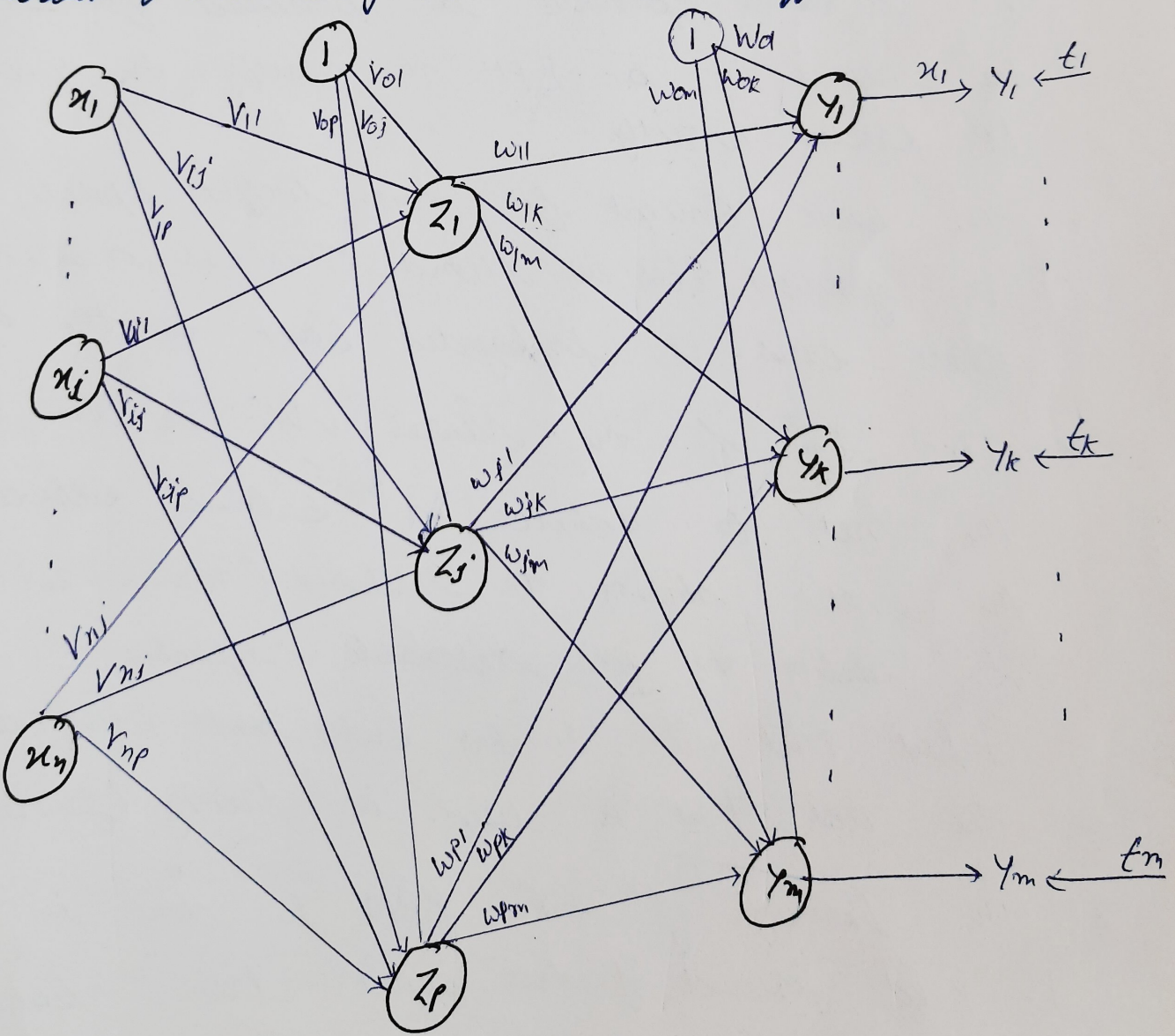
BACK PROPAGATION NETWORK

* It is one of the most important developments in Neural Networks
* This learning Algorithm is applied to Multi Layer feed forward Networks Consisting of processing elements with Continous differentiable Activation function.
* The Networks Associated with back propagation learning algorithm are also called back Propagation Networks
* For a given set of training I/P O/P pair, this algorithm provides a procedure for changing the weights in a BPN to classify the given I/P pattern Correctly.
* The basic Concept for this weight update Algo is gradient descent Method. This is a Method where error is propagated back to the hidden unit
* The Aim of the Neural N/w is to train the net to achieve a balance between the net's ability to respond (memorization) and its ability to give resonable responses to the Input that is similar but not identical to the one that is used in training (Generalization)
* The Training of the BPN is done in 3 stages
  1) Feed forward of the Input training pattern
  2) Calculation
  3) BACK Propagation of error, and updation of weights.

⇒ Architecture of BPN:

* A BPN N/w is a Multilayer, feed-forward neural N/w consisting of an input layer, a hidden layer & an output layer
* The Neurons present in the hidden & o/p layers have biases which are connected from the unit whose activations is always 1.
* The I/P are sent to the BPN & the output obtained from the net could be either binary (0,1) or bipolar (-1, +1)
* The Activation function could be any function which increases monotonically and is also differentiable.



Terminology used in flow chart and in Training Algorithm are as follows

x = I/P training vector $(x_1 \dots x_j \dots x_n)$

t = target output vector $(t_1 \dots t_k \dots t_m)$

α = Learning Rate

x_i = Input Unit i (since the I/P Layer uses Identity Activation function, the I/P & O/P signals here are same)

v_{oj} = bias on jth hidden Unit

w_{ok} = bias on kth output unit

z_j = Hidden unit j, The net Input to z_j is

$$z_{inj} = v_{oj} + \sum_{i=1}^{n} x_i \cdot v_{ij}$$

and the output is

$$z_j = f(z_{inj})$$

y_k = Output unit k. The net Input to y_k is

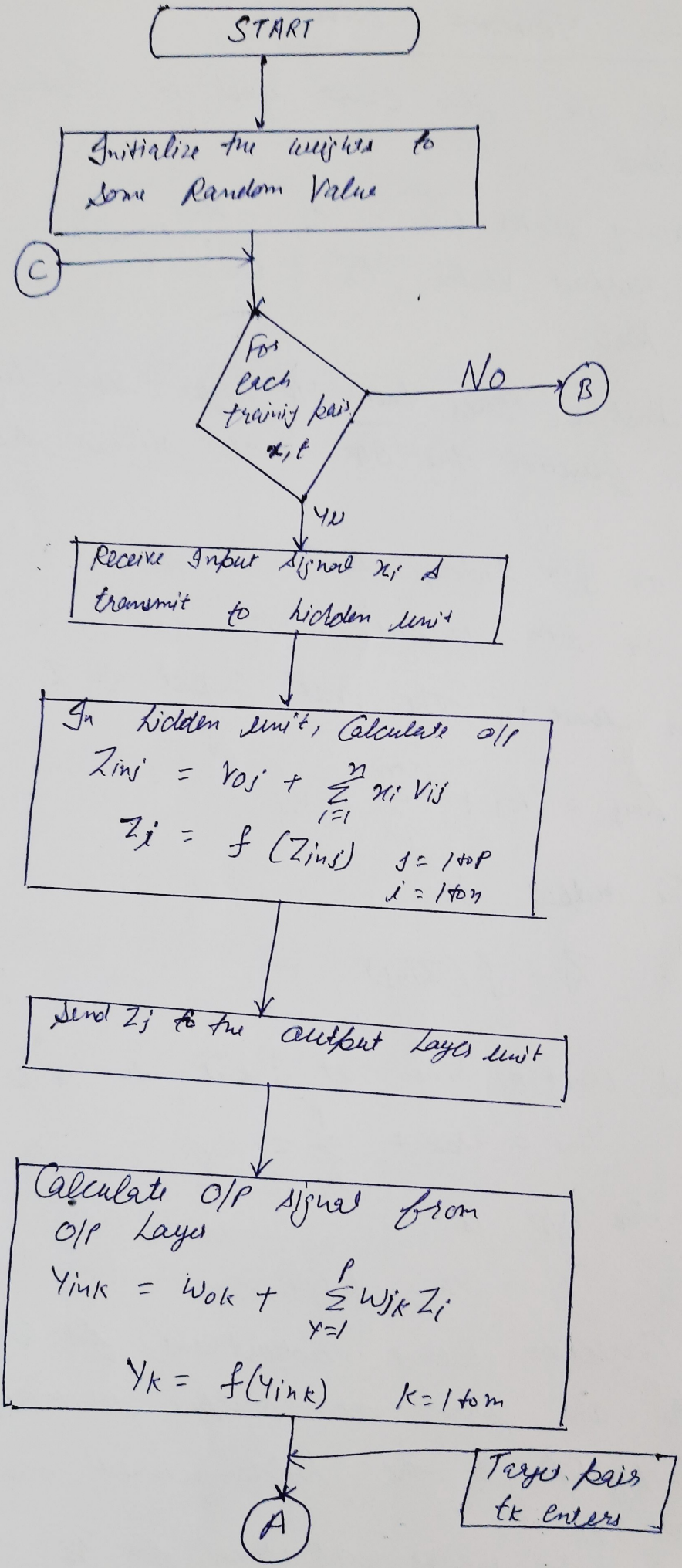
$$y_{ink} = w_{ok} + \sum_{j=1}^{p} z_j w_{jk}$$

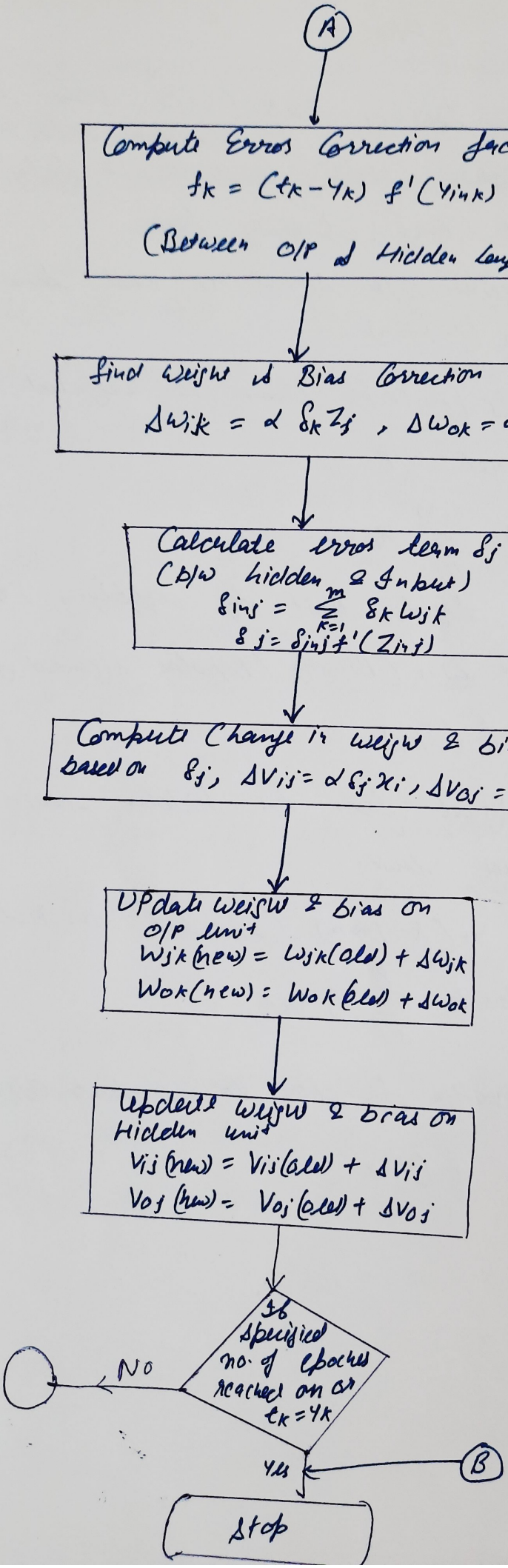
and the O/P is

$$y_k = f(y_{ink})$$

δ_k = Error Correction weight adjustment for w_{jk} that is due to an error at output unit y_k. which is back propagated to the hidden unit that feed into unit y_k

δ_j = Error Correction weight Adjustment for v_{ij} that is due to the back propagation of error to the hidden unit z_j


START
Initialize the weights to some Random Value
C
For each training pair x, t
No
B
Yes
Receive Input Signal x_i & transmit to hidden unit
In hidden unit, Calculate o/p
$Z_{inj} = v_{oj} + \sum_{i=1}^{n} x_i v_{ij}$
$Z_j = f(Z_{inj})$ $j = 1 \text{ to } p$ $i = 1 \text{ to } n$
Send Z_j to the output layer unit
Calculate O/P signal from O/P Layer
$y_{ink} = w_{ok} + \sum_{j=1}^{p} w_{jk} Z_j$
$Y_k = f(y_{ink})$ $k = 1 \text{ to } m$
Target pair tk enters
A


A
Compute Errors Correction factors
$\delta_k = (t_k - y_k) f'(y_{in_k})$
(Between O/P & Hidden Layer)
Find weight & Bias Correction term
$\Delta w_{jk} = \alpha \delta_k z_j$, $\Delta w_{ok} = \alpha \delta_k$
Calculate errors term δ_j
(b/w hidden & Input)
$\delta_{in_j} = \sum_{k=1}^{m} \delta_k w_{jk}$
$\delta_j = \delta_{in_j} f'(z_{in_j})$
Compute Change in weight & bias
based on δ_j, $\Delta v_{ij} = \alpha \delta_j x_i$, $\Delta v_{oj} = \alpha \delta_j$
Update weight & bias on
O/P unit
$w_{jk}(new) = w_{jk}(old) + \Delta w_{jk}$
$w_{ok}(new) = w_{ok}(old) + \Delta w_{ok}$
Update weight & bias on
Hidden unit
$v_{ij}(new) = v_{ij}(old) + \Delta v_{ij}$
$v_{oj}(new) = v_{oj}(old) + \Delta v_{oj}$
Is specified no. of epochs reached or $t_k = y_k$
NO
Yes
B
Stop

TRAINING ALGORITHM FOR BPN

Step 0: Initialize weights & Learning rate (Take some small random value)

Step 1: perform step 2-9 when stopping condition is false

Step 2: Perform steps 3-8 for each training pair

Step 3: Each I/P unit receive I/P signal x_i and sends it to the hidden unit ($i = 1$ to n)

Step 4: Each hidden unit Z_j ($j = 1$ to p) sums its weighted I/P signals to calculate net I/P

$$Z_{inj} = V_{oj} + \sum_{i=1}^{n} x_i v_{ij}$$

Calculate O/P of the hidden unit by applying its activation function over Z_{inj} (Binary / Bipolar sigmoid fcn)

$$Z_j = f(Z_{inj})$$

and send the O/P signal from the hidden unit to the I/P of O/P layer unit

Step 5: For each O/P unit Y_k ($k = 1$ to m) Calculate the net input

$$Y_{ink} = W_{ok} + \sum_{j=1}^{p} Z_j w_{jk}$$

and apply the Activation function to compute O/P signal.

$$Y_k = f(Y_{ink})$$

Back Propagation of error (Phase II)

Step 6: Each output unit y_k ($k = 1$ to m) receives a target pattern corresponding to the i/p training pattern & computes the error correction term

$$\delta_k = (t_k - y_k)\, f'(y_{in_k})$$

The derivative $f'(y_{in_k})$ can be calculated as in s/w. On the basis of the calculated error correction term, update the change in weights & bias

$$\Delta w_{jk} = \alpha\, \delta_k\, z_j \qquad \Delta w_{ok} = \alpha\, \delta_k$$

Also send δ_k to the hidden layer backward

Step 7: Each hidden unit (z_j $j = 1$ to p) sums its delta i/p from the output units

$$\delta_{in_j} = \sum_{k=1}^{m} \delta_k\, w_{jk}$$

The term δ_{in_j} gets multiplied with its derivative of $f(z_{in_j})$ to calculate the error term

$$\delta_j = \delta_{in_j} f'(z_{in_j})$$

The derivative $f'(z_{in_j})$ can be calculated depending on whether binary / bipolar sigmoidal function is used. On the basis of calculated δ_j, update the change in weight & bias:

$$\Delta v_{ij} = \alpha\, \delta_j\, x_i$$

$$\Delta v_{oj} = \alpha\, \delta_j$$

Weights & bias updation (Phase III):

Step 8: Each O/P unit (Y_k, $k = 1$ to m) updates the bias & weights

$$W_{jk}(new) = w_{jk}(old) + \Delta w_{jk}$$
$$W_{ok}(new) = W_{ok}(old) + \Delta W_{ok}$$

Each hidden unit (Z_j, $j = 1$ to p) updates it's bias & weights:

$$V_{ij}(new) = V_{ij}(old) + \Delta V_{ij}$$
$$V_{oj}(new) = V_{oj}(old) + \Delta V_{oj}$$

Step 9: Check for stopping condition. The stopping condition may be certain No of Epochs reached or when the Actual o/p equals the target o/p.

TESTING ALGO. IN BACK PROPAGATION NETWORK.

Step 0: Initialize the weights. The weights are taken from Training Algo.

Step 1: Perform steps 2-4 for each Input Vector

Step 2: Set the Activation of Input unit x_i ($i = 1$ to n)

Step 3: Calculate the net Input to hidden unit z and it's output for $j = 1$ to p

$$Z_{inj} = V_{oj} + \sum_{i=1}^{n} x_i V_{ij} \qquad Z_j = f(Z_{inj})$$

Step 4: Now Compute the output of the output layer unit for $k = 1$ to m

$$Y_{ink} = W_{ok} + \sum_{j=1}^{p} Z_j w_{jk}$$

$$Y_k = f(Y_{ink})$$

Use Sigmoid Activation function for Calculating the o/p.

FACTOR AFFECTING BACK PROPAGATION NETWOR

* The Training of a BPN BASED ON THE CHOICE of Various Parameter

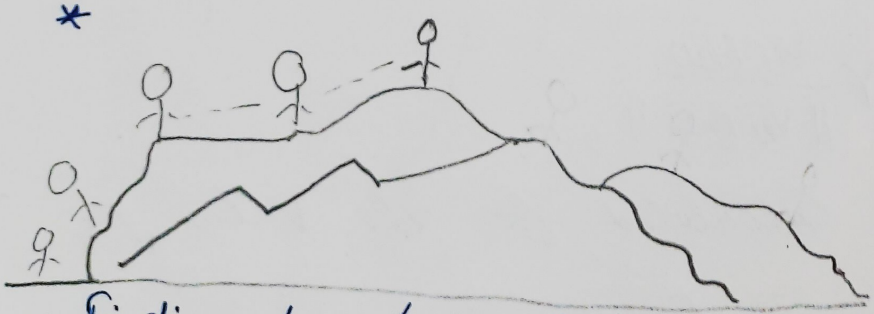
* The Convergence of the BPN is Based on some Important Learning factor such as the Initial weight, The Learning rate, The Updation rule, The size & nature of Training set, & its Architecture.

1 Initial weights: * Initial weights are Initialized at small random Value.

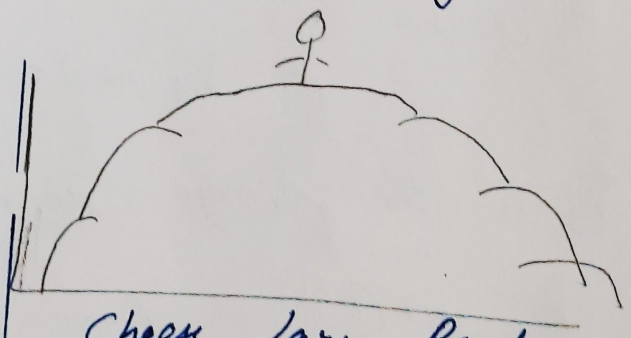
* The Choice of the Initial weights determine how fast the N/w Converge

* The Initial weights Can not be Very high because the Sigmoid Activation function Used in BPN may get saturated from the begining itself and the system may be stuck at a Local minima or at a very flat plateau at the starting point.

*



Finding the lowest point in a Hilly Landscape / Choose some small Random Value for weights Initially



Choose Large Random Value Initially

* One Method of choosing the weights W_{ij} is Choosing it's Range

$$\left[\frac{-3}{\sqrt{a_i}}, \frac{3}{\sqrt{a_i}}\right]$$

Where a_i is the Number of processing element j that feed forward to processing element i.

* The Initialization can also be done by a Method called Nyugen - Widrow Initialization.
* This type of Initialization leads to faster Convergence of Network.
* This Method is used for Improving the Learning ability of the Hidden layers.
* The Random Initialization of weights Connecting Input neurons to the Hidden neuron is obtained by the Equation

$$V_{ij}(new) = \gamma \frac{V_{ij}(old)}{\| V_{ij}(old) \|}$$

V_j is the Average weight Calculated for all Value of i

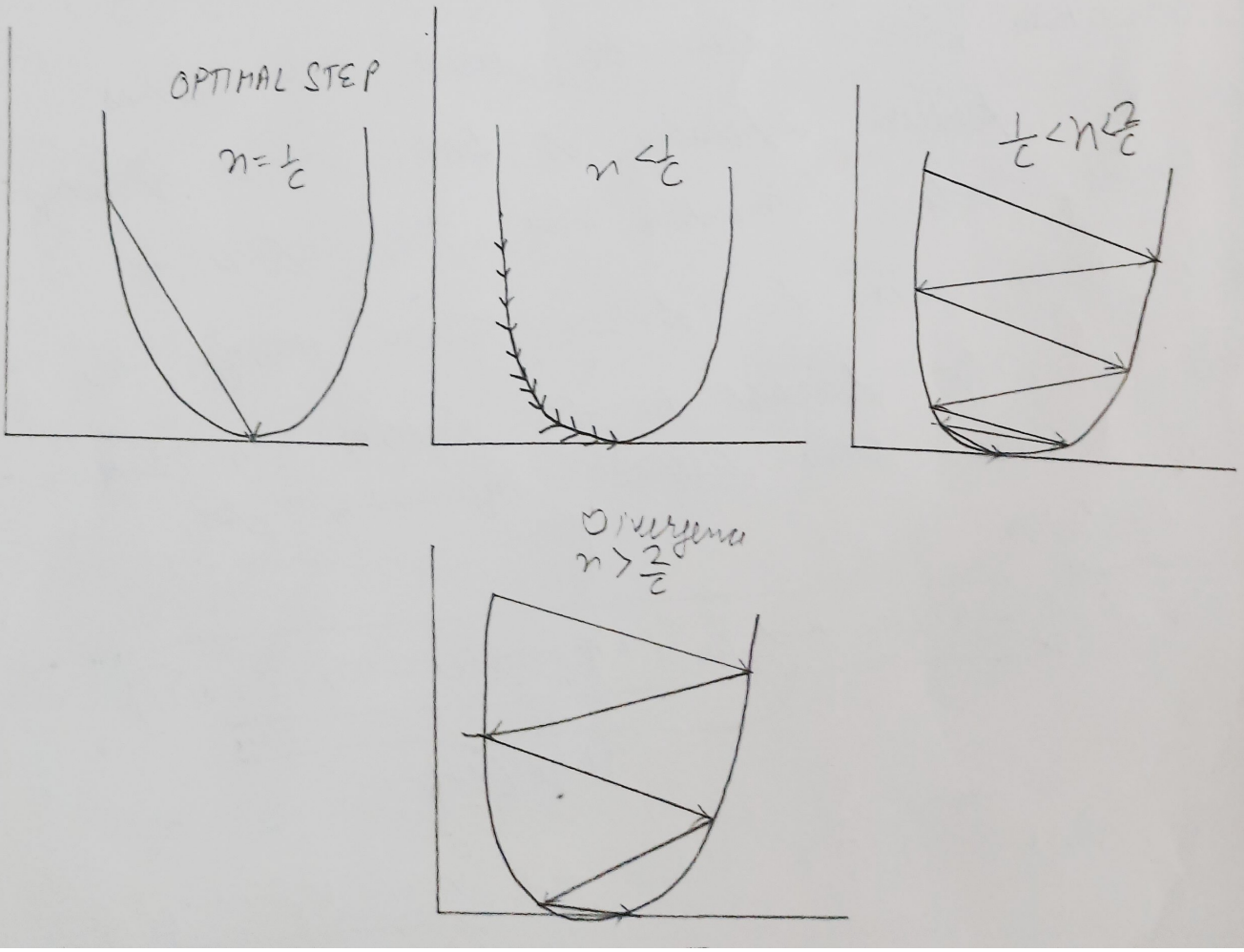
$\gamma = 0.7 (P)^{1/n}$ ("n" is the No of Input neurons &

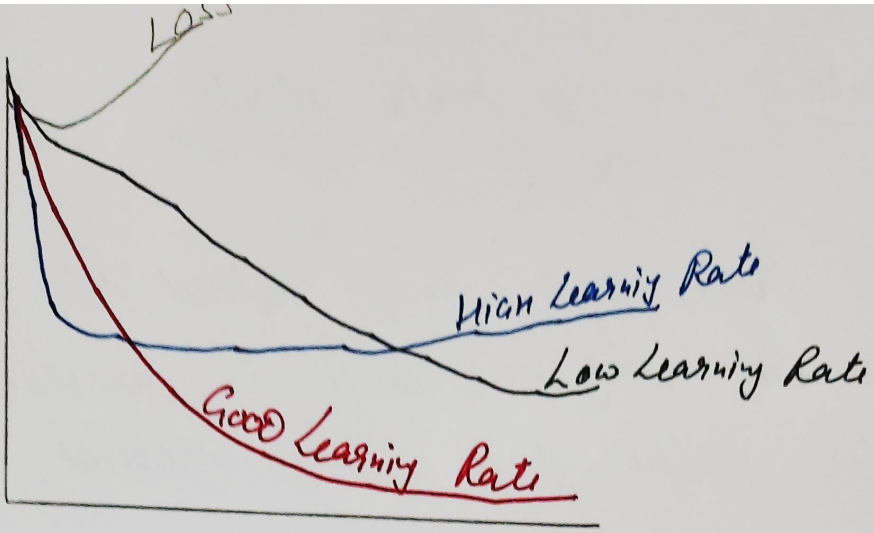
$\gamma \rightarrow$ Scale factor P is the No of Hidden neurons)

2 Learning Rate α: Learning rate affects the Convergence of the BPN.

* A large Value of α may speed up the Convergence but might result in overshooting the Minima
* Smaller Value of α may result undershoot.

* The Range of α from 10^{-3} to 10 has been used successfully for several back propagation algorithm experiment.
* Thus a large learning rate leads to rapid learning. but there is an oscillation of weights. While the lower learning rate leads to slower learning





3 Momentum factor: (η)

* The Momentum factor is denoted by $\eta \in [0,1]$ and the value of 0.9 is often used for the Momentum factor
* This approach is more useful when some training data are very different from the majority of data.
* A Momentum factor can be used with either pattern by pattern updating or batch mode updating
* In Batch mode, Momentum has the effect of complete averaging over the pattern. Even though the averaging is only partial in the pattern by pattern mode. It leaves some useful information for weight updation.

$$w_{jk}(t+1) = w_{jk}(t) + \underbrace{\alpha \delta_k z_j + \eta [w_{jk}(t) - w_{jk}(t-1)]}_{\Delta w_{jk}(t+1)}$$

and

$$v_{ij}(t+1) = v_{ij}(t) + \underbrace{\alpha \delta_j x_i + \eta [v_{ij}(t) - v_{ij}(t-1)]}_{\Delta v_{ij}(t+1)}$$

Momentum also help in fast convergence

4 GENERALIZATION

* A Network is said to be Generalized when it Sensibly Interpolates with Input Networks that are new to the Network
* When there are many trainable parameters for the given amount of training data the Network learn but not Generalize & with small no of trainable parameters the N/w fails to learn the training data & perform very poorly on test data
* Solution to this problem is to monitor the error on the test set & terminate when the error increase
* For Improving the ability of the Network to generalize from a training data set to a test data set
* It is desirable to make small changes in the I/p space of a pattern without changing the output components.
* However Computationally, this Method is very expensive
* Small nets are preffered than larger one

5 Number of Training data:

* The training data should be sufficient & proper.
* There exist a thumb rule, which states that the training data should cover the entire expected Input space. while training, training-Vectos pair should be selected randomly from the set

⇒ Assume, the Input space was being Linearly separable into 'L' disjoint regions with their boundaries being part of Hyper plane.

⇒ 'T' be the Lower bound of the No of Training patterns

→ Choosing T such that T/L >>1 will allow the network to discriminate pattern classes using fine piecewise Hyperplane partitioning.

6 Number of Hidden Layer nodes:

* The Number of Hidden units required for an application need to be determined seperately.
* The size of a Hidden Layer is usually determined Experimentally.
* If the network does not Converge to a solution, It may need more Hidden nodes. On the other Hand if the N/w Converges, the user may try a very few Hidden nodes & then settle finally on a size based on overall system performance

APPLICATIONS of BACK PROPAGATION NETWORK

* It is used in the field of speech Recognition
* The neural network is trained to enunciate each letter of a word & a sentence
* It is used in the field of Character & face recognition.