Your grade: 100%

Your latest: **100%** • Your highest: **100%** • To pass you need at least 80%. We keep your highest score.

Next item →

1. In logistic regression given the input $\mathbf{x}$, and parameters $w \in \mathbb{R}^{n_x}, b \in \mathbb{R}$, how do we generate the output $\hat{y}$? **1 / 1 point**

[✓ $\sigma(W\,\mathbf{x} + b)$.]
$\sigma(W\,\mathbf{x})$
$W\,\mathbf{x} + b$
$\tanh(W\,\mathbf{x} + b)$

Correct
Right, in logistic regression we use a linear function $W\mathbf{x} + b$ followed by the sigmoid function σ, to get an output y, referred to as \hat{y}, such that $0 < \hat{y} < 1$.

2. Suppose that $\hat{y} = 0.9$ and $y = 1$. What is the value of the "Logistic Loss"? Choose the best option. **1 / 1 point**

0.005
[✓ 0.105]
$+\infty$
$\mathcal{L}(\hat{y}, y) = -\left(\hat{y}\ \log y + (1-\hat{y})\ \log(1-y)\right)$

Correct
Yes. Since $\mathcal{L}(\hat{y}, y) = -\left(y\ \log \hat{y} + (1-y)\ \log(1-\hat{y})\right)$, for the given values we get $\mathcal{L}(\hat{y}, y) = -\left(1\ \log 0.9 + 0\ \log 0.1\right)$

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? **1 / 1 point**

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

Correct

4. Consider the following random arrays a and b, and c: **1 / 1 point**

$a = np.random.randn(3,4)$ # $a.shape = (3,4)$

$b = np.random.randn(1,4)$ # $b.shape = (1,4)$

$c = a + b$

What will be the shape of c?

The computation cannot happen because it is not possible to broadcast more than one dimension.
c.shape = (3, 1)
[✓ c.shape = (3, 4)]
c.shape = (1, 4)

Correct
Yes. Broadcasting is used, so row b is copied 3 times so it can be summed to each row of a.

5. Consider the two following random arrays a and b: **1 / 1 point**

$a = np.random.randn(4,3)$ # $a.shape = (4,3)$

$b = np.random.randn(3,2)$ # $b.shape = (3,2)$

$c = a * b$

What will be the shape of c?

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

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).

6. Suppose our input batch consists of 8 grayscale images, each of dimension 8x8. We reshape these images into feature column vectors $\mathbf{x}^j$. Remember that $X = \left[\mathbf{x}^{(1)}\mathbf{x}^{(2)} \cdots \mathbf{x}^{(8)}\right]$. What is the dimension of X?

1 / 1 point

- (8, 8, 8)
- [✓ (64, 8)]
- (8, 64)
- (512, 1)

Correct
Yes. After converting the 8x8 gray scale images to a column vector we get a vector of size 64, thus X has dimension $(64, 8)$.

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

1 / 1 point

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, 45)]
- c.shape = (12288, 150)
- The computation cannot happen because the sizes don't match. It's going to be "Error"!
- c.shape = (150,150)

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"

8. Consider the following code snippet:

1 / 1 point

$a.shape = (4, 3)$

$b.shape = (4, 1)$

```
for i in range(3):
 for j in range(4):
c[i][j] = a[j][i] + b[j]
```

How do you vectorize this?

- c = a.T + b
- c = a + b
- c = a + b.T
- [✓ c = a.T + b.T]

Yes. a[j][i] being used for a[i][j] indicates we are using a.T, and the element in the row j is used in the column j thus we are using b.T.

9. Consider the following code:

1 / 1 point

$a = np.random.randn(3, 3)$

$b = np.random.randn(3, 1)$

$c = a * b$

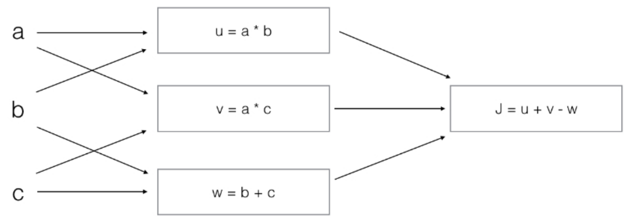
What will be c? (If you're not sure, feel free to run this in python to find out).

- [✓ 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)]
- 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)

Correct

10. Consider the following computation graph.

1 / 1 point



What is the output J?

[✓ $J = (a-1)*(b+c)$]

$J = a*b+b*c+a*c$

$J = (b-1)*(c+a)$

$J = (c-1)*(b+a)$

✓ **Correct**

Yes.

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