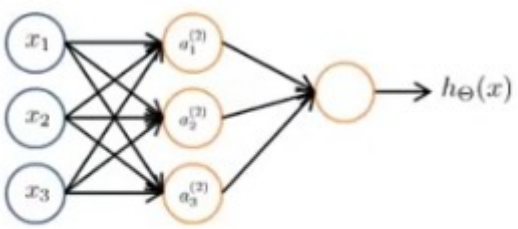
~ Model Representation II ~

Forward propagation: Vectorized implementation

We consider the following neural network. Previously we said that the sequence of steps that we need in order to compute the output of a hypotheses is the equations given below where we compute the activation values of the three hidden neurons uses and then we use those to compute the final output of our hypotheses $h_\theta(x)$.



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

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

We're going to define a few extra terms:

$$a_1^{(2)} = g(z_1^{(2)})$$
$$a_2^{(2)} = g(z_2^{(2)})$$
$$a_3^{(2)} = g(z_3^{(2)})$$

So we have that $a_1^{(2)}$, which this term is equal to $g(z_1^{(2)})$. The superscript 2, what that means is that the $z_1^{(2)}$ and $a_1^{(2)}$ as well, the superscript 2 in parentheses means that these are values associated with layer 2, that is with the hidden layer in the neural network.

where:

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

So the z values are just a linear combination, a weighted linear combination, of the input values x_0, x_1, x_2, x_3 that go into a particular neuron. With this observation we're going to be able to vectorize the computation of the neural network.

$$x = \begin{bmatrix} x_0 \\ x_1 \\ \cdots \\ x_n \end{bmatrix} z^{(j)} = \begin{bmatrix} z_1^{(j)} \\ z_2^{(j)} \\ \cdots \\ z_n^{(j)} \end{bmatrix}$$

We can now vectorize the computation of $a_1^{(2)}$, $a_2^{(2)}$, $a_3^{(2)}$ as follows:

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

$$a^{(2)} = g(z^{(2)})$$

Just to be clear $z^{(2)}$, is a three-dimensional vector ($z^{(2)} \in \mathbb{R}^3$) and $a^{(2)}$ is also a three-dimensional vector ($a^{(2)} \in \mathbb{R}^3$). To make our notation a little more consistent with what we'll do later, the input layer we have the inputs x, but we can also thing it is as in activations of the first layers.

Setting $x = a^{(1)}$, we can rewrite the equation as:

$$z^{(j)} = \Theta^{(j-1)} a^{(j-1)}$$

To take care of the extra bias unit in the neural network, what we're going to do is add an extra neuron $a_0^{(2)}$, where $a_0^{(2)}$ is always equal to 1. We now have that $a^{(2)}$ is going to be a four dimensional feature vector ($a^{(2)} \in \mathbb{R}^4$) because we just added the extra feature ($a_0^{(2)}$ which is equal to 1 corresponding to the bias unit in the hidden layer).

And finally, to compute the actual value output of our hypotheses $h_\Theta(x)$, we then simply need to compute $z^{(3)}$ where:

$$z^{(3)} = \Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)} .$$

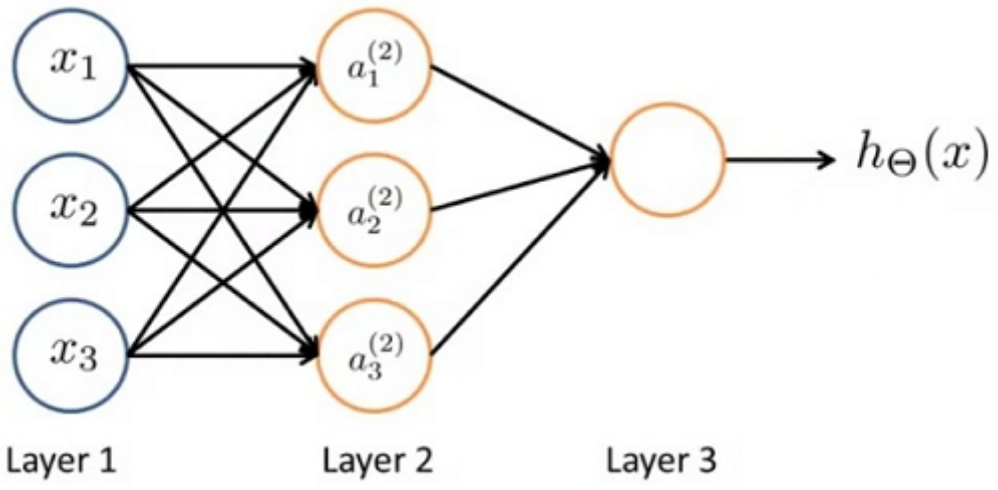
$$z^{(3)} = \Theta^{(2)} a^{(2)}$$

$$h_\Theta(x) = a^{(3)} = g(z^{(3)})$$

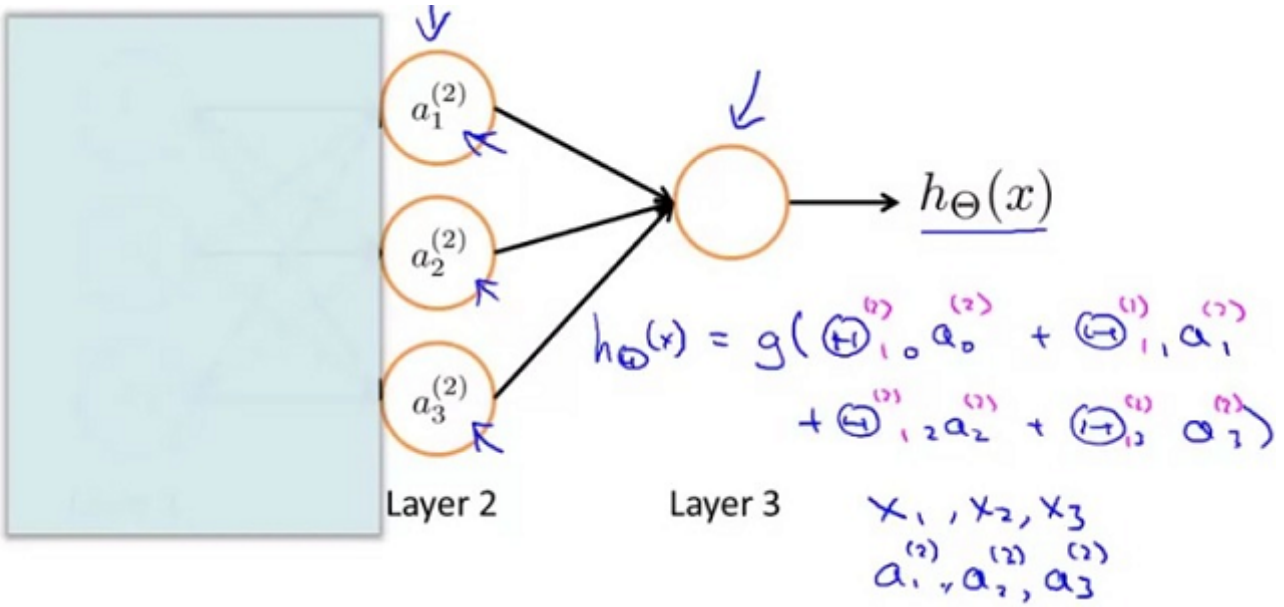
This process of computing $h_\Theta(x)$ is also called **forward propagation** and is called that because we start of with the activations of the **input-units** and then we sort of **forward-propagate** that to the hidden layer and compute the activations of the hidden layer and then we sort of forward propagate that and compute the activations of the output layer, but this process of computing the activations from the input then the hidden then the output layer, and that's also called forward propagation

Neural Network learning its own features

This forward propagation view also helps us to understand what Neural Networks might be doing. Diagram below looks a lot like logistic regression.



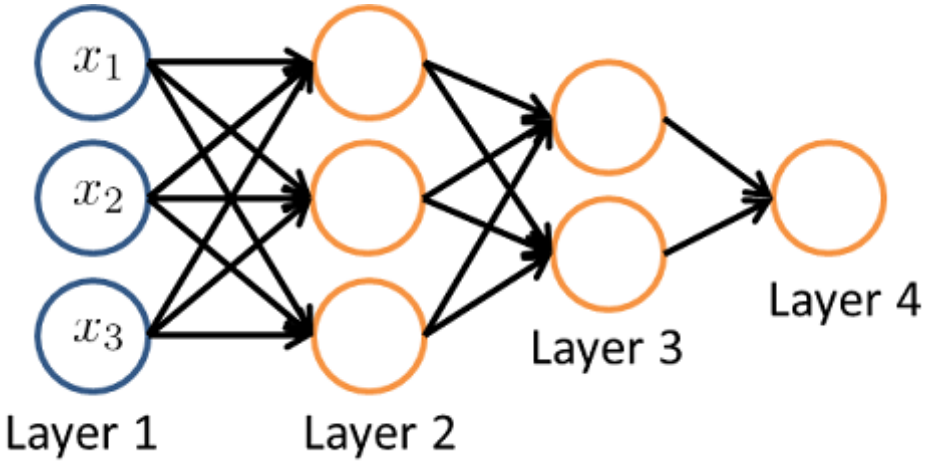
Let's say we cover up the left path of the picture, the resulting diagram below looks alot like logistic regression. What we're doing is using a logistic regression unit and we're using that to make a prediction $h_\Theta(x)$. What the neural network is doing is just like logistic regression, except that rather than using the original features x_1, x_2, x_3, is using the new features $a_1^{(2)}$, $a_2^{(2)}$, $a_3^{(2)}$.



And the great thing about this, is that the features $a_1^{(2)}$, $a_2^{(2)}$, $a_3^{(2)}$, they themselves are learned as functions of the input. Concretely, the function mapping from layer 1 to layer 2, that is determined by some other set of parameters, $\Theta^{(1)}$.

So it's as if the neural network, instead of being constrained to feed the features x_1, x_2, x_3 to logistic regression. **It gets to learn its own features, $a_1^{(2)}$, $a_2^{(2)}$, $a_3^{(2)}$, to feed into the logistic regression!**

Video Question: Consider the network:



Let $a^{(1)} = x \in \mathbb{R}^{n+1}$ denote the input (with $a_0^{(1)} = 1$.

How would you compute $a^{(2)}$?

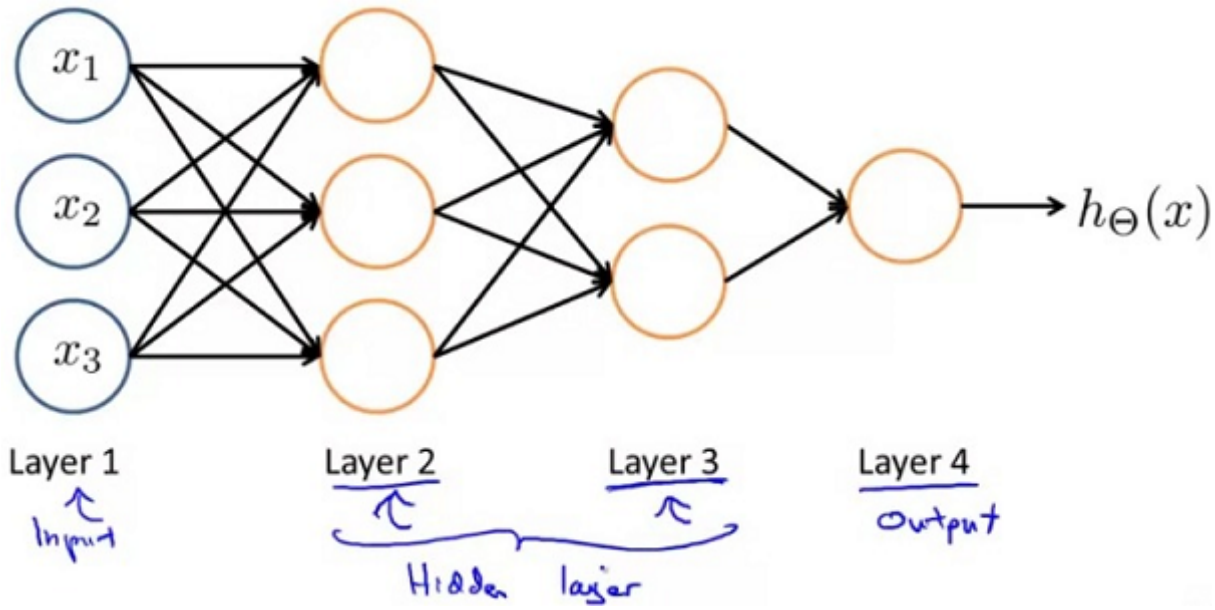
- $a^{(2)} = \Theta^{(1)} a^{(1)}$
- $z^{(2)} = \Theta^{(2)} a^{(1)};\ a^{(2)} = g(z^{(2)})$

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

- $z^{(2)} = \Theta^{(2)} g(a^{(1)});\ a^{(2)} = g(z^{(2)})$

Other network architectures

We can have neural networks with other types of diagrams as well, and the way that neural networks are connected, that's called the **architecture**. So the term architecture refers to how the different neurons are connected to each other. The following image is an example of a different neural network architecture.



Summary

To re-iterate, the following is an example of a neural network:

$$\begin{aligned} a_1^{(2)} &= g(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3) \\ a_2^{(2)} &= g(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3) \\ a_3^{(2)} &= g(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3) \\ h_\Theta(x) = a_1^{(3)} &= g(\Theta_{10}^{(2)}a_0^{(2)} + \Theta_{11}^{(2)}a_1^{(2)} + \Theta_{12}^{(2)}a_2^{(2)} + \Theta_{13}^{(2)}a_3^{(2)}) \end{aligned}$$

In this section we'll do a vectorized implementation of the above functions. We're going to define a new variable $z_k^{(j)}$ that encompasses the parameters inside our g function. In our previous example if we replaced by the variable z for all the parameters we would get:

$$\begin{aligned} a_1^{(2)} &= g(z_1^{(2)}) \\ a_2^{(2)} &= g(z_2^{(2)}) \\ a_3^{(2)} &= g(z_3^{(2)}) \end{aligned}$$

In other words, for layer $j = 2$ and node k, the variable z will be:

$$z_k^{(2)} = \Theta_{k,0}^{(1)}x_0 + \Theta_{k,1}^{(1)}x_1 + \cdots + \Theta_{k,n}^{(1)}$$

The vector representation of x and z^j is:

$$x = \begin{bmatrix} x_0 \\ x_1 \\ \cdots \\ x_n \end{bmatrix} z^{(j)} = \begin{bmatrix} z_1^{(j)} \\ z_2^{(j)} \\ \cdots \\ z_n^{(j)} \end{bmatrix}$$

Setting $x = a^{(1)}$, we can rewrite the equation as:

$$z^{(j)} = \Theta^{(j-1)}a^{(j-1)}$$

We are multiplying our matrix $\Theta^{(j-1)}$ with dimensions $s_j \times (n+1)$ (where s_j is the number of our activation nodes) by our vector $a^{(j-1)}$ with height $(n+1)$. This gives us our vector $z^{(j)}$ with height s_j. Now we can get a vector of our activation nodes for layer j as follows:

$a^{(j)} = g(z^{(j)})$

Where our function g can be applied element-wise to our vector $z^{(j)}$.

We can then add a bias unit (equal to 1) to layer j after we have computed $a^{(j)}$. This will be element $a_0^{(j)}$ and will be equal to 1. To compute our final hypothesis, let's first compute another z vector:

$z^{(j+1)} = \Theta^{(j)} a^{(j)}$

We get this final z vector by multiplying the next theta matrix after $\Theta^{(j-1)}$ with the values of all the activation nodes we just got. This last theta matrix $\Theta^{(j)}$ will have only **one row** which is multiplied by one column $a^{(j)}$ so that our result is a single number. We then get our final result with:

$h_\Theta(x) = a^{(j+1)} = g(z^{(j+1)})$

Notice that in this **last step**, between layer j and layer $j+1$, we are doing **exactly the same thing** as we did in logistic regression. Adding all these intermediate layers in neural networks allows us to more elegantly produce interesting and more complex non-linear hypotheses.