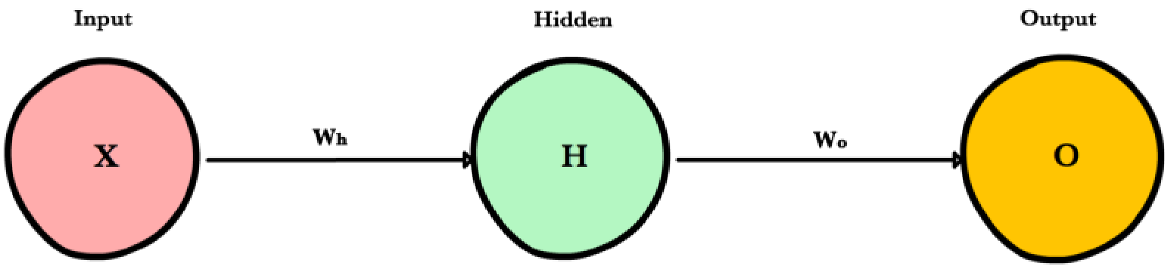
Forward Propagation

Forward propagation is the way neural networks make predictions given inputs because data is *forward propagated* thought the network. In a feedforward neural network, the sum of the products of the inputs and their weights are calculated. This is then fed to the output. As it makes it's way through the network is makes many small computations using activation functions (X, H, and O below). These computations take place at each layer within the network and propagate their results forward to the next layer. Between each layer are *weights* (Wh and Wo below) that are multiplied against the propagated values before they are computed by the activation unit. This continues until the final layer, the output layer, produces the prediction.



Basic steps:

> 1. Calculate the weighted input to the hidden layer by multiplying X by the hidden weight Wh
> 2. Apply the activation function and pass the result to the final layer
> 3. Repeat step 2 except this time X is replaced by the hidden layer's output, H

Matrix Dimensions: Passing Values Between Layers

The values passed between layers in a neural network a represented as a matrix of values (or weights) as denoted by Θ or W where:

> $\Theta^{(j)}_{mn}$ is a matrix of values controlling function mapping from layer j to the next layer, $j+1$ and m and n are the row and column of the matrix value.
>
> $\Theta^{(2)}_{12}$ is a matrix of values controlling function mapping from layer 2 to layer 3 and the value at $1, 2$.

Input thetas to an activation function are superscripted with the index of the calling layer where:

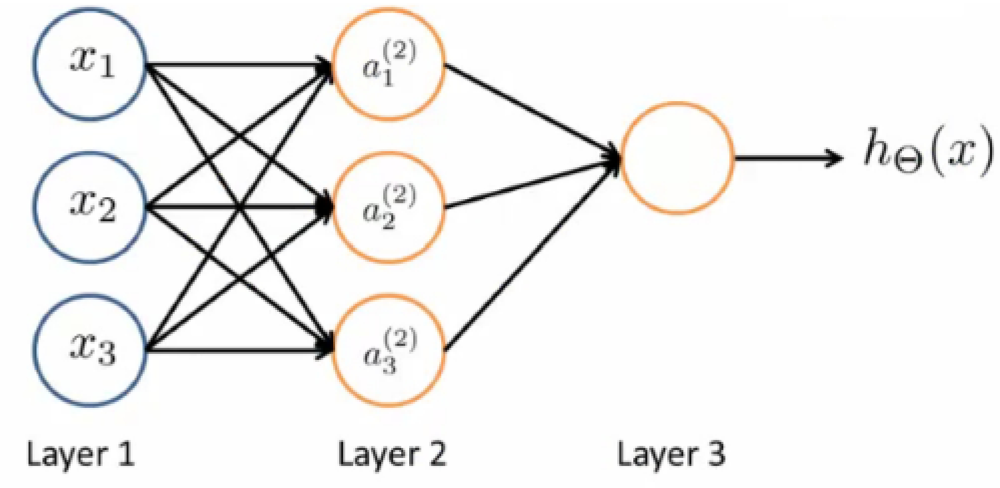
> $\Theta^{(2)}$ matrix would be input to activation function $a^{(3)}_i$ which would output a $\theta^{(3)}$ matrix.

In neural networks, one of the major challenges is getting matrix dimensions right. If a neural network has s_j activation functions in layer j and s_{j+1} in layer $j+1$, then $\theta^{(j)}$ will be of dimension s_{j+1} x $(s_j + 1)$. We say $+1$ because we are adding a bias unit of 1 in the same way that's done for linear and logistic regression. In other words, the number of rows (m) is the number of activation functions in the next layer and the number of columns (n) is the number of activation functions plus 1 from the current layer. For example:

> If layer 2 contains 20 activation functions and layer 3 contains 30 activation functions
>
> then, the dimensions of the Θ matrix would be 30 x 21.

Bottom line is, the *rows* for the matrix must be equal to the number of activation functions in the previous layer and the number of *columns* must be equal to the number of activation functions in the next layer.

Layer Computation: Activation Function Calculations



For layer 1, which are our x values, we can reference as $a^{(1)}j$ just as if it were the results from any other activation function:

> **Layer 1 (Input Layer)**
>
> $a_j^{(1)} = x_j$

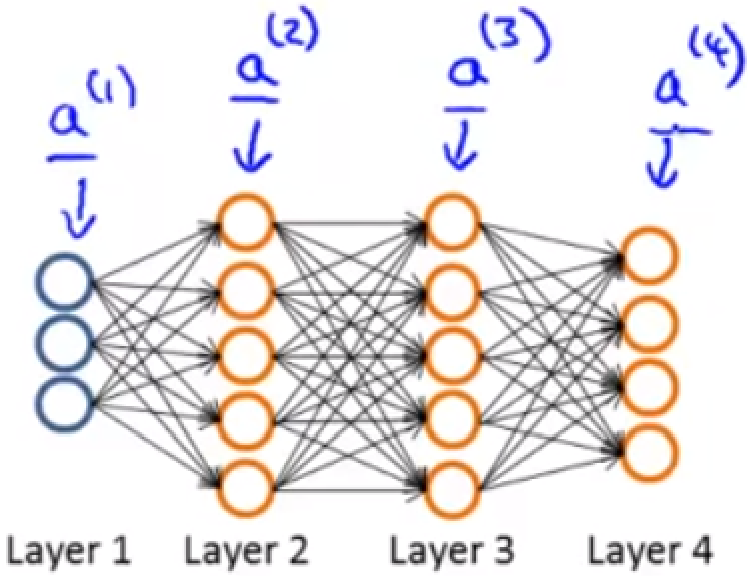
In referencing the diagram above, below is an example with the three activation functions in layer 2 each using a *Sigmoid* activation function g:

> **Layer 2**
>
> $a_1^{(2)} = g(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3) = g(z_1^{(2)})$
>
> $a_2^{(2)} = g(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3) = g(z_2^{(2)})$
>
> $a_3^{(2)} = g(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3) = g(z_3^{(2)})$

The hypothesis function in layer 3 would be:

> **Layer 3 (Output Layer)**
>
> $h_\Theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)}a_0^{(2)} + \Theta_{11}^{(2)}a_1^{(2)} + \Theta_{12}^{(2)}a_2^{(2)} + \Theta_{13}^{(2)}a_3^{(2)}) = g(z^{(3)})$

4 Layer Example



In summary, for a 4 layer neural network with a single x value it would be:

> $a^{(1)} = x$
>
> $z^{(2)} = \Theta^{(1)} a^{(1)}$
>
> $a^{(2)} = g(z^{(2)})$ plus $a_0^{(2)}$
>
> $z^{(3)} = \Theta^{(2)} a^{(2)}$
>
> $a^{(3)} = g(z^{(3)})$ plus $a_0^{(3)}$
>
> $z^{(4)} = \Theta^{(3)} a^{(3)}$
>
> $a^{(4)} = h_\Theta(x) = g(z^{(4)})$

Vectorized Calculations

The following illustrates the input, vectorized computation for layer 2 and output in layer 3. This process is also called *forward propagation*.

Layer 1 (input) is expressed as a 4 dimensional vector (matrix) in this case which includes a bias column (matrix values are arbitrary). Note that layer 1 can also be reference as $a^{(1)}$:

> $$a^{(1)} = x = \begin{bmatrix} 1 & 1 & 2 & 3 \\ 1 & 4 & 5 & 6 \\ 1 & 7 & 8 & 9 \end{bmatrix}$$

Our theta values, used was weights for layer 2, would be expressed as a 4 dimensional vector which serves as the *mapping* between layer 1 and layer 2. The *mapping* happens in the black lines in the illustration above.

> $$\Theta^{(1)} = \begin{bmatrix} \Theta_{10}^{(1)} & \Theta_{11}^{(1)} & \Theta_{12}^{(1)} & \Theta_{13}^{(1)} \\ \Theta_{20}^{(1)} & \Theta_{21}^{(1)} & \Theta_{22}^{(1)} & \Theta_{23}^{(1)} \\ \Theta_{30}^{(1)} & \Theta_{31}^{(1)} & \Theta_{32}^{(1)} & \Theta_{33}^{(1)} \end{bmatrix}$$

In layer 2, we want to call our Sigmoid function with the input passed in to layer 1 where:

> $z^{(2)} = \Theta^{(1)} x = \Theta^{(1)} a^{(1)}$

Therefore our "new x values" used as input to layer 3 would be:

> $a^{(2)} = g(z^{(2)}) = g(\Theta^{(1)} x) = g(\Theta^{(1)} a^{(1)})$

In layer 3, we would have a different set of weights and to compute our hypothesis, which service as the *mapping* between layer 2 and layer 3. We would our output for layer 2 as follows where $a_0^{(2)}$ is the bias unit of 1 where:

> $z^{(3)} = \Theta^{(2)} a^{(2)}$

Expanded, would be:

> $h_\Theta(x) = g(z^{(3)}) = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$