Neural Network Basics

Quiz, 10 questions

<div style="border:1px solid">

1

point

</div>

1.

What does a neuron compute?

○ A neuron computes a function g that scales the input x linearly (Wx + b)

○ A neuron computes the mean of all features before applying the output to an activation function

○ A neuron computes a linear function (z = Wx + b) followed by an activation function

○ A neuron computes an activation function followed by a linear function (z = Wx + b)

<div style="border:1px solid">

1

point

</div>

2.

Which of these is the "Logistic Loss"?

○ $\mathcal{L}^{(i)}(\hat{y}^{(i)}, y^{(i)}) = -(y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}) \ )$

○ $\mathcal{L}^{(i)}(\hat{y}^{(i)}, y^{(i)}) = |\ y^{(i)} - \hat{y}^{(i)}\ |^2$

○ $\mathcal{L}^{(i)}(\hat{y}^{(i)}, y^{(i)}) = max(0, y^{(i)} - \hat{y}^{(i)})$

○ $\mathcal{L}^{(i)}(\hat{y}^{(i)}, y^{(i)}) = |\ y^{(i)} - \hat{y}^{(i)}\ |$

<div style="border:1px solid">

1

point

</div>

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((32*32,3))

Neural Network Basics

x = img.reshape((32*32*3,1))

○   x = img.reshape((3,32*32))

○   x = img.reshape((1,32*32,*3))

| 1 |
| point |

4.

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

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

| 1 |
| point |

5.

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

Neural Network Basics

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

What will be the shape of "c"?

○ c.shape = (3, 3)

○ c.shape = (4, 3)

○ c.shape = (4,2)

○ The computation cannot happen because the sizes don't match. It's going to be "Error"!

| 1 |
| point |

6.

Suppose you have n_x input features per example. Recall that $X = [x^{(1)} x^{(2)} ... x^{(m)}]$. What is the dimension of X?

○ $(1, m)$

○ (n_x, m)

○ (m, n_x)

○ $(m, 1)$

| 1 |
| point |

7.

Recall that "np.dot(a,b)" performs a matrix multiplication on a and b, whereas "a*b" performs an element-wise

Neural Network Basics
multiplication.

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

```
1   a = np.random.randn(12288, 150) # a.shape = (12288, 150)
2   b = np.random.randn(150, 45) # b.shape = (150, 45)
3   c = np.dot(a,b)
```

What is the shape of c?

○   c.shape = (150,150)

○   c.shape = (12288, 45)

○   The computation cannot happen because the sizes don't match. It's going to be "Error"!

○   c.shape = (12288, 150)

1
point

8.
Consider the following code snippet:

```
1   # a.shape = (3,4)
2   # b.shape = (4,1)
3
4   for i in range(3):
5     for j in range(4):
6       c[i][j] = a[i][j] + b[j]
```

How do you vectorize this?

○   c = a.T + b.T

○   c = a.T + b

○   c = a + b.T

○   c = a + b

1
point

Neural Network Basics

9.
Consider the following code?

Quiz, 10 questions

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

○   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)
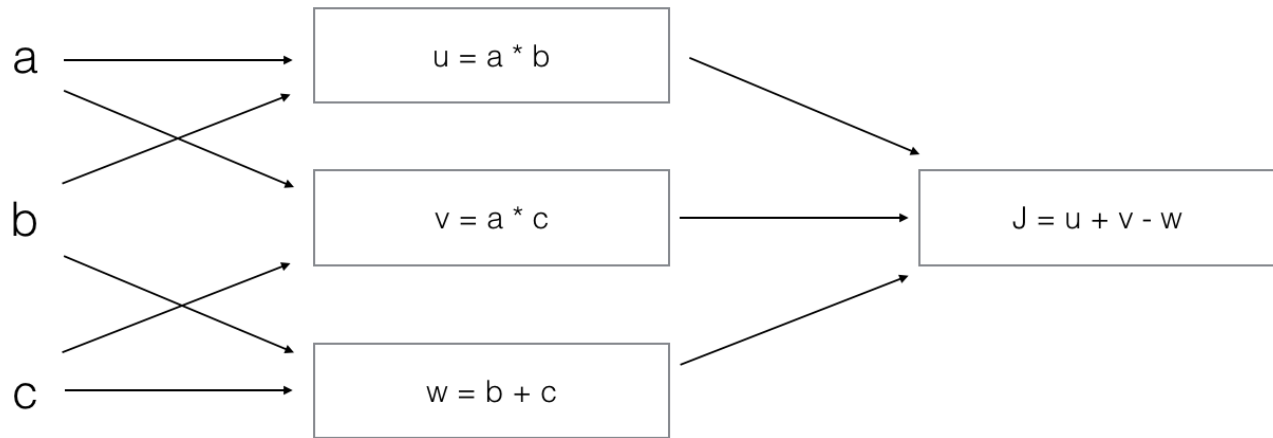

| 1 |
| point |

10.

Consider the following computation graph.

Neural Network Basics

Quiz, 10 questions

What is the output J?

○ J = (c - 1)*(b + a)

○ J = (a - 1) * (b + c)

○ J = a*b + b*c + a*c

○ J = (b - 1) * (c + a)

Upgrade to submit