Congratulations! You passed!

Grade received 80%

Latest Submission Grade 80%

To pass 80% or higher

Go to next item

1. Which of the following are true? (Check all that apply.)

1 / 1 point

- [ ] $a^{[2](12)}$ denotes activation vector of the 12^{th} layer on the 2^{nd} training example.
- [ ] $a_4^{[2]}$ is the activation output of the 2^{nd} layer for the 4^{th} training example
- [x] X is a matrix in which each column is one training example.

Correct

- [x] $a^{[2](12)}$ denotes the activation vector of the 2^{nd} layer for the 12^{th} training example.

Correct

- [x] $a^{[2]}$ denotes the activation vector of the 2^{nd} layer.

Correct

- [x] $a_4^{[2]}$ is the activation output by the 4^{th} neuron of the 2^{nd} layer

Correct

- [ ] X is a matrix in which each row is one training example.

Expand

Correct
Great, you got all the right answers.

2. The tanh activation is not always better than sigmoid activation function for hidden units because the mean of its output is closer to zero, and so it centers the data, making learning complex for the next layer. True/False?

1 / 1 point

- (●) False
- ( ) True

Expand

Correct
Yes. As seen in lecture the output of the tanh is between -1 and 1, it thus centers the data which makes the learning simpler for the next layer.

3. Which of these is a correct vectorized implementation of forward propagation for layer l, where $1 \le l \le L$?

1 / 1 point

- (●) $Z^{[l]} = W^{[l]}A^{[l-1]} + b^{[l]}$
  $A^{[l]} = g^{[l]}(Z^{[l]})$
- ( ) $Z^{[l]} = W^{[l]}A^{[l]} + b^{[l]}$
  $A^{[l+1]} = g^{[l+1]}(Z^{[l]})$
- ( ) $Z^{[l]} = W^{[l]}A^{[l]} + b^{[l]}$

$A^{[l+1]} = g^{[l]}(Z^{[l]})$

- $Z^{[l]} = W^{[l-1]}A^{[l]} + b^{[l-1]}$
  $A^{[l]} = g^{[l]}(Z^{[l]})$

Expand

Correct

4. The use of the ReLU activation function is becoming more rare because the ReLU function has no derivative for $c = 0$. True/False?

1 / 1 point

- True
- False

Expand

Correct

Yes. Although the ReLU function has no derivative at $c = 0$ this rarely causes any problems in practice. Moreover it has become the default activation function in many cases, as explained in the lectures.

5. Consider the following code:

1 / 1 point

#+begin_src python

x = np.random.rand(3, 2)

y = np.sum(x, axis=0, keepdims=True)

#+end_src

What will be y.shape?

- (2,)
- (3,)
- (1, 2)
- (3, 1)

Expand

Correct

Yes. By choosing the axis=0 the sum is computed over each column of the array, thus the resulting array is a row vector with 2 entries. Since the option keepdims=True is used the first dimension is kept, thus (1, 2).

6. Suppose you have built a neural network with one hidden layer and tanh as activation function for the hidden layers. Which of the following is a best option to initialize the weights?

1 / 1 point

- Initialize all weights to a single number chosen randomly.
- Initialize the weights to large random numbers.
- Initialize the weights to small random numbers.
- Initialize all weights to 0.

Expand

Correct

The use of random numbers helps to "break the symmetry" between all the neurons allowing them to compute different functions. When using small random numbers the values $z^{[k]}$ will be close to zero thus the activation values will have a larger gradient speeding up the training process.

7. Logistic regression's weights should be initialized randomly rather than to all zeros, because if you initialize to all zeros, then logistic regression will fail to learn a useful decision boundary because it will fail to "break symmetry"

1 / 1 point

zeros, then logistic regression will fail to learn a useful decision boundary because it will fail to "break symmetry", True/False?

- ( ) True
- (•) False

Expand

Correct
Yes, Logistic Regression doesn't have a hidden layer. If you initialize the weights to zeros, the first example x fed into the logistic regression will output zero but the derivatives of the Logistic Regression depend on the input x (because there's no hidden layer) which is not zero. So at the second iteration, the weights' values follow x's distribution and are different from each other if x is not a constant vector.

8. You have built a network using the tanh activation for all the hidden units. You initialize the weights to relatively large values, using np.random.randn(..,..)*1000. What will happen?

1 / 1 point

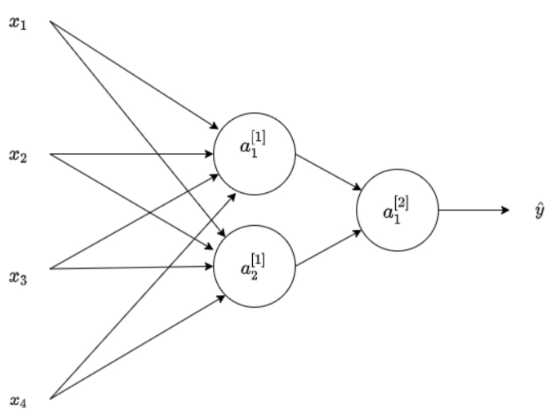
- ( ) This will cause the inputs of the tanh to also be very large, thus causing gradients to also become large. You therefore have to set α to a very small value to prevent divergence; this will slow down learning.
- ( ) This will cause the inputs of the tanh to also be very large, causing the units to be "highly activated" and thus speed up learning compared to if the weights had to start from small values.
- (•) This will cause the inputs of the tanh to also be very large, thus causing gradients to be close to zero. The optimization algorithm will thus become slow.
- ( ) So long as you initialize the weights randomly gradient descent is not affected by whether the weights are large or small.

Expand

Correct
Yes. tanh becomes flat for large values; this leads its gradient to be close to zero. This slows down the optimization algorithm.

9. Consider the following 1 hidden layer neural network:

0 / 1 point



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

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

Correct
Yes. $b^{[k]}$ is a column vector and has the same number of rows as neurons in the k-th layer.

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

! **This should not be selected**

No. The number of rows in

$$W^{[k]}$$

is the number of neurons in the k-th layer and the number of columns is the number of inputs of the layer.

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

! **This should not be selected**

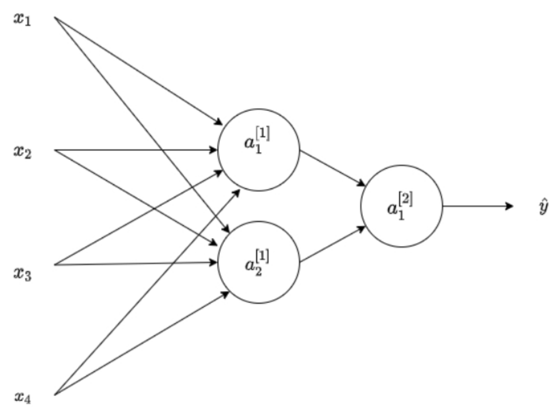
Expand

Incorrect

You didn't select all the correct answers

10. Consider the following 1 hidden layer neural network:

0 / 1 point



What are the dimensions of $Z^{[1]}$ and $A^{[1]}$?

- ( ) $Z^{[1]}$ and $A^{[1]}$ are (2, 1)
- (•) $Z^{[1]}$ and $A^{[1]}$ are (4, m)
- ( ) $Z^{[1]}$ and $A^{[1]}$ are (2, m)
- ( ) $Z^{[1]}$ and $A^{[1]}$ are (4, 1)

Expand

Incorrect

No. The $Z^{[1]}$ and $A^{[1]}$ are calculated over a batch of training examples. The number of columns in $Z^{[1]}$ and $A^{[1]}$ is equal to the number of examples in the batch, m. And the number of rows in $Z^{[1]}$ and $A^{[1]}$ is equal to the number of neurons in the first layer.