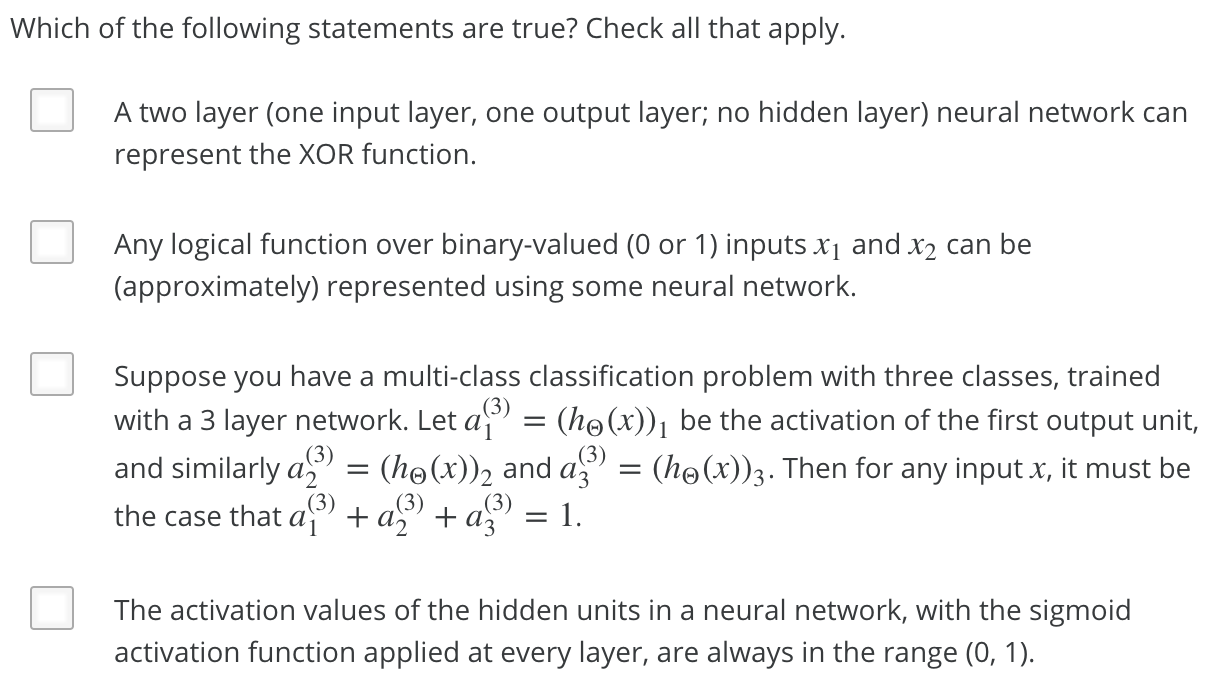
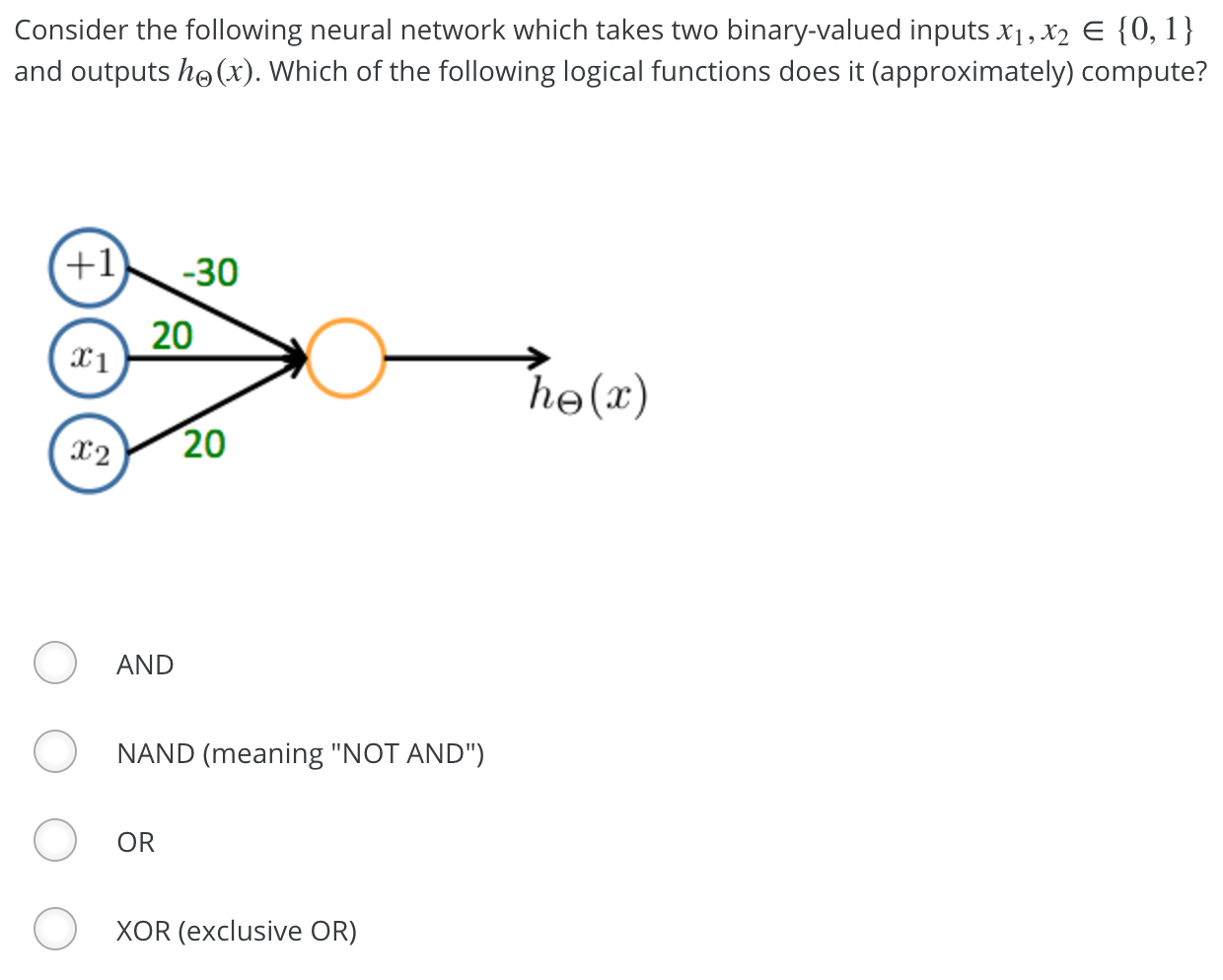
**Machine Learning Week 4 Quiz 1 (Neural Networks: Representation) Stanford Coursera**

**Question 1**

[](https://github.com/mGalarnyk/datasciencecoursera/blob/master/Stanford_Machine_Learning/Week1/data/NeuralRepresentationQ1.png)

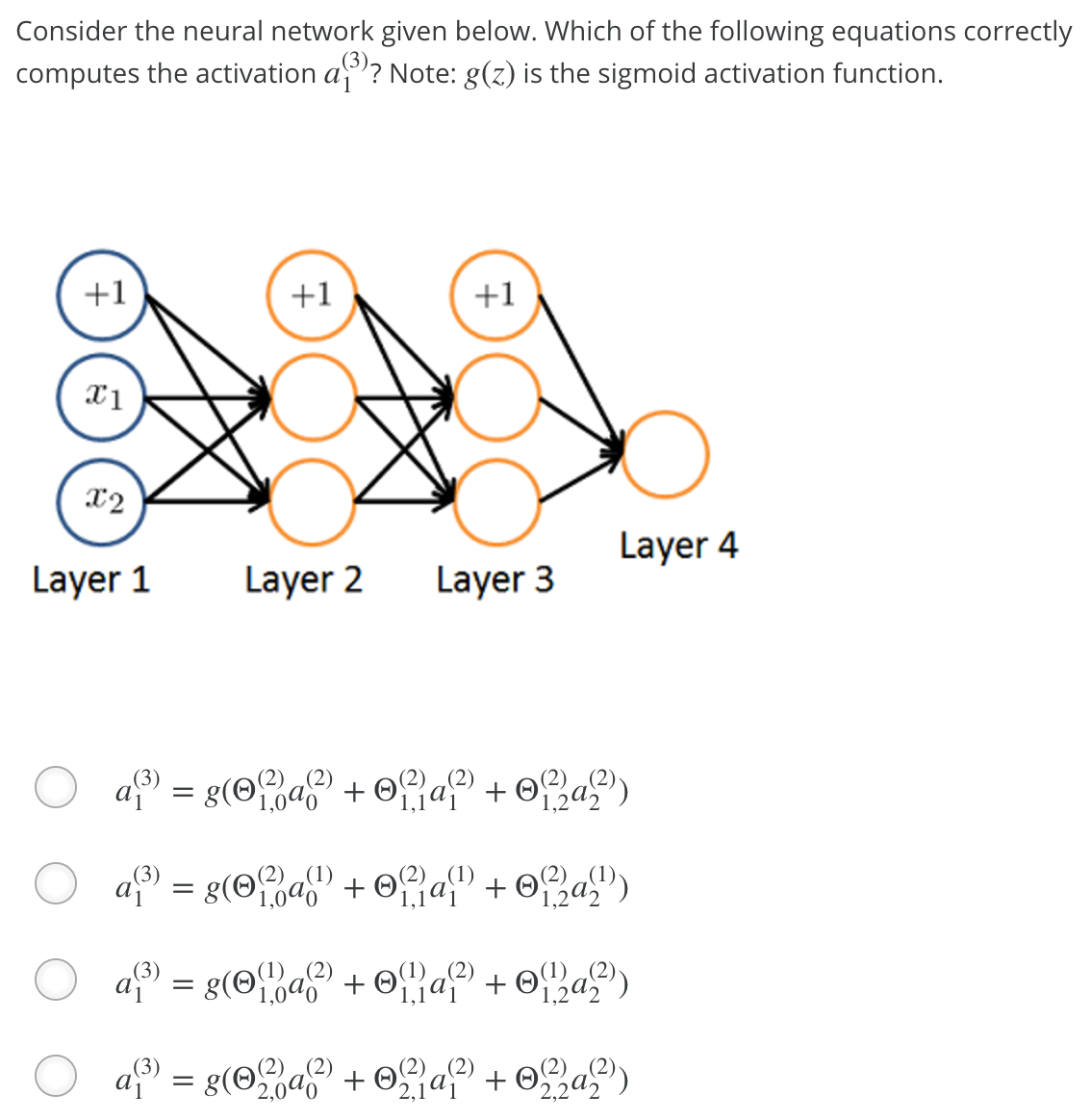
| **True or False** | **Statement** | **Explanation** |
| --- | --- | --- |
| False | A two layer (one input layer, one output layer; no hidden layer) neural network can represent the XOR function. | We must compose multiple logical operations by using a hidden layer to represent the XOR function. |
| True | Any logical function over binary-valued (0 or 1) inputs x1 and x2 can be (approximately) represented using some neural network. | Since we can build the basic AND, OR, and NOT functions with a two layer network, we can (approximately) represent any logical function by composing these basic functions over multiple layers. |
| False | Suppose you have a multi-class classification problem with three classes, trained with a 3 layer network. Let a(3)1=(hΘ(x))1 be the activation of the first output unit, and similarly a(3)2=(hΘ(x))2 and a(3)3=(hΘ(x))3. Then for any input x, it must be the case that a(3)1+a(3)2+a(3)3=1. | The outputs of a neural network are not probabilities, so their sum need not be 1. |
| True | 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). | None Needed |

**Question 2**

[](https://github.com/mGalarnyk/datasciencecoursera/blob/master/Stanford_Machine_Learning/Week1/data/NeuralRepresentationQ2.png)

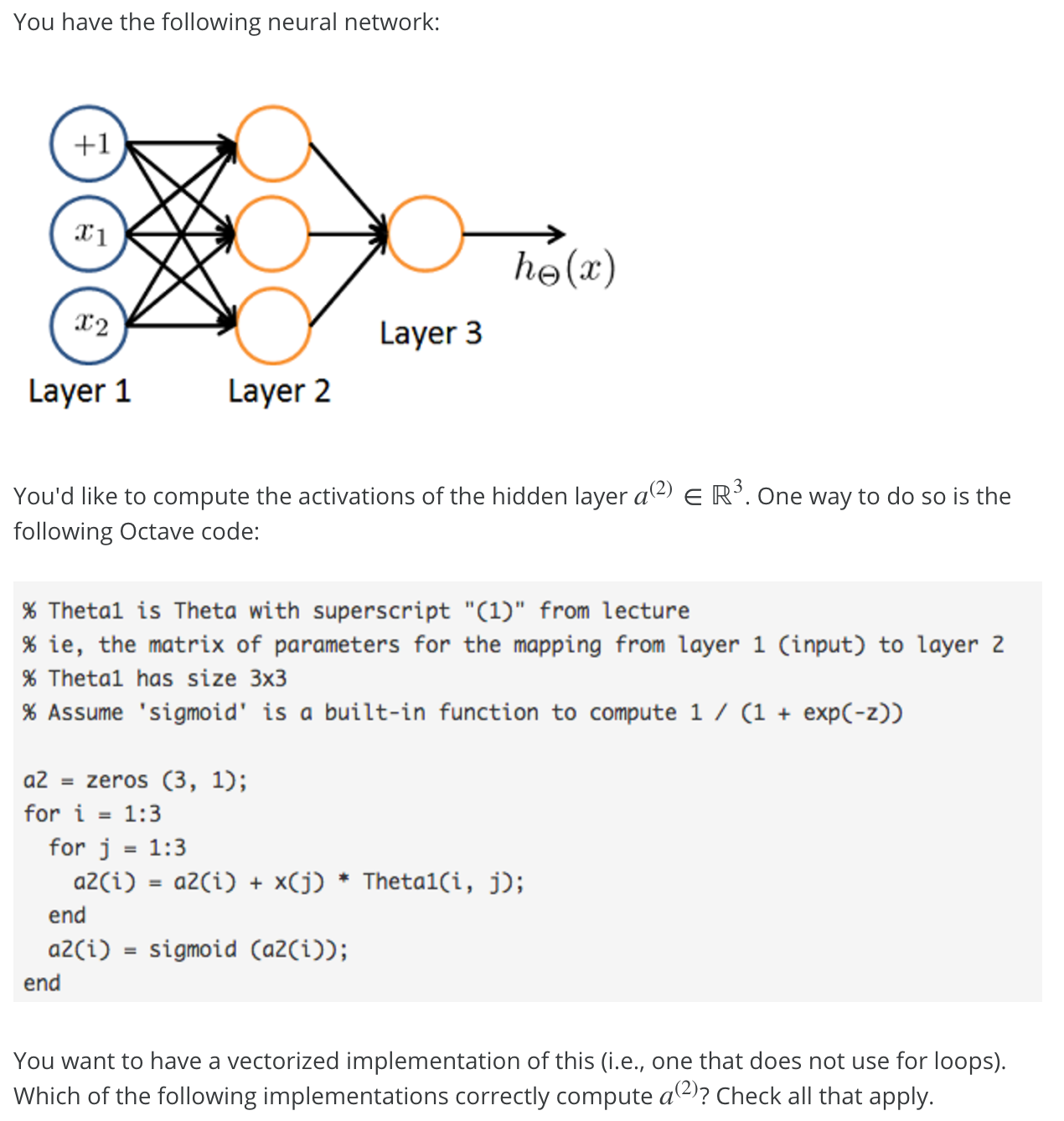
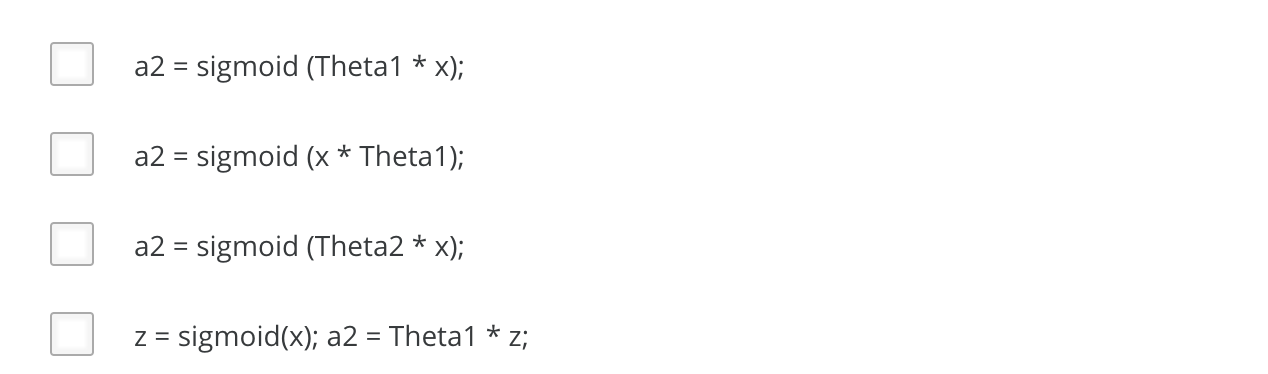
| **Answer** | **Explanation** |
| --- | --- |
| **AND** | [Answer Image](https://github.com/mGalarnyk/datasciencecoursera/blob/master/Stanford_Machine_Learning/Week1/data/NeuralRepresentationQ2answer.jpg) |

**Question 3**

[](https://github.com/mGalarnyk/datasciencecoursera/blob/master/Stanford_Machine_Learning/Week1/data/NeuralRepresentationQ3.png)

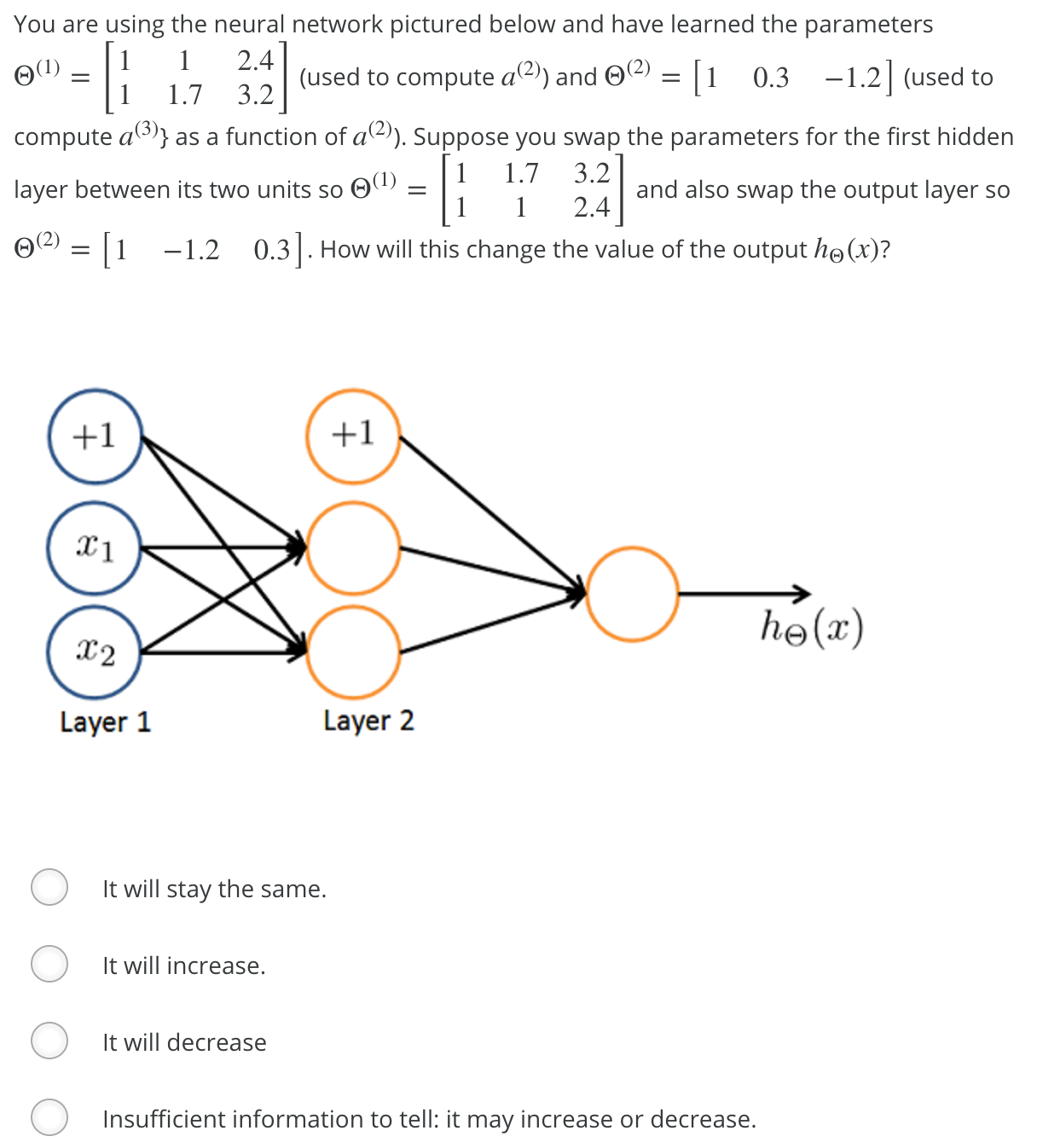
| **Answer** | **Explanation** |
| --- | --- |
| [Answer Image](https://github.com/mGalarnyk/datasciencecoursera/blob/master/Stanford_Machine_Learning/Week1/data/NeuralRepresentationQ3answer.png) | This correctly uses the first row of Θ(2) and includes the "+1" term of a(2)0 |

**Question 4**

[](https://github.com/mGalarnyk/datasciencecoursera/blob/master/Stanford_Machine_Learning/Week1/data/NeuralRepresentationQ4p1.png) [](https://github.com/mGalarnyk/datasciencecoursera/blob/master/Stanford_Machine_Learning/Week1/data/NeuralRepresentationQ4p2.png)

| **Answer** | **Explanation** |
| --- | --- |
| a2 = sigmoid (Theta1 \* x); | In the lecture's notation a(2) = g(Θ(1)x), so this version computes it directly, as the sigmoid function will act element-wise. |

**Question 5**

[](https://github.com/mGalarnyk/datasciencecoursera/blob/master/Stanford_Machine_Learning/Week1/data/NeuralRepresentationQ5.png)

| **Answer** | **Explanation** |
| --- | --- |
| It will stay the same. | Swapping Θ(1) swaps the hidden layers output a^{(2)}. But the swap of Θ(2) cancels out the change, so the output will remain unchanged. |