Parimi Lakshmi Priyanka (lxp160730)

Maddi Jaya Padma shri (jxm166230)

Assignment - III

1) Question 4.3

$w_0 + w_1 x_1 + w_2 x_2 > 0$

Perceptron A weight values : $w_0 = 1 \quad w_1 = 2 \quad w_2 = 1 \Rightarrow 1 + 2x_1 + x_2 > 0$

Perceptron B weight values : $w_0 = 0 \quad w_1 = 2 \quad w_2 = 1 \Rightarrow 2x_1 + x_2 > 0$

A is more general than perceptron B if and only if

~~(∀x∈X)[(B(x)→1~~

$$(\forall x \in X)\ [\, B(x) = 1 \longrightarrow A(x) = 1 \,]$$

$$B(x) = 1 \Rightarrow 2x_1 + x_2 > 0$$

$$\Rightarrow 1 + 2x_1 + x_2 > 0$$

$$\Rightarrow A(x) = 1$$

∴ True. A is more general than Perceptron B.

2) Given a gradient descent training rule for a single neuron

$$o = w_0 + w_1(x_1 + x_1^2) + \cdots + w_n(x_n + x_n^2)$$

Here, error function is given as $E = \frac{1}{2} \sum_{k \in outputs} (t_k - o_k)^2$

Weight update : $w_i \leftarrow w_i + \Delta w_i$

$$\Delta w_i = \eta \frac{\partial E}{\partial w_i}$$

consider the case of w_0

$$\Delta w_0 = -\eta \frac{\partial E}{\partial w_0} \rightarrow (1)$$

$$\frac{\partial E}{\partial w_0} = \frac{\partial}{\partial w_0}\left(\frac{1}{2}\sum_{k \in output}(t_k - O_k)^2\right)$$

$$= \frac{1}{2} \times 2 \sum (t_k - O_k) \times \frac{\partial}{\partial w_0} \sum (t_k - (w_0 + w_1 x_1 + w_1 x_1^2 + \dots + w_n x_n + w_n x_n^2))$$

$$= \sum_{k \in output} (t_k - O_k) \times -1$$

$$= -\sum (t_k - O_k)$$

substituting in (1)

$$\Delta w_0 = -\eta - \sum (t_k - O_k)$$

$$= \eta \sum_{k \in output} (t_k - O_k)$$

Similarly if we consider for the other values of i i.e

$w_1, w_2, w_3 \dots w_n$

$$\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i} \times \frac{1}{2} \sum_{k \in output} (t_k - O_k)^2$$

$$= \frac{1}{2} \times \sum 2 (t_k - O_k) \times \frac{\partial}{\partial w_i} (t_k - O_k)$$

$$\cancel{= \sum (t_k - O_k) \times \frac{\partial}{\partial w_i} (t_k - (w_0 + w_1 x_1 + w_1 x_1^2 + \dots + w_n x_n + w_n}$$

Scanned by CamScanner

$$= \Sigma (t_k - o_k) \times \frac{\partial}{\partial w_i} (t_k - (w_0 + w_1 x_1 + w_1 x_1^2 + \dots + w_i x_i + w_i x_i^2 + \dots + w_n x_n + w_n x_n^2)$$

$$= \Sigma (t_k - o_k) \times (-(x_{ik} + x_{ik}^2))$$

$$= \cancel{-(x_{ik} + x_{i2k}}$$

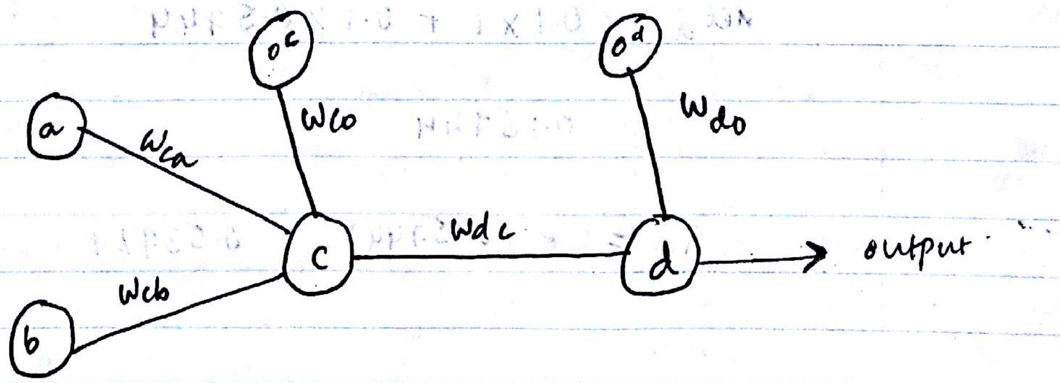
$$= -(x_{ik} + x_{ik}^2) \sum_{k \in outputk} (t_k - o_k)$$

$$\Delta w_i = \eta \, \Sigma (t_k - o_k)(x_{ik} + x_{ik}^2)$$

3) Two-layer feedforward ANN with two inputs a and b, one hidden unit c and one output unit d.

$\eta = 0.3 \quad \alpha = 0.9$

a	b	d
1	0	1
0	1	0



⇒ First input to the artificial neural network is (1,0) and target output is 1.

Forward pass:

For node A:

$net_a = 1$ Output: $X_a = 1$

For node B:

$net_b = 0$ Output: $x_b = 0$

For node c:

$net_c = w_{co} x_0 + w_{ca} x_1 + w_{cb} x_2$

$= 0.1 \times 1 + 0.1 \times 1 + 0.1 \times 0$

$= 0.2$

Output: $x_c = \sigma(0.2) = 0.5498$

For node d:

$net_d = 0.1 \times 1 + 0.1 \times 0.5498$

$= 0.15498$

$x_d = \sigma(0.15498) = 0.5386$

Backward pass:

For node D:

$$\delta_D = x_d(1-x_d)(t_d-x_d)$$

$$= 0.5386(1-0.5386)(1-0.5386)$$

$$= 0.1146$$

For node C:

$$\delta_C = x_c(1-x_c)(w_{dc}\delta_d)$$

$$= 0.5498(1-0.5498)(0.1 \times 0.1146)$$

$$= 0.5498 \times 0.1 \times 0.1146 \times (1-0.5498)$$

$$= 0.00283$$

$$\Delta w_{dc} = \eta \delta_d x_c$$

$$= 0.3 \times 0.1146 \times 0.5498$$

$$= 0.0189$$

For node B:

$$\cancel{\delta_b = x_b(1-x_b)(w_{cb}\delta_c)}$$

$$= \cancel{-2}0$$

$$\Delta w_{cb} = \eta \delta_{cb} x_b = 0$$

For node A:

$$\delta_a = x_a(1-x_a)(w_{cb}\delta_c)$$

$$= 1(1-1)(w_{cb}\delta_c)$$

$$= 0.$$

$$\Delta w_{ca} = \eta \delta_c X_1$$

$$= 0.3 \times 0.00283 \times 1$$

$$= 0.000849$$

$$\Delta w_{co} = \eta \delta_c x_{co}$$

$$= 0.3 \times 0.00283 \times 1$$

$$= 0.000849$$

$$\Delta w_{do} = \eta \delta d\, x_{do}$$

$$= 0.3 \times 0.1146 \times 1$$

$$= 0.03438$$

New weights:

$$w_{do} = 0.1 + 0.03438 = 0.13438$$

$$w_{dc} = 0.1 + 0.0189 = 0.1189$$

$$w_{co} = 0.1 + 0.000849 = 0.100849$$

$$w_{ca} = 0.1 + 0.000849 = 0.100849$$

$$w_{cb} = 0.1 + 0 = 0.1$$

Now the input is training example 2:

$(a,b) \rightarrow (0,1)$

Target output = 0.

Forward pass:

For node A:

$Net_a = 0$.

Output: $x_a = 0$.

For node B:

$Net_b = 1$

Output: $x_b = 1$.

For node c:

$net_c = W_{c0}x_{c0} + W_{ca}x_a + W_{cb}x_b$

$= 0.100849 \times 1 + 0.100849 \times 0 + 0.1 \times 1$

$= 0.100849 + 0.1$

$= 0.200849$

Output: $x_c = \sigma(0.200849)$

$= 0.55004$

for node D:

$$net_d = w_{do} x_{do} + w_{cd} x_c$$

$$= 0.13438 \times 1 + 0.1189 \times 0.5504$$

$$= 0.1348 + 0.06544$$

$$= 0.20024$$

output x_d : $\sigma(0.20024)$

$$= 0.5498$$

Backward pass:

$$\delta_d = (0.5498)(1-0.5498)(0-0.5498)$$

$$= -0.1360$$

$$\delta_c = (0.55004)(1-0.5504)(0.1189 \times -0.1360)$$

$$= -0.0040$$

Updating weights:

$\eta = 0.3$ $\alpha = 0.9$

$\Delta w_{ji}(n) = \eta \delta_j x_{ji} + \alpha \Delta w_{ji}(n-1)$. This is the weight update on n^{th} iteration.

$\Delta w_{do} = (0.3 \times -0.1360 \times 1) + (0.9 \times 0.03438)$

$= -0.0408 + 0.0309$

$= -0.009$

$\Delta w_{ca} = (0.3)(0.04)(0) + 0.9 \times 0.000849$

$= 0 + 0.0007641$

$= 0.0007641$

$\Delta w_{cb} = (0.3)(-0.004)(1) + (0.9 \times 0]$

$= -0.0012$

$\Delta w_{co} = 0.3 \times -0.004 \times 1 + (0.9 \times 0.000849)$

$= -0.0012 + 0.0007641$

$= -0.000435$

$\Delta w_{dc} = (0.3 \times -0.136 \times 0.55004) + (0.9 \times 0.0189)$

$= -0.02244 + 0.01701$

$= -0.00543$

Final weights:

$w_{do} = 0.13438 + (-0.009) = 0.125$

$w_{cd} = 0.1189 + (-0.0054) = 0.1135$

$w_{co} = 0.100849 + (-0.000435) = 0.100414$

$w_{ca} = 0.100849 + 0.0007641 = 0.10161$

$w_{cb} = 0.1 + (-0.0012) = 0.0988$

(4) Considering tanh in place of sigmoid function.

Error term : $\frac{1}{2} \sum_{k \in outputs} (t_k - o_k)^2$

$$\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial w_{ji}}$$

$$\frac{\partial E_d}{\partial w_{ji}} = \frac{\partial E_d}{\partial net_j} \cdot \underbrace{\frac{\partial net_j}{\partial w_{ji}}}_{x_{ji}}$$

$\frac{\partial E_d}{\partial net_j}$ should be calculated.

For output unit :

$$\frac{\partial E_d}{\partial net_j} = \frac{\partial E_d}{\partial o_j} \frac{\partial o_j}{\partial net_j}$$

Here $\frac{\partial E_d}{\partial o_j} = -(t_j - o_j)$

Second term is the derivative of tanh function.

$$\frac{\partial o_j}{\partial net_j} = \frac{\partial}{\partial net_j}(\tanh) = 1 - \tanh^2(x)$$

$$= 1 - (o_j)^2$$

$-\delta_j \Rightarrow \frac{\partial E_d}{\partial net_j} = -(t_j - o_j)(1 - o_j^2)$

Scanned by CamScanner

For hidden unit:

$$\frac{\partial E_d}{\partial net_j} = \sum_{k \in Downstream(j)} \frac{\partial E_d}{\partial net_k} \frac{\partial net_k}{\partial net_j}$$

$$= \sum -\delta_k \frac{\partial net_k}{\partial net_j}$$

$$= \sum -\delta_k \frac{\partial net_k}{\partial o_j} \frac{\partial o_j}{\partial net_j}$$

$$= \sum -\delta_k w_k (1-o_j^2)$$

Putting it all together.

$$\delta_j = (1-o_j^2) \sum_{k \in Downstream} \delta_k w_{kj}$$

Finally, the weight update rule for

output:

$$\delta_j \leftarrow (1-o_j^2)(t_j - o_j)$$

hidden:

$$\delta_j \leftarrow (1-o_j^2) \sum_{k \in Downstream} \delta_k w_{kj}$$

5) $E(\vec{w}) = \frac{1}{2} \sum_{d \in D} \sum_{k \in outputs} (t_{kd} - o_{kd})^2 + \gamma \sum_{i,j} w_{ji}^2$

Weight update rule:

$$\cancel{w_{ij} \leftarrow w_{ij} + \Delta w_{ij}} \qquad \Delta w_{ji} = -\eta \frac{\partial E(\vec{w})}{\partial w_{ji}}$$

$$w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$$

$$\frac{\partial E(\vec{w})}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} \frac{1}{2} \sum \sum (t_{kd} - o_{kd})^2 + \frac{\partial}{\partial w_{ji}} \gamma \sum_{i,j} w_{ji}^2$$

$$= -(t_j - o_j)(1 - o_j)(o_j) x_{ji} + 2\gamma w_{ji}$$

$$\therefore \quad w_{ji} \leftarrow w_{ji} + \eta (t_j - o_j)(1 - o_j)(o_j) x_{ji} - 2\gamma \eta w_{ji}$$

$$[\because (t_j - o_j)(1 - o_j) o_j = \delta_j]$$

$$\therefore \quad w_{ji} \leftarrow w_{ji} + \eta \delta_j x_{ji} - 2\gamma \eta w_{ji}$$

$$w_{ji} \leftarrow \eta \delta_j x_{ji} + w_{ji} (1 - 2\gamma \eta)$$

For hidden layer:

In the same way, weight update rule is

$$w_{ji} \leftarrow \eta \delta_j x_{ji} + w_{ji} (1 - 2\eta\gamma)$$

Here $\delta_j = o_j (1 - o_j) \sum_{k \in downstream(j)} \delta_k w_{kj}$

The above update rule

The above equations show that the update rule can be implemented by multiplying each weight by some constant before performing the standard gradient descent update.