1. Which of the following is stored in the 'cache' during forward propagation for latter use in backward propagation? 1 / 1 point

- (●) $Z^{[l]}$
- ( ) $W^{[l]}$
- ( ) $b^{[l]}$

Correct
Yes. This value is useful in the calculation of $dW^{[l]}$ in the backward propagation.

2. During the backpropagation process, we use gradient descent to change the hyperparameters. True/False? 1 / 1 point

- ( ) True
- (●) False

Correct
Correct. During backpropagation, we use gradient descent to compute new values of $W^{[l]}$ and $b^{[l]}$. These are the parameters of the network.

3. Considering the intermediate results below, which layers of a deep neural network are they likely to belong to? 1 / 1 point

- ( ) Input layer of the deep neural network.
- (●) Later layers of the deep neural network.
- ( ) Middle layers of the deep neural network.

Early layers of the deep neural network.

Correct

Correct. The deep layers of a neural network are typically computing more complex features such as the ones shown in the figure.

4. Vectorization allows you to compute forward propagation in an L-layer neural network without an explicit for-loop (or any other explicit iterative loop) over the layers l=1, 2, …,L. True/False?

1 / 1 point

- (selected) False
- True

Correct

Forward propagation propagates the input through the layers, although for shallow networks we may just write all the lines ($a^{[2]} = g^{[2]}(z^{[2]}), z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$, ...) in a deeper network, we cannot avoid a for loop iterating over the layers: ($a^{[l]} = g^{[l]}(z^{[l]}), z^{[l]} = W^{[l]}a^{[l-1]} + b^{[l]}$, ...).

5. Assume we store the values for $n^{[l]}$ in an array called layer_dims, as follows: layer_dims = $[n_x$, 4,3,2,1]. So layer 1 has four hidden units, layer 2 has 3 hidden units, and so on. Which of the following for-loops will allow you to initialize the parameters for the model?

1 / 1 point

- ```
  for i in range(len(layer_dims)-1):

  parameter['W' + str(i+1)] = np.random.randn(layer_dims[i], layer_dims[i+1]) * 0.01

  parameter['b' + str(i+1)] = np.random.randn(layer_dims[i+1], 1) * 0.01
  ```
- (selected)
  ```
  for i in range(len(layer_dims)-1):

  parameter['W' + str(i+1)] = np.random.randn(layer_dims[i+1], layer_dims[i]) * 0.01

  parameter['b' + str(i+1)] = np.random.randn(layer_dims[i+1], 1) * 0.01
  ```
-
  ```

parameter['W' + str(i+1)] = np.random.randn(layer_dims[i+1], layer_dims[i]) * 0.01

parameter['b' + str(i+1)] = np.random.randn(layer_dims[i+1], 1) * 0.01

○

for i in range(1, len(layer_dims)/2):

parameter['W' + str(i)] = np.random.randn(layer_dims[i], layer_dims[i-1]) * 0.01

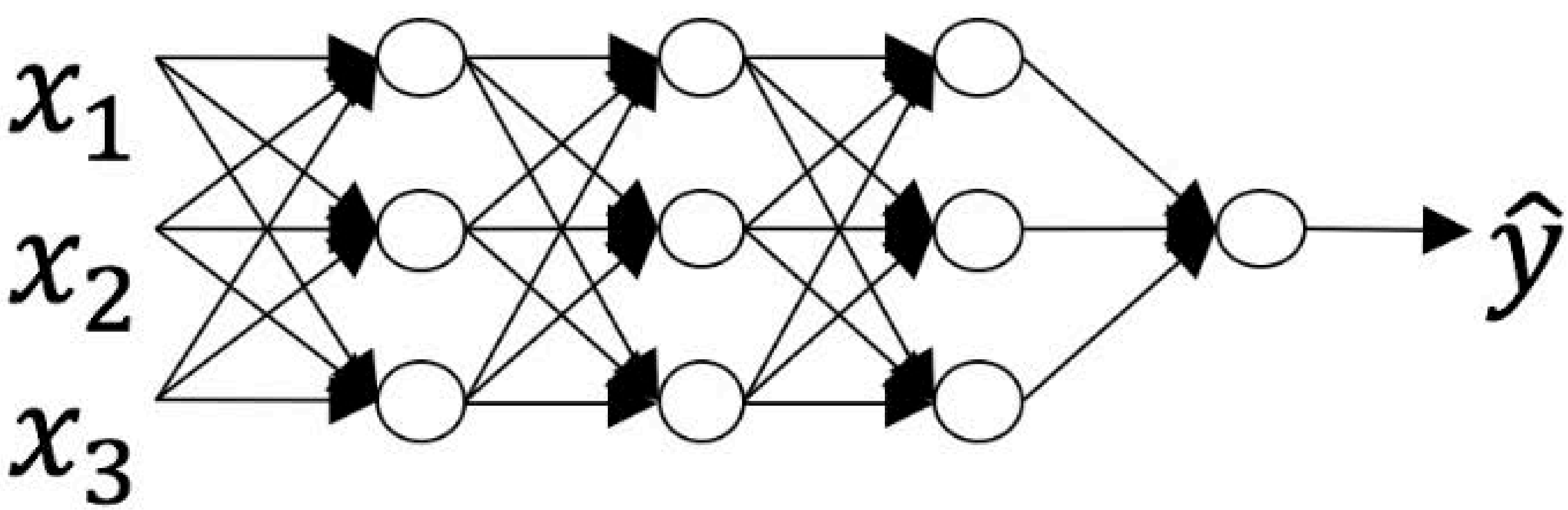
parameter['b' + str(i)] = np.random.randn(layer_dims[i], 1) * 0.01

Correct

Yes. This iterates over 0, 1, 2, 3 and assigns to $W^{[l]}$ the shape $(n^{[l]}, n^{[l-1]})$.

6. Consider the following neural network.

1 / 1 point



How many layers does this network have?

- ○ The number of layers L is 3. The number of hidden layers is 3.
- ◉ The number of layers L is 4. The number of hidden layers is 3.
- ○ The number of layers L is 4. The number of hidden layers is 4.
- ○ The number of layers L is 5. The number of hidden layers is 4.

Correct

Yes. As seen in lecture, the number of layers is counted as the number of hidden layers + 1. The input and output layers are not counted as hidden layers.

7. If L is the number of layers of a neural network then $dZ^{[L]} = A^{[L]} - Y$. True/False?

1 / 1 point

- [ ] False
- [x] True

Correct
Correct. The gradient of the output layer depends on the difference between the value computed during the forward propagation process and the target values.

8. For any mathematical function you can compute with an L-layered deep neural network with N hidden units there is a shallow neural network that requires only $\log N$ units, but it is very difficult to train.

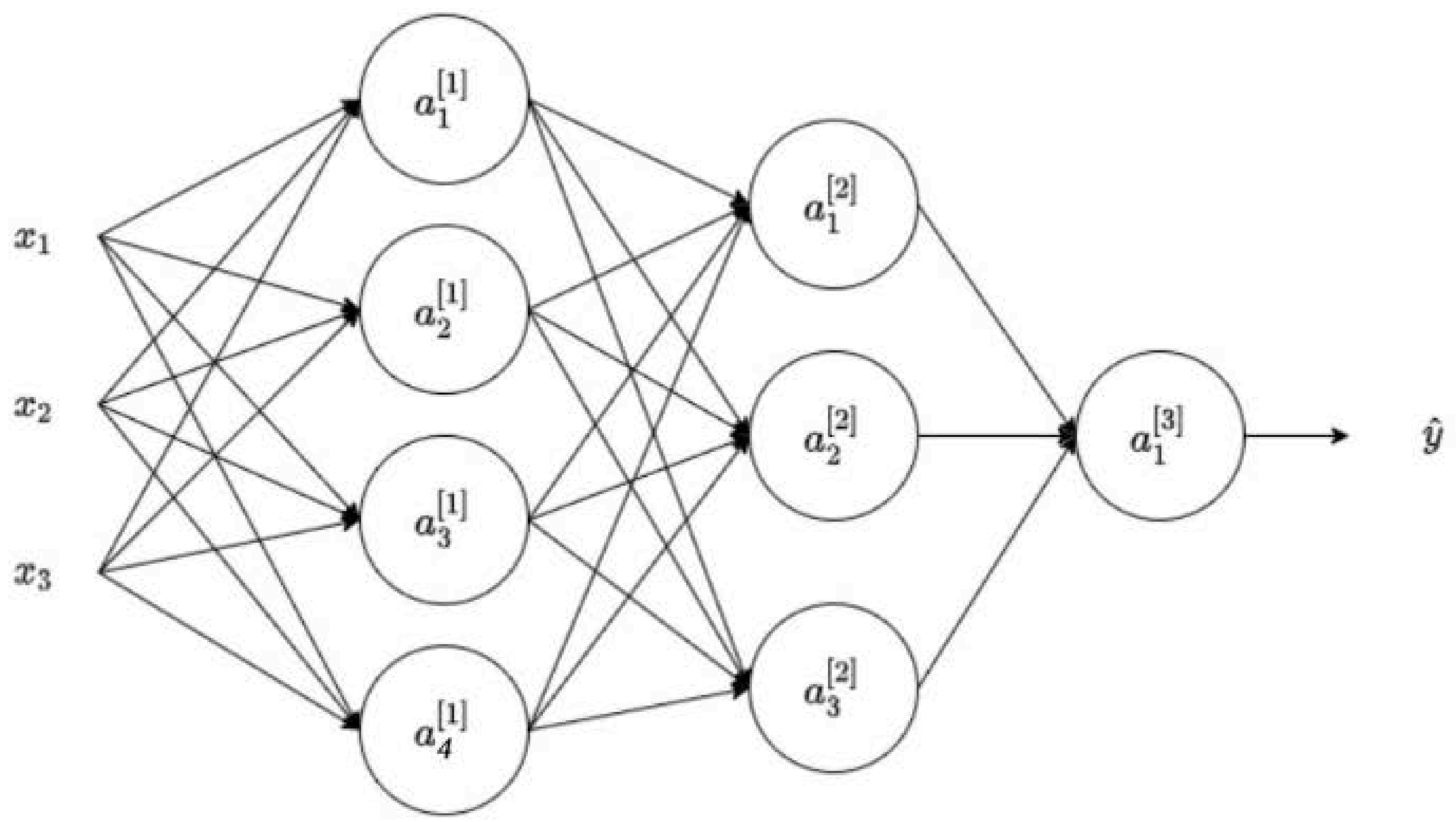
1 / 1 point

- [ ] True
- [x] False

Correct
Correct. On the contrary, some mathematical functions can be computed using an L-layered neural network and a given number of hidden units; but using a shallow neural network the number of necessary hidden units grows exponentially.

9. Consider the following 2 hidden layers neural network:

1 / 1 point



Which of the following statements is true? (Check all that apply).

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

> **Correct**
> Yes. More generally, the shape of $W^{[l]}$ is $(n^{[l]}, n^{[l-1]})$.

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

> **Correct**
> Yes. More generally, the shape of $b^{[l]}$ is $(n^{[l]}, 1)$.

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

> **Correct**
> Yes. More generally, the shape of $W^{[l]}$ is $(n^{[l]}, n^{[l-1]})$.

- [ ] $b^{[1]}$ will have shape (1, 4)

10. Whereas the previous question used a specific network, in the general case what is the dimension of $b^{[l]}$, the bias vector associated with layer l? **1 / 1 point**

- ○ $b^{[l]}$ has shape $(1, n^{[l]})$
- ◉ $b^{[l]}$ has shape $(n^{[l]}, 1)$
- ○ $b^{[l]}$ has shape $(n^{[l+1]}, 1)$
- ○ $b^{[l]}$ has shape $(1, n^{[l-1]})$

> **Correct**
> True. $b^{[l]}$ is a column vector with the same number of rows as units in the respective layer.