Congratulations! You passed!

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

Go to next item

1. What is the "cache" used for in our implementation of forward propagation and backward propagation? 0 / 1 point

- ○ It is used to cache the intermediate values of the cost function during training.
- ○ 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.
- ○ 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.
- ○ It is used to keep track of the hyperparameters that we are searching over, to speed up computation.

Incorrect
You didn't select an answer.

2. Among the following, which ones are "hyperparameters"? (Check all that apply.) 1 / 1 point

- ☑ number of iterations

Correct

- ☐ weight matrices $W^{[l]}$
- ☐ activation values $a^{[l]}$
- ☑ learning rate α

Correct

- ☐ bias vectors $b^{[l]}$
- ☑ number of layers L in the neural network

Correct

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

Correct

3. Which of the following statements is true? 1 / 1 point

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

Correct

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

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(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
```

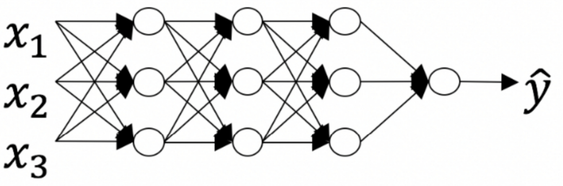
○
```
for i in range(1, len(layer_dims)/2):
    parameter['W' + str(i)] = np.random.randn(layer_dims[i], layer_dims[i-1]) * 0.01
    parameter['b' + str(i)] = np.random.randn(layer_dims[i-1], 1) * 0.01
```

○
```
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
```

Correct

6. Consider the following neural network. 1 / 1 point



How many layers does this network have?

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

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.