축하합니다! 통과하셨습니다!

받은 학점 100% 최신 제출물 학점 100% 통과 점수: 80% 이상

다음 항목으로 이동

1. What is stored in the 'cache' during forward propagation for latter use in backward propagation? 1 / 1점

- $W^{[l]}$
- $A^{[l]}$
- $b^{[l]}$
- $Z^{[l]}$

더 보기

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

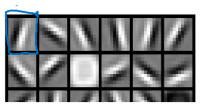
2. During the backpropagation process, we use gradient descent to change the hyperparameters. True/False? 1 / 1점

- True
- False

더 보기

맞습니다
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? 1 / 1점



더 보기

맞습니다
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? 1 / 1점

- True
- False

더 보기

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

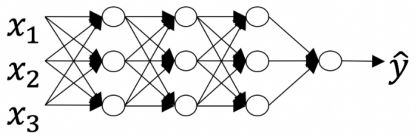
5. Assume we store the values for $n^{[l]}$ in an array called layer_dims, as follows: layer_dims = [n_x, 4,3,2,1]. So layer 1 has four hidden units, layer 2 has 3 hidden units and so on. Which of the following for-loops will allow you to initialize the parameters for the model? 1 / 1점

- 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)):
  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)/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

더 보기

맞습니다

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



How many layers does this network have?

- 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 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.

더 보기

맞습니다
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. 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 / 1점

- False
- True

더 보기

맞습니다
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.

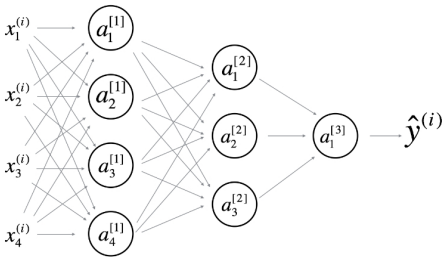
8. For any mathematical function you can compute with an L-layered deep neural network with N hidden units there is a shallow neural network that requires only $\log N$ units, but it is very difficult to train. 1 / 1점

- False
- True

더 보기

맞습니다
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 layer neural network: 1 / 1점



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

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

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

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

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

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

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

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

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

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

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

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

더 보기

맞습니다
Great, you got all the right answers.

10. In the general case if we are training with m examples what is the shape of $A^{[l]}$? 1 / 1점

- $(n^{[l]}, m)$
- $(m, n^{[l+1]})$
- $(n^{[l+1]}, m)$
- $(m, n^{[l]})$

더 보기

맞습니다
Yes. The number of rows in $A^{[l]}$ corresponds to the number of units in the l-th layer.