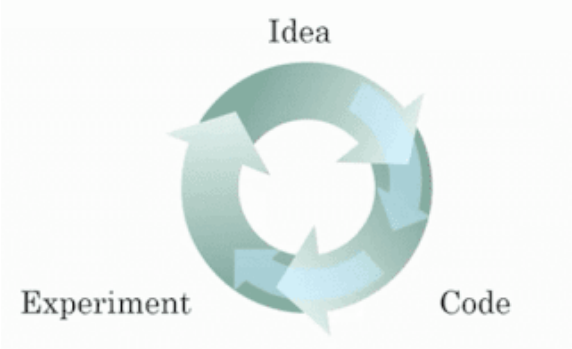
Neural Networks and Deep Learning (Week 1)

1. What does the analogy “AI is the new electricity” refer to?
    - AI runs on computers and is thus powered by electricity, but it is letting computers do things not possible before.
    - **Similar to electricity starting about 100 years ago, AI is transforming multiple industries.**
      Correct.
      Yes. AI is transforming many fields from the car industry to agriculture to supply-chain...
    - Through the “smart grid”, AI is delivering a new wave of electricity.
    - AI is powering personal devices in our homes and offices, similar to electricity
2. Which of these are reasons for Deep Learning recently taking off? (Check the three options that apply.)
    - **Deep learning has resulted in significant improvements in important applications such as online advertising, speech recognition, and image recognition.**
      Correct
    - Neural Networks are a brand new field.
    - **We have access to a lot more computational power.**
      Correct
      Yes! The development of hardware, perhaps especially GPU computing, has significantly improved deep learning algorithms' performance.
    - **We have access to a lot more data.**
      Correct
      Yes! The digitalization of our society has played a huge role in this.
3. Recall this diagram of iterating over different ML ideas. Which of the statements below are true? (Check all that apply.)



    - **Being able to try out ideas quickly allows deep learning engineers to iterate more quickly.**
      Correct
    - **Faster computation can help speed up how long a team takes to iterate to a good idea.**
      Correct
    - It is faster to train on a big dataset than a small dataset.

- **Recent progress in deep learning algorithms has allowed us to train good models faster (even without changing the CPU/GPU hardware).**
  Correct
  Yes. For example, we discussed how switching from sigmoid to ReLU activation functions allows faster training.

4. When an experienced deep learning engineer works on a new problem, they can usually use insight from previous problems to train a good model on the first try, without needing to iterate multiple times through different models. True/False?
   - True
   - **False**
     Correct
     Yes. Finding the characteristics of a model is key to have good performance. Although experience can help, it requires multiple iterations to build a good model.
5. Which one of these plots represents a ReLU activation function?
   - Figure 1:

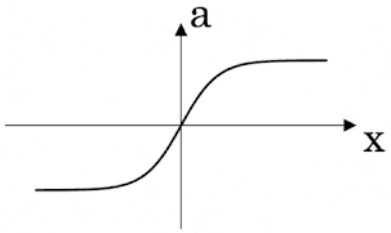
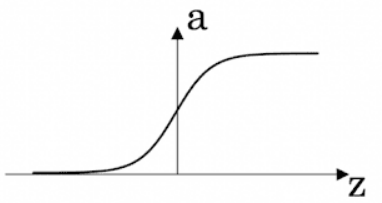
     

     Figure 2:

   - **Figure 3:**

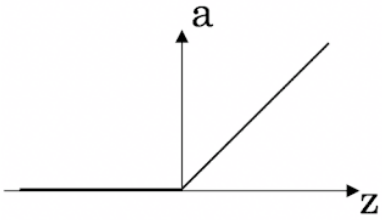
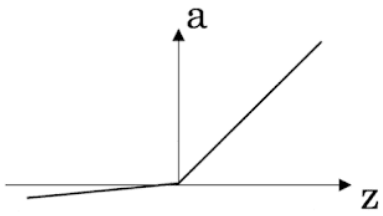
     

     Figure 4:

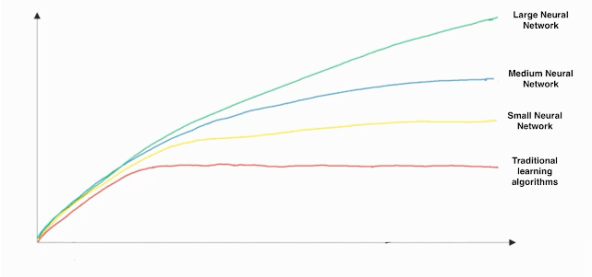
     

     Correct

6. Images for cat recognition is an example of “structured” data because it is represented as a structured array in a computer. True/False?
   - True
   - **False**
     Correct
7. A demographic dataset with statistics on different cities' population, GDP per capita, economic growth is an example of “unstructured” data because it contains data coming from different sources. True/False?
   - True
   - **False**
     Correct
     A demographic dataset with statistics on different cities' population, GDP per capita, economic growth is an example of “structured” data by opposition to image, audio or text datasets.
8. Why is an RNN (Recurrent Neural Network) used for machine translation, say translating English to French? (Check all that apply.)
   - **It can be trained as a supervised learning problem.**
     Correct
     Yes. We can train it on many pairs of sentences x (English) and y (French).

- It is strictly more powerful than a Convolutional Neural Network (CNN).
- **It is applicable when the input/output is a sequence (e.g., a sequence of words).**
  Correct
  Yes. An RNN can map from a sequence of english words to a sequence of french words.
- RNNs represent the recurrent process of Idea->Code->Experiment->Idea->....

1. In this diagram which we hand-drew in lecture, what do the horizontal axis (x-axis) and vertical axis (y-axis) represent?



- **x-axis is the amount of data**
  y-axis (vertical axis) is the performance of the algorithm.
  Correct
- x-axis is the performance of the algorithm
  y-axis (vertical axis) is the amount of data.
- x-axis is the amount of data
  y-axis is the size of the model you train.
- x-axis is the input to the algorithm
  y-axis is outputs.

2. Assuming the trends described in the previous question's figure are accurate (and hoping you got the axis labels right), which of the following are true? (Check all that apply.)
   - Decreasing the training set size generally does not hurt an algorithm's performance, and it may help significantly.
   - **Increasing the training set size generally does not hurt an algorithm's performance, and it may help significantly.**
     Correct
     Yes. Bringing more data to a model is almost always beneficial.
   - **Increasing the size of a neural network generally does not hurt an algorithm's performance, and it may help significantly.**
     Correct
     Yes. According to the trends in the figure above, big networks usually perform better than small network
   - Decreasing the size of a neural network generally does not hurt an algorithm's performance, and it may help significantly.

Neural Networks and Deep Learning (Week 2) Quiz

1. What does a neuron compute?
   - A neuron computes an activation function followed by a linear function (z = Wx + b)
   - A neuron computes the mean of all features before applying the output to an activation function

- A neuron computes a function g that scales the input x linearly (Wx + b)
- **A neuron computes a linear function (z = Wx + b) followed by an activation function**

  Correct

  Correct, we generally say that the output of a neuron is a = g(Wx + b) where g is the activation function (sigmoid, tanh, ReLU, ...).

2. Which of these is the "Logistic Loss"?
   - $\mathcal{L}^{(i)}(\hat{y}^{(i)}, y^{(i)}) = \mid y^{(i)} - \hat{y}^{(i)} \mid$
   - $\mathcal{L}^{(i)}(\hat{y}^{(i)}, y^{(i)}) = -(y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}))$

     Correct

     Correct, this is the logistic loss you've seen in lecture!
   - $\mathcal{L}^{(i)}(\hat{y}^{(i)}, y^{(i)}) = \mid y^{(i)} - \hat{y}^{(i)} \mid^2$
   - $\mathcal{L}^{(i)}(\hat{y}^{(i)}, y^{(i)}) = max(0, y^{(i)} - \hat{y}^{(i)})$
3. Suppose img is a (32,32,3) array, representing a 32x32 image with 3 color channels red, green and blue. How do you reshape this into a column vector?
   - x = img.reshape((1,32*32,*3))
   - **x = img.reshape((32*32*3,1))**

     Correct
   - x = img.reshape((3,32*32))
   - x = img.reshape((32*32,3))
4. Consider the two following random arrays "a" and "b":

```
a = np.random.randn(2, 3) # a.shape = (2, 3)
b = np.random.randn(2, 1) # b.shape = (2, 1)
```

```
c = a + b
```

What will be the shape of "c"?

- The computation cannot happen because the sizes don't match. It's going to be "Error"!
- c.shape = (2, 1)
- c.shape = (3, 2)
- **c.shape = (2, 3)**

  Correct

  Yes! This is broadcasting. b (column vector) is copied 3 times so that it can be summed to each column of a.

7. Consider the two following random arrays "a" and "b":

```
a = np.random.randn(4, 3) # a.shape = (4, 3)
b = np.random.randn(3, 2) # b.shape = (3, 2)
```

```
c = a*b
```

What will be the shape of "c"?

- c.shape = (4, 3)
- c.shape = (4,2)
- c.shape = (3, 3)
- **The computation cannot happen because the sizes don't match. It's going to be "Error"!**

  Correct

  Indeed! In numpy the "*" operator indicates element-wise multiplication. It is different from "np.dot()". If you would try "c = np.dot(a,b)" you would get c.shape = (4, 2).

10. Suppose you have n_x input features per example. Recall that $X = [x^{(1)}x^{(2)}...\ x^{(m)}]$. What is the dimension of X?
    - $(m, 1)$
    - (n_x, m)
      Correct
    - $(1, m)$
    - (m, n_x)
11. Recall that "np.dot(a,b)" performs a matrix multiplication on a and b, whereas "a*b" performs an element-wise multiplication.
    Consider the two following random arrays "a" and "b":

```
a = np.random.randn(12288, 150) # a.shape = (12288, 150)
b = np.random.randn(150, 45) # b.shape = (150, 45)
```

```
c = np.dot(a,b)
```

What is the shape of c?

- c.shape = (12288, 150)
- **c.shape = (12288, 45)**
  Correct
  Correct, remember that a np.dot(a, b) has shape (number of rows of a, number of columns of b). The sizes match because :
  "number of columns of a = 150 = number of rows of b"
- The computation cannot happen because the sizes don't match. It's going to be "Error"!
- c.shape = (150,150)

14. Consider the following code snippet:

```
# a.shape = (3,4)
# b.shape = (4,1)

for i in range(3):
  for j in range(4):
```

```
    c[i][j] = a[i][j] + b[j]
```

How do you vectorize this?

- c = a.T + b
- c = a.T + b.T
- c = a + b
- **c = a + b.T**
  Correct

1. Consider the following code:

```
a = np.random.randn(3, 3)
b = np.random.randn(3, 1)
```

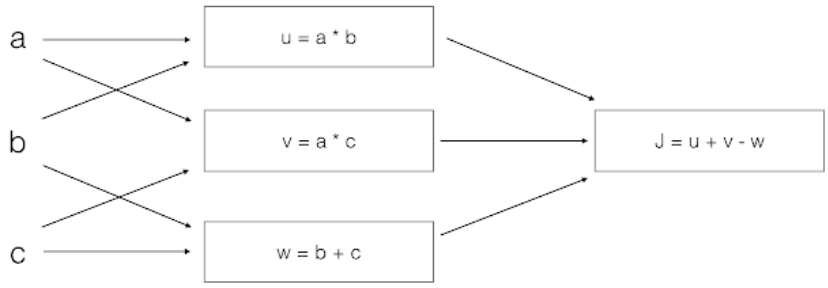
```
c = a*b
```

What will be c? (If you're not sure, feel free to run this in python to find out).

- **This will invoke broadcasting, so b is copied three times to become (3,3), and * is an element-wise product so c.shape will be (3, 3)**
  Correct

- This will invoke broadcasting, so b is copied three times to become (3, 3), and $*$ invokes a matrix multiplication operation of two 3x3 matrices so c.shape will be (3, 3)
- This will multiply a 3x3 matrix a with a 3x1 vector, thus resulting in a 3x1 vector. That is, c.shape = (3,1).
- It will lead to an error since you cannot use "*" to operate on these two matrices. You need to instead use np.dot(a,b)

4. Consider the following computation graph.



What is the output J?

- J = (c - 1)*(b + a)
- **J = (a - 1) * (b + c)**
  Correct
  Yes. J = u + v - w = a*b + a*c - (b + c) = a * (b + c) - (b + c) = (a - 1) * (b + c).
- J = a*b + b*c + a*c
- J = (b - 1) * (c + a)

Neural Networks and Deep Learning (Week 3) Quiz

1. Which of the following are true? (Check all that apply.)
   - X **is a matrix in which each column is one training example.**
     Correct
   - $a^{[2](12)}$ denotes activation vector of the 12th layer on the 2nd training example.
   - $a_4^{[2]}$ is the activation output of the 2nd layer for the 4th training example 4
   - $\boldsymbol{a^{[2](12)}}$ **denotes the activation vector of the 2nd layer for the 12th training example.**
     Correct
   - $\boldsymbol{a_4^{[2]}}$ **is the activation output by the 4th neuron of the 2nd layer 4**
     Correct
   - X is a matrix in which each row is one training example.
   - $\boldsymbol{a^{[2]}}$ **denotes the activation vector of the 2nd layer.**
     Correct

2. The tanh activation usually works better than sigmoid activation function for hidden units because the mean of its output is closer to zero, and so it centers the data better for the next layer. True/False?
   - **True**
     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.

- False

3. Which of these is a correct vectorized implementation of forward propagation for layer l, where $1 \leq l \leq L$?
   - $Z^{[l]} = W^{[l]} A^{[l-1]} + b^{[l]}$
     $A^{[l]} = g^{[l]}(Z^{[l]})$
     Correct
   - $Z^{[l]} = W^{[l-1]} A^{[l]} + b^{[l-1]}$
     $A^{[l]} = g^{[l]}(Z^{[l]})$
   - $Z^{[l]} = W^{[l]} A^{[l]} + b^{[l]}$
     $A^{[l+1]} = g^{[l+1]}(Z^{[l]})$
   - $Z^{[l]} = W^{[l]} A^{[l]} + b^{[l]}$
     $A^{[l+1]} = g^{[l]}(Z^{[l]})$
4. You are building a binary classifier for recognizing cucumbers (y=1) vs. watermelons (y=0). Which one of these activation functions would you recommend using for the output layer?
   - ReLU
   - Leaky ReLU
   - **sigmoid**
     Correct
     Yes. Sigmoid outputs a value between 0 and 1 which makes it a very good choice for binary classification. You can classify as 0 if the output is less than 0.5 and classify as 1 if the output is more than 0.5. It can be done with tanh as well but it is less convenient as the output is between -1 and 1.
   - tanh
5. Consider the following code:

```
A = np.random.randn(4,3)
```

```
B = np.sum(A, axis = 1, keepdims = True)
```

What will be B.shape? (If you're not sure, feel free to run this in python to find out).

- **(4, 1)**
  Correct
  Yes, we use (keepdims = True) to make sure that A.shape is (4,1) and not (4,). It makes our code more rigorous.
- (4,)
- (, 3)
- (1, 3)

7. Suppose you have built a neural network. You decide to initialize the weights and biases to be zero. Which of the following statements is true?
   - **Each neuron in the first hidden layer will perform the same computation. So even after multiple iterations of gradient descent each neuron in the layer will be computing the same thing as other neurons.**
     Correct
   - Each neuron in the first hidden layer will perform the same computation in the first iteration. But after one iteration of gradient descent they will learn to compute different things because we have "broken symmetry".
   - Each neuron in the first hidden layer will compute the same thing, but neurons in different layers will compute different things, thus we have accomplished "symmetry breaking" as described in lecture.

- The first hidden layer's neurons will perform different computations from each other even in the first iteration; their parameters will thus keep evolving in their own way.

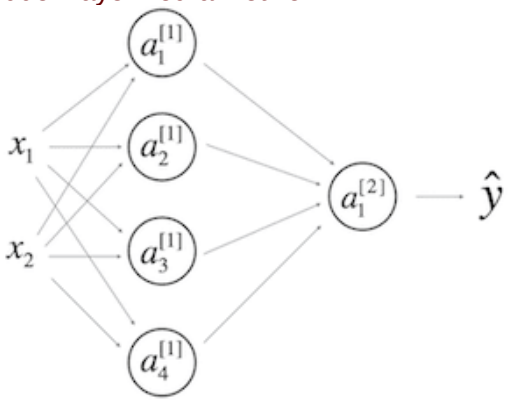
8. Logistic regression's weights w should be initialized randomly rather than to all zeros, because if you initialize to all zeros, then logistic regression will fail to learn a useful decision boundary because it will fail to "break symmetry", True/False?
   - True
   - **False**

     Correct

     Yes, Logistic Regression doesn't have a hidden layer. If you initialize the weights to zeros, the first example x fed in 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.
9. You have built a network using the tanh activation for all the hidden units. You initialize the weights to relative large values, using np.random.randn(..,..)*1000. What will happen?
   - 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 also become large. You therefore have to set α to be very small to prevent divergence; this will slow down learning.
   - It doesn't matter. So long as you initialize the weights randomly gradient descent is not affected by whether the wights are large or small
   - **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.**

     Correct

     Yes. tanh becomes at for large values, this leads its gradient to be close to zero. This slows down the optimization algorithm.
1. Consider the following 1 hidden layer neural network



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

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

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

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

  Correct

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

  Correct

2. In the same network as the previous question, what are the dimensions of $Z^{[1]}$ and $A^{[1]}$?
   - $Z^{[1]}$ and $A^{[1]}$ are (1,4)
   - $Z^{[1]}$ and $A^{[1]}$ are (4,1)
   - **$Z^{[1]}$ and $A^{[1]}$ are (4,m)**

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

Neural Networks and Deep Learning (Week 4) Quiz

1. What is the "cache" used for in our implementation of forward propagation and backward propagation?
   - It is used to cache the intermediate values of the cost function during training.
   - **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.
   - 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.
   - It is used to keep track of the hyperparameters that we are searching over, to speed up computation.
2. Among the following, which ones are "hyperparameters"? (Check all that apply.)
   - **learning rate α**

     Correct
   - **number of layers L in the neural network**

     Correct
   - weight matrices $W^{[l]}$
   - bias vectors $b^{[l]}$
   - **number of iterations**

     Correct
   - activation values $a^{[l]}$
   - **size of the hidden layers $n^{[l]}$**

     Correct
3. 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.**

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

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?
   - 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 layers, 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?

   ```
   for(i in range(1, len(layer_dims)/2)):
     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
   ```

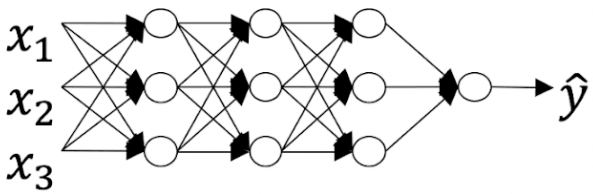
   ```
   for(i in range(1, len(layer_dims)/2)):
     parameter['W' + str(i)] = np.random.randn(layers[i], layers[i-1])) * 0.01
     parameter['b' + str(i)] = np.random.randn(layers[i-1], 1) * 0.01
   ```

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

   ```
   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
   ```

   - **Correct**

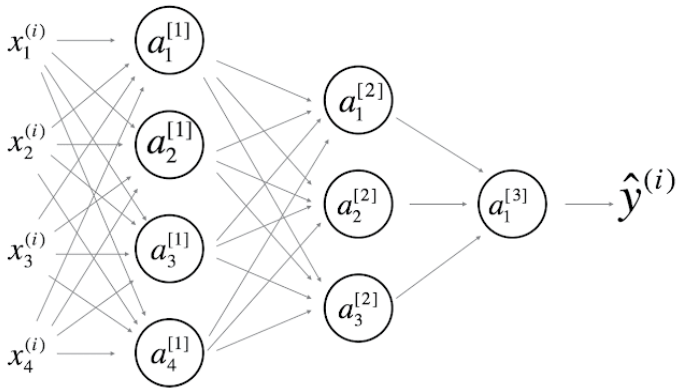
6. Consider the following neural network.



How many layers does this network have?

- **The number of layers 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.
- 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.

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?
   - **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.
   - False
8. 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**
     Correct
   - False
9. Consider the following 2 hidden layer neural network:



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 $W^{[l]}$ is $(n^{[l]}, n^{[l-1]})$.
- **$b^{[1]}$ will have shape (4, 1)**
  Correct
  Yes. More generally, the shape of $b^{[l]}$ is $(n^{[l]}, 1)$.
- $W^{[1]}$ will have shape (3, 4)
- $b^{[1]}$ will have shape (3, 1)
- **$W^{[2]}$ 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)
- $W^{[2]}$ will have shape (3, 1)
- **$b^{[2]}$ will have shape (3, 1)**

  Correct

  Yes. More generally, the shape of $b^{[l]}$ is $(n^{[l]}, 1)$.
- $W^{[3]}$ will have shape (3, 1)
- **$b^{[3]}$ will have shape (1, 1)**

  Correct

  Yes. More generally, the shape of $b^{[l]}$ is $(n^{[l]}, 1)$.
- **$W^{[3]}$ will have shape (1, 3)**

  Correct

  Yes. More generally, the shape of $W^{[l]}$ is $(n^{[l]}, n^{[l-1]})$.
- $b^{[3]}$ will have shape (3, 1)

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?

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

  Correct

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