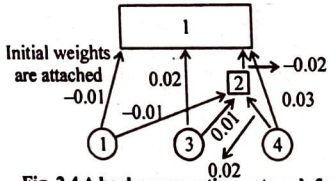

Fig. 2.4 A back propagation network for XOR problem

Back propagation network for learning the exclusive OR function.

The weights are initialized randomly as follows.

$W_{13} = 0.02$, $W_{14} = 0.03$, $W_{12} = -0.02$, $W_{23} = 0.01$, $W_{24} = 0.02$, $W_{16} = -0.01$, $W_{26} = -0.01$

Calculate of activation. Consider a training instance with the input vector = (1,1) and the desired output vector = 0

$O_3 = O_4 = 1$

$$O_2 = \frac{1}{\left[1+e^{-(1\times0.01+1\times0.02-1\times0.01)}\right]} = 0.505$$

$$O_1 = \frac{1}{\left[1+e^{-(0.505\times-0.02+1\times0.02+1\times0.03-1\times0.01)}\right]}$$

$$= 0.508$$

weight training : Assume that the learning rate

$n = 0.3$

$\delta_1 = 0.508n\ (1\text{-}0.508)\ (0\text{-}0.508) = 0.127$

$\Delta\ W_{13} = 0.3 \times (-0.127) \times 1 = -0.038$

$\delta_2 = 0.505\ (1\text{-}0.505)\ (-0.127 \times -0.02)$

$= 0.0006$

$\Delta\ W_{23} = 0.3 \times 0.006 \times 1 = 0.0002$

The rest of the weight adjustments are omitted. note that the threshold is adjusted likewise. It takes may iteration like this before the learning process stops. The following set of final weights gives the mean required error of less than 0.01.

$W_{13} = 4.98$, $W_{14} = 4.98$, $W_{12} = -11.30$, $W_{23} = 5.62$, $W_{24} = 5.62$, $W_{16} = -2.16$, $W_{26} = -8.83$

Q.9 Discuss in defail the training algorithm used in multilayer feed forward back propagation network.

Ans. See Section 2.3, heading Training in Algorithm.

2.4 SELECTION OF PARAMETERS

For efficient operation of back propagation network, it is important to select appropriate parameters used for training.

(i) **Initial Weights–** It will influence whether the net reaches a global minima of the error and if so how rapidly it converges. If the initial weight is too large the initial input signals to each hidden or output unit will fall in the saturation region where the derivation of the sigmoid has a very small value (f(net) = 0) If initial weights are too small, the net input to a hidden or output will approach zero, which then causes extremely slow learning. To get the best result the initial weights are set to random numbers between -0.5 and 0.5 or between -1 and 1. The intialization of weights can be done randomly and there is also a specific approach, which is discussed below.

Nguyen-Widrow Initialization – Faster learning of a BPN can be obtained by using Nguyen-windrow (Nw) initialization. This approach is based on a geometrical analysis of the response of the hidden neurons to a single input. This method is designed to improve the learning ability of the hidden units

$$\beta = 0.7(P)^{1/n} = 0.7\sqrt[n]{P}$$

where,

n = no. of input units,

P = number of hidden units,

β = Scale factors

Steps in Nw Initialization– For each hidden unit, initialize its weight vector (from the input units)

$V_{ij(old)}$ = random number between -0.5 and 0.5

$$\text{Compute } \| V_{j(old)} \| = \sqrt{V_{ij}^{(old)2} + V_{2j}^{(old)2} + ... + V_{ni}^{(old)}}$$

Reinitialize weight : $V_{ij} = \frac{\beta V_{ij}(old)}{||V_j(old)||}$

Set bias : V_{oj} = random number between - β and β.

The Nguyen-widrow analysis is based on the activation function

$$\tan h(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

which is closely related to the bipolar sigmoidal activation function.

Selection of learning Rate – A high learning rate leads to rapid learning but the weights may oscillate, while a lower learning rate leads to slower learning method suggested for adopting learning rate are as follows:

(i) Start with a high learning rate and steadily decrease it. Chages in the weight vector must be small in order to reduce oscillations or any divergence.

(ii) A simple suggestion is to increase the learning rate in order to improve performance and to decrase the learning rate in order to worsen the performance.

(iii) Another method is to double the learning rate until the error value wor sens.

2.5 LEARNING IN BACK PROPAGATION–

There are two types of learning.

(i) Sequential learning or pre-Pattern method.

(ii) Batch learning or Pre-epoch method.

In sequential learning a given input pattern is propagated forward, the error is determined and back propagated and the weights are updates. In batch learning the weight are updates only after the entire set of training network has been presented to the network. Thus the weight update is only performed after every epoch.

if P = Patterns in one epoch, then

$$\Delta W = \frac{1}{P}\sum_{p=1}^{\infty} \Delta Wp$$

In some cases, it is advantageous to accumalate the weight corrections terms for several patterns and make a single weight adjustment for each weight rather than updating the weights after each pattern is presented. This procedure has a 'smoothing effect". In some cases the smoothing may increase the chances of convergence to a local minimum.

Training a Net – The motivation for applying back propagation net is to achieve a balance between memorization and generalization, it is not necessarily advantages to continue training until the error reaches a minimum value. The two disjoints sets of data used during training are:

(i) Set of training patterns.

(ii) Set of training testing patterns

The weight adjustments are based on the training pattern. As along as error the for validation decreases training continues. whenever the error begins to increase the net is starting to memorize the training patterns. At this point training is terminated.

Numbers of training pairs– The number of training pairs also plays an important role during training of the nets. A simple thumb rule is used to find number of training pairs.

Consider a net trained to classify the fraction 1-(e/q) of the trained patterns correctly. This means it wire also classify (1-e) of the testing pattern correctly by using the following condition.

If there are enough training patterns the net will be able to generalize as desired. Enough training pattern is given by W/p = e, where

P_w = number of training patterns.

and W = number of weights to be trained.

The value of "e" lies between 0 to 1/8. The necessary condition is given by

$$P > \frac{|w|}{1-a}$$

Where a = expected accuracy for the test set.

Sufficient condition is given as

$$P > \frac{|w|}{1-a}\log\frac{n}{1-a}$$

where n = number of nodes.

Number of Hidden Units– If the activation function can vary with the function, then it can be seem that a n-input, m-output function requires at most 2n+1 hidden units. If more number of hidden layers are present, then the calculation for the δ's are repeated for each additional hidden layer present, summing all the δ's for units present in the previous layer that is fed into the current layer for which δ is being calculated.

Momentum factor– In BPN, the weight change is in a direction that is a combination of current gradient and the previous gradient. This approach is beneficial when some training data are very different from a majority of the data. A small learning rate is used to avoid major disruption of the direction of learning. when very unusual pair of training pattern is presented. If the momentum in is added to the weight update formula, the convergence is faster. The weights from one or more previous training pattern must be saved in order to use momentum. For the BPN with momentum, the new weights for training step + 2, is based on t and t+1. It is found that momentum allows the net to perform large weight adjustments as long as the correction proceeds in the same general direction for several patterns. Thus using momentum, the net does not proceed in the direction of the gradient put travels in the direction of the combination of the current gradient and the previous direction for which the weight correction is made. The main purpose of the momentum is to accelerate the convergence of error propagation algorithm. This method makes the current weight adjustment with a fraction of the recent weight adjustment.

The weight updating formula for BPN with momentum is,

$$W_{jk}(t+1) = W_{jk}^{(t)} + \alpha\delta_k z_j + \mu\,[W_{jk}^{(t)} - W_{jk}^{(t-1)}]$$

$$V_{jk}(t+1) = V_{jk}^{(t)} + \alpha\delta_k x_i + \mu\,[V_{jk}(+) - V_{jk}^{(t-1)}]$$

μ is called the momentum factor. It ranges form $0 < \mu < 1$.

limitations of momentum factor

(i) learning rate places the upper limit on the amount by rate places the upper limit on the amount by which the weight can be changed.

(ii) Momentum factor can cause weight changes to be in a direction that would increase the error.

2.6 MERITS AND DEMERITS OF BACK PROPAGATION NETWORK

The merits and demerits of the back propagation network are as listed best below:

Merits : The mathematical formula present here can be applied to any network and does not require any special mention of the features of the function to be learnt.

2. The computing time is reduced if the weight chosen are small at the beginning.
3. The batch update of weights exist, which provides a smoothing effect on the weight correction terms.

Demerits– 1. The number of learning steps may be high, and also the learning phase has intensive calculations.

2. The selection of the number of hidden modes in the network is a problem. if the number of hidden neurons is small, then the function to be learnt may not be possibly represented, as the capacity of network is small. If the number of hidden neurons is increased, the number of independent variable of the error function also increases and the computing time also increase repidly.
3. For complex problem it may require days or weeks to train the network or it may not train at all long training time results in non-optimum stepsize.
4. The network may get trapped in a local minima even though there is a much deeper minimum nearby.
5. The training may sometimes cause temporal instability to the system.

2.7 APPLICATION OF BACK PROPAGATION NETWORK

Back prapagation has a wide variety of application and a few of these are mentioned to demonstrate the power of this method.

1. **Image Compression**– Image compression is a process of reducing the numbers of bits required to represent the image. A multilayer network with 3 hidden layers is used for this purpose.
   ---> The three hidden layers are termed as combiner layer and decombinear layer respectively. From input layer to combiner layer and decombiner layer to output layer, connections are designed as fully connected neural net.

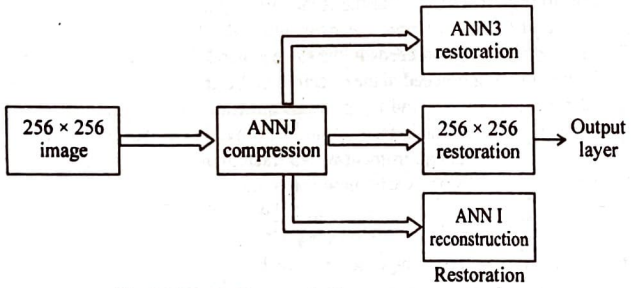

Fig. 2.5 Block diagram for image compression

---> Then the training process is carried out. Images are compressed and then reconstructed by using two different ANN stages.
----> Optimum learning rate and momentum factor are taken to be 0.01 and 0.1 respectively.

2. **Data Compression**– Beak propagation net can be used to compress data by training a set top function as an auto associative set with fewer hidden units then there are input or output Units.
   ---> consider a set of character with each character represent or a grid of 7×9 pixels. Since each character is represented in the input data as a vector with 63 binary components, the input layer of neural net has 63 input units. The hidden layer is given a smaller number of units.
   ---> the net is considered to have learn pattern if all output values are with in a specified tolerance of desired values. Then, the response of unit is considered current if its activation is greater than the tolerance.
3. **Control System**– The application of back propagation networks in speed control of induction motors has emerged as one of the challenging tasks in control systems area.

---> The network is used in the controller and feed back path. The feedback closed loop control system is used to ontrol the speed of induction motor.
---> The BPN circuit is used to train the net for controllers and also for neural network estimation to estimate torque, flux & flux angle.

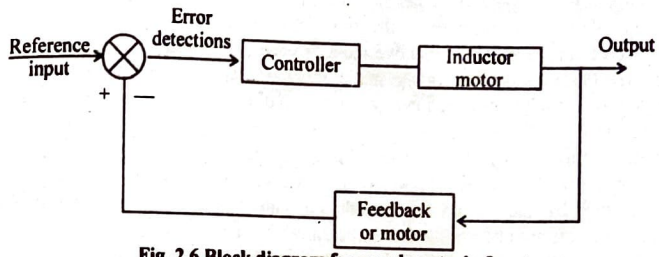

Fig. 2.6 Block diagram for speed control of motor

4. **Pattern Recognition**– Back propagation net is used for recognizing handwritten zip codes. Even when an application is based on standard training algorithm, it is imported to customize the architecture in order to improve the performance of application.
   ----> This back propagation net has several hidden layers but the pattern of connections from one layer to next is quite localized.
5. **Speech Recognition**– Rosenburg produced a spectacular, sucess with NET talk, system that converted printed English text into highly intelligible speech. His tape recording of the training process bare a strong resemblance to the sounds of a child at various stages as learning to speech.

2.8 Kohonen Self Organizing Feature maps–

Q.10 Illustrate Kohonan network with an example. **(KUK, March 2010, 2012)**

Q.11 Describe the self organizing feature mapping (SOFM) algorithm to adaptively transfer an involving signal pattern of arbitray dimension in to a discrete map. **(KUK, May 2011)**

Ans. kohonen worked in the development of the theory of competition, as a result the competitive processing elements are referred to as Kohonen unit. these self organizing maps can also be termed as topology preserving maps.

In a topology-preserving map, units located physical next to each other will respond to classes of input vectors that are likewise located next to each other. Although it is easy to visualize units located to each other in a two-dimensional array, it is not easy to determine which classes of vectors are next to each other in a high dimensional space.

The self organizing map, developed by kohonen, groups the input data into cluster which are commonly used for unsupervised training. In the SOM, all the units in the neighborhood that receive positive feedback from the winning unit participate in the learning process. Even if a neighboring unit is weight is orthogonal to the input vector, its weight vector will still change in response to the input vector. This simple addition to the competitive process is sufficient to account for the order mapping. The topological structure property is observed in the brain, but is not found in any other artificial neural network expect SOM. There are 'n' cluster units, arranged in a one in two dimasiomnal array and the input signals are n-tuples. In self organizing process, the cluster unit whose weight vector matches the input pattern closely is selected as

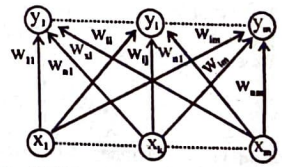

Fig. 2.7 Kohonen self organizing map

the winner. The winning and the neighboring units update their weights. The neighboring units weights vectors are not close to the input pattern but differ to save extent. Hence, the connection weights do not multiply the signal sent from the input units to the cluster units.

Architecture– The architecture of the Kohonen self organizing map is shown in figure 2.7. This architecture is similar to the competitive net architecture.

The key principle for map formation is that training should take place over an extended region of the network centered on the maximally active mode. Hence, the concept of neighborhood should be defined for the net. This may be fixed by the spatial relation between nodes with in the self organizing layer. Two neighborhood schemes are shown based on two dimensions array in the form of rectangular and hexagonal grids.

Training Algorithm– Initially, the weights and learning rate and set. The input vectors to be clustered are presented to the network once the input vector unit are given, based on the initial weights, the winner unit is calculated either by Euclidean distance method or sum of products method. Based on the winner unit selection, the weights are updates for that particular winner unit using competitive learning rule. An epoch is said to be completed once all the input vectors are presented to the network. By updating the learning rate, several epochs of training may be performed.

Step 1: Set topological neighborhood parameters. Set learning rate, initialize weights.
Step 2: While stopping condition is false, do steps 3-9.
Step 3: For each input vector x, do steps 4-6
Step 4: For each j, compute squared Euclidean distance

$$D(j) = \sum (W_{ij} - X_i)^2 \quad i = 1 \text{ to } n \text{ and}$$
$$j = i \text{ to } m.$$

Step 5: Find index J, when D(j) is minimum.
Step 6: For all units j, with in the specified neighbohood of J, and for all i, update the weights.

$$W_{ij\,(new)} = W_{ij(old)} + \alpha [x_i - W_{ij}(old)].$$

Step 7: Update the learning rate.
Step 8: Reduce the radius of topological neighborhood at specified times.
Step 9: Test the stopping condition.

The map formation occurs in two phases–
(i) Initial formation of perfect order.
(ii) Final Convergence.

The second phase takes a longer duration than the first phase and requires a small value of learning rate. The learning rate is a slowly decreasing function of time and the radius of the neighborhood around a cluster unit also decreases as the clustering process goes on. The initial weights are assumed with random values.

Q.12 Train the Kohonan networks when the weights are:
W_1 = (0.440, 0.870, 0.220)
W_2 = (0.530, 0.270, 0.801)
W_3 = (0.424, 0.566, 0.707) and
Input X = (0.208, 0.590, 0.780) and eta = 0.3 **(KUK, May 2009)**

Ans. We calculate the dot product as follows:

$$X.WH_1 = 0.894$$
$$X.WH_2 = 0.974$$
$$X.WH_3 = 0.777$$

So, unit H_2 is the winner and its weights vector is updated by (assuming that the learning rate n = 0.3 and that the radius of neighbourhood is 0.)

$$\Delta WH_2 = n(X - WH_2) = [-0.065, 0.007, 0.022]$$
$$\text{New} \quad WH_2 = [0.359, 0.573, 0.729]$$

Unlike the perceptron, the kohonen network uses single pairs learning rather than multipass feedback and is potetially fast. This fact suggest its suitability for real time application.

Training the Grossberg Layers –

1. An I/P vector is applied, Kohonen O/P Is are established, & GB O/P's are calculated as in normal OP.

2. Next each weight is adjusted only if it connected to winning Kohonen,. neuron. [Each weight connected to winning K. neuron is adjusted]. Amount of weight adjustment α diff b/w weight and desired O/P of the G.B. neuron to which it connects.

$$V_{ij}\,new = v_{ij}\,old + B\,(Y_j - V_{ij}\,old)\,k$$
$$K_i = O/P \text{ of Kohonen neuron i.}$$
$$Y_j = J^{th} \text{ component of desired O/P vector.}$$
$$\text{Initially} \quad \beta = 0.1 \text{ \& generally reduced as training progresses.}$$

Weights of gross L. Converges to the average values of desired O/Ps.
Weights of gross K.L. converges to the avg. values of desired I/P's.

2.9 Counter Propagation network (CPN)

Q.13 Explain the training procedure for Kohonon and Grossberg layers of counter propagation network. Explain any one application of counter propagation network. (KUK, June 2007)
Q.14 Explain the architecture of counter propagation network and also derive expression for the weight updation involved in counter propagation. (KUK, June 2008)
Q.15 Define Grossberg layer training of counter propagation network. (KUK May, 2010)
Q.16 Give Brief description of counter propagation networks. (KUK, May 2010)

Ans. Counter propagation Network (CPN) – The counter propagation Network (CPN) developed by Robert Hecht Nielsen is beyond the representational limits of single layer networks. This is a multilayer networks based on the various combining structures of input, clustesing and output layers. Compared to the back propagation network, it reduces the time by one hundred times. CPN is different from the back propogation network in the sense that it provides solution for those applications which cannot have layer iterations. As a result CPN can be used for data compression, approximation of function, Pattern association, Pattern completion and signal enhancement applications.

Counter propagation is a combination of two well-known algorithms the self organazing map of Kohonen and the Grossberg out star.Together they have the properties which are absent in either of the one.

The Counter propagation network functions as a table capable of generalization. The training process associates input vector with corresponding output vector. The net will approximate a function, if the data given represents function values.

Counter propagation networks are trained in two stges
Step 1: The input vectors are clustered on the basis of Euclidean distances or by the dot product method.
Stap 2: The desired response is obtained by adopting the weights from the cluster units to the output units.

Counter propagation network is classified in two types.
1. Full Counter propagation network.
2. Forward only Counter propagation network.

CPN is inferior to back propagation for most mapping network application. Its advantages are that in is simple and forms a good statistical model of its input vector environment. The CPN trains rapidly.if appropriately applied it can save large amount of computing time.

The Full CPN possess the generalization capability which allows it to produce a correct output even when it is given an input vector that is partial incomplete or partially incorrect. Full CPN can represent large number of vector pairs, x : y by constructing a look up table.

Architecture– The architecture of the full CPN is shown in fig. 2.8. The given vector pairs are x:y and the approximated vector pairs may be x* : y *.

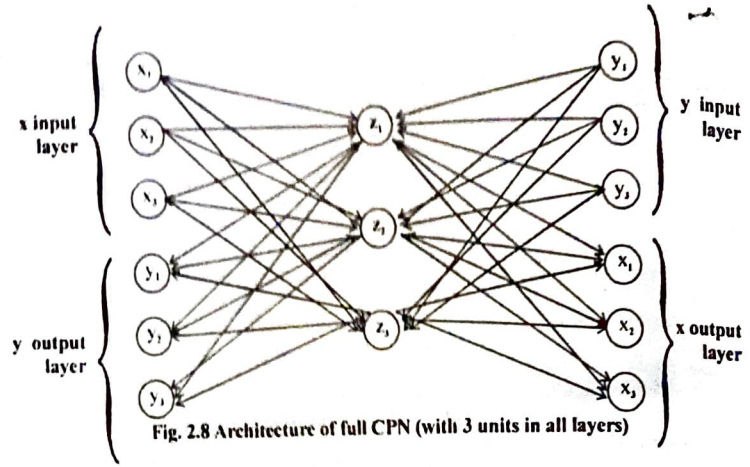

Fig. 2.8 Architecture of full CPN (with 3 units in all layers)

In fig. 2.8 the weights are not mentioned to avoid cramping in the architecture. The CPN functions in two modes. Normal mode, where the input vector is accepted and output vector is produced and training mode, where the input vector is applied and the weights are adjusted to obtain desired output vector.

The architecture of a counter propagation network resembles an instar and outstar model. basically, it has two input layers and two output layers with hidden layer common to the input and output layers. The model which connects the input layers to the hidden layers is called Instar model and the model which connects the hidden layer to the output layer is called outstar model. The weights are updated both in the Instar and outstar model. The network is a fully interconnected network.

Training Phases of Full CPN :

The full CPN training is achieved in two phases.

First Phase : This phase of training may be called as Instar modeled training. The active units here are the units in the X-input, Z-cluster and Y- input layers.

Generally in CPN, the cluster unit does not assume any topology, but the winning unit is allowed to learn. This winning unit uses our standard kohonen learning rule for its weights updation. The rule is given by:

$$\begin{aligned} V_{ij}(new) &= V_{ij}(old) + \alpha (x_i - V_{ij}(old)) \\ &= (1-\alpha)\, V_{ij}(old) + \alpha\, x_i \\ i &= 1 \text{ to } n \\ W_{kj}(new) &= W_{kj}(old) + \beta (j_k - W_{jk}(old)) \\ &= (1-\beta)\, W_{k(old)} + \beta y_k;\ K = 1 \text{ to } m \end{aligned}$$

Second Phase– In this phase, we can find only the J unit remaining active in the cluster layer. The weight from the winning cluster unit J to the putput unit are adjusted. So that vector of activation of units in the y output layer. y*. is approximation of input vector y : and x* is an approximation of input vector X. This phase may be called the outstar modeled training. The weight updation is done by the Grossberg learning rule. which is used only for outstar learning. In outstar learning, no competition is assumed among the units. and the learning occurs for all units in a particular leyer. Thee weight updation rule is given as.

$$\begin{aligned} U_{jk}(new) &= u_{jk}(old) + \alpha (y_k - u_{jk}(old)) \\ &= (1-a)\, \alpha_{jk}(old) + a y_k;\ k = 1 \text{ to } m. \\ t_{ji}(new) &= t_{ji}(old) + b (x_i + t_{ji}(old)) \\ &= (1-b)\, t_{ji}(old) + b x_i;\ i = 1 \text{ to } n. \end{aligned}$$

The weight change indicated is the learning rate times the error:

1. Kohonan layer classifies the I/P vectors into groups that are similar.
2. It is accomplished by adjusting the KOH. L. weights so that similar I/P vectors activate the same K. Neuron.
3. Its is then responsibility of G. L. to produce desired O/P.

Preprocessing the I/P vectors–

1. Normalize I/P vectors (desirable but not mandatory) before apply to N/W.

This is done by dividing each component of vector by vectors length.

$x = (x_1, x_2, x_3, \ldots\ldots\ldots x_n)$

vector length $= \sqrt{X_1^2 + X_2^2 + \ldots\ldots\ldots X_n^2 = t}$

unit vector $x^1 = \left(\frac{x_1}{t_1}, \frac{x_2}{t_2}, \frac{x_3}{t_3}, \ldots\ldots\ldots \frac{x_n}{t}\right)$

i.e. $X_i^1 = \frac{x_i}{\sqrt{X_1^2 + X_2^2 + \ldots\ldots\ldots X_n^2}}$

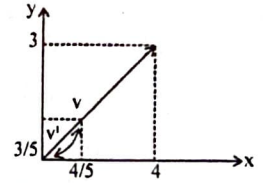

Fig. 2.9 Normalization of input vector

1. To train Kohonan layer I/P vector is applied & its product is calculated with the weight vector associated with each Kohonan layer neuron.
2. Neuron with highest dot product = winner & its weights are adjusted.
3. Since Dot product [used to calculate NET] is measure of similarity b/w I/P vector is weight vector the training process actually consists of selecting the K. Neuron whose weight vector is most similar to I/P vector, & making it still more similar.

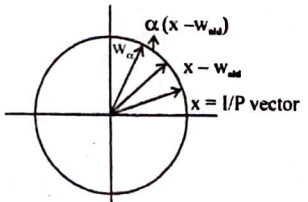

Fig. 2.10 Weight Adjustment in kohonen layer

This is unsupervised training [There is no Teacher] Network self organized by adjusting weights so that K neuron has maximum O/P for a given by input vector.

Training Equation

$$W_{new} = w_{old} + \alpha [x - iw_{old}]$$

New value of w/t connecting an I/P components to winning neuron.

α Training rate coefficient

It starts at about 0.7 & gradually reduced during training

A $\alpha [x - W_{ij}]$

Each wt associated with winning Kohonen new is changed by amount proportional to diff. between its value & value of input to which it connects.

Initializing Weight Vector.

1. Randomize the weights to small numbers for Kohonen Training, randomized Wt: Vectors should be normalize.

2. Distribute the weight vectors according to the density of I/P vectors that must be separated, there by placing more weight vectors in the vicinity of large number I/P vector's. This is impractical to implement directly.

3. Connex combination method. Sets all the weight to the same Value= $\frac{1}{\sqrt{n}}$ n: number of I/P's & hence the number of components in each wt vector. Each component x_i of I/P.

$$\left[\alpha x_i + \frac{1}{\sqrt{n}}(1-\alpha)\right]$$

where n = no. of I/P's
--Initially α is given very small values.
--As n/w trains, α is gradually increased to a limit of L.

2.10 APPLICATIONS OF COUNTER PROPAGATION

Q.17 Explain various applications of counter propagation networks. (KUK, May 2009

Ans. There are various application of CPN

1. **Data Compression–** CPN can be used to compress data prior to transmission, thereby reducing the no. of bits must be sent.
2. Suppose an image is to be transmitted. It can be divided into subimages as shown in figure.
3. Each subimages is further divided into pixels.
4. Each subimage is now a vector of pixels. For simplicity assume each pixel can be either 0 or 1.

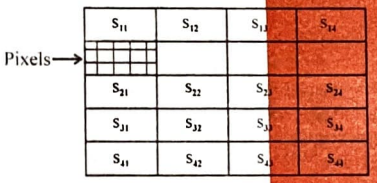

Fig. 2.11 Sub images

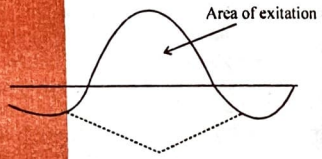

Fig. 2.12 Area of inbibition

5. If there are n pixels in a sub image, then n bits will be required to transmit it.
6. CPN can be used to perform vector quantization.
7. The set of sub image vectors is used as input to train the kohonen layer in accretive mode in which only one neuron is activated.
8. G.L. weights are adjusted to produce the binary code of the index of winning kohonen neuron. Suppose 7th neuron in K.L. is activated the G.L. will be to produce output 000.....111 (binary code for 7). It is the shorter bit string that is transmitted.
9. At receiving end, an identically trained CPN accepts the binary code and produces the inverse fn, an approximation of the original subimage.
10. This method has been applied both to speech & images fielding data compression ratio 101 & 10001.

2. Clustering– Clustering is basically the grouping of objects according to their similarity. We can see how the kohonon network can perform clustering through a competitive learning mechanism called "winner take all".

The node with the largest activation level is declared

This node is the only node will generate the output signal, and all other nodes are supressed to the zero activation level. Kohonen network uses intra layer connection to moderate this competitive. The output of each node acts as a inhibitory input to the other nodes but is actually excitatory in its neighbourhood.

The complex scheme for moderating competition with in a layer, is known as lateral inhibition. The inhibitory effect of a node can also decrease with the distance from it and assume the appearance of a maxican sombrero. The exact size of the neighbourhood varies as learning goes on. It starts large and slowly reduced, making the rays of charge sharper. After training the weight vector of each node encodes the information of group of similar input patterns. Given an input vector, it is assigned to the node with the maximum activation.

3. Satistical modelling– The Kohonen network has the ability to extract the statistical properties of the input data set. It can specifically estimate the probability density function of the input data. This property is very useful since we can apply the kohonen network to statistical Modeling. Kohonen can show it with a fully trained network, if an output vector is randomly selected according to the probability density function of the input data set, then the probability of the vector being closet to any given, weight vector 15 1/k when K is the number of Kohonen neurons.

The Kohonen network should be trained on data that is satistically representative or meaningful to the total input on the other hand, modelling capacity is affected by the network size. In general the smaller the network, the less accurate the satistical model.

Under the winner take all strategy only one neuron gets activated. For each input pattern. This is called the aocretive mode. Another mode called the interpolative mode permits a group of neurous having the highest activation to be winner at one time. This allow the output of the network to be an interpolation of more than one best answer and is thus capable of representing more complex patterns and producing more accurate results.

2.11 IMAGE CLASSIFICATION

Q.18 Explain the concept of image classification of counter propogation network. (KUK, May 2010)

Ans. Here we look how CPN can be used to classify images into categories.

The problem is to determine the angle of rotation of the principle axis of an objects in two dimensions, directly from the raw video images of the objects. In this case the objects is the model of the space shuttle which can rotated 360° about a single axis of rotation. Numerical algorithm as well as pattern matching technique exist will solve this problem. The neural network solution passess some interesting advantages.

Figure show is diagram of the system archtecture. For the space craft orientation system. The architecture is an example of how n neural network can be embedded as a part of overall system.

The system uses a CPN having 1026 input units [1024 for the image and 2 for the training inputs] 12 hidden units, and 2 output units. The unit on the middle layer learn to divide the input vector are used to train the network. These 12 vectors represents images of the shuttle at 30 degree increaments [0°, 30°,.....330°] since there are 12 categories and 12 training vectors, training of competitive layer consists of setting each units weight equal to one of the input vector.

Now this network is limited to classifying all inputs patterns into only one of 12 categories. An input pattern representing a rotation of 32 degrees, for example, it would be classified aas a 30 degree pattern

by this network. One way to remove this deficiency would be to add more units on the middle layer allowing for a fine categorization of the input images. An alternative approach is to allow the output units to perform an interpolation. For patterns that do not match one of the training patterns to with in a tolerance.

Now the ouput layer units calculate their output values according to Eq. $Y^1K = \Sigma W_{kj} Z_j$

In the normal case, where the ith hidden units wins $Y'k = W_{ki}$, since $Z_j = 1$ for $j = 1$ and $Z_j = 0$ otherwise Suppose two competitive units shared in winning the ones with the two closet matching input patterns is, that zj×cosθj for the two winning units. If we restrict the total output from the middle layers to unity, then the putput values from the output layer would be

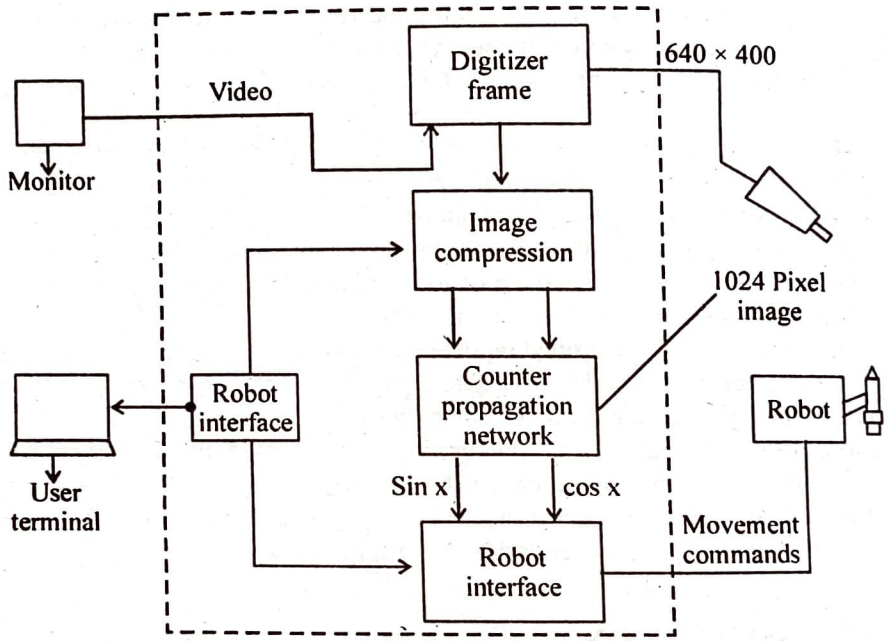

Fig. 2.13 System Architecture for the space craft orientation system

$$y_k^i = W_{ki} + W_{kj} j z_j$$

Where the ith and jth units on the middle layer
Where the winner 1 and

$$Z_i + Z_j = 1$$

The network output is a linear interpolation of the outputs that would be obtained from the two patterns that exactly matched two hidden units that shared the victory.

Using the techniques, the network will classify successfully input patterns representing rotation angles. It had been seen during the training period. In our experiment, the average error was approximately $\pm 30^0$. Since simple scheme is used, the error varied from 0 to 10 degree.

One of the benefit of using the neural network approach to pattern matching is robustness in the presence of noise or of contradictory data.