-1, yin<0

Adaline Network (Adaptive linear Neuron)

Network that uses linear activation are called linear unit. A ~~single~~ network with single linear unit is adaline.

* It uses bipolar activation fn.
* Bias act like adjustable weight
* Only one o/p unit.
* Trained using delta rule/least mean square rule (LMS)/Widraw-HOFF rule.
* Delta rule is used to minimize the mean square error b/w the activation and the target.
* Delta rule is derived from ~~Rosen~~ Gradient descent method.
* Delta rule updates the weight b/w the connection so as to minimise the differen

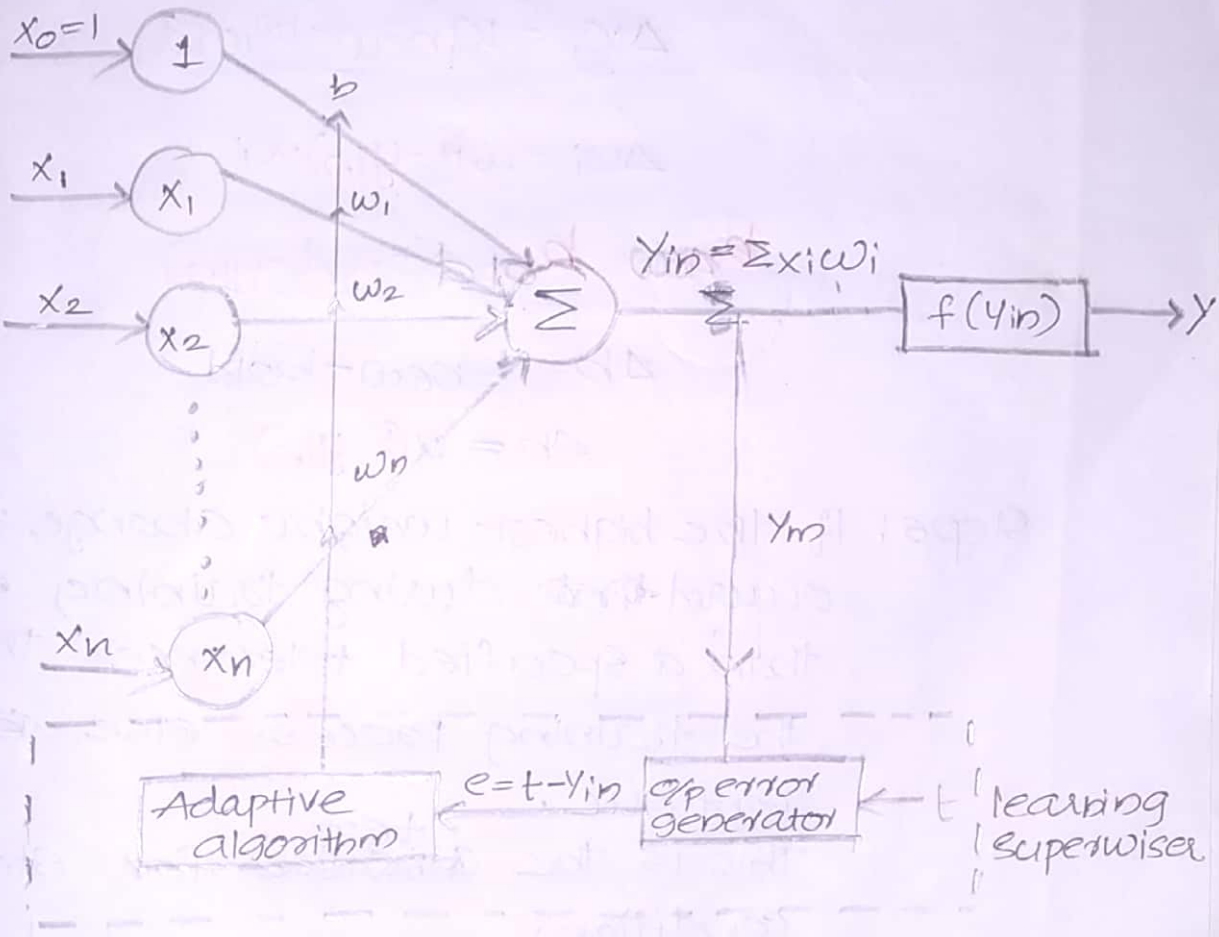
b/w net input to o/p unit and target value.

* The aim of the delta rule is to minimise the error over all training patterns.
* Delta rule for adjusting the weight of ith pattern where i=1 to n is

$$\Delta w_i = \alpha (t - y_{in}) x_i$$

(error)

Adaline Model



Training Algorithm

Step 0: weight & bias are set to some random values (not zero). Set the learning rate parameter, α $(0.1 < \alpha \leq 1.0)$

Step 1: Perform step 2-6 when stopping condition is false.

Step 2: Perform step 3-5 for each bipolar training pair (s:t)

Step 3: Set the activation for the i/p unit i=1 to n

ie, $x_i = s_i$

Step 4: Calculate the net i/p

$$y_{in} = b + \sum_{i=1}^{n} x_i w_i$$

Step 5: Updates the weights & bias for i=1 to n

$$w_{inew} = w_{iold} + \alpha (t - y_{in}) x_i$$

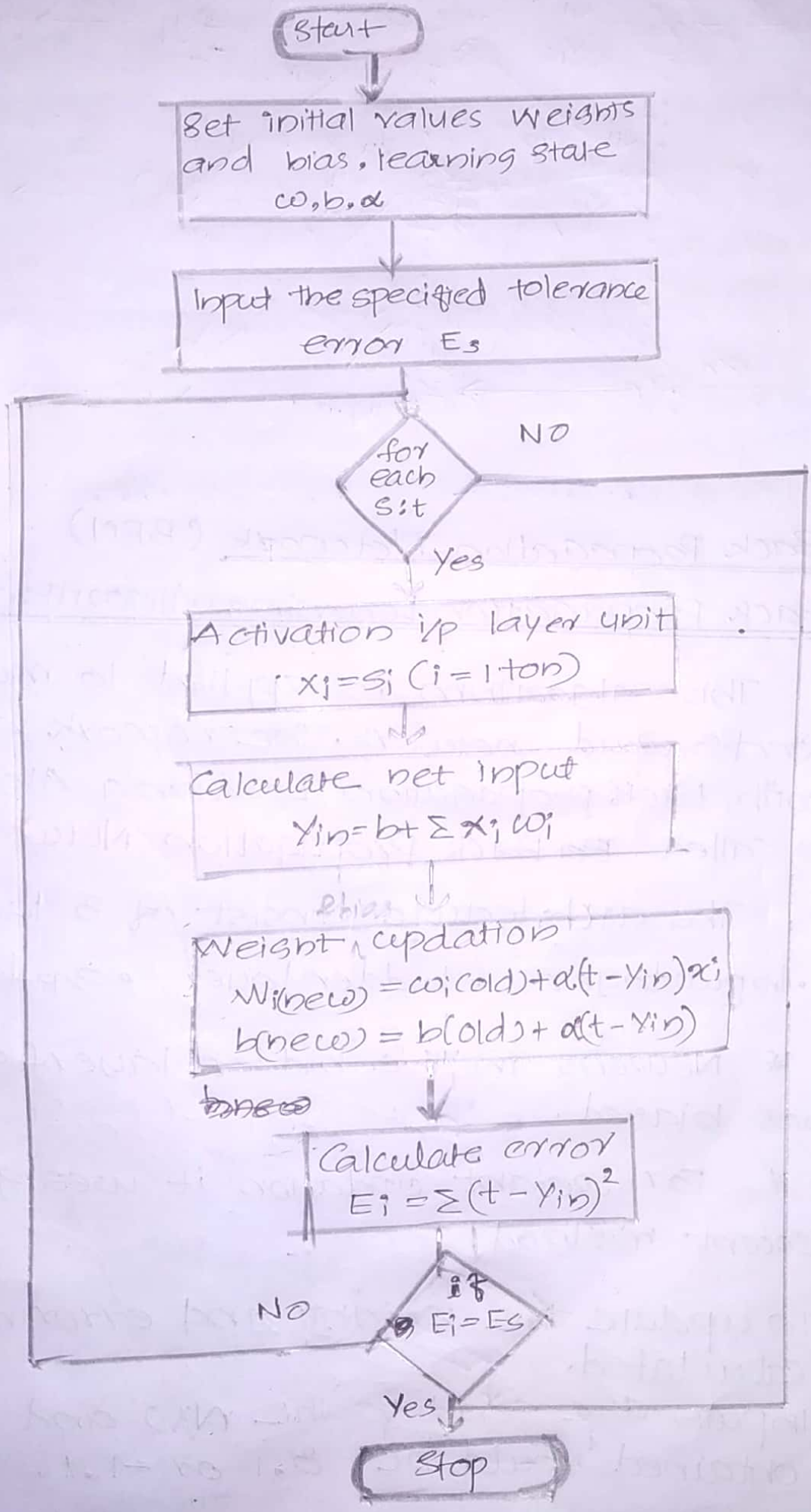
$$\Delta w_i = \alpha (t - y_{in}) x_i$$

$$b_{new} = b_{old} + \alpha (t - y_{in})$$

$$\Delta b = \alpha (t - y_{in})$$

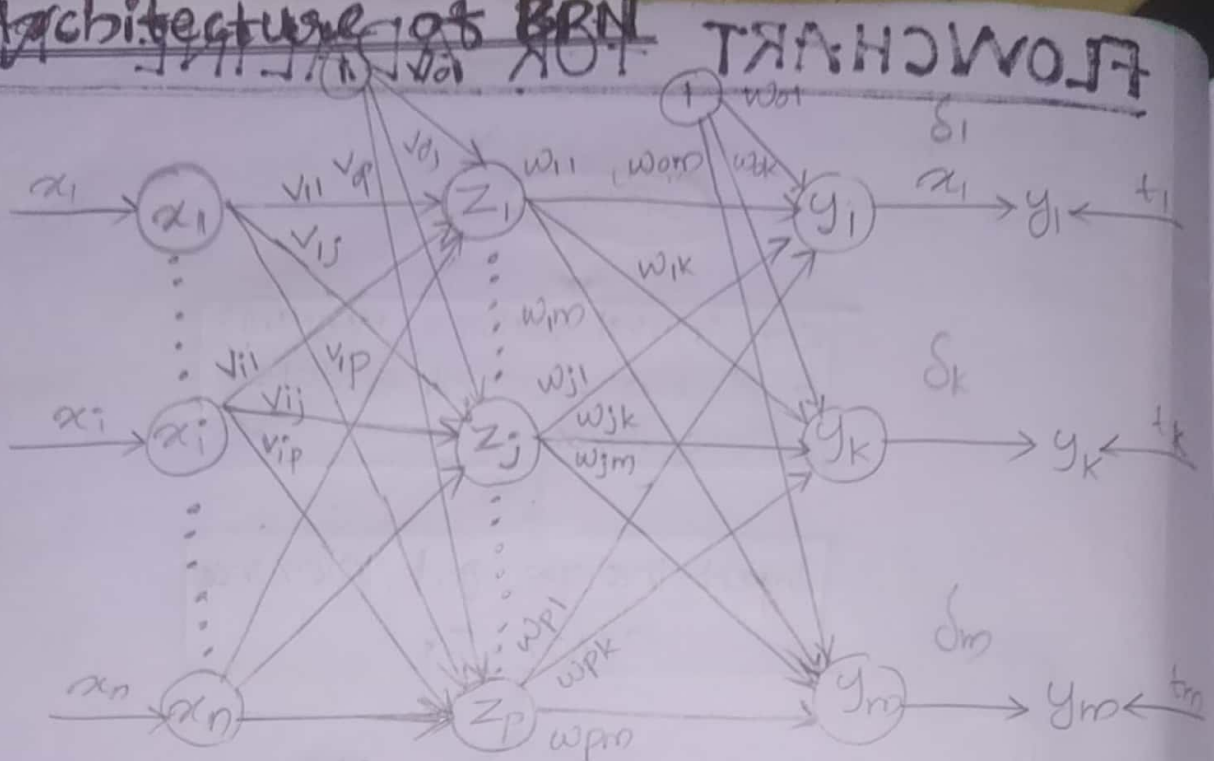
Step 6: If the highest weight change that occured during training smaller than a specified tolerance then stop the training process else continue. This is the test for stopping condition

FLOWCHART FOR ADALINE N/W



23/09/14

Architecture of BPN



Back Propagation Network (BPN)

Back Propagation Learning Algorithm

This algorithm is applied to multi layer feedforward network. The network associated with back propagation Learning Algorithm is called Back Propagation Network.

The arthitecture consist of 3 layers.

- Input layer
- hidden layer
- o/p layer

* Neurons in the hidden layer & o/p layer has biased.

* For weight updation it uses gradient descent method.

* To update the weight and error must be calculated.
* Inputs are send to the N/W and o/p obtained could be 0,1 or -1,1.

x_i — i^{th} neuron in the i/p unit/layer

v_{oj} — Bias on the j^{th} neuron in the hidden layer

w_{ok} — Bias on the k^{th} neuron in the o/p layer

z_j — j^{th} neuron in the hidden layer

y_k — k^{th} neuron in the o/p layer.

δ_k: error correction weight adjustment for w_{jk}, ie, due to the error at the o/p unit y_k

δ_j: error correction weight adjustment for v_{ij} due to the back propagation of the error in the hidden layer z_j.

→ There are 3 stages in the training of BPN

1. Feed forward phase
2. Back propagation of error
3. Weight and bias updation.

Training Algorithm

Step 0: Initialize weights, bias and learning rate.

Step 1: Performs step 2 to 9 when stopping condition is false.

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

Feed forward phase (phase 1)

Step 3: Each i/p unit receives the i/p signals x_i and send to the hidden unit.

Step 4: Each hidden unit z_j calculate its net input

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

Step : Calculate the o/p of the hidden unit by applying the activation fn: ie,

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

And send the o/p signal from the hidden layer to the o/p layer.

Step 5: For each neuron in the o/p layer

Calculate the net i/p

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

Back propagation of error (Phase: 2)

Step 6: For each o/p unit y_k where $k=1$ to m receives a target pattern corresponding to the i/p training pattern and compute the error correction term.

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

where, $f'(y_{ink}) = f(y_{in})(1 - f(y_{in}))$

Step 7: On the basis of the calculated error update the change in weight and bias

$$\Delta w_{jk} = \alpha \delta_k z_j \quad \text{Change in weight}$$

$$\Delta w_{ok} = \alpha \delta_k \quad \text{Change in bias}$$

Also send δ_k to the hidden layer backwards

Step 7 : For each hidden unit $z_j, j=1$ to p sum its delta i/p from the o/p unit.

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

Now we calculate $\delta_j = \delta_{inj} \cdot f'(z_{inj})$

On the basis of calculated δ_j update the change in weight and bias.

~~$\delta V_{ij} = \alpha$~~

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

$$\Delta V_{0j} = \alpha \delta_j$$

Weight and bias updation (phase : 3)

Step 8 : each o/p unit $y_k, k=1$ to m update the bias & weight.

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

$$w_{0k(new)} = w_{0k(old)} + \Delta w_{0k}$$

For each unit z_j $j=1$ to p update the bias & weight.

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

$$V_{0j(new)} = V_{0j(old)} + \Delta V_{0j}$$

Step 9 : Check for the stopping condition. The stopping condition may be certain no. of epochs reach or when the actual o/p equals the target o/p.