Module 2

Adaline Madaline

Syllabus

Training Techniques for ANNs 10 CO2

2.1 Introduction to supervised and unsupervised learning, Adaline and Madaline

2.2 Hebbian learning, Perceptron Learning, Delta learning rule, Widrow Hoff learning, Winner take all Learning Rule , Out star learning

2.3 Multilayer Feedforward Network, Error Back Propagation Training, Learning factors.

Adaptive Linear Neuron (Adaline)

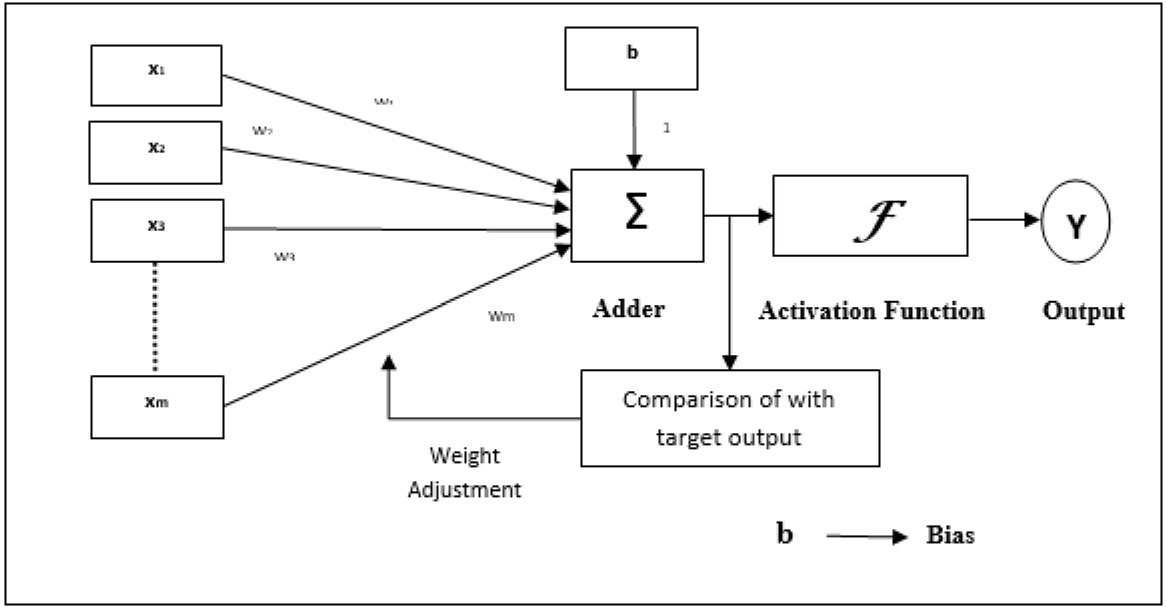
Adaline which stands for Adaptive Linear Neuron, is a network having a single linear unit.

It was developed by Widrow and Hoff in 1960.

Some important points about Adaline are as follows –

- It uses bipolar activation function.
- It tries to minimize the Mean-Squared Error (MSE) between the actual output and the desired/target output.
- The weights and the bias are adjustable.

Architecture



Algorithm

Step 1: Initialize the following to start the training –Weights, Bias, Learning rate α

Step 2: While the stopping condition is False do steps 3 to 7.

Step 3: for each training set perform steps 4 to 6.

Step 4: Set activation of input unit $x_i = s_i$ for (i=1 to n).

Step 5: compute net input to output unit $y_{in} = \sum w_i x_i + b$

Here, b is the bias and n is the total number of neurons.

Step 6: Update the weights and bias for i=1 to n

$$w_i(new) = w_i(old) + \alpha(t - y_{in})x_i$$
$$b(new) = b(old) + (t - y_{in})$$

and calculate $error : (t - y_{in})^2$

Step 7: Test the stopping condition. The stopping condition may be when the weight changes at a low rate or no change.

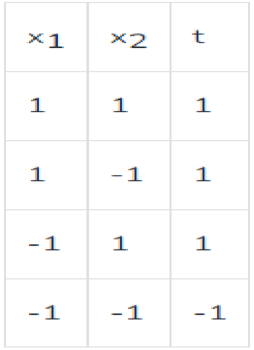
Problem:Design OR gate using Adaline Network.

Solution :

Initially, all weights are assumed to be small random values, say 0.1, and set learning rule to 0.1.

Also, set the least squared error to 2.

The weights will be updated until the total error is greater than the least squared error.



x1	x2	t
1	1	1
1	-1	1
-1	1	1
-1	-1	-1

Calculate the net input $y_{in} = \sum w_i x_i + b$

(when x_1=x_2=1) $y_{in} = 0.1 \times 1 + 0.1 \times 1 + 0.1 = 0.3$

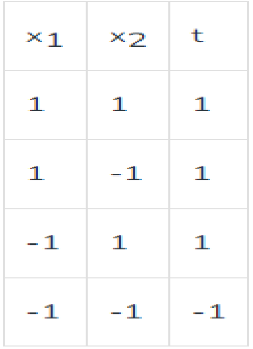
Now compute, $(t-y_{in})$=(1-0.3)=0.7, calculate the error $error = (t - y_{in})^2 = 0.7^2 = 0.49$

Now, update the weights and bias

$$w_i(new) = w_i(old) + \alpha(t - y_{in})x_i$$
$$b(new) = b(old) + (t - y_{in})$$

$$w_1(new) = 0.1 + 0.1(1 - 0.3)1 = 0.17$$
$$w_2(new) = 0.1 + 0.1(1 - 0.3)1 = 0.17$$

$$b(new) = 0.1 + 0.1(1 - 0.3) = 0.17$$



x1	x2	t
1	1	1
1	-1	1
-1	1	1
-1	-1	-1

x_1	x_2	t	y_{in}	$(t-y_{in})$	Δw_1	Δw_2	Δb	w_1 (0.1)	w_2 (0.1)	b (0.1)	$(t-y_{in})$^2
1	1	1	0.3	0.7	0.07	0.07	0.07	0.17	0.17	0.17	0.49
1	-1	1	0.17	0.83	0.083	-0.083	0.083	0.253	0.087	0.253	0.69
-1	1	1	0.087	0.913	-0.0913	0.0913	0.0913	0.1617	0.1783	0.3443	0.83
-1	-1	-1	0.0043	-1.0043	0.1004	0.1004	-0.1004	0.2621	0.2787	0.2439	1.01

This is epoch 1 where the total error is 0.49 + 0.69 + 0.83 + 1.01 = 3.02 so more epochs will run until the total error becomes less than equal to the least squared error i.e 2.

```
# Predict from the evaluated weight and bias of adaline
def prediction(X,w,b):
        y=[]
        for i in range(X.shape[0]):
                x = X[i]
                y.append(sum(w*x)+b)
        return y
prediction(x,w,b)
```

```
[array([1.0192042]),
array([0.99756877]),
array([0.99756877]),
array([-0.99756877])]
```

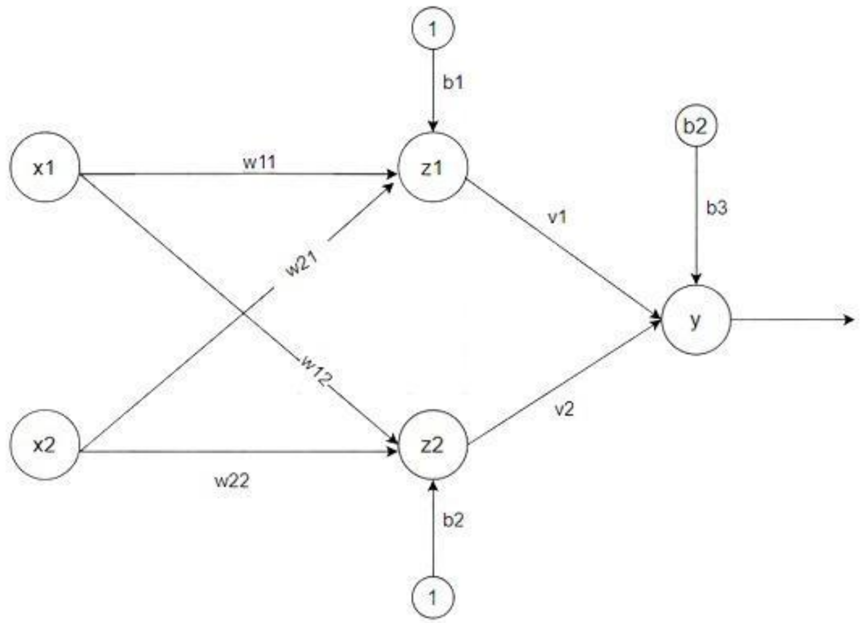
Output:

```
Error : [2.33228319]
Error : [1.09355784]
Error : [0.73680883]
Error : [0.50913731]
Error : [0.35233593]
Error : [0.24384625]
Error : [0.16876305]
Error : [0.11679891]
Error : [0.08083514]
Error : [0.05594504]
Error : [0.0387189]
Error : [0.02679689]
Error : [0.01854581]
Error : [0.01283534]
Error : [0.00888318]
Error : [0.00614795]
Error : [0.00425492]
Error : [0.00294478]
Error : [0.00203805]
Error : [0.00141051]
Error : [0.0009762]
weight : [0.01081771 0.01081771 0.98675106] Bias :
[0.01081771]
```

Madaline (Multiple Adaptive Linear Neuron) :

- The Madaline(supervised Learning) model consists of many Adaline in parallel with a single output unit. The Adaline layer is present between the input layer and the Madaline layer hence Adaline layer is a hidden layer. The weights between the input layer and the hidden layer are adjusted, and the weight between the hidden layer and the output layer is fixed.
- It may use the majority vote rule, the output would have an answer either true or false. Adaline and Madaline layer neurons have a bias of ‘1’ connected to them. use of multiple Adaline helps counter the problem of non-linear separability.

Architecture



- There are three types of a layer present in Madaline
- First input layer contains all the input neurons, the Second hidden layer consists of an adaline layer, and weights between the input and hidden layers are adjustable and the third layer is the output layer the weights between hidden and output layer is fixed they are not adjustable.

Algorithm

Step 1: Initialize weight and set learning rate α.

$v_1=v_2=0.5$, $b=0.5$

other weight may be a small random value.

Step 2: While the stopping condition is False do steps 3 to 9.

Step 3: for each training set perform steps 4 to 8.

Step 4: Set activation of input unit xi = si for (i=1 to n).

Step 5: compute net input of Adaline unit

$z_{in1} = b_1 + x_1w_{11} + x_2w_{21}$

$z_{in2} = b_2 + x_1w12 + x_2w_{22}$

Step 6: for output of remote Adaline unit using activation function given below:

Activation function f(z) =$1 \text{ if } z \geq 0 \ (and) \ (-1) \text{ if } z < 0$

z1=f(zin1)

z2=f(zin2)

Algorithm

Step 7: Calculate the net input to output.

$y_{in} = b_3 + z_1v_1 + z_2v_2$

Apply activation to get the output of the net

$y=f(y_{in})$

Step 8: Find the error and do weight updation

if $t \neq y$ then t=1 update weight on z(j) unit whose next input is close to 0.

if $t = y$ no updation

$w_{ij}(new) = w_{ij}(old) + \alpha(t-z_{inj})x_i$

$b_j(new) = b_j(old) + \alpha(t-z_{inj})$

if t=-1 then update weights on all unit z_k which have positive net input

Step 9: Test the stopping condition; weights change all number of epochs.