# Week4

## Table of Contents

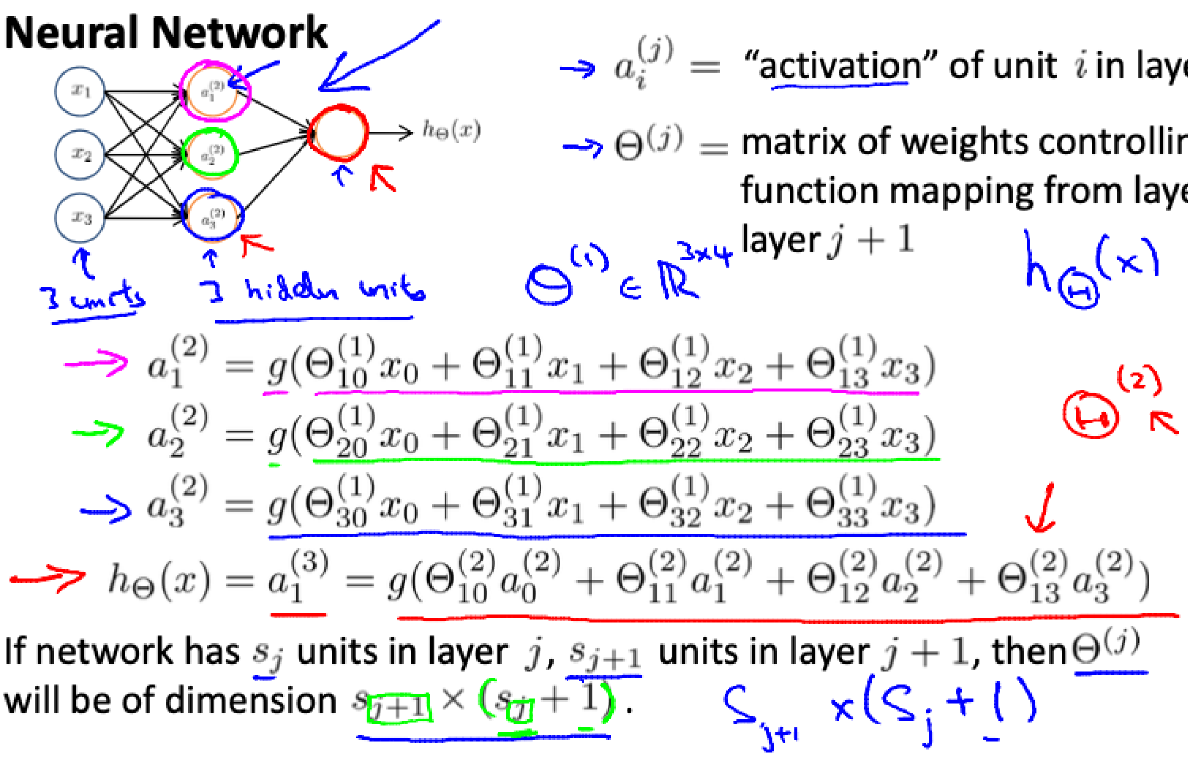


# Neural Networks: Representation

## Model Represent I

Neuron model: Logistic unit Our hypothesis output is the logistic function applied to the sum of the values of our activation nodes, which have been multiplied by yet another parameter matrix Θ(2) containing the weights for our second layer of nodes.

- Each layer gets its own matrix of weight.



# Model Representation II

Forward propagation: Vectorized implementation

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:

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

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

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

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 $z^{(j)}$.

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

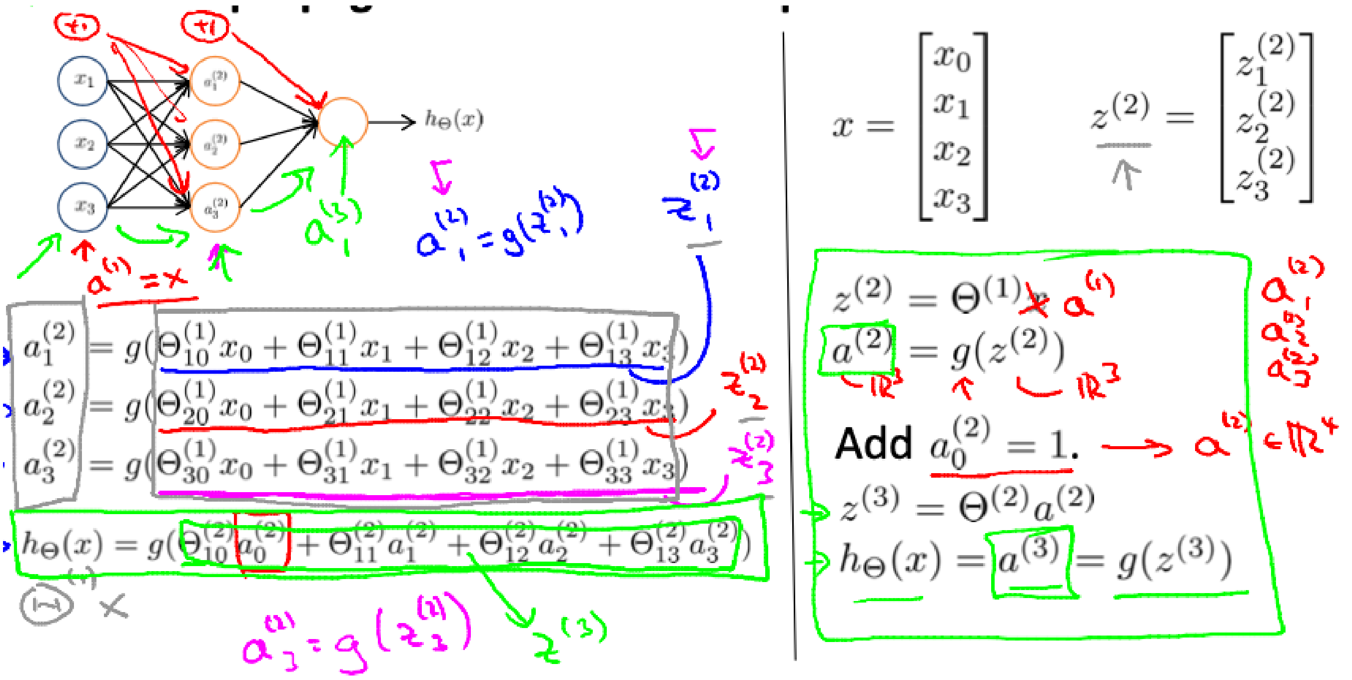
We can then add a bias unit (equal to 1) to layer j after we have computed $a^{(j)}$. This will be element $a_0^{(j)}$ and will be equal to 1. To compute our final hypothesis, let's first compute another z vector:

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

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.




## Examples and Intuitions

Use neural net to implement AND, NOR and OR.

The $\Theta^{(1)}$ matrices for AND, NOR, and OR are:

$$
\begin{aligned}
&AND: \\
&\quad \Theta^{(1)} = \begin{bmatrix} -30 & 20 & 20 \end{bmatrix} \\
&NOR: \\
&\quad \Theta^{(1)} = \begin{bmatrix} 10 & -20 & -20 \end{bmatrix} \\
&OR: \\
&\quad \Theta^{(1)} = \begin{bmatrix} -10 & 20 & 20 \end{bmatrix}
\end{aligned}
$$

We can combine these to get the XNOR logical operator (which gives 1 if $x_1$ and $x_2$ are both 0 or both 1).

$$
\begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} \rightarrow \begin{bmatrix} a_1^{(2)} \\ a_2^{(2)} \end{bmatrix} \rightarrow \begin{bmatrix} a^{(3)} \end{bmatrix} \rightarrow h_\Theta(x)
$$

For the transition between the first and second layer, we'll use a $\Theta^{(1)}$ matrix that combines the values for AND and NOR:

$$
\Theta^{(1)} = \begin{bmatrix} -30 & 20 & 2010 & -20 & -20 \end{bmatrix}
$$

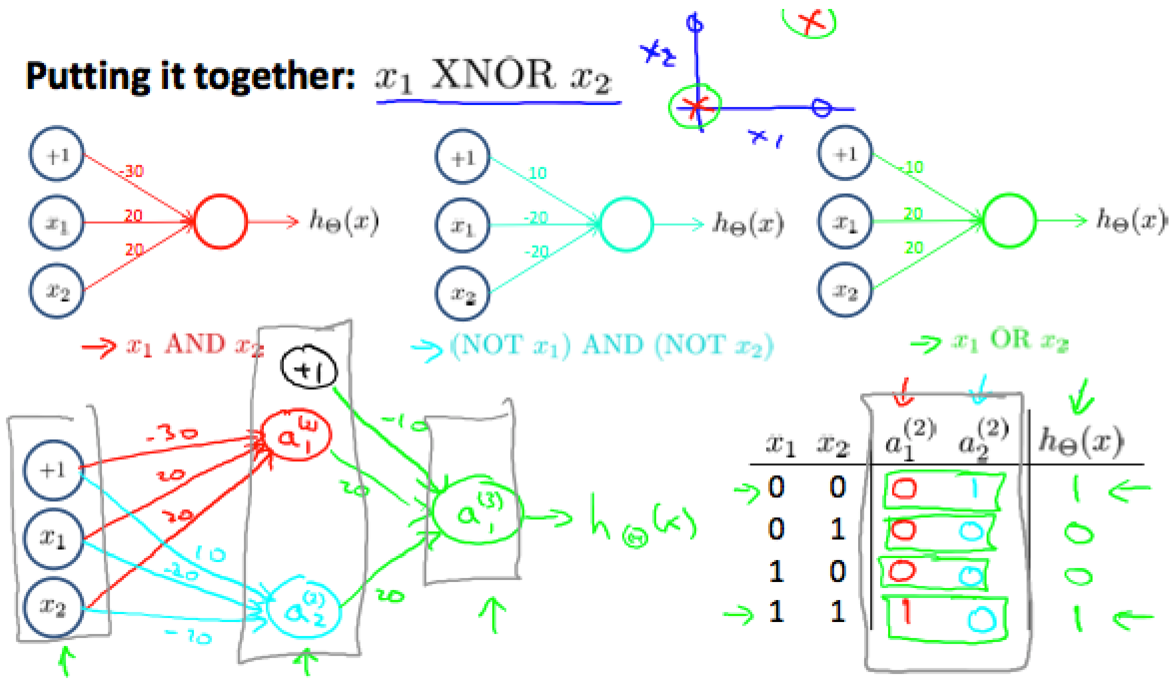
For the transition between the second and third layer, we'll use a $\Theta^{(2)}$ matrix that uses the value for OR:

$$
\Theta^{(2)} = \begin{bmatrix} -10 & 20 & 20 \end{bmatrix}
$$

Let's write out the values for all our nodes:

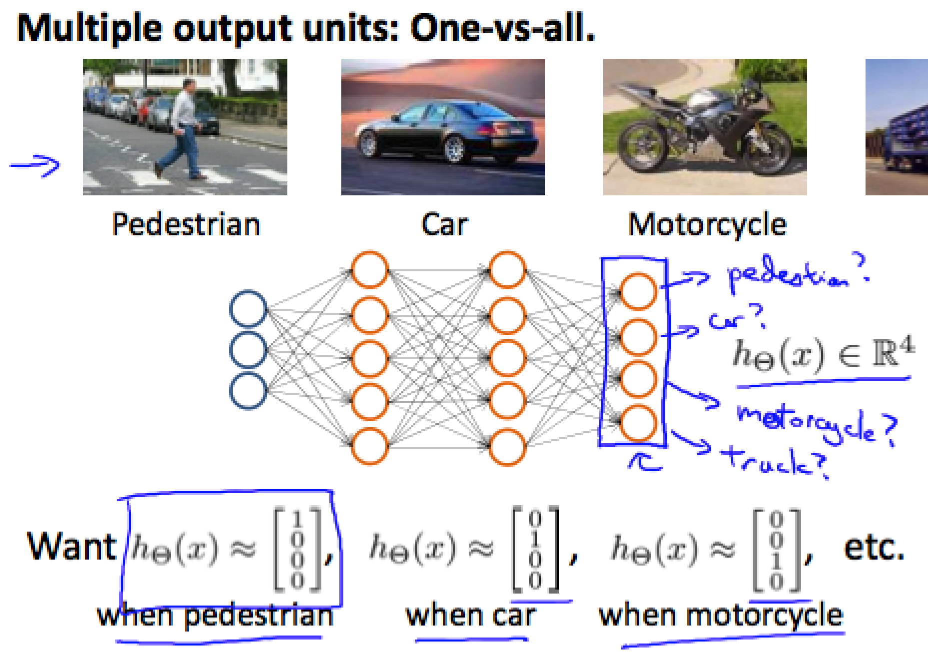
$$
\begin{aligned}
a^{(2)} &= g(\Theta^{(1)} \cdot x) \\
a^{(3)} &= g(\Theta^{(2)} \cdot a^{(2)}) \\
h_\Theta(x) &= a^{(3)}
\end{aligned}
$$

Then we have the XNOR operator using a hidden layer with two nodes. The following summarizes the above algorithms:




## Multiclass Classification

To classify data into multiple classes, we let our hypothesis function return a vector of values. Say we wanted to classify our data into one of four categories. We will use the following example to see how this classification is done. This algorithm takes as input an image and classifies it accordingly:



We can define our set of resulting classes as y:

$$y^{(i)} = \begin{bmatrix}1\\0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\\0\end{bmatrix}, \begin{bmatrix}0\\0\\0\\1\end{bmatrix},$$

Each $y^{(i)}$ represents a different image corresponding to either a car, pedestrian, truck, or motorcycle. The inner layers, each provide us with some new information which leads to our final hypothesis function. The setup looks like:

$$\begin{bmatrix}x_0\\x_1\\x_2\\\cdots\\x_n\end{bmatrix} \rightarrow \begin{bmatrix}a_0^{(2)}\\a_1^{(2)}\\a_2^{(2)}\\\cdots\end{bmatrix} \rightarrow \begin{bmatrix}a_0^{(3)}\\a_1^{(3)}\\a_2^{(3)}\\\cdots\end{bmatrix} \rightarrow \cdots \rightarrow \begin{bmatrix}h_\Theta(x)_1\\h_\Theta(x)_2\\h_\Theta(x)_3\\h_\Theta(x)_4\end{bmatrix}$$

Our resulting hypothesis for one set of inputs may look like:

$$h_\Theta(x) = [0010]$$

In which case our resulting class is the third one down, or $h_\Theta(x)_3$, which represents the motorcycle.

# Quiz

1.

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

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

- [ ] Suppose you have a multi-class classification problem with three classes, trained with a 3 layer network. Let $a_1^{(3)} = (h_\Theta(x))_1$ be the activation of the first output unit, and similarly $a_2^{(3)} = (h_\Theta(x))_2$ and $a_3^{(3)} = (h_\Theta(x))_3$. Then for any input $x$, it must be the case that $a_1^{(3)} + a_2^{(3)} + a_3^{(3)} = 1$.

- [ ] A two layer (one input layer, one output layer; no hidden layer) neural network can represent the XOR function.

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

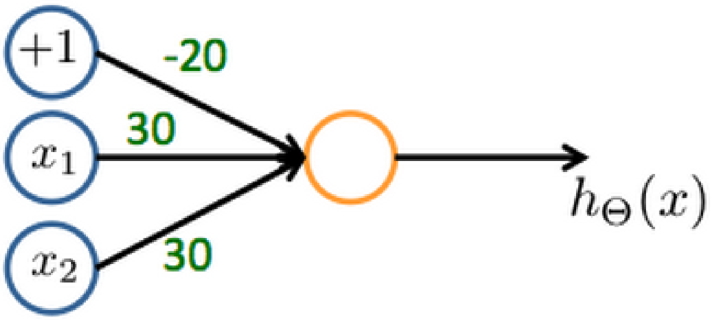
A and D

B: every input x has the chance to output 1.

C: at least 3 layer.

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?

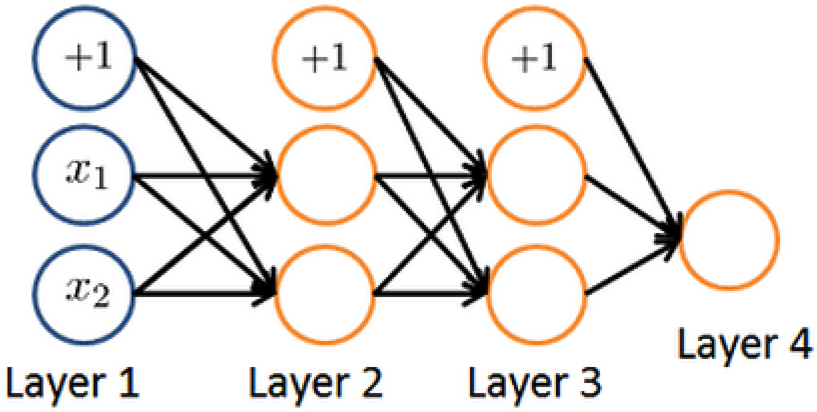



- OR
- AND
- NAND (meaning "NOT AND")
- XOR (exclusive OR)

A

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.

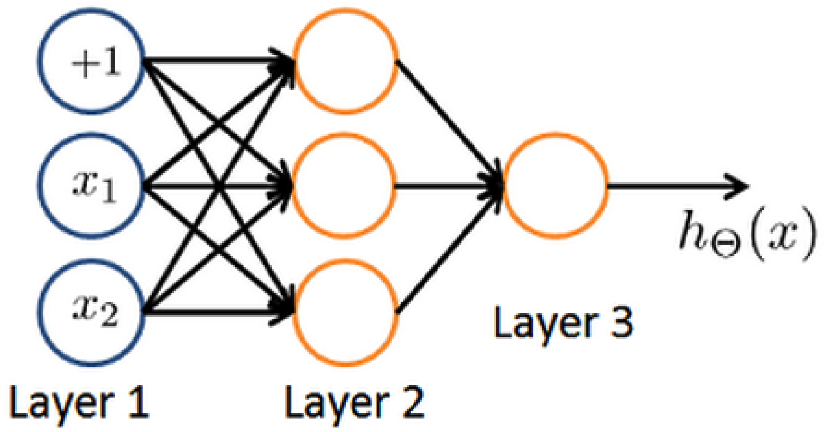



- $a_1^{(3)} = g(\Theta_{1,0}^{(2)} a_0^{(2)} + \Theta_{1,1}^{(2)} a_1^{(2)} + \Theta_{1,2}^{(2)} a_2^{(2)})$
- $a_1^{(3)} = g(\Theta_{1,0}^{(1)} a_0^{(1)} + \Theta_{1,1}^{(1)} a_1^{(1)} + \Theta_{1,2}^{(1)} a_2^{(1)})$
- $a_1^{(3)} = g(\Theta_{1,0}^{(1)} a_0^{(2)} + \Theta_{1,1}^{(1)} a_1^{(2)} + \Theta_{1,2}^{(1)} a_2^{(2)})$
- The activation $a_1^{(3)}$ is not present in this network.

A

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) * Theta1(i, j);
  end
  a2(i) = sigmoid (a2(i));
end
```

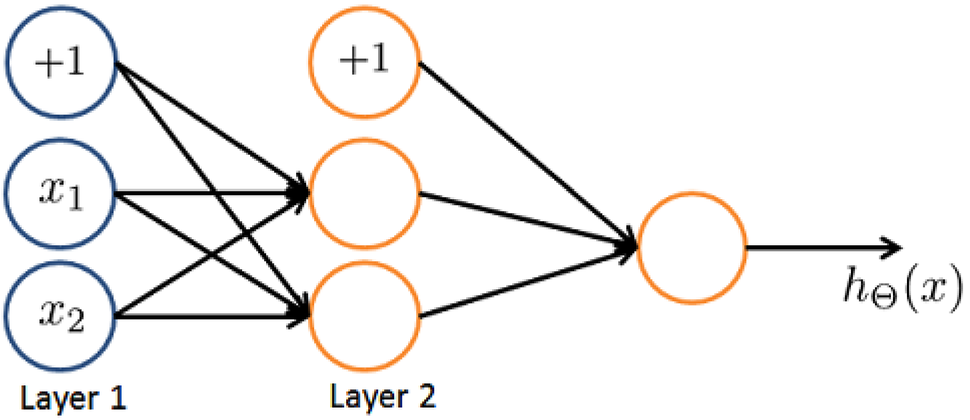
You want to have a vectorized implementation of this (i.e., one that does not use for loops). Which of the following implementations correctly compute $a^{(2)}$? Check all that apply.

- [ ] a2 = sigmoid (Theta1 * x);
- [ ] a2 = sigmoid (x * Theta1);
- [ ] a2 = sigmoid (Theta2 * x);
- [ ] z = sigmoid(x); a2 = Theta1 * z;

A

5.

You are using the neural network pictured below and have learned the parameters $\Theta^{(1)} = \begin{bmatrix} 1 & 2.1 & 1.3 \\ 1 & 0.6 & -1.2 \end{bmatrix}$ (used to compute $a^{(2)}$) and $\Theta^{(2)} = \begin{bmatrix} 1 & 4.5 & 3.1 \end{bmatrix}$ (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 $\Theta^{(1)} = \begin{bmatrix} 1 & 0.6 & -1.2 \\ 1 & 2.1 & 1.3 \end{bmatrix}$ and also swap the output layer so $\Theta^{(2)} = \begin{bmatrix} 1 & 3.1 & 4.5 \end{bmatrix}$. How will this change the value of the output $h_\Theta(x)$?




- It will stay the same.
- It will increase.
- It will decrease
- Insufficient information to tell: it may increase or decrease.

A.