## Congratulations! You passed!

Next Item

1 / 1 point

1. What is the "cache" used for in our implementation of forward propagation and backward propagation?

- It is used to keep track of the hyperparameters that we are searching over, to speed up computation.
- It is used to cache the intermediate values of the cost function during training.
- **We use it to pass variables computed during forward propagation to the corresponding backward propagation step. It contains useful values for backward propagation to compute derivatives.**

**Correct**
Correct, the "cache" records values from the forward propagation units and sends it to the backward propagation units because it is needed to compute the chain rule derivatives.

- We use it to pass variables computed during backward propagation to the corresponding forward propagation step. It contains useful values for forward propagation to compute activations.

1 / 1 point

2. Among the following, which ones are "hyperparameters"? (Check all that apply.)

- [x] size of the hidden layers $n^{[l]}$

**Correct**

- [x] number of layers $L$ in the neural network

**Correct**

- [ ] bias vectors $b^{[l]}$

**Un-selected is correct**

- [x] learning rate $\alpha$

**Correct**

- [ ] activation values $a^{[l]}$

**Un-selected is correct**

- [ ] weight matrices $W^{[l]}$

**Un-selected is correct**

- [x] number of iterations

**Correct**

1 / 1 point

3. Which of the following statements is true?

- **The deeper layers of a neural network are typically computing more complex features of the input than the earlier layers.**

**Correct**

- ( ) The earlier layers of a neural network are typically computing more complex features of the input than the deeper layers.

---

✔ 1 / 1 point

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: ($a^{[l]} = g^{[l]}(z^{[l]})$, $z^{[l]} = W^{[l]}a^{[l-1]} + b^{[l]}$, ...).

---

✔ 1 / 1 point

5. Assume we store the values for $n^{[l]}$ in an array called layers, 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)/2)):
    parameter['W' + str(i)] = np.random.randn(layers[i], layers[i
        -1])) * 0.01
    parameter['b' + str(i)] = np.random.randn(layers[i], 1) * 0.01
```

- ( )
```
for(i in range(1, len(layer_dims)/2)):
    parameter['W' + str(i)] = np.random.randn(layers[i], layers[i
        -1])) * 0.01
    parameter['b' + str(i)] = np.random.randn(layers[i-1], 1) * 0
        .01
```

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

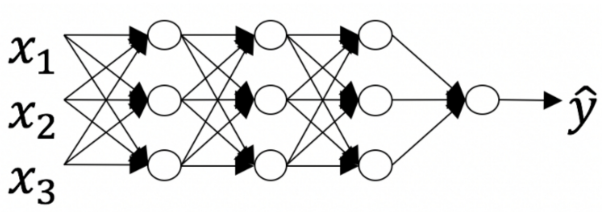
- (•)
```
for(i in range(1, len(layer_dims))):
    parameter['W' + str(i)] = np.random.randn(layers[i], layers[i
        -1])) * 0.01
    parameter['b' + str(i)] = np.random.randn(layers[i], 1) * 0.01
```

**Correct**

---

✔ 1 / 1 point

6. Consider the following neural network.




How many layers does this network have?

- (•) The number of layers $L$ is 4. 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.

- ( ) 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 4.
- The number of layers $L$ is 5. The number of hidden layers is 4.

---

✔ 1 / 1 point

7. During forward propagation, in the forward function for a layer $l$ you need to know what is the activation function in a layer (Sigmoid, tanh, ReLU, etc.). During backpropagation, the corresponding backward function also needs to know what is the activation function for layer $l$, since the gradient depends on it. True/False?

- (●) True

**Correct**
Yes, as you've seen in the week 3 each activation has a different derivative. Thus, during backpropagation you need to know which activation was used in the forward propagation to be able to compute the correct derivative.

- ( ) False

---

✔ 1 / 1 point

8. There are certain functions with the following properties:

(i) To compute the function using a shallow network circuit, you will need a large network (where we measure size by the number of logic gates in the network), but (ii) To compute it using a deep network circuit, you need only an exponentially smaller network. True/False?

- (●) True

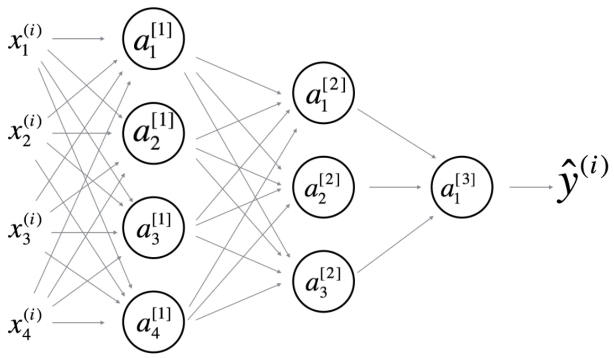
**Correct**

- ( ) False

---

✔ 1 / 1 point

9. Consider the following 2 hidden layer neural network:




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

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

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

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

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

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

**Un-selected is correct**

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

**Un-selected is correct**

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

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

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

**Un-selected is correct**

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

**Un-selected is correct**

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

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

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

**Un-selected is correct**

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

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

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

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

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

**Un-selected is correct**

---

✔ 1 / 1 point

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*?

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

**Correct**
True

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

---