The ADALINE

A neuron with a learning rule based on the Least Mean Square or Delta Rule as devised by Widrow and Hoff (1962).

As before, define the internal response of the neuron as the dot product of the d-element input vectors, x_i, $i = 1, n$ and the weight vector, w_i, $i = 1, d$, $net_i = a^t x$. Let the output, z, be defined as a continuous function of the internal response. For example,

$$z = \frac{1}{1 + \exp^{-net_i}}$$

or

$$z = \frac{2}{1 + \exp^{-net_i}} - 1$$

so that z varies from -1 to +1.

Single Neuron Analysis

The error is the difference between the expected value, t_i, and the internal response, net_i, $e_i = t_i - net_i$. Since this value can have a sign, it makes since to use the following instead:

$$J(w) = E(w) = \frac{1}{2}(t_i - net_i)^2$$

Our learning rule will be of the form:

$$w(k+1) = w(k) + \Delta w_k$$

where

$$\Delta w = -\eta \frac{\partial J(w)}{\partial w}$$

Single Neuron Analysis

Using the

$$\begin{aligned}
\frac{\partial J(w)}{\partial w} &= -(t - net)\frac{\partial net(w)}{w} \\
&= -(t - net)\frac{\partial w^t x}{w} \\
&= -(t - net)(x) \\
\Delta w &= -(-(t - net)(x)) \; = \; (t - net)(x)
\end{aligned}$$

So, for input i,

$$w(k+1) \; = \; w(k) + \eta_k(t_i \; - \; w^t x)(x_i)$$

or for batch application:

$$w(k+1) \quad = \quad w(k) + \eta_k \sum_{i=1}^{n}(t_i \; - \; net_i)(x_i)$$

ADALINE Example

$$x = \left| \begin{array}{ccccc|ccccc} 1 & 2 & 2 & 3 & 3 & 5 & 5 & 6 & 6 & 7 \\ 3 & 2 & 4 & 3 & 4 & 0 & 3 & 2 & 4 & 2 \end{array} \right|$$

$$\eta = 0.5,\ b = 1, w(1) = \left| \begin{array}{c} 0.3 \\ 0.7 \end{array} \right|$$

k = 1,

$$x = \left| \begin{array}{c} 3 \\ 3 \end{array} \right|, t = 0$$

$$\begin{aligned} net &= w^t x = 0.78 \\ z &= \frac{1}{1 + \exp^{-0.78}} = 0.37 \\ e &= 0.5 * (t - z)^2 = 0.07 \\ w &= \left| \begin{array}{c} 0.30 \\ 0.70 \end{array} \right| + 0.5 * (0 - 0.37) \left| \begin{array}{c} 3 \\ 1 \end{array} \right| = \left| \begin{array}{c} -0.46 \\ 0.089 \end{array} \right| \end{aligned}$$

ADALINE Example

k = 2,

$$x = \begin{vmatrix} 5 \\ 0 \end{vmatrix}, t = 1$$

$$net = w^t x = -2.29$$

$$z = \frac{1}{1 + \exp^{-2.29}} = -0.82$$

$$e = 0.5 * (t - z)^2 = 1.65$$

$$w = \begin{vmatrix} -0.46 \\ 0.09 \end{vmatrix} + 0.5 * (0 - (-0.82)) \begin{vmatrix} 5 \\ 0 \end{vmatrix} = \begin{vmatrix} 0.32 \\ -0.95 \end{vmatrix}$$

ADALINE Example

k = 3,

$$x = \begin{vmatrix} 1 \\ 3 \end{vmatrix}, t = 0$$

$$net = w^t x = -2.52$$

$$z = \frac{1}{1 + \exp^{-2.52}} = -0.85$$

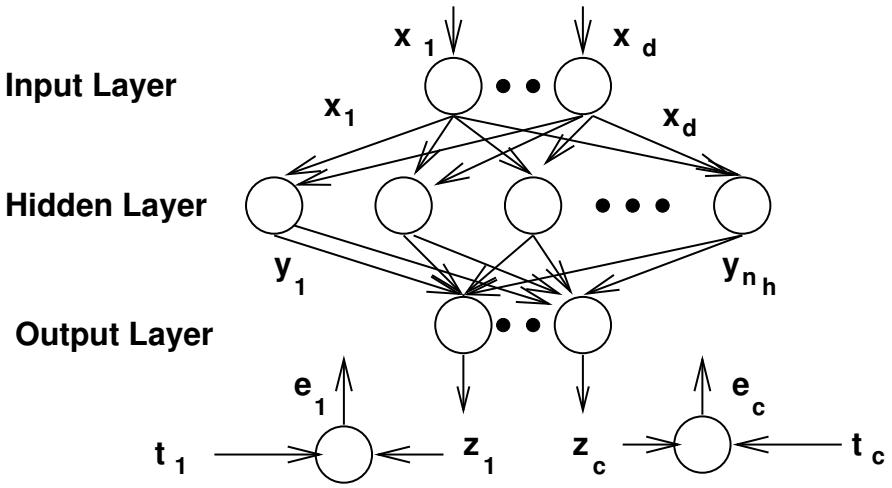
$$e = 0.5 * (t - z)^2 = 0.36$$

$$w = \begin{vmatrix} 0.32 \\ -0.95 \end{vmatrix} + 0.5 * (0 - (-0.85)) \begin{vmatrix} 1 \\ 3 \end{vmatrix} = \begin{vmatrix} 0.44 \\ -0.89 \end{vmatrix}$$

And so on. The final solution is:

$$w = \begin{vmatrix} 0.92 \\ -0.39 \end{vmatrix}$$

Multilayer ADALINE



1. x_i are the inputs, $i = 1, d$.
2. y_j are the outputs of the hidden units, $j = 1, h$.
3. z_k are the outputs of the output units, $k = 1, c$.
4. t_k are the target outputs $k = 1, c$.
5. w_{ji} is the weight on the link from input unit i to unit j in the hidden layer.
6. W_{kj} is the weight on the link from hidden unit j to unit k in the output layer.

Feedforward

At hidden layer h_j,

$$net_j = w_j^t x = \sum_{i=1}^{d} w_{ji} x_i$$

$$y_j = f(net_j) = f(\sum_{i=1}^{d} w_{ji} x_i)$$

At the output layer o_k,

$$net_k = W_k^t y = \sum_{j=1}^{c} W_{kj} y_j$$

$$= \sum_{j=1}^{c} W_{kj} f(\sum_{i=1}^{d} w_{ij} x_i)$$

$$z_k = f(net_k) = f(\sum_{j=1}^{c} W_{kj} f(\sum_{i=1}^{d} w_{ij} x_i)$$

$$)$$

Feedforward Example

Multilayer Error

As before, we need the rate of change of the error with respect to w, $\partial J(w)/\partial(w)$, but if we use z_k, we no longer have a direct relationship. So we use the chain rule,

$$\begin{aligned}
\Delta w_k &= -\eta \nabla J(w_k) \\
\nabla J(w_k) &= \frac{\partial J(w_k)}{\partial w_k} = \frac{\partial J(w_k)}{\partial net_k}\frac{\partial net_k}{\partial w_k} \\
J(w_k) &= \frac{1}{2}(t_k - z_k)^2 \\
&= \frac{1}{2}(t_k - net_k)^2
\end{aligned}$$

So,

$$\begin{aligned}
\frac{\partial J(w_k)}{\partial w_k} &= (t_k - z_k)(x) \\
\frac{\partial J(w_k)}{\partial net_k} &= (t_k - z_k) f'(net_k)
\end{aligned}$$

Multilayer Error Continued

Let, the sensitivity of unit k be defined as:

$$\begin{aligned}\delta_k &= -\partial J(w_k)/\partial net_k \\ &= (t_k - net_k) f'(net_k) \\ &= (t_k - z_k) f'(net_k)\end{aligned}$$

$$f'(net_k) = \frac{\partial f(net)}{\partial net}$$

which for $\frac{1}{1 + \exp^{-w^t y}}$ is:

$$f'(net) = f(net)(1 - f(net)) = z(1 - z)$$

Output Layer Learning

For an output layer node, k, with input vector, y_k:

$$\begin{aligned} \partial net_k/\partial w_k &= \frac{\partial(w_k^t y_k)}{\partial w_k} \\ &= y_k \end{aligned}$$

So,

$$\begin{aligned} \Delta W_{kj} &= -\frac{\partial J(w_k)}{\partial w_k} = -\frac{\partial J(w_k)}{\partial net_k}\frac{\partial net_k)}{\partial w_k} \\ &= -\eta\delta_k y \\ &= -\eta(t_k - z_k)f'(net_k)y_k \\ &= -\eta(t_k - z_k)z_k(1-z_k)y_k \end{aligned}$$

Hidden Layer Learning

The error from the output unit has to be propagated backward from the output layer. If output node k, has error e_k, then that error should be distributed over the nodes that input to it according to some function $f(e_k, w_k j, y_j)$. Of course, we expect to minimize our error function, but we can directly do so since we only have an indirect relationship between w_j and e_k. From before, the rate of change of the error at node k with respect to the weight vector at node j is:

$$\frac{\partial e_k}{w_j} = \frac{\partial e_k}{\partial net_j}\frac{\partial net_j}{\partial w_j}$$

The second term is already known to be x. Using the chain rule on the first term, we can get:

$$\frac{\partial e_k}{net_j} = \sum_{k=1}^{c} \frac{\partial e_k}{\partial net_k}\frac{\partial net_k}{\partial net_j}$$

Hidden Layer Learning

$$\delta_j = \sum_{i=1}^{c} \delta_k \frac{\partial net_k}{\partial net_j}$$

$$\frac{\partial net_k}{\partial net_j} = \sum_{i=j,}^{n_h} \frac{w_{kj} y_j}{\partial net_j}$$

Since $y_j = f(net_j)$,

$$\frac{\partial net_k}{\partial net_j} = w_{kj} f'(net_j)$$

So we have,

$$\delta_j = \left[\sum_{k=1}^{c} \delta_k W_{kj} \right] f'(net_j)$$

Hidden Layer Learning

If we define

$$e_j = \left[\sum_{k=1}^{c} \delta_k w_{kj}\right]$$

Then,

$$\delta_j = e_j f'(net_j)$$

And,

$$\begin{aligned}
\Delta w_{ji} &= -\eta x_i \delta_j \\
&= -\eta \left[\sum_{i=1}^{c} W_{kj}\delta_k\right] f'(net_j) x_i \\
&= -\eta \left[\sum_{i=1}^{c} W_{kj}\delta_k\right] y(1-y) x_i
\end{aligned}$$

Multiple Hidden Layers

If there are multiple hidden layers, the equations above apply to the last hidden layer with x_i replaced with y_i from the previous layer.

For the layers between the first and last hidden layers, these equations apply with δ_k replaced with $delta_j$ from the succeeding hidden layer.

The Backpropagation Algorithm

Feedforward:

1. Feed the input into the network.
2. For each hidden layer, calculate net_j and y_j, $j = 1, n_h$.
3. At the output layer, calculate net_k z_k and

The Backpropagation Algorithm

Backpropagation:

1. At the output layer, $\delta_k = (t - z)f'(net)$ and $\Delta W_{kj} = \eta \delta_k y_k$ and create a new set of weights, W(m+1).
2. For each hidden layer,

   (a) $$\delta_j = \left[\sum_{k=1}^{c} \delta_k W_{kj}\right] f'(net_j)$$

   (b) $$\Delta w_{ji} = \eta \delta_j x_i$$

   where x_i is replaced by y_i for every hidden layer except the first.

   (c) Calculate $w_j(m+1)$

Backpropagation Algorithm

The term:

$$e_j = \left[\sum_{k=1}^{c} \delta_k W_{kj} \right]$$

from the calculation of δ_j is called the error term for the hidden unit as it represents the error assigned to the node.

In the batch form, process all the inputs and accumulate the δ_j before modifying the weight vectors. This avoids oscillation, but slows convergence. Samples should randomized and white noise can be added to improve robustness.

Example

Backpropagation Example