Congratulations! You passed!

Grade received 90% | Latest Submission Grade 90% | To pass 80% or higher

Go to next item

1. We use the "cache" in our implementation of forward and backward propagation to pass useful values to the next layer in the forward propagation. True/False?

0 / 1 point

- ( ) False
- (•) True

Expand

Incorrect
Incorrect. The "cache" is used in our implementation to store values computed during forward propagation to be used in backward propagation.

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

1 / 1 point

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

  ✓ Correct

- [x] learning rate

  α

  α

  ✓ Correct

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

  ✓ Correct

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

  ✓ Correct

Expand

Correct
Great, you got all the right answers.

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

1 / 1 point

- ( ) Middle layers of the deep neural network

- Middle layers of the deep neural network.
- Later layers of the deep neural network.
- Early layers of the deep neural network.
- Input layer of the deep neural network.

Expand

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

- True
- False

Expand

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

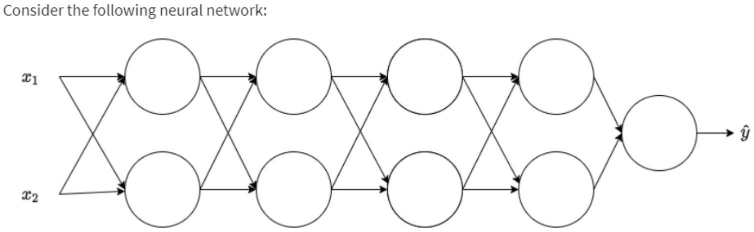
- for i in range(len(layer_dims)-1):
  parameter['W' + str(i+1)] = np.random.randn(layer_dims[i], layer_dims[i+1]) * 0.01
  parameter['b' + str(i+1)] = np.random.randn(layer_dims[i+1], 1) * 0.01
- for i in range(1, len(layer_dims)/2):
  parameter['W' + str(i)] = np.random.randn(layer_dims[i], layer_dims[i-1])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

Expand

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 6
- The number of layers L is 2

- The number of layers L is 2.
- The number of layers L is 4.
- The number of layers L is 5.

Expand

Correct
Yes. The number of layers is the number of hidden layers + 1.

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?

1 / 1 point

- True
- False

Expand

Correct
Yes, as you've seen in 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.

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

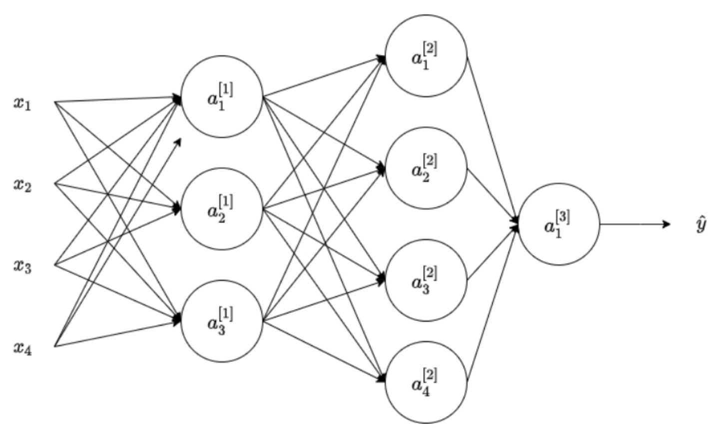
- True
- False

Expand

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 are true? (Check all that apply).

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

$W^{[2]}$

will have shape (3, 4)

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

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

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

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

Expand

Correct
Great, you got all the right answers.

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?

1 / 1 point

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

Expand

Correct
True