1. Which of the following are true? (Check all that apply.) 1 / 1 point

- [x] $a^{[2]}$ denotes the activation vector of the 2^{nd} layer.

  Correct

- [x] $a_4^{[2]}$ is the activation output by the 4^{th} neuron of the 2^{nd} layer

  Correct

- [ ] $a^{[2](12)}$ denotes activation vector of the 12^{th} layer on the 2^{nd} training example.
- [ ] X is a matrix in which each row is one training example.
- [x] $a^{[2](12)}$ denotes the activation vector of the 2^{nd} layer for the 12^{th} training example.

  Correct

- [x] X is a matrix in which each column is one training example.

  Correct

- [ ] $a_4^{[2]}$ is the activation output of the 2^{nd} layer for the 4^{th} training example

2. The tanh activation is not always better than sigmoid activation function for hidden units because the mean of its output is closer to zero, and so it centers the data, making learning complex for the next layer. True/ False? 1 / 1 point

- ( ) True
- (•) False

Correct

Yes. As seen in lecture the output of the tanh is between -1 and 1, it thus centers the data which makes the learning simpler for the next layer.

3. Which of these is a correct vectorized implementation of forward propagation for layer l, where $1 \leq l \leq L$? **1 / 1 point**

- $Z^{[l]} = W^{[l-1]} A^{[l]} + b^{[l-1]}$
- $A^{[l]} = g^{[l]}(Z^{[l]})$

- $Z^{[l]} = W^{[l]} A^{[l-1]} + b^{[l]}$
- $A^{[l]} = g^{[l]}(Z^{[l]})$

- $Z^{[l]} = W^{[l]} A^{[l]} + b^{[l]}$
- $A^{[l+1]} = g^{[l]}(Z^{[l]})$

- $Z^{[l]} = W^{[l]} A^{[l]} + b^{[l]}$
- $A^{[l+1]} = g^{[l+1]}(Z^{[l]})$

Correct

4. When building a binary classifier for recognizing cats (y=1) vs raccoons (y=0). Is better to use the sigmoid function as activation function for the hidden layers. True/False **1 / 1 point**

- False
- True

Correct

Yes. Using tanh almost always works better than the sigmoid function for hidden layers.

5. Consider the following code: 1 / 1 point

A = np.random.randn(4,3)

B = np.sum(A, axis = 1, keepdims = True)

What will be B.shape? (If you're not sure, feel free to run this in python to find out).

- ○ (4,)
- ○ (1, 3)
- ◉ (4, 1)
- ○ (3,)

Correct
Yes, we use (keepdims = True) to make sure that A.shape is (4,1) and not (4,). It makes our code more robust.

6. Suppose you have built a neural network. You decide to initialize the weights and biases to be zero. Which of the following statements is true? 1 / 1 point

- ○ Each neuron in the first hidden layer will perform the same computation in the first iteration. But after one iteration of gradient descent they will learn to compute different things because we have "broken symmetry".
- ○ Each neuron in the first hidden layer will compute the same thing, but neurons in different layers will compute different things, thus we have accomplished "symmetry breaking" as described in the lecture.
- ○ The first hidden layer's neurons will perform different computations from each other even in the first

iteration; their parameters will thus keep evolving in their own way.

- (●) Each neuron in the first hidden layer will perform the same computation. So even after multiple iterations of gradient descent, each neuron in the layer will be computing the same thing as other neurons.

Correct

7. Logistic regression's weights w should be initialized randomly rather than to all zeros, because if you initialize to all zeros, then logistic regression will fail to learn a useful decision boundary because it will fail to "break symmetry", True/False? 1 / 1 point

- ( ) True
- (●) False

Correct
Yes, Logistic Regression doesn't have a hidden layer. If you initialize the weights to zeros, the first example x fed into the logistic regression will output zero but the derivatives of the Logistic Regression depend on the input x (because there's no hidden layer) which is not zero. So at the second iteration, the weights' values follow x's distribution and are different from each other if x is not a constant vector.

8. You have built a network using the tanh activation for all the hidden units. You initialize the weights to relatively large values, using np.random.randn(..,..)*1000. What will happen? 1 / 1 point

- ( ) This will cause the inputs of the tanh to also be very large, thus causing gradients to also become large. You therefore have to set α to a very small value to prevent divergence; this will slow down learning.
- ( ) So long as you initialize the weights randomly gradient descent is not affected by whether the weights are large or small.

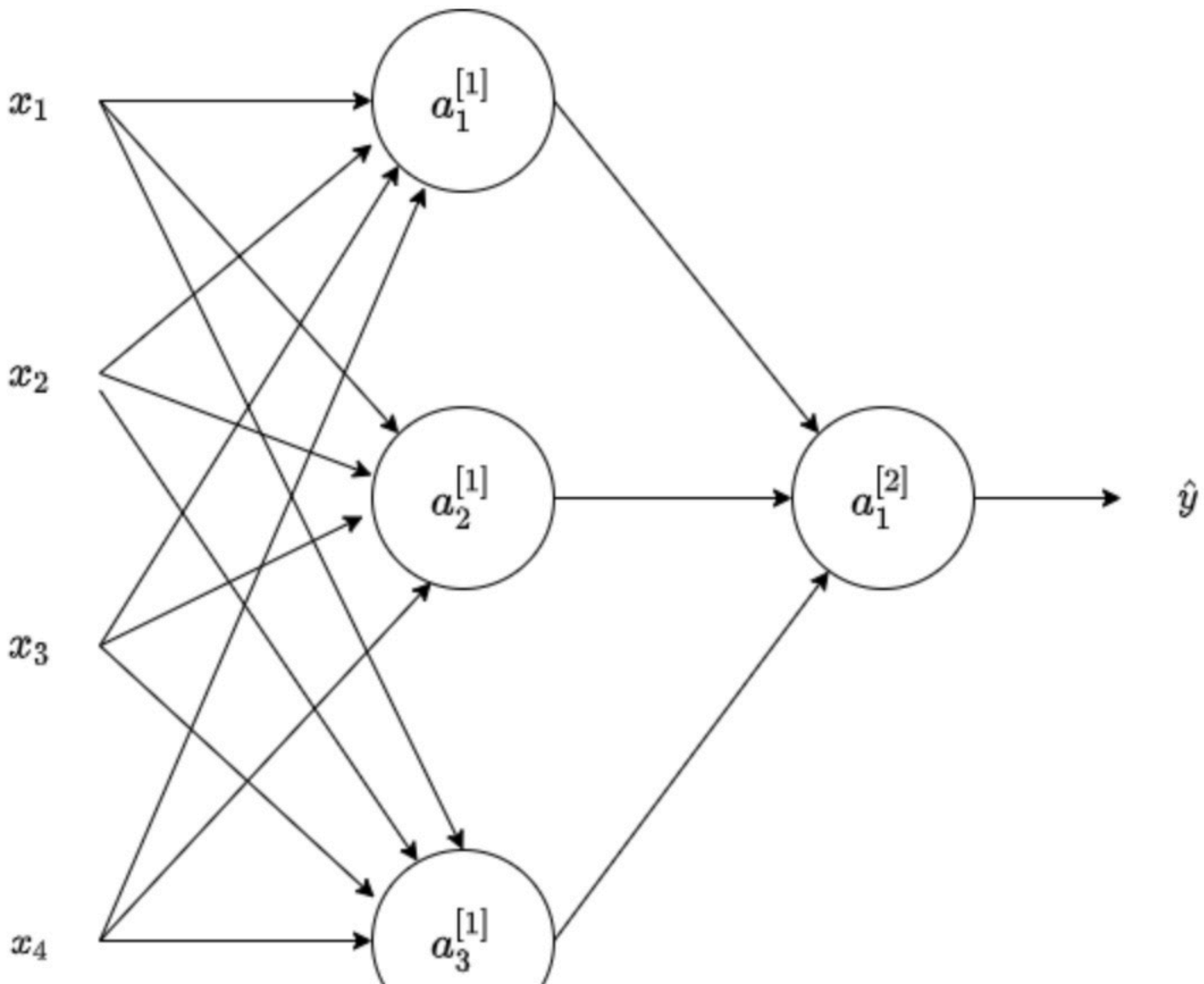
This will cause the inputs of the tanh to also be very large, causing the units to be “highly activated” and thus speed up learning compared to if the weights had to start from small values.

This will cause the inputs of the tanh to also be very large, thus causing gradients to be close to zero. The optimization algorithm will thus become slow.

Correct

Yes. tanh becomes flat for large values; this leads its gradient to be close to zero. This slows down the optimization algorithm.

9. Consider the following 1 hidden layer neural network: 1 / 1 point



Which of the following statements are True? (Check all that apply).

- [x] $W^{[1]}$ will have shape (3, 4).

Correct
Yes. The number of rows in $W^{[k]}$ is the number of neurons in the k-th layer and the number of columns is the number of inputs of the layer.

- [ ] $W^{[1]}$ will have shape (4, 3).
- [ ] $b^{[2]}$ will have shape (3, 1)
- [ ] $b^{[1]}$ will have shape (1, 3)
- [x] $b^{[1]}$ will have shape (3, 1).

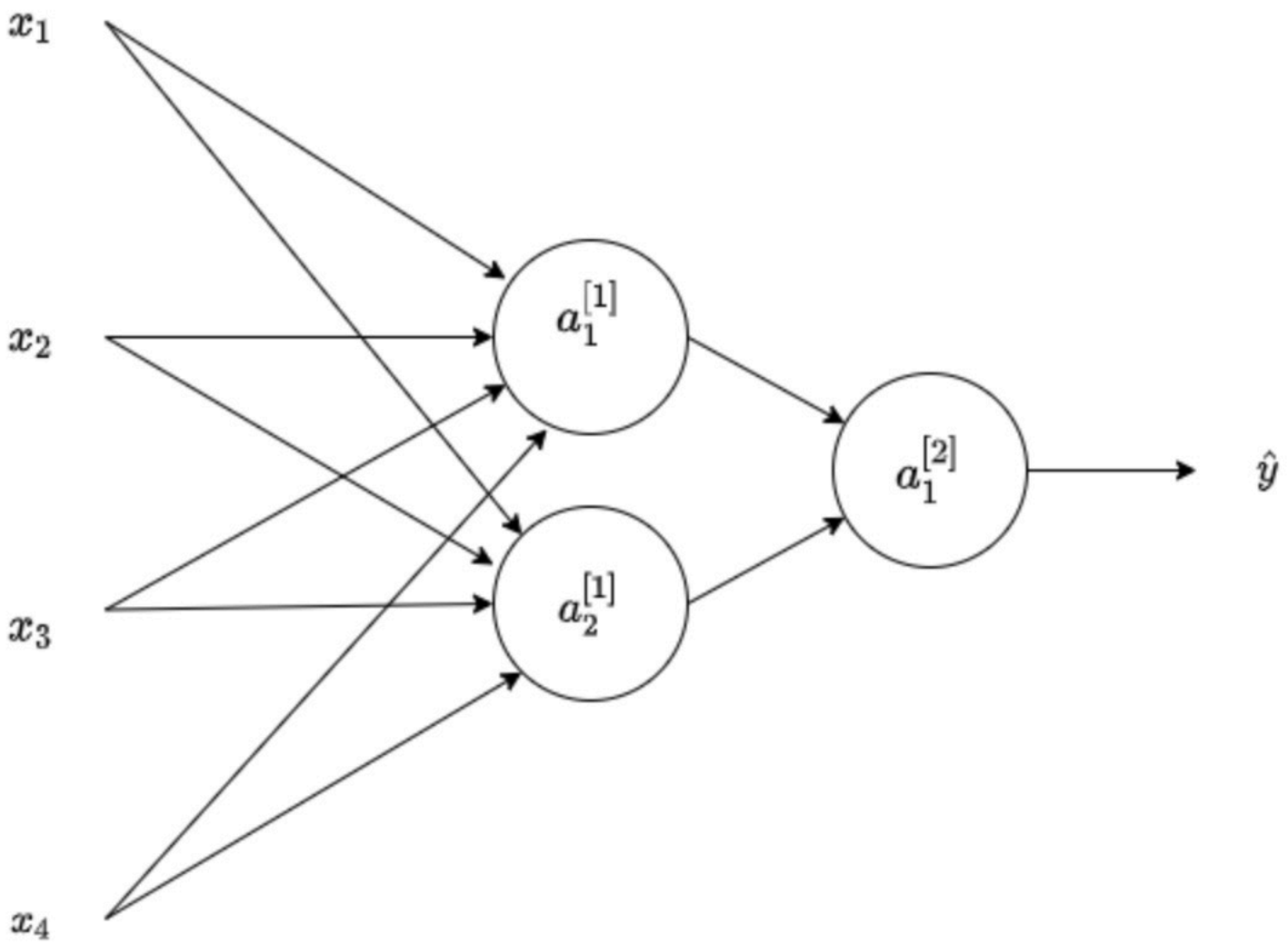
Correct
Yes. $b^{[k]}$ is a column vector and has the same number of rows as neurons in the k-th layer.

- [x] $b^{[2]}$ will have shape (1,1)

Correct
Yes. $b^{[k]}$ is a column vector and has the same number of rows as neurons in the k-th layer.

10. Consider the following 1 hidden layer neural network:

1 / 1 point



What are the dimensions of $Z^{[1]}$ and $A^{[1]}$?

- ○ $Z^{[1]}$ and $A^{[1]}$ are (4, 1)
- ○ $Z^{[1]}$ and $A^{[1]}$ are (2, 1)
- ◉ $Z^{[1]}$ and $A^{[1]}$ are (2, m)
- ○ $Z^{[1]}$ and $A^{[1]}$ are (4, m)

Correct

Yes. The $Z^{[1]}$ and $A^{[1]}$ are calculated over a batch of training examples. The number of columns in $Z^{[1]}$ and $A^{[1]}$ is equal to the number of examples in the batch, m. And the number of rows in $Z^{[1]}$ and $A^{[1]}$ is equal to the number of neurons in the first layer.