Your grade: 100%

Your latest: **100%** • Your highest: **100%** • To pass you need at least 80%. We keep your highest score.

Next item →

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

1 / 1 point

- ○ $b^{[l]}$
- ◉ $Z^{[l]}$
- ○ $W^{[l]}$

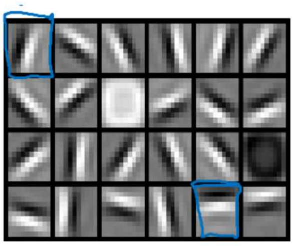
Correct
Yes. This value is useful in the calculation of $dW^{[l]}$ in the backward propagation.

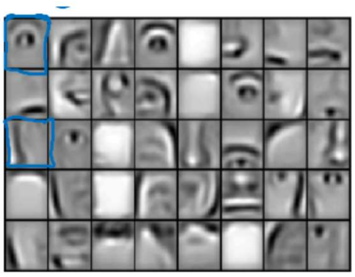
2. During the backpropagation process, we use gradient descent to change the hyperparameters. True/False?

- ○ 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. Which of the following is more likely related to the early layers of a deep neural network?

◉ 

○ 

○ 

Correct
Yes. The early layer of a neural network usually computes simple features such as edges and lines.

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?

- ○ True
- ◉ False

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?

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

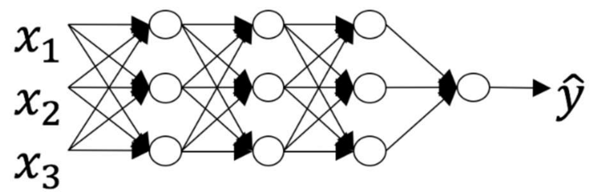
◉
```
for i in range(1, len(layer_dims)):
parameter['W' + str(i)] = np.random.randn(layer_dims[i], layer_dims[i-1]) * 0.01
parameter['b' + str(i)] = np.random.randn(layer_dims[i], 1) * 0.01
```

○
```
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
```

○
```
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], 1) * 0.01
```

Correct

6. Consider the following neural network.



How many layers does this network have?

- [x] 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.
- [ ] The number of layers L is 3. The number of hidden layers is 3.

> **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?

- [ ] False
- [x] True

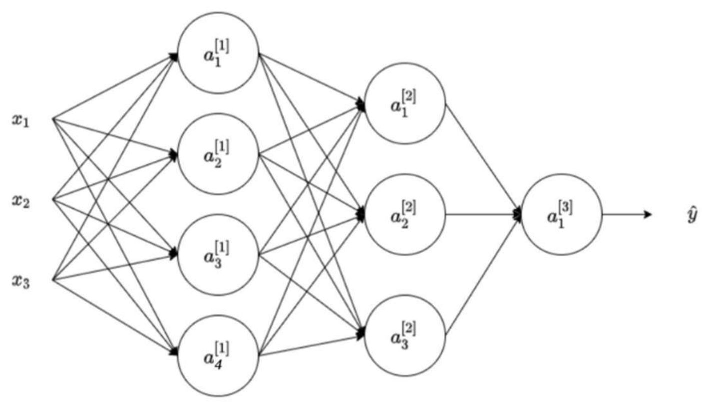
> **Correct**
> Correct. The gradient of the output layer depends on the difference between the value computed during the forward propagation process and the target values.

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.

- [x] False
- [ ] True

> **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:



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

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

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

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

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

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

10. Whereas the previous question used a specific network, in the general case what is the dimension of W^{[l]}, the weight matrix associated with layer l?

- [x] $W^{[l]}$ has shape $(n^{[l]}, n^{[l-1]})$
- [ ] $W^{[l]}$ has shape $(n^{[l]}, n^{[l+1]})$
- [ ] $W^{[l]}$ has shape $(n^{[l+1]}, n^{[l]})$
- [ ] $W^{[l]}$ has shape $(n^{[l-1]}, n^{[l]})$

> **Correct**
> True