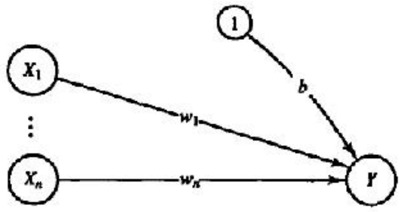
3- ADALINE (adaptive linear Neuron) [Widrow & Hoff, 1960]

- Typically, Adaline uses bipolar (-1, +1) activation for its input signals and its target outputs (although it is not restricted to such value).
- The weights from the input units to the Adaline are adjustable.
- In general the Adaline can be trained using the delta rule (also known as the least mean square (LMS) or Widrow–Hoff rule.
- In Adaline there is only one output unit.



The structure of Adaline

$$net = b + \sum_i x_i w_i$$

After training, if the net is being used for pattern classification in which the desired output is either +1 or -1, a threshold function is applied to the net input to obtain the activation.

$$y = f(net) = \begin{cases} 1 & \text{if } net \geq \theta \\ -1 & \text{if } net < \theta \end{cases}$$

$$b\,(new) = b\,(old) + \alpha(t - net)$$
$$w_i\,(new) = w_i\,(old) + \alpha(t - net)x_i$$

- For a single neuron, the suitable value of the learning rate **α** is to be: $0.1 \leq n*\alpha \leq 1$, where n is the total number of the input units.
- The learning rule minimizes the mean squared error between the activation and the target value.

$$E = \sum_{p=1}^{m} (t_p - \sum_{i=0}^{n} (x_{i,p} w_i)^2$$

$$E = \sum_{p=1}^{m} (t_p - net_p)^2$$

Where t_p is the associated target for the input pattern ***p*.** As an example, if the neural net represents logic gate with two input, then the total squared error is

$$E = \sum_{p=1}^{m} (t_p - (x_{1,p} w_1 + x_{2,p} w + w_0))^2$$

For AND function gate, if $w_1 = w_2 = 1$ and b = -1.5, then,

Pattern	binary		bipolar		
P	$x_{1,p}$	$x_{2,p}$	t_p	net_p	E_p
1	1	1	1	0.5	0.25
2	1	0	-1	-.5	0.25
3	0	1	-1	-.5	0.25
4	0	0	-1	-.5	0.25

E= $\sum E_p$= 1

The separating line is: **$x_2 = -x_1 + 1.5$** (i.e., the weights that minimize this error are: w1=w2=1, b=-1.5)

4- Delta learning rule:

4.1- delta rule for single output unit:

The delta rule change the weights of the neural connections so as to minimize the difference between the net input "net" and the target value "t",

$$\Delta w_i = \alpha\, (t - net)\, x_i \qquad \text{...........(1)}$$

$$net = \sum_{i=1}^{n} x_i w_i \qquad \text{...............(2)}$$

the squared error for a particular training pattern is:

$$E = (t - net)^2 \qquad \text{...........(3)}$$

- ***E*** is a function of all of the weights, w_i, i=1,2, …., n
- The gradient of ***E*** is the vector consisting of the partial derivatives of ***E*** with respect to each of the weights

$$\nabla E = \frac{\partial E}{\partial w_i} \qquad \text{...........(4)}$$

- The gradient gives the direction of most rapid increase in ***E***, the opposite direction gives the most rapid decrease in the error, i. e., the error can be reduced by adjusting the weight w_i in the direction of $-\partial E/\partial w$.

Since, $net = \sum x_i w_i$; and $E = (t - net)^2$

Thus,

$$\nabla E = \frac{\partial E}{\partial w_i} = -2(t - net)\frac{\partial net}{\partial w_i} \qquad \text{..............(5)}$$

$$= -2\,(t - net)\, x_i \qquad \text{...........(6)}$$

Thus the local error will be reduced most rapidly by adjusting the weight to the delta rule:

$$\boldsymbol{\Delta w_i = \alpha\ (t - net)\ x_i}$$

4.2- Delta rule for several output units:

Delta rule can be extended to more than single output unit, then for the output unit $\boldsymbol{y_J}$ we have:

$$\boldsymbol{\Delta w_{IJ} = \alpha\ (t_J - net_J)\ x_I}$$

The squared error for a particular training pattern is

$$E = \sum_{J=1}^{m} (t_J - net_J)^2$$

Again,

$$\nabla E = \frac{\partial E}{\partial w_{iJ}} = \frac{\partial}{\partial w_{iJ}} \sum_{j=1}^{m} (t_j - net_j)^2$$

$$= \frac{\partial}{\partial w_{iJ}} (t_J - net_J)^2$$

The weight $\boldsymbol{W_{IJ}}$ influence the error only on output unit $\boldsymbol{y_J}$ and:

$$\boldsymbol{net_J = \sum_{i=1}^{n} x_i w_{iJ}}$$

$$\nabla E = \frac{\partial E}{\partial w_{iJ}} = -2(t_J - net_J) \frac{\partial net_J}{\partial w_{iJ}}$$

$$\boldsymbol{= -2\ (t_J - net_J)\ x_I}$$

Adjusting the weights according to delta rule for a given learning rate:

$$\boldsymbol{\Delta w_{IJ} = \alpha\ (t_J - net_J)\ x_I}$$

5- MADALINE

Adalines can be combined so that the output from some of them becomes input for others of them, then the net becomes multilayer. Such a multilayer net, known as a MADALINE.

In this section we will discuss a MADALINE with one hidden layer (composed of two hidden Adaline units) and one output Adaline unit. Generalizations to more hidden units, more output units, and more hidden layers, are straightforward.

Architecture

A simple MADALINE net is illustrated in the following figure. The use of the hidden units, Z_1 and Z_2 give the net computational capabilities not found in single layer nets, but also complicate the training process.

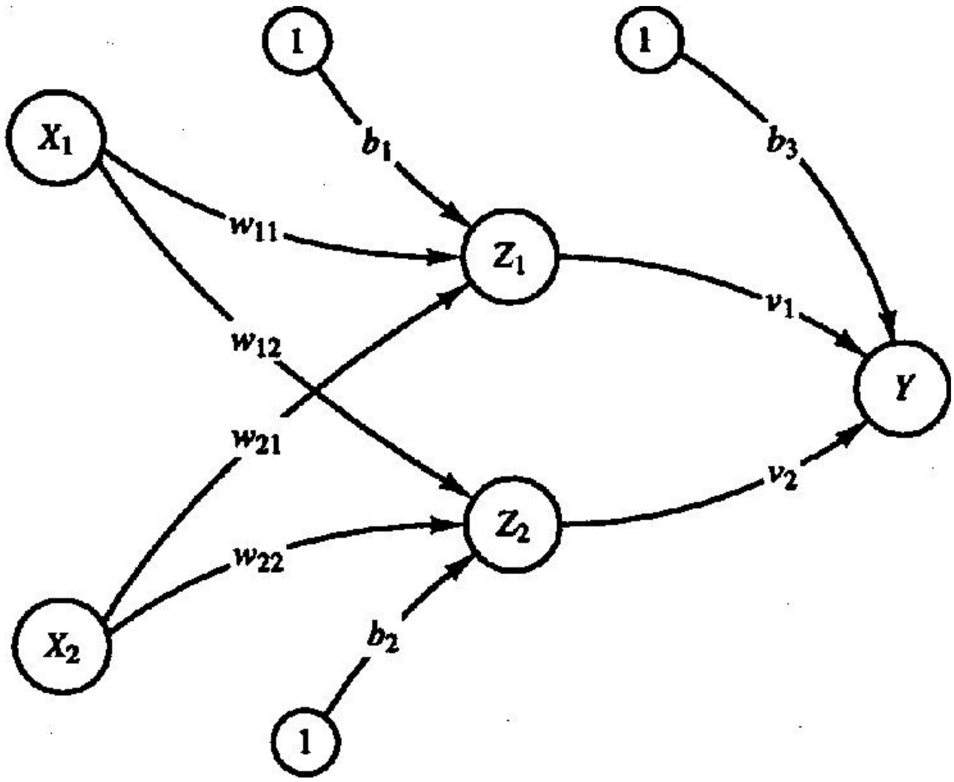


Figure Madaline **with two hidden** Adaline **and one output** Adaline.

Algorithm

In the MRI algorithm (**M**adaline **R**ule **I**: the original form of MADALINE training) [Widrow and Hoff, 1960]:

1- only the weights for the hidden Adalines are adjusted; the weights for the output unit are fixed. (MRII, allows training for weights in all layers of the net).

2- the weights v_1 and v_2 and the bias b_3 that feed into the output unit Y are determined so that the response of unit Y is 1 if the signal it receives from either Z_1 or Z_2 (or both) is 1, and is -1 if both Z_1 and Z_2 send a signal of -1. In other words, the unit Y performs the logic function OR on the signals it receives from Z_1 and Z_2 The weights into Y are:

$v_1 = 0.5$, $v_2 = 0.5$, $b_3 = 0.5$

3- the weights on the first hidden Adaline w_{11}, w_{21}, and the bias b_1 and the weights on the second hidden Adaline w_{12}, w_{22}, and b_2 are adjusted according to the algorithm.

4- the activation function for units Z_1 and Z_2 and Y is:

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

5- Set the learning rate **α** as in the **Adaline** training algorithm (a small value between 0.1 and 1).

6- Compute net input to each Adaline unit:

$$Z\text{-}in_1 = b_1 + x_1w_{11} + x_2w_{21}$$

$$Z\text{-}in_2 = b_2 + x_1w_{12} + x_2w_{22}$$

7- Determine output of each hidden unit:

$$Z_1 = f(Z\text{-}in_1)$$

$$Z_2 = f(Z\text{-}in_2)$$

8- Determine output of net:

$$y\text{-}in = b_3 + Z_1v_1 + Z_2v_2$$

$$y = f(y\text{-}in)$$

9- Determine error and update weights according to the following:

9.1 if $t = y$, no weight updates are performed.

9.2 if $t \neq y$, then:

if $t = 1$, then: update weights on Z_J, the unit whose net input is closest to 0

$$b_J(new) = b_J(old) + \alpha(1 - Z\text{-}in_J)$$

$$w_{iJ}(new) = w_{iJ}(old) + \alpha(1 - Z\text{-}in_J)x_i$$

if $t = -1$, then: update weights on all units Z_k that have positive net input ($Z\text{-}in_k > 0$):

$$b_k(new) = b_k(old) + \alpha(-1 - Z\text{-}in_k)$$

$$w_{ik}(new) = w_{ik}(old) + \alpha(-1 - Z\text{-}in_k)x_i$$

10- If weight changes have stopped (or reached an acceptable level), , then stop; otherwise continue.

Step 9 is motivated by the desire to (1) update the weights only if an error occurred and (2) update the weights in such a way that it is more likely for the net to produce the desired response.

If t = 1 and error has occurred, it means that all Z units had value - 1 and at least one Z unit needs to have a value of +1. Therefore, we take Z_J to be the

Z unit whose net input is closest to 0 and adjust its weights (using Adaline training with a target of + 1)

$$b_J(new) = b_J(old) + \alpha(1 - Z\text{-}in_J)$$

$$w_{iJ}(new) = w_{iJ}(old) + \alpha(1 - Z\text{-}in_J)x_i$$

If $t = -1$ and error has occurred, it means that at least one Z unit had value +1 and all Z units must have value -1. Therefore, we adjust the weights on all of the Z units with positive net input,(using Adaline training with a target of -1)

$$b_k(new) = b_k(old) + \alpha(-1 - Z\text{-}in_k)$$

$$w_{ik}(new) = w_{ik}(old) + \alpha(-1 - Z\text{-}in_k)x_i$$

Example: illustrate the use of the **MRI** algorithm to train a **MADALLNE to** solve the XOR problem, having the following:

Weights into Z_1			Weights into Z_2			Weights into Y		
w_{11}	w_{21}	b_1	w_{12}	w_{22}	b_2	v_1	v_2	b_3
.05	**.2**	**.3**	**.1**	**.2**	**.15**	**.5**	**.5**	**.5**

Sol:
Only the computations for the first weight updates are shown. The training patterns are:

x_1	x_2	t
1	1	-1
1	-1	1
-1	1	1
-1	-1	-1

1- For the first training pair $x_1 = 1$, $x_2 = 1$, $t = -1$

$Z\text{-}in_1 = .3 + .05 + .2 = .55$
$Z\text{-}in_2 = .15 + .1 + .2 = .45$
$Z_1 = 1$
$Z_2 = 1$
$y\text{-}in = .5 + .5 + .5 = 1.5$
$y = 1$
$t - y = -1 -1 = -2, \neq 0$, then an error occurred

since t = -1, and both Z units have positive net inputs,
update the weights on unit Z_1 as follows:

$$b_1(new) = b_1(old) + \alpha(-1 - Z\text{-}in_1)$$
$$= 0.3 + (0.5)(-1 - 0.55) = -0.475$$
$$w_{11}(new) = w_{11}(old) + \alpha(-1 - Z\text{-}in_1)x_1$$
$$= 0.05 + (0.5)(-1 - 0.55) = -0.725$$
$$w_{21}(new) = w_{21}(old) + \alpha(-1 - Z\text{-}in_1)x_2$$
$$= 0.2 + (0.5)(-1 - 0.55) = -0.575$$

update the weights on unit Z_2 as follows:

$$b_2(new) = b_2(old) + \alpha(-1 - Z\text{-}in_2)$$
$$= 0.15 + (0.5)(-1 - 0.45) = -0.575$$
$$w_{12}(new) = w_{12}(old) + \alpha(-1 - Z\text{-}in_2)x_1$$
$$= 0.1 + (0.5)(-1 - 0.45) = -0.625$$
$$w_{22}(new) = w_{22}(old) + \alpha(-1 - Z\text{-}in_2)x_2$$
$$= 0.2 + (0.5)(-1 - 0.45) = -0.525$$

After four epochs of training , the final weights found to be:

w_{11}	w_{21}	b_1	w_{12}	w_{22}	b_2
-0.73	**1.53**	**-0.99**	**1.27**	**-1.33**	**-1.09**

Geometric interpretation of weights:

The positive response region for the Madaline trained is the union of the regions where each of the hidden units has a positive response. The decision boundary for each hidden unit can be calculated:

For hidden unit $\boldsymbol{Z_1}$, the boundary line is:

$$x_2 = -\frac{w_{11}}{w_{21}} x_1 - \frac{b_1}{w_{21}}$$

$$= \frac{0.73}{1.53} x_1 + \frac{0.99}{1.53}$$

$$= 0.48\, x_1 + 0.65$$

And for hidden unit $\boldsymbol{Z_2}$, the boundary line is:

$$x_2 = -\frac{w_{12}}{w_{22}} x_1 - \frac{b_2}{w_{22}}$$

$$= \frac{1.27}{1.33} x_1 + \frac{1.09}{1.33}$$

$$= 0.96\, x_1 - 0.82$$

The response diagram for the MADALINE is illustrated in the shown figure