$\partial E / \partial w1 =$ (∂ E / ∂h1) · (∂h1 / ∂z_{h1}) · (∂z_{h1} / ∂ w1)

$\partial E / \partial w1 =$ (y1 - t1) · y1 · (1 - y1) · w5 · (∂h1 / ∂z_{h1}) · (∂z_{h1} / ∂ w1)

$\partial E / \partial w1 =$ (y1 - t1) · y1 · (1 - y1) · w5 · h1 · (1 - h1) · (∂z_{h1} / ∂ w1)

$\partial E / \partial w1 =$ (y1 - t1) · y1 · (1 - y1) · w5 · h1 · (1 - h1) · x1

Δw1 = (y1 - t1) · y1 · (1 - y1) · w5 · h1 · (1 - h1) · x1

Δw1 = (y1 - t1) · y1 · (1 - y1) · w5 · h1 · (1 - h1) · x1

Δw1 = Error · Output Node · Interneuron Weight · Hidden Layer Node · Activation

E(y1) vs. epoch

E(y1) vs epoch
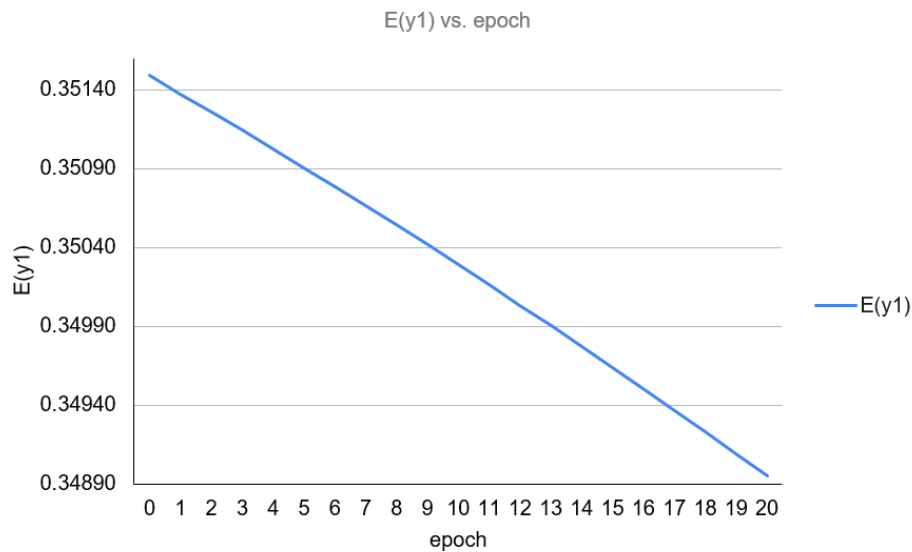
Do the results and charts make sense? Why?

Yes these results and charts make sense. The goal of this exercise was to use the backpropagation algorithm to alter the w1 weight in such a way that reduces the error. This was made evident with the decreasing slope in the first chart, showing that as the epochs increase the error decreases (for the 1,1 input).

Why do the charts show their respective trends?

As stated before, the first chart shows its trend of a decreasing slope because the error decreases as the epochs increase. The second chart shows the error continually decreasing and increasing. This is so because we are now feeding in all four inputs of the xor table, rather than the same one input. This results in the network learning one input pretty well which causes the error to decrease, and then the input of x1 and x2 change, resulting in the error increasing because the network hasn't learned that input well enough yet. We also need to keep in mind we are only changing one weight in this example, whereas changing multiple weights at a time will result in the network learning the inputs faster and the wave's amplitude won't be as large as the epochs increase.