|  |  |
| --- | --- |
| What is the "cache" used for in our implementation of forward propagation and backward propagation? | 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. |
| Among the following, which ones are "hyperparameters"? (Check all that apply.) | bias vectors b^{[l]}b  [l]    Un-selected is correct  **size of the hidden layers n^{[l]}n**  **[l]**    **Correct**  **number of iterations**  **Correct**  activation values a^{[l]}a  [l]    Un-selected is correct  weight matrices W^{[l]}W  [l]    Un-selected is correct  **learning rate \alphaα**  **Correct**  **number of layers LL in the neural network**  **Correct** |
| 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. |
| Vectorization allows you to compute forward propagation in an LL-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]})*a*[2]=*g*[2](*z*[2]), z^{[2]}= W^{[2]}a^{[1]}+b^{[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]})*a*[*l*]=*g*[*l*](*z*[*l*]), z^{[l]} = W^{[l]}a^{[l-1]} + b^{[l]}*z*[*l*]=*W*[*l*]*a*[*l*−1]+*b*[*l*], ...). |
| Assume we store the values for n^{[l]}*n*[*l*] in an array called layers, as follows: layer\_dims = [n\_x*nx*​, 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(layers[i], layers[i-1])) \* 0.01  parameter[‘b’ + str(i)] = np.random.randn(layers[i], 1) \* 0.01 |
| Consider the following neural network.  https://d3c33hcgiwev3.cloudfront.net/imageAssetProxy.v1/cwmw1nrfEeeJIwrF5BVsIg_e9a22da9e380c0350d2dfd47dcf34503_Screen-Shot-2017-08-06-at-12.42.46-PM.png?expiry=1557273600000&hmac=DW7AbKyylQZOsOaDFklSiujs0u4Cu3aEshlAvejv7wU | The number of layers L*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. |
| During forward propagation, in the forward function for a layer l*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*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. |
| 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 |
| Consider the following 2 hidden layer neural network:  https://d3c33hcgiwev3.cloudfront.net/imageAssetProxy.v1/8sF12nrfEeeumw4MySoK5g_36df26a0659f76c6566ff4f3706e6ad2_Screen-Shot-2017-08-05-at-12.50.32-PM.png?expiry=1557273600000&hmac=fGL6MSa-H6Hn4wWy_lOMNEDT8CAJpyU45MLexL8F2T0  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]}W  [l]  is (n^{[l]}, n^{[l-1]})(n  [l]  ,n  [l−1]  ).  b^{[1]}b  [1]  will have shape (4, 1)  Correct  Yes. More generally, the shape of b^{[l]}b  [l]  is (n^{[l]}, 1)(n  [l]  ,1).  W^{[1]}W  [1]  will have shape (3, 4)  Un-selected is correct  b^{[1]}b  [1]  will have shape (3, 1)  Un-selected is correct  W^{[2]}W  [2]  will have shape (3, 4)  Correct  Yes. More generally, the shape of W^{[l]}W  [l]  is (n^{[l]}, n^{[l-1]})(n  [l]  ,n  [l−1]  ).  b^{[2]}b  [2]  will have shape (1, 1)  Un-selected is correct  W^{[2]}W  [2]  will have shape (3, 1)  Un-selected is correct  b^{[2]}b  [2]  will have shape (3, 1)  Correct  Yes. More generally, the shape of b^{[l]}b  [l]  is (n^{[l]}, 1)(n  [l]  ,1).  W^{[3]}W  [3]  will have shape (3, 1)  Un-selected is correct  b^{[3]}b  [3]  will have shape (1, 1)  Correct  Yes. More generally, the shape of b^{[l]}b  [l]  is (n^{[l]}, 1)(n  [l]  ,1).  W^{[3]}W  [3]  will have shape (1, 3)  Correct  Yes. More generally, the shape of W^{[l]}W  [l]  is (n^{[l]}, n^{[l-1]})(n  [l]  ,n  [l−1]  ).  b^{[3]}b  [3]  will have shape (3, 1)  Un-selected is correct |
| 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*l*? | *W*[*l*] has shape (n^{[l]}, n^{[l-1]})(*n*[*l*],*n*[*l*−1]) |