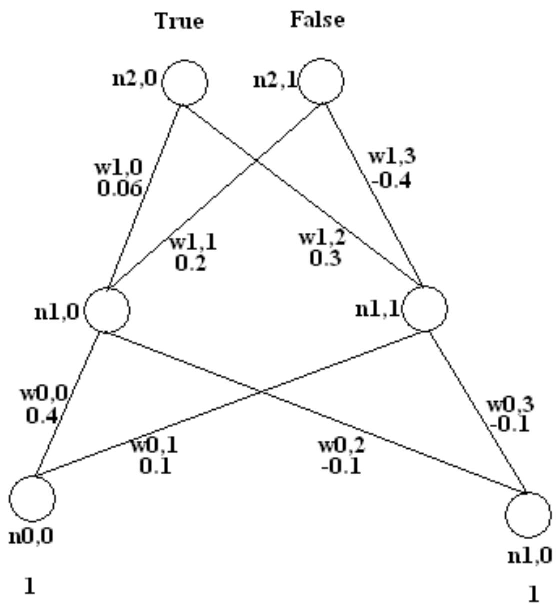
Sample Forward and Backward pass for Backprop Learning



Pattern data for XOR

n0,0	*n0,1*	*Output n2,0*	*Output n2,1*
1	1	0	1
1	0	1	0
0	1	1	0
0	0	0	1

β= Learning rate = 0.45
α= Momentum term = 0.9

$f(\mathbf{x}) = 1.0 / (1.0 + \exp(-x))$

NOTE : XOR problem would not really be encoded with two output nodes. I'm only doing it here for illustration purposes.

Forward Pass

n0,0 = 1 & **n0,1** = 1   {Load the input nodes}

Middle nodes take weighted input so :

n1,0 = f((1 * 0.4) + (1 * -0.1))
= f(0.3)
= 0.5744425

n1,1 = f((1 * 0.1) + (1 * -0.2))
= f(-0.1)
= 0.4750208

Output nodes take weighted input so:

n2,0 = f((0.5744425 * 0.06) + (0.4750208 * 0.3))
= f(0.03446655 + 0.14250624)
= f(0.17697279)
= **0.54412806**   Output for node **n2,0**

n2,1 = f((0.5744425 * 0.2) + (0.4750208 * -0.4))
= f(0.1148885 + -0.1900083)
= f(-0.0751198)
= **0.48122886**   Output for node **n2,1**

Backward pass

Calculate the error on each output node

n2,0_ERROR = n2,0 * (1.0 – n2,0) * (n2,0_DESIRED – n2,0)
= 0.54412806 * (1.0 - 0.54412806) * (n2,0_DESIRED - 0.54412806)
= 0.54412806 * (1.0 - 0.54412806) * (0 - 0.54412806)
= 0.54412806 * 0.45587194 * -0.54412806
= **-0.134972442**

n2,1_ERROR = n2,1 * (1.0 - n2,1) * (n2,1_DESIRED - n2,1)
= 0.48122886 * (1.0 - 0.48122886) * (n2,1_DESIRED - 0.48122886)
= 0.48122886 * (1.0 - 0.48122886) * (1 - 0.48122886)
= 0.48122886 * 0.51877114 * 0.51877114
= **0.129509993**

Once we have calculated the error on the output nodes we can adjust their weights so:

w1,0Δ = β * n2,0_ERROR * n1,0
= 0.45 * -0.134972442 * 0.5744425
= **-0.034890258**

w1,0 = w1,0 + Δ + (α * Δ(t-1)) {Δ(t-1) = previous delta change for w1,0}
= 0.06 + -0.034890258 + (0.9 * 0)
= **0.025109742** new weight value for w1,0

NOTE : Store Δ for next time round

w1,2Δ = β * n2,0_ERROR * n1,1
= 0.45 * -0.134972442 * 0.4750208
= **-0.028851623**

w1,2 = w1,2 + Δ + (α * Δ(t-1)) {Δ(t-1) = previous delta change for w1,2}
= -0.4 + -0.028851623 + (0.9 * 0)
= **-0.428851623** new weight value for w1,2

NOTE : Store Δ for next time round

w1,1Δ = β * n2,1_ERROR * n1,0
= 0.45 * 0.129509993 * 0.5744425
= **0.03347822**

w1,1 = w1,1 + Δ + (α * Δ(t-1)) {Δ(t-1) = previous delta change for w1,1}
= 0.2 + 0.03347822 + (0.9 * 0)
= **0.23347822** new weight value for w1,1

NOTE : Store Δ for next time round

Now that we have adjusted the weights from the middle layer to the output layer we must calculate the middle layer error values and adjust their weights so :

n1,0_ERROR = (n2,0_ERROR * w1,0) + (n2,1_ERROR * w1,1) {weighted error from layer 2}
= (-0.134972442 * 0.025109742) + (0.1295099993 * 0.23347822)
= 0.026848641

n1,1_ERROR = (n2,0_ERROR * w1,2) + (n2,1_ERROR * w1,3) {weighted error from layer 2}
= {follow same method as above}

Once we have calculated the error on the middle nodes we can adjust their weights so:

W0,0Δ = β * n1,0_ERROR * n0,0
= 0.45 * 0.026848641 * 1
= **0.012081888**

w0,0 = w0,0 + Δ + (α * Δ(t-1)) {Δ(t-1) = previous delta change for w0,0}
= 0.4 + **0.012081888** + (0.9 * 0)
= 0.412081888 new weight value for w0,0

NOTE : Store Δ for next time round

Adjust the rest of the weights for the middle layer using the same method.