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)

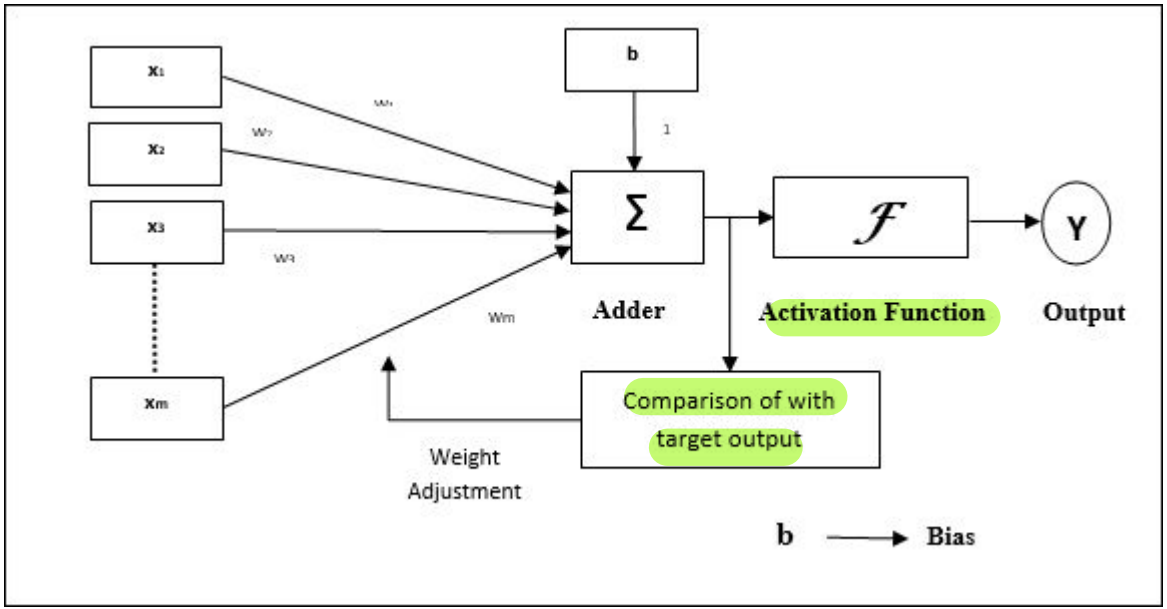
Adaline which stands for Adaptive Linear Neuron, is a network having a single linear unit.

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.

x_1	x_2	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)
```

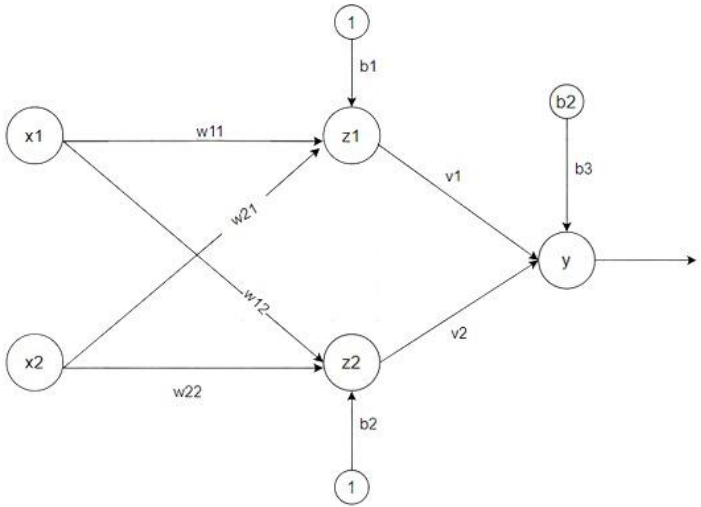
```
[array([1.0192042]),
array([0.99756877]),
array([0.99756877]),
array([-0.99756877])]
```

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 weights between the input layer and the hidden layer are adjusted, and the weight between the hidden layer and the output layer is fixed.
- Adaline and Madaline layer neurons have a bias of ‘1’ connected to them.



Algorithm

Step 1: Initialize weights, bias and learning rate α to small random values
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

$$z_{in1} = b_1 + x_1 w_{11} + x_2 w_{21}$$
$$z_{in2} = b_2 + x_1 w12 + x_2 w_{22}$$

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.

$$y_{in} = b_3 + z_1 v_1 + z_2 v_2$$

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.

Using Madaline network, implement XOR function with bipolar inputs and targets. Assume the required parameters for training of the network.

Solution: The training pattern for XOR function is given in Table

x_1	x_2	1	t
1	1	1	−1
1	−1	1	1
−1	1	1	1
−1	−1	1	−1

input sample, X1 = 1, X2 = I, target t = -1, and learning rate a equal to 0.5:

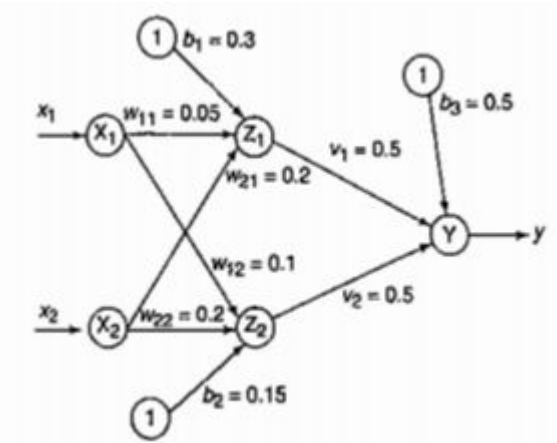


Figure : Network architecture of Madaline for XOR functions .(initial weights given).

- **Calculate net input to the hidden units:**

$$z_{in1} = b_1 + x_1 w_{11} + x_2 w_{21}$$
$$= 0.3 + 1 \times 0.05 + 1 \times 0.2 = 0.55$$
$$z_{in2} = b_2 + x_1 w_{12} + x_2 w_{22}$$
$$= 0.15 + 1 \times 0.1 + 1 \times 0.2 = 0.45$$

Calculate the output z_1, z_2 by applying the activations over the net input computed. The activation function is given by

$$f(z_{in}) = \begin{cases} 1 & \text{if } z_{in} \geq 0 \\ -1 & \text{if } z_{in} < 0 \end{cases}$$

Hence,

$$z_1 = f(z_{in1}) = f(0.55) = 1$$
$$z_2 = f(z_{in2}) = f(0.45) = 1$$

- After computing the output of the hidden units, then find the net input entering into the output unit:

$$y_{in} = b_3 + z_1 v_1 + z_2 v_2$$
$$= 0.5 + 1 \times 0.5 + 1 \times 0.5 = 1.5$$

- Apply the activation function over the net input y_{in} to calculate the output y:

$$y = f(y_{in}) = f(1.5) = 1$$

- Since $t \neq y$, weight updation has to be performed. Also since $t = -1$, the weights are updated on z_1 and z_2 that have positive net input. Since here both net inputs z_{in1} and z_{in2} are positive, updating the weights and bias on both hidden units, we obtain

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

$$w_{11}(\text{new}) = w_{11}(\text{old}) + \alpha(t - z_{in1})x_1$$
$$= 0.05 + 0.5(-1 - 0.55) \times 1 = -0.725$$
$$w_{12}(\text{new}) = w_{12}(\text{old}) + \alpha(t - z_{in2})x_1$$
$$= 0.1 + 0.5(-1 - 0.45) \times 1 = -0.625$$
$$b_1(\text{new}) = b_1(\text{old}) + \alpha(t - z_{in1})$$
$$= 0.3 + 0.5(-1 - 0.55) = -0.475$$

$$w_{21}(\text{new}) = w_{21}(\text{old}) + \alpha(t - z_{in1})x_2$$
$$= 0.2 + 0.5(-1 - 0.55) \times 1 = -0.575$$
$$w_{22}(\text{new}) = w_{22}(\text{old}) + \alpha(t - z_{in2})x_2$$
$$= 0.2 + 0.5(-1 - 0.45) \times 1 = -0.525$$
$$b_2(\text{new}) = b_2(\text{old}) + \alpha(t - z_{in2})$$
$$= 0.15 + 0.5(-1 - 0.45) = -0.575$$

Inputs			Target											
x_1	x_2	1	(t)	z_{in1}	z_{in2}	z_1	z_2	y_{in}	y	w_{11}	w_{21}	b_1	w_{12}	w_{22}	b_2
EPOCH-1															
1	1	1	−1	0.55	0.45	1	1	1.5	1	−0.725	−0.58	−0.475	−0.625	−0.525	−0.575
1	−1	1	1	−0.625	−0.675	−1	−1	−0.5	−1	0.0875	−1.39	0.34	−0.625	−0.525	−0.575
−1	1	1	1	−1.1375	−0.475	−1	−1	−0.5	−1	0.0875	−1.39	0.34	−1.3625	0.2125	0.1625
−1	−1	1	−1	1.6375	1.3125	1	1	1.5	1	1.4065	−0.069	−0.98	−0.207	1.369	−0.994
EPOCH-2															
1	1	1	−1	0.3565	0.168	1	1	1.5	1	0.7285	−0.75	−1.66	−0.791	−0.207	−1.58
1	−1	1	1	−0.1845	−3.154	−1	−1	−0.5	−1	1.3205	−1.34	−1.068	−0.791	0.785	−1.58
−1	1	1	1	−3.728	−0.002	−1	−1	−0.5	−1	1.3205	−1.34	−1.068	−1.29	0.785	−1.08
−1	−1	1	−1	−1.0495	−1.071	−1	−1	−0.5	−1	1.3205	−1.34	−1.068	−1.29	1.29	−1.08
EPOCH-3															
1	1	1	−1	−1.0865	−1.083	−1	−1	−0.5	−1	1.32	−1.34	−1.07	−1.29	1.29	−1.08
1	−1	1	1	1.5915	−3.655	1	−1	0.5	1	1.32	−1.34	−1.07	−1.29	1.29	−1.08
−1	1	1	1	−3.728	1.501	−1	1	0.5	1	1.32	−1.34	−1.07	−1.29	1.29	−1.08
−1	−1	1	−1	−1.0495	−1.701	−1	−1	−0.5	−1	1.32	−1.34	−1.07	−1.29	1.29	−1.08