

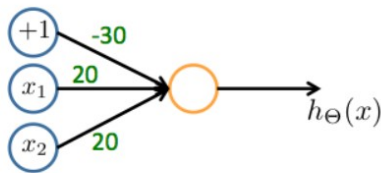
1. Which of the following statements are true? Check all that apply.

1 point

- ☒ Any logical function over binary-valued (0 or 1) inputs x_1 and x_2 can be (approximately) represented using some neural network.
- ☐ A two layer (one input layer, one output layer; no hidden layer) neural network can represent the XOR function.
- ☒ 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 $a_1^{(3)} + a_2^{(3)} + a_3^{(3)} = 1$.

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?

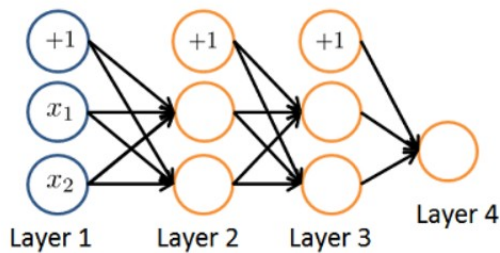
1 point



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

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.

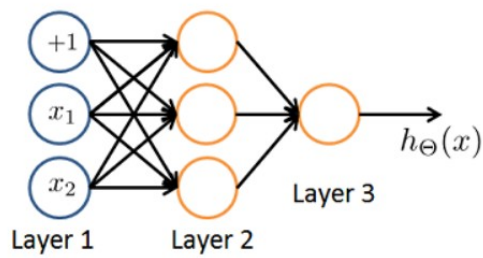
1 point



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

4. You have the following neural network:

1 point



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))

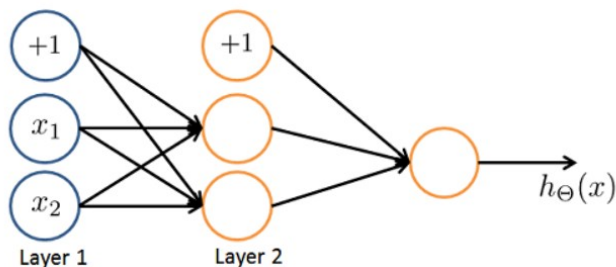
a2 = zeros (3, 1);
for i = 1:3
    for j = 1:3
        a2(i) = a2(i) + x(j) * Theta1(i, j);
    end
    a2(i) = sigmoid (a2(i));
end
```

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.

- ☒ $a2 = \text{sigmoid}(\text{Theta1} * x);$
- ☐ $a2 = \text{sigmoid}(x * \text{Theta1});$
- ☐ $a2 = \text{sigmoid}(\text{Theta2} * x);$
- ☐ $z = \text{sigmoid}(x); a2 = \text{Theta1} * z;$

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 $\Theta^{(2)} = [1 \quad 4.5 \quad 3.1]$ (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 $\Theta^{(2)} = [1 \quad 3.1 \quad 4.5]$. How will this change the value of the output $h_{\Theta}(x)$?

1 point



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