Neural Network Learning Methods

Delta Learning Rule

- Supervised learning, only applicable for continuous activation function
- The learning signal r is called delta and defined as:

$$r = [d_i - f(\mathbf{w}_i^t \mathbf{x})] \cdot f'(\mathbf{w}_i^t \mathbf{x})$$

- Derived by calculating the gradient vector with respect to $\mathbf{w}i$ of the squared error.

$$E = \frac{1}{2}[d_i - f(\mathbf{w}_i^t \mathbf{x})]^2$$

$$\nabla E = -[d_i - f(\mathbf{w}_i^t \mathbf{x})] f'(\mathbf{w}_i^t \mathbf{x}) \cdot \mathbf{x}$$

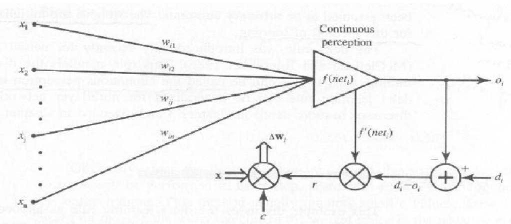
$$\frac{\partial E}{\partial w_{ij}} = -[d_i - f(\mathbf{w}_i^t \mathbf{x})] f'(\mathbf{w}_i^t \mathbf{x}) x_j$$

$$\Delta \mathbf{w}_i = -\eta \nabla E = \eta [d_i - f(\mathbf{w}_i^t \mathbf{x})] f'(\mathbf{w}_i^t \mathbf{x}) \cdot \mathbf{x}$$

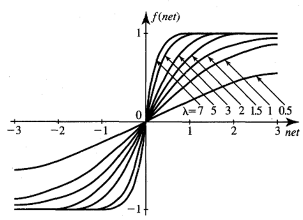
$$\Delta w_{ij} = \eta [d_i - f(\mathbf{w}_i^t \mathbf{x})] f'(\mathbf{w}_i^t \mathbf{x}) x_j$$

Delta Learning Rule

- The weight initialization is random
- Also called continuous perceptron training rule



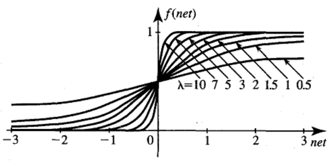
Continuous Activation Functions



$$f(net) \triangleq \frac{2}{1 + \exp(-\lambda net)} - 1$$

$$f'(net) = \frac{1}{2}(1 - o^2)$$

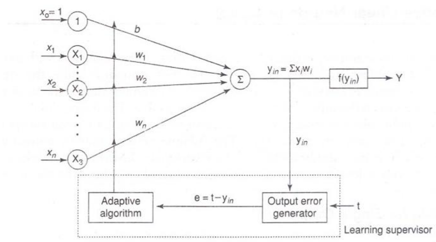
Continuous Activation Functions



$$f(net) \triangleq \frac{1}{1+\exp(-\lambda net)}$$

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

ADALINE MODEL



Delta Rule(Widrow-Hoff Rule)

Step 0: Weights and bias are set to some random values but not zero. Set the learning rate parameter α.

Step 1: Perform Steps 2–6 when stopping condition is false.

Step 2: Perform Steps 3–5 for each bipolar training pair $s:t$.

Step 3: Set activations for input units $i = 1$ to n.

$$x_i = s_i$$

Step 4: Calculate the net input to the output unit.

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

Delta Rule(Widrow-Hoff Rule)

Step 5: Update the weights and bias for $i = 1$ to n:

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

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

The range of learning rate can be between 0.1 to 1.0.

Delta Rule(Widrow-Hoff Rule)

Implement OR function with bipolar inputs and targets using Adaline network.

Solution: The truth table for OR function with bipolar inputs and targets is shown below.

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

(Learning rate=0.1, Initial weights =all 0.1)

Delta Rule(Widrow-Hoff Rule)

The initial weights are taken to be $w_1 = w_2 = b = 0.1$ and the learning rate $\alpha = 0.1$
For the first input sample, $x_1 = 1, x_2 = 1, t = 1$, we calculate the net input as

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

$$= b + \sum_{i=1}^{2} x_i w_i$$

$$y_{in} = b + x_1 w_1 + x_2 w_2$$

$$y_{in} = 0.1 + 1 \times 0.1 + 1 \times 0.1$$

$$y_{in} = 0.3$$

Delta Rule(Widrow-Hoff Rule)

Now compute $(t - y_{in}) = (1 - 0.3) = 0.7$.
Updating the weights we obtain,

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

where $\alpha(t - y_{in})x_i$ is called as weight change Δw_i. The new weights are obtained as

$$w_1(\text{new}) = w_1(\text{old}) + \Delta w_1$$

$$= 0.1 + 0.1 \times 0.7 \times 1$$

$$= 0.1 + 0.07$$

$$w_1(\text{new}) = 0.17$$

Delta Rule(Widrow-Hoff Rule)

$$w_2(\text{new}) = w_2(\text{old}) + \Delta w_2$$

$$= 0.1 + 0.1 \times 0.7 \times 1$$

$$w_2(\text{new}) = 0.17$$

$$b(\text{new}) = b(\text{old}) + \Delta b$$

$$= 0.1 + 0.1 \times 0.7$$

$$b(\text{new}) = 0.17$$

$$E = (t - y_{in})^2 = (0.7)^2 = 0.49$$

The final weights after presenting first input sample are

$$w = [0.17\ 0.17\ 0.17]$$

and error $E = 0.49$.

Delta Rule(Widrow-Hoff Rule)

Inputs x_1 x_2 1	Target t	Net input y_{in}	$t-y_{in}$	Weight changes Δw_1	Δw_2	Δb	Weights w_1 (0.1	w_2 0.1	b 0.1)	Error $(t-y_{in})^2$
EPOCH-1										
1 1 1	1	0.3	0.7	0.07	0.07	0.07	0.17	0.17	0.17	0.49
1 −1 1	1	0.17	0.83	0.083	−0.083	0.083	0.253	0.087	0.253	0.69
−1 1 1	1	0.087	0.913	−0.0913	0.0913	0.0913	0.1617	0.1783	0.3443	0.83
−1 −1 1	−1	0.0043	−1.0043	0.1004	0.1004	−0.1004	0.2621	0.2787	0.2439	1.01
EPOCH-2										
1 1 1	1	0.7847	0.2153	0.0215	0.0215	0.0215	0.2837	0.3003	0.2654	0.046
1 −1 1	1	0.2488	0.7512	0.7512	−0.0751	0.0751	0.3588	0.2251	0.3405	0.564
−1 1 1	1	0.2069	0.7931	−0.7931	0.0793	0.0793	0.2795	0.3044	0.4198	0.629
−1 −1 1	−1	−0.1641	−0.8359	0.0836	0.0836	−0.0836	0.3631	0.388	0.336	0.699

Delta Rule(Widrow-Hoff Rule)

EPOCH-3										
1 1 1	1	1.0873	−0.0873	−0.087	−0.087	−0.087	0.3543	0.3793	0.3275	0.0076
1 −1 1	1	0.3025	+0.6975	0.0697	−0.0697	0.0697	0.4241	0.3096	0.3973	0.487
−1 1 1	1	0.2827	0.7173	−0.0717	0.0717	0.0717	0.3523	0.3813	0.469	0.515
−1 −1 1	−1	−0.2647	−0.7353	0.0735	0.0735	−0.0735	0.4259	0.4548	0.3954	0.541
EPOCH-4										
1 1 1	1	1.2761	−0.2761	−0.0276	−0.0276	−0.0276	0.3983	0.4272	0.3678	0.076
1 −1 1	1	0.3389	0.6611	0.0661	−0.0661	0.0661	0.4644	0.3611	0.4339	0.437
−1 1 1	1	0.3307	0.6693	−0.0669	0.0669	0.0699	0.3974	0.428	0.5009	0.448
−1 −1 1	−1	−0.3246	−0.6754	0.0675	0.0675	−0.0675	0.465	0.4956	0.4333	0.456
EPOCH-5										
1 1 1	1	1.3939	−0.3939	−0.0394	−0.0394	−0.0394	0.4256	0.4562	0.393	0.155
1 −1 1	1	0.3634	0.6366	0.0637	−0.0637	0.0637	0.4893	0.3925	0.457	0.405
−1 1 1	1	0.3609	0.6391	−0.0639	0.0639	0.0639	0.4253	0.4654	0.5215	0.408
−1 −1 1	−1	−0.3603	−0.6397	0.064	0.064	−0.064	0.4893	0.5204	0.4575	0.409

Delta Rule(Widrow-Hoff Rule)

Epoch	Total mean square error
Epoch 1	3.02
Epoch 2	1.938
Epoch 3	1.5506
Epoch 4	1.417
Epoch 5	1.377

Delta Rule Example

The set of input training vectors is as follows:

$$\mathbf{x}_1 = \begin{bmatrix} 1 \\ -2 \\ 0 \\ -1 \end{bmatrix}, \quad \mathbf{x}_2 = \begin{bmatrix} 0 \\ 1.5 \\ -0.5 \\ -1 \end{bmatrix}, \quad \mathbf{x}_3 = \begin{bmatrix} -1 \\ 1 \\ 0.5 \\ -1 \end{bmatrix}$$

Initial weights

$$\mathbf{w}^1 = \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0.5 \end{bmatrix}$$

Learning rate =1

o/p= [-1 -1 1]

Step 1 Input is vector $\mathbf{x}_1$, initial weight vector is $\mathbf{w}^1$:

$$net^1 = \mathbf{w}^{1t}\mathbf{x}_1 = 2.5$$
$$o^1 = f(net^1) = 0.848$$
$$f'(net^1) = \frac{1}{2}[1 - (o^1)^2] = 0.140$$
$$\mathbf{w}^2 = c(d_1 - o^1)f'(net^1)\mathbf{x}_1 + \mathbf{w}^1$$
$$= \begin{bmatrix} 0.974 & -0.948 & 0 & 0.526 \end{bmatrix}^t$$

Step 2 Input is vector $\mathbf{x}_2$, weight vector is $\mathbf{w}^2$:

$$net^2 = \mathbf{w}^{2t}\mathbf{x}_2 = -1.948$$
$$o^2 = f(net^2) = -0.75$$
$$f'(net^2) = \frac{1}{2}[1 - (o^2)^2] = 0.218$$
$$\mathbf{w}^3 = c(d_2 - o^2)f'(net^2)\mathbf{x}_2 + \mathbf{w}^2$$
$$= \begin{bmatrix} 0.974 & -0.956 & 0.002 & 0.531 \end{bmatrix}^t$$

Step 3 Input is x_3, weight vector is $\mathbf{w}^3$:

$$net^3 = \mathbf{w}^{3t}\mathbf{x}_3 = -2.46$$
$$o^3 = f(net^3) = -0.842$$
$$f'(net^3) = \frac{1}{2}[1 - (o^3)^2] = 0.145$$
$$\mathbf{w}^4 = c(d_3 - o^3)f'(net^3)\mathbf{x}_3 + \mathbf{w}^3$$
$$= \begin{bmatrix} 0.947 & -0.929 & 0.016 & 0.505 \end{bmatrix}^t$$

Perform two training steps of the network as in Figure 2.24 using the delta learning rule for $\lambda = 1$ and $c = 0.25$. Train the network using the following data pairs

$$\left(\mathbf{x}_1 = \begin{bmatrix} 2 \\ 0 \\ -1 \end{bmatrix},\quad d_1 = -1\right),\quad \left(\mathbf{x}_2 = \begin{bmatrix} 1 \\ -2 \\ -1 \end{bmatrix},\quad d_2 = 1\right)$$

The initial weights are $\mathbf{w}^1 = [\,1\ 0\ 1\,]^t$. [*Hint:* Use $f'(net) = (1/2)(1 - o^2)$ and $f(net)$ as in (2.3a).]