Setup

The purpose of this document is to help give a visual understanding of the mathematics within a multi-layer neural network. We begin with very basic examples so that the mathematics can be more easily understood, and then we increase the complexity slowly so the leap from one case to the next isn't too significant. This document is not meant to be a complete lesson in neural networks but a supplement to another text. Please refer to the book available at neuralnetworksanddeeplearning.com in order to gain the knowledge we assume in this explanation. This document aims only at explaining the math and code for particular cases of NN.

1. 2 inputs, 0 hidden layers, one output
2. 2 inputs, 0 hidden layers, two output
3. 2 inputs, 1 hidden layer with two nodes each, one output
4. 2 inputs, 1 hidden layer with two nodes each, two output
5. 2 inputs, 2 hidden layer with two nodes each, one output
6. 2 inputs, 2 hidden layer with two nodes each, two output

We denote

input layer nodes by x_i with bias node x_0,

hidden layer (n) nodes by $h_i^{(n)}$ with bias node h_0,

weights in the 0th layer by $w_{ij}^{(1)}$, weights in the 1st layer by $w_{ij}^{(2)}$, and weights in the 2nd layer by $w_{ij}^{(3)}$, and

output layer nodes by o_j.

We limit ourselves to maximum 2 nodes/hidden layers to simplify computation by hand. As you will see below, writing each calculation out is not feasible. Therefore, you may want to use the power of linear algebra to make these more compact. This document is meant to aid those who may not have this experience.

In addition, we note here that the multi-layer neural network method studied here is backpropogation with vanilla gradient descent

$$w = w - \gamma \cdot \frac{dJ}{dw}$$

as the learning rule for the weights, and the sigmoid function

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

is used as the activation function for each node in hidden and output layers. Please note that the derivative of the sigmoid function has a very nice form. That is,

$$\frac{d\sigma}{dx} = \sigma(x)(1 - \sigma(x)).$$

This is a significant piece of information that is used when calculating the derivatives in our method. Finally, the cost function we choose to minimize is

$$J(\vec{o}) = \frac{1}{2}||\vec{o} - \vec{t}||_2^2$$

where $\vec{o} = (o_1, \ldots, o_j)$ is the vector of estimated output and $\vec{t} = (t_1, \ldots, t_j)$ is the vector of true output.

The primary idea of backpropogation: the input are sent forward into the network, activating the neurons at hidden layers and at the output layer; we take the output activation and compare to the truth that we know; we take this error and send it back through the network to correct the only things we have control over – the weights.

The choice of (1) learning rule, (2) activation function, and (3) cost function will change the results we show here.

Case 1

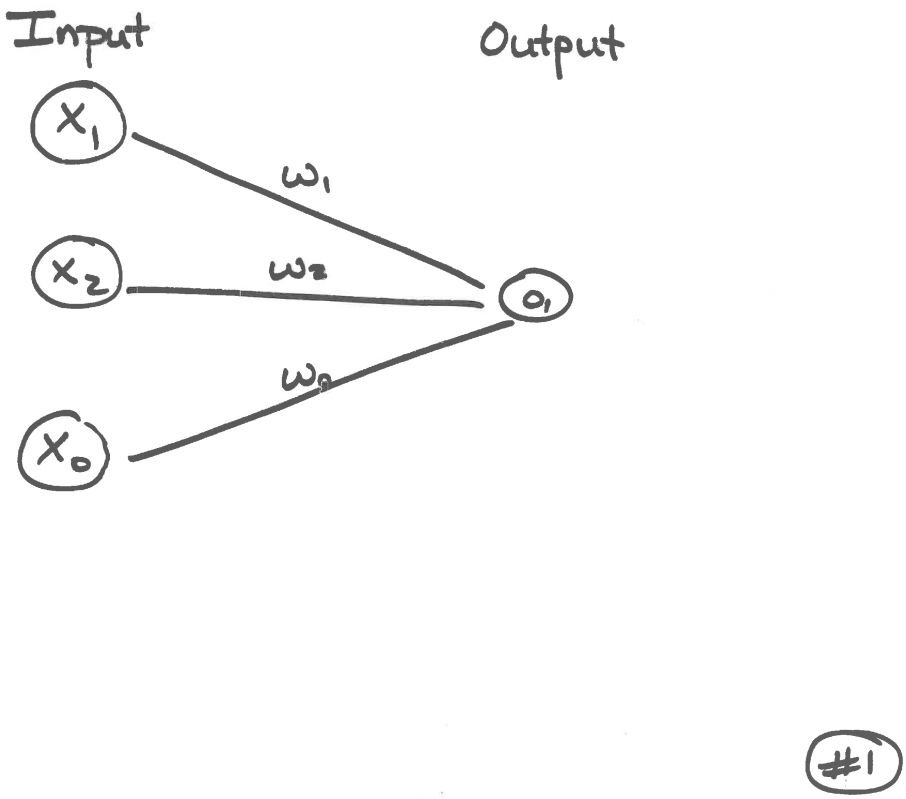

Figure 1: 2 inputs, 0 hidden layers, one output

In case 1 we don't have hidden layers. You can trace the following steps along the network in Figure 1.

Feedforward:

$$o_1 = \sigma\left(w_0 \cdot x_0 + w_1 \cdot x_1 + w_2 \cdot x_2\right)$$

Error(s):

$$J(o_1) = 0.5(o_1 - t_1)^2$$

Derivatives:

$$\frac{dJ}{do_1} = (o_1 - t_1)$$

$$\frac{do_1}{dw_i} = o_1 \cdot (1 - o_1) \cdot x_i$$

Update:

$$w_i = w_i - \gamma \left(\frac{dJ}{do_1} \cdot \frac{do_1}{dw_i}\right)$$

Case 2

In this case, we'll be following the same steps as before, but now we have two output. Please refer to Figure 2 to trace and understand the steps.

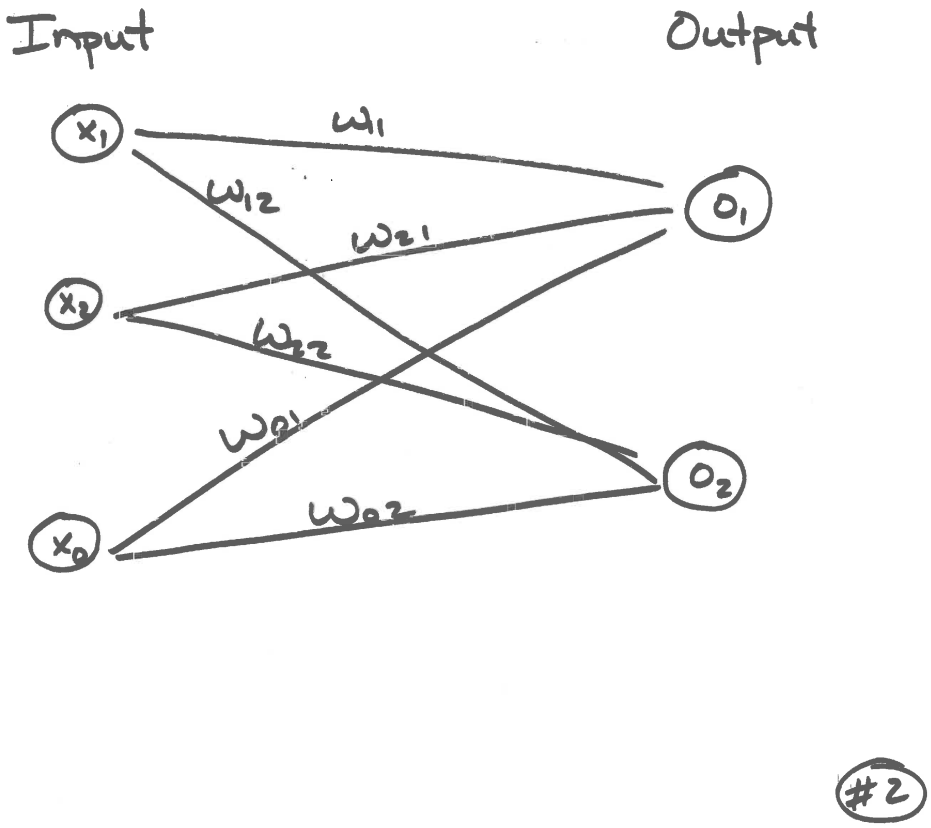

Figure 2: 2 inputs, 0 hidden layers, two output

Feedforward:

$$o_1 = \sigma\left(w_{01} \cdot x_0 + w_{11} \cdot x_1 + w_{21} \cdot x_2\right)$$

$$o_2 = \sigma\left(w_{02} \cdot x_0 + w_{12} \cdot x_1 + w_{22} \cdot x_2\right)$$

Error(s):

$$J(o_1, o_2) = 0.5(o_1 - t_1)^2 + 0.5(o_2 - t_2)^2$$

Derivatives:

$$\frac{dJ}{do_1} = (o_1 - t_1) \qquad \frac{dJ}{do_2} = (o_2 - t_2)$$

$$\frac{do_1}{dw_{i1}} = o_1 \cdot (1 - o_1) \cdot x_i \qquad \frac{do_2}{dw_{i2}} = o_2 \cdot (1 - o_2) \cdot x_i$$

Please note we used the chain rule for $\frac{do_1}{dw_{i1}}$ and $\frac{do_2}{dw_{i2}}$.

Update:

$$w_{i1} = w_{i1} - \gamma \left(\frac{dJ}{do_1} \cdot \frac{do_1}{dw_{i1}} \right)$$

$$w_{i2} = w_{i2} - \gamma \left(\frac{dJ}{do_2} \cdot \frac{do_2}{dw_{i2}} \right)$$

Case 3

Please refer to Figure 3 to trace and understand the steps. Remember, we rely on the chain rule since each hidden node and output node are composed functions.

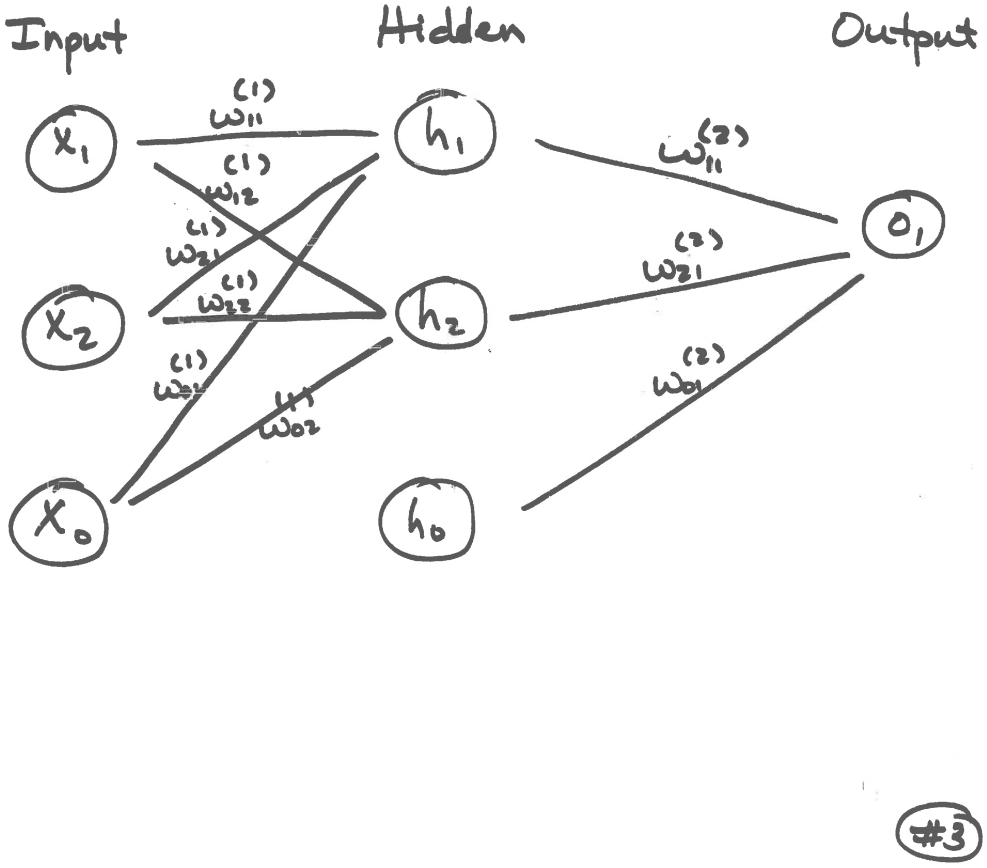

Figure 3: 2 inputs, 1 hidden layer with two nodes each, one output

Feedforward:

$$h_1 = \sigma \left(w_{01}^{(1)} \cdot x_0 + w_{11}^{(1)} \cdot x_1 + w_{21}^{(1)} \cdot x_2 \right)$$

$$h_2 = \sigma \left(w_{02}^{(1)} \cdot x_0 + w_{12}^{(1)} \cdot x_1 + w_{22}^{(1)} \cdot x_2 \right)$$

$$o_1 = \sigma \left(w_{01}^{(2)} \cdot h_0 + w_{11}^{(2)} \cdot h_1 + w_{21}^{(2)} \cdot h_2 \right)$$

Error(s):

$$J(o_1) = 0.5(o_1 - t_1)^2$$

Derivatives:

$$\frac{dJ}{do_1} = (o_1 - t_1) \qquad \frac{do_1}{dh_j} = o_1(1-o_1) \cdot w_{j1}^{(2)}$$

$$\frac{do_1}{dw_{i1}^{(2)}} = o_1(1-o_1) \cdot h_i \qquad \frac{dh_j}{dw_{ij}^{(1)}} = h_j \cdot (1-h_j) \cdot x_i$$

Update:

$$w_{i1}^{(1)} = w_{i1}^{(1)} - \gamma \left(\frac{dJ}{do_1} \cdot \frac{do_1}{dh_j} \cdot \frac{dh_j}{dw_{i1}^{(1)}} \right)$$

$$w_{i2}^{(1)} = w_{i2}^{(1)} - \gamma \left(\frac{dJ}{do_1} \cdot \frac{do_1}{dh_j} \cdot \frac{dh_j}{dw_{i2}^{(1)}} \right)$$

$$w_{i1}^{(2)} = w_{i1}^{(2)} - \gamma \left(\frac{dJ}{do_1} \cdot \frac{do_1}{dw_{i1}^{(2)}} \right)$$

Case 4

Please refer to Figure 4 to trace and understand the steps. Remember, we rely on the chain rule since each hidden node and output node are composed functions.

Feedforward:

$$h_1 = \sigma \left(w_{01}^{(1)} \cdot x_0 + w_{11}^{(1)} \cdot x_1 + w_{21}^{(1)} \cdot x_2 \right)$$

$$h_2 = \sigma \left(w_{02}^{(1)} \cdot x_0 + w_{12}^{(1)} \cdot x_1 + w_{22}^{(1)} \cdot x_2 \right)$$

$$o_1 = \sigma \left(w_{01}^{(2)} \cdot h_0 + w_{11}^{(2)} \cdot h_1 + w_{21}^{(2)} \cdot h_2 \right)$$

$$o_2 = \sigma \left(w_{02}^{(2)} \cdot h_0 + w_{12}^{(2)} \cdot h_1 + w_{22}^{(2)} \cdot h_2 \right)$$

Error(s):

$$J(o_1, o_2) = 0.5(o_1 - t_1)^2 + 0.5(o_2 - t_2)^2$$

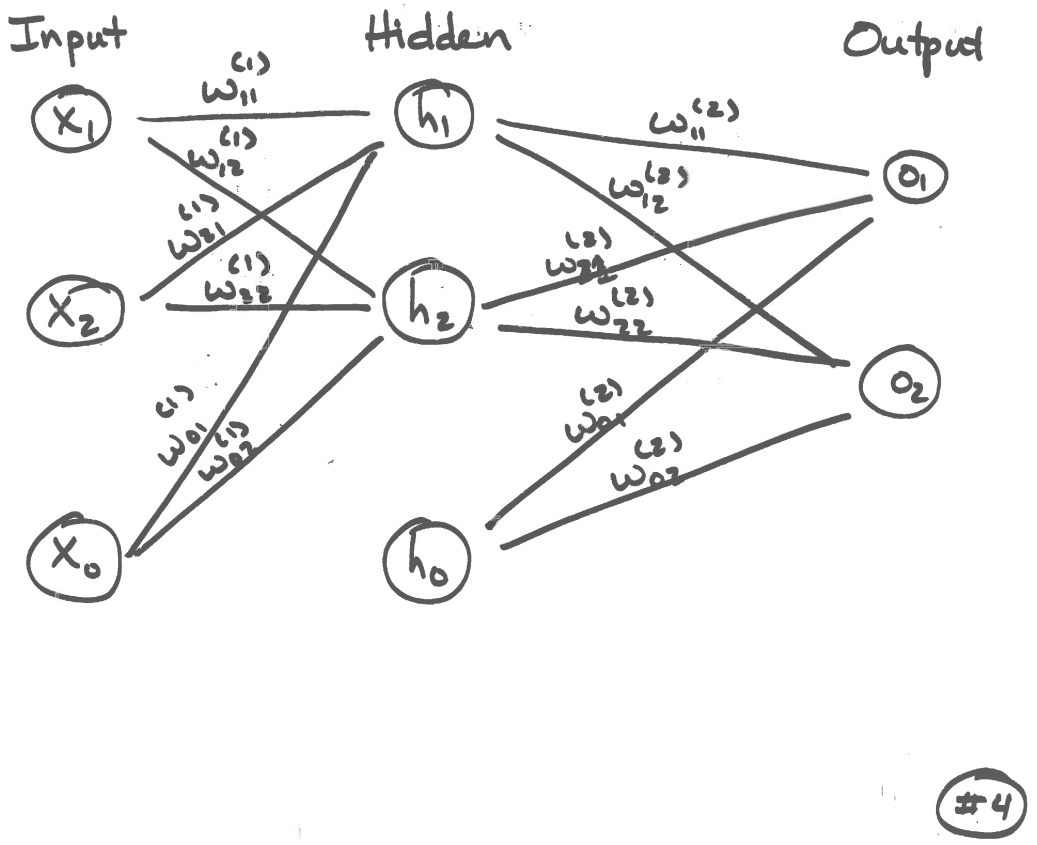

Figure 4: 2 inputs, 1 hidden layer with two nodes each, two output

Derivatives:

$$\frac{dJ}{dw_{i1}^{(2)}} = (o_1 - t_1) \cdot o_1(1 - o_1)h_i$$

$$\frac{dJ}{dw_{i2}^{(2)}} = (o_2 - t_2) \cdot o_2(1 - o_2)h_i$$

$$\frac{dJ}{dw_{i1}^{(1)}} = \frac{dJ}{do_1} \cdot \frac{do_1}{dh_1} \cdot \frac{dh_1}{dw_{i1}^{(1)}} + \frac{dJ}{do_2} \cdot \frac{do_2}{dh_1} \cdot \frac{dh_1}{dw_{i1}^{(1)}}$$

$$= (o_1 - t_1) \cdot o_1(1 - o_1) \cdot w_{11}^{(2)} \cdot h_1(1 - h_1)x_i + (o_2 - t_2) \cdot o_2(1 - o_2) \cdot w_{12}^{(2)} \cdot h_1(1 - h_1)x_i$$

$$\frac{dJ}{dw_{i2}^{(1)}} = \frac{dJ}{do_1} \cdot \frac{do_1}{dh_2} \cdot \frac{dh_2}{dw_{i2}^{(1)}} + \frac{dJ}{do_2} \cdot \frac{do_2}{dh_2} \cdot \frac{dh_2}{dw_{i2}^{(1)}}$$

$$= (o_1 - t_1) \cdot o_1(1 - o_1) \cdot w_{21}^{(2)} \cdot h_2(1 - h_2)x_i + (o_2 - t_2) \cdot o_2(1 - o_2) \cdot w_{22}^{(2)} \cdot h_2(1 - h_2)x_i$$

Update:

$$w_{i1}^{(2)} = w_{i1}^{(2)} - \gamma \left(\frac{dJ}{dw_{i1}^{(2)}} \right)$$

$$w_{i2}^{(2)} = w_{i2}^{(2)} - \gamma \left(\frac{dJ}{dw_{i2}^{(2)}} \right)$$

$$w_{i1}^{(1)} = w_{i1}^{(1)} - \gamma \left(\frac{dJ}{dw_{i1}^{(1)}} \right)$$

$$w_{i2}^{(1)} = w_{i2}^{(1)} - \gamma \left(\frac{dJ}{dw_{i2}^{(1)}} \right)$$

Case 5

Please refer to Figure 5 to trace and understand the steps. Remember, we rely on the chain rule since each hidden node and output node are composed functions.

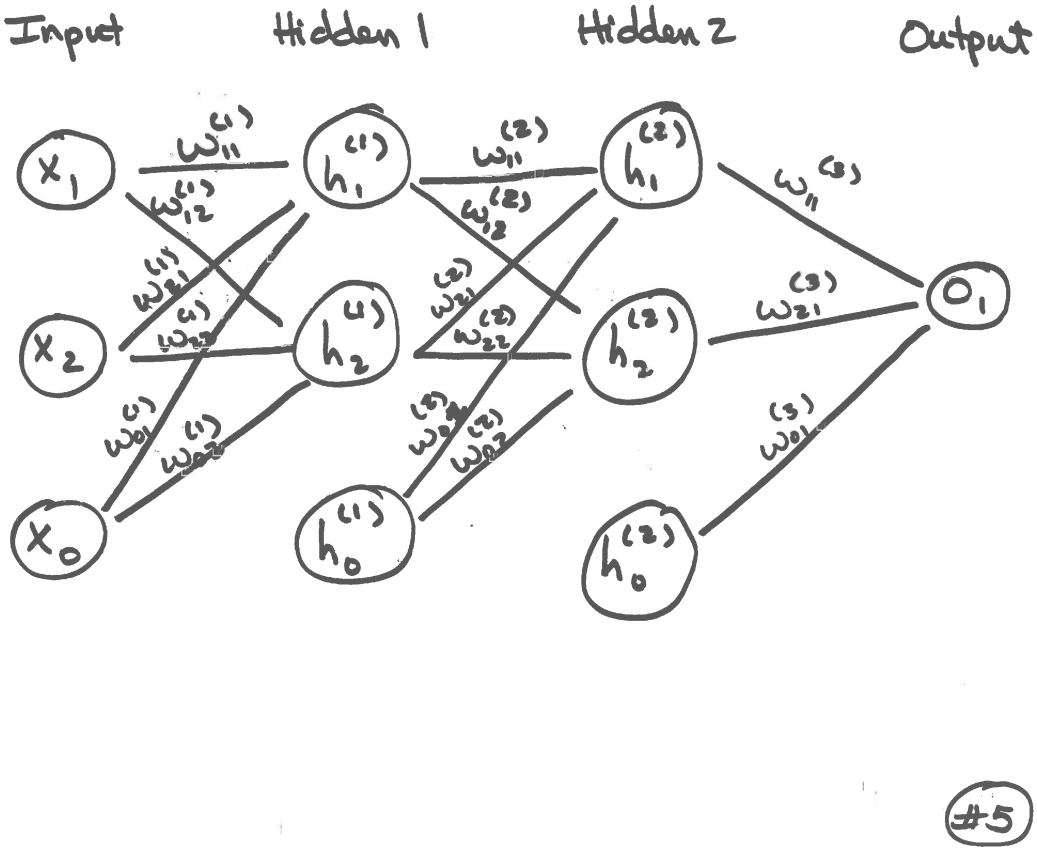

Figure 5: 2 inputs, 2 hidden layer with two nodes each, one output

Feedforward:

$$h_1^{(1)} = \sigma\left(w_{01}^{(1)} \cdot x_0 + w_{11}^{(1)} \cdot x_1 + w_{21}^{(1)} \cdot x_2\right)$$

$$h_2^{(1)} = \sigma\left(w_{02}^{(1)} \cdot x_0 + w_{12}^{(1)} \cdot x_1 + w_{22}^{(1)} \cdot x_2\right)$$

$$h_1^{(2)} = \sigma\left(w_{01}^{(2)} \cdot h_0^{(1)} + w_{11}^{(2)} \cdot h_1^{(1)} + w_{21}^{(2)} \cdot h_2^{(1)}\right)$$

$$h_2^{(2)} = \sigma\left(w_{02}^{(2)} \cdot h_0^{(1)} + w_{12}^{(2)} \cdot h_1^{(1)} + w_{22}^{(2)} \cdot h_2^{(1)}\right)$$

$$o_1 = \sigma\left(w_{01}^{(3)} \cdot h_0^{(2)} + w_{11}^{(3)} \cdot h_1^{(2)} + w_{21}^{(3)} \cdot h_2^{(2)}\right)$$

Error(s):

$$J(o_1) = 0.5(o_1 - t_1)^2$$

Derivatives:

$$\frac{dJ}{dw_{i1}^{(3)}} = (o_1 - t_1) \cdot o_1(1 - o_1) h_i^{(2)}$$

$$\frac{dJ}{dw_{i1}^{(2)}} = (o_1 - t_1) \cdot o_1(1 - o_1) \cdot h_1^{(2)} \cdot w_{11}^{(3)} \cdot (1 - h_1^{(2)}) \cdot h_i^{(1)}$$

$$\frac{dJ}{dw_{i2}^{(2)}} = (o_1 - t_1) \cdot o_1(1 - o_1) \cdot h_2^{(2)} \cdot w_{21}^{(3)} \cdot (1 - h_2^{(2)}) \cdot h_i^{(1)}$$

$$\begin{aligned}\frac{dJ}{dw_{i1}^{(1)}} &= (o_1 - t_1) \cdot o_1(1 - o_1) \cdot h_1^{(2)} \cdot w_{11}^{(3)} \cdot (1 - h_1^{(2)}) \cdot w_{11}^{(2)} \cdot h_1^{(1)}(1 - h_1^{(1)}) \cdot x_i \\ &+ (o_1 - t_1) \cdot o_1(1 - o_1) \cdot h_2^{(2)} \cdot w_{21}^{(3)} \cdot (1 - h_2^{(2)}) \cdot w_{12}^{(2)} \cdot h_1^{(1)}(1 - h_1^{(1)}) \cdot x_i\end{aligned}$$

$$\begin{aligned}\frac{dJ}{dw_{i2}^{(1)}} &= (o_1 - t_1) \cdot o_1(1 - o_1) \cdot h_1^{(2)} \cdot w_{11}^{(3)} \cdot (1 - h_1^{(2)}) \cdot w_{21}^{(2)} \cdot h_2^{(1)}(1 - h_2^{(1)}) \cdot x_i \\ &+ (o_1 - t_1) \cdot o_1(1 - o_1) \cdot h_2^{(2)} \cdot w_{21}^{(3)} \cdot (1 - h_2^{(2)}) \cdot w_{22}^{(2)} \cdot h_2^{(1)}(1 - h_2^{(1)}) \cdot x_i\end{aligned}$$

The updates for each set of weights still use the gradient descent method discussed above.

Case 6

For the last case, we leave you with a diagram so that you might attempt to determine the steps based on the above. We hope at this point the pattern is recognizable and the code in NNcasebycase.py is now understandable.

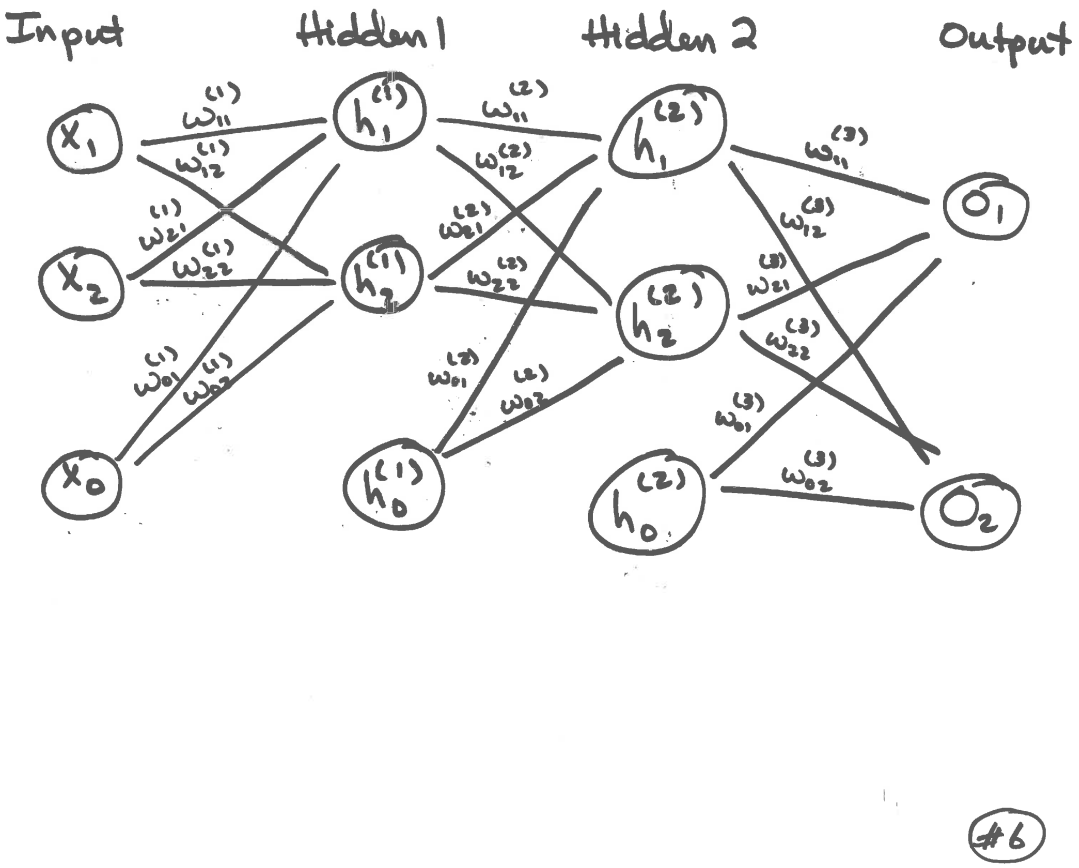

Figure 6: 2 inputs, 2 hidden layer with two nodes each, two output