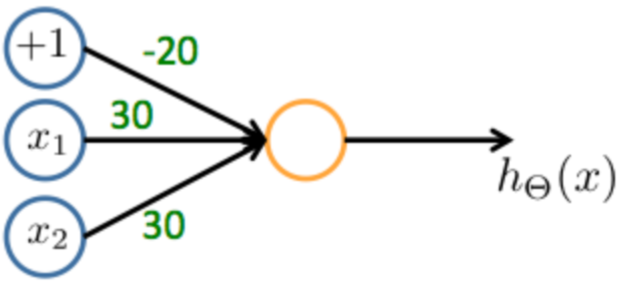
Week 4-1: Neural Networks

1. Which of the following statements are true? Check all that apply.

   (a) Any logical function over binary-valued (0 or 1) inputs x_1 and x_2 can be (approximately) valued represented using some neural network.

   (b) The activation values of the hidden units in a neural network, with the sigmoid function applied at every layer, are always in the range (0, 1).

2. Consider the following neural network which takes two binary-valued inputs $x_1, x_2 \in \{0, 1\}$ and outputs $h_\Theta(x)$. Which of the following logical functions does it approximately compute?

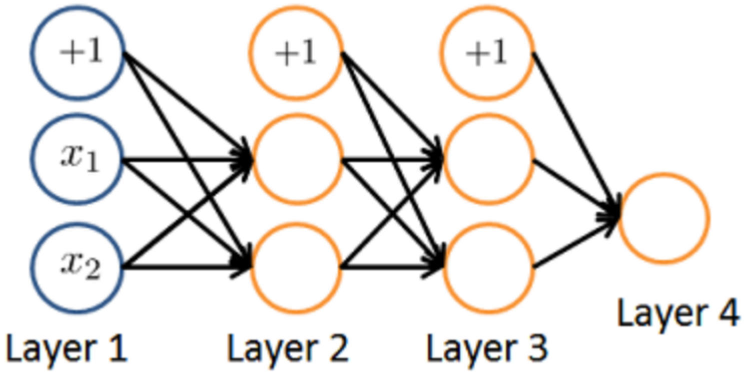


Possible Values:

x_1	x_2	$h_\Theta(x)$	Logistic Output
0	0	-20	0
0	1	10	1
0	1	10	1
1	1	40	1

To calculate $h_\Theta(x)$: evaluate $g(z)$ at each value, as its the sigmoid activation function. Thus, we say $h_\Theta(x) = g(\Theta_{ij}^{(1)} + \Theta_{ij}^{(2)} + \Theta_{ij}^{(3)})$ in this example. So, for the first row of x_1 and x_2, $h_\Theta(x) = -20x_0 + 30x_1 + 30x_2$ and $x_1 = 0$ and $x_2 = 0$. So, $h_\Theta(x) = g(-20(1) + 30(0) + 30(0)) = g(-20)$. As $g(z)$ is the sigmoid function, negative values < -4.6 and positive values > 4.6 are essentially equal to 0 and 1 respectively. Therefore, in this case, $h_\Theta(x) = 0$. Do the same for the rest, and we see that the output is 1 for either x_1 or x_2 being 1.

3. Consider the neural network given below. Which of the following equations correctly computes the activation $a_1^{(3)}$? Note: $g(z)$ is the sigmoid activation function.



> We can identify from the super and subscripts that $a_1^{(3)}$ refers to the first non-bias (node that is not +1) node in layer 3. Thus the calculation of $a_1^{(3)}$ will be derived from a linear combination of Θs from layer 2.
> $\therefore a_1^{(3)} = g\left(\Theta_{1,0}^{(2)} a_0^{(2)} + \Theta_{1,1}^{(2)} a_1^{(2)} + \Theta_{1,2}^{(2)} a_2^{(2)}\right)$

4. You have the following neural network. You'd like to compute the activations of the hidden layer $a^{(2)} \in \mathbb{R}^3$. One way to do so is the following Octave code:

```
% Theta1 is Theta with superscript "(1)" from lecture
% ie, the matrix of parameters for the mapping from layer 1 (input)
% to layer 2
% Theta1 has size 3x3
% Assume 'sigmoid' is a built-in function to compute 1/(1+exp(-z))

a2 = zeros(3, 1);
for i = 1:3
    for j = 1:3
        a2(i) = a2(i) + x(j) + Theta(i, j);
    end
    a2(i) = sigmoid(a2(i));
end
```

You want to have a vectorized implementation of this (i.e., one that does not use any loops). Which of the following implementations correctly compute $a^{(3)}$? Check all that apply.

> (a)
> ```
> z = Theta1 * x;
> a2 = sigmoid(z);
> ```

5. You are using the neural network pictured below and have learned the parameters $\Theta^{(1)} = \begin{bmatrix} 1 & -1.5 & 3.7 \\ 1 & 5.1 & 2.3 \end{bmatrix}$ (used to compute $a^{(2)}$) and $\Theta^{(2)} = \begin{bmatrix} 1 & 0.6 & -0.8 \end{bmatrix}$ (used to compute $a^{(3)}$ as a function of $a^{(2)}$). Suppose you swap the parameters for the first hidden layer between its two units so $\Theta^{(1)} = \begin{bmatrix} 1 & 5.1 & 2.3 \\ 1 & -1.5 & 3.7 \end{bmatrix}$ and also swap the output layer so $\Theta^{(2)} = \begin{bmatrix} 1 & -0.8 & 0.6 \end{bmatrix}$. How will this change the value of the output $h_\Theta(x)$?

Swapping $\Theta^{(1)}$ swaps the hidden layers output of $a^{(2)}$. But the swap of $\Theta^{(2)}$ cancels out the change, so the result will remain unchanged.
It will stay the same.